Multiple Objective Optimization and Implications for Single Objective Optimization

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

dem Fachbereich C - Mathematik und Naturwissenschaften
Bergische Universität Wuppertal

vorgelegt von

Jochen Gorski
aus Neumarkt in der Oberpfalz

Wuppertal, 2010

Als Dissertation genehmigt vom
Fachbereich C - Mathematik und Naturwissenschaften
der Bergische Universität Wuppertal

Diese Dissertation kann wie folgt zitiert werden:

urn:nbn:de:hbz:468-20101129-120637-5
[http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:hbz:468-20101129-120637-5]

Tag der mündlichen Prüfung:	28. Juli 2010
Vorsitzende der Prüfungskommission:	Prof. Dr. Kathrin Klamroth
Erstgutachter:	Prof. Dr. Kathrin Klamroth
Zweitgutachter:	Prof. Dr. Horst W. Hamacher

Contents

Zusammenfassung

Diese Arbeit beschäftigt sich mit Problemen aus der multikriteriellen Optimierung und beschreibt, wie multikriterielle Optimierungsmethoden zur Lösung von einkriteriellen Problemen genutzt werden können. Während aus traditioneller Sicht meist Lösungsmethoden aus der einkriteriellen Optimierung dazu benutzt werden, effiziente Lösungen eines gegebenen multikriteriellen Problems zu bestimmen, gehen wir in dieser Arbeit den umgekehrten Weg. Hierfür werden zwei unterschiedliche Ansätze verfolgt. Auf der einen Seite benutzen wir Lösungskonzepte aus der multikriteriellen Optimierung direkt dazu, optimale Lösungen für einkriterielle Probleme zu bestimmen. Auf der anderen Seite verfolgen wir das Ziel, Lösungsverfahren für einkriterielle Probleme zu verbessern, indem wir durch eine multikriterielle Beschreibung des gegebenen Problems zusätzliche Informationen in die Problemlösung einfließen lassen.
In den einführenden Kapiteln 1 bis 3 werden zunächst die theoretischen Grundlagen der in dieser Arbeit behandelten Ideen anhand allgemeiner ein- und multikriterieller Optimierungsprobleme vorgestellt. Dabei gehen wir detaillierter auf einkriterielle Probleme ein, die sich als ε-Constraint oder Gewichtete-Summen-Ansatz eines multikriteriellen Problems interpretieren lassen. Zusätzlich präsentieren wir eine erweiterte Suchstrategie für die alternierende Block-Such-Methode für einkriterielle Probleme, die auf einer multikriteriellen Erweiterung des Problems beruht.
Die Arbeit gliedert sich im Anschluss in zwei thematisch unterschiedliche Abschnitte. Der erste Abschnitt (Kapitel 4 bis 10) beschäftigt sich mit *kombinatorischer Optimierung.* Neben den oben genannten Aspekten beleuchten wir hier zusätzlich die Frage des Zusammenhangs der effizienten Menge von kombinatorischen Problemen. Der zweite Abschnitt (Kapitel 11 bis 13) widmet sich hingegen *bikonvexen Optimierungsproblemen.* Dabei steht vor allem das Verbindungsstandort-Problem aus der Standort-Theorie im Mittelpunkt.

Der erste Teil des ersten Abschnitts (Kapitel 4 bis 6) geht v.a. der Frage nach, wie multikriterielle Lösungsansätze zur Bestimmung optimaler Lösungen für einkriterielle kombinatorische Optimierungsprobleme genutzt werden können. Dabei diskutieren wir neben einkriteriellen Problemen mit zusätzlichen Nebenbedingungen auch Probleme, deren Zielfunktion sich als Summe verschiedenartiger Funktionstypen, wie Summen- und Bottleneck-Zielfunktionen, schreiben lässt. Diese Probleme werden in der Literatur oftmals auch als algebraische Summationsprobleme bezeichnet. Hierfür wiederholen wir in Kapitel 4 zunächst die wichtigsten Resultate für Probleme mit Summen- und Bottleneck-Zielfunktionen sowohl für den einkriteriellen als auch für den multikriteriellen Fall. Neben einem kurzen Literaturüberblick präsentieren wir in diesem Kapi-

tel zusätzlich ein verallgemeinertes Lösungsverfahren für multikriterielle Bottleneck-Probleme.
Kapitel 5 beschäftigt sich mit verallgemeinerten Bottleneck-Problemen. Anstelle des größten Kostenkoeffizienten einer zulässigen Lösung soll nun der k.-größte Koeffizient minimiert werden. Dies führt zum Begriff der k-max-Optimierungsprobleme. Wir zeigen für den einkriteriellen Fall, dass sich diese Probleme mit Hilfe einer Folge von binären Summationsproblemen lösen lassen. Für den multikriteriellen Fall präsentieren wir einen modifizierten ε-Constraint-Ansatz, um effiziente Lösungen des Problems zu generieren.
Kapitel 6 widmet sich ausführlicher multikriteriellen Ansätzen zur Bestimmung optimaler Lösungen von einkriteriellen kombinatorischen Optimierungsproblemen. Im ersten Teil werden zunächst einkriterielle Probleme mit einer zusätzlichen Nebenbedingung betrachtet. Wir präsentieren einen verallgemeinerten bikriteriellen Ansatz für dieses Problem, mit dessen Hilfe eine optimale Lösung des Ausgangsproblems bestimmt werden kann. Der zweite Teil des Kapitels beschäftigt sich mit algebraischen Summationsproblemen. Wir zeigen, wie die in den Kapiteln 4 und 5 vorgestellten Algorithmen für multikriterielle Probleme mit Summen-, Bottleneck und k-max-Zielfunktionen dazu benutzt werden können, um diese Probleme zu lösen. Dabei nehmen wir zunächst Bezug auf spezielle Probleme aus der Literatur und diskutieren anschließend daraus abgeleitete, verallgemeinerte Problemstellungen.
Der zweite Teil des ersten Abschnitts (Kapitel 7 bis 10) befasst sich vorwiegend mit Zusammenhangseigenschaften der effizienten Menge von multikriteriellen kombinatorischen Optimierungsproblemen. Da es sich bei kombinatorischen Optimierungsproblemen um diskrete Probleme handelt, benutzen wir graphentheoretische Konzepte um den Zusammenhang der effizienten Menge zu definieren. Ist die effiziente Menge eines gegebenen Problems zusammenhängend, so kann diese mit Hilfe einfacher Algorithmen, basierend auf Nachbarschaftssuche, komplett bestimmt werden. In Kapitel 7 beschäftigen wir uns zunächst mit geeigneten Definitionen der Benachbartheit von effizienten Lösungen. Danach untersuchen wir den Zusammenhang der effizienten Menge für eine Vielzahl klassischer kombinatorischer Probleme und zeigen, dass die effiziente Menge für diese Probleme im Allgemeinen nicht zusammenhängend ist. Anhand einer numerischen Studie für unterschiedliche Varianten von Rucksack-Problemen gehen wir weiterhin auf die Häufigkeit des Auftretens dieses Phänomens ein.
Kapitel 8 beschäftigt sich mit Zusammenhangseigenschaften der effizienten Menge für kombinatorische Probleme mit Bottleneck-Zielfunktionen. Wir weisen anhand eines allgemein gültigen Gegenbeispiels nach, dass auch in diesem Fall die effiziente Menge für viele Klassen von kombinatorischen Problemen nicht zusammenhängend ist. Der zweite Teil des Kapitels 8 widmet sich bikriteriellen Rucksack-Problemen mit Bottleneck-Zielfunktionen. Dabei zeigen wir unter anderem, dass - im Gegensatz zur effizienten Menge - die Menge der schwach-effizienten Lösungen dieses Problems stets zusammenhängend ist.
Die abschließenden Kapiteln 9 und 10 beschäftigen sich mit speziellen Klassen von kombinatorischen Problemen, für die der Nachweis des Zusammenhangs der effizienten Menge erbracht werden kann.
In Kapitel 9 werden dabei zunächst Rucksack-Probleme mit binären Gewichten behandelt. Ausgehend von einem Rucksack-Problem mit zwei Gleichheits-Nebenbedingungen auf den binären Gewichten, entwickeln wir einen Greedy-Algorithmus, mit dessen Hil-

fe die nicht-dominierte Menge des unrestringierten trikriteriellen Optimierungsproblems mit zwei binären Zielfunktionen effizient berechnet werden kann. Anhand numerischer Untersuchungen weisen wir nach, dass es mit Hilfe des Algorithmus unter anderem möglich ist, Instanzen mit über einer Million Items und 180 Milliarden nicht-dominierten Punkten in weniger als 30 Minuten zu lösen. Zusätzlich benutzen wir diesen Algorithmus, um den Zusammenhang der effizienten Menge für dieses spezielle Problem zu beweisen.
Kapitel 10 widmet sich einer speziellen Klasse von bikriteriellen Matroid-Problemen, bei der eine der beiden Zielfunktionen als binär vorausgesetzt wird. Auch für dieses Problem ist es möglich, den Zusammenhang der effizienten Menge mittels eines modifizierten Lösungsverfahrens aus der Literatur nachzuweisen.

Der zweite Abschnitt dieser Arbeit widmet sich bikonvexen Optimierungsproblemen. Eine Funktion heißt dabei bikonvex, wenn sich die gegebene Variablenmenge der Funktion so in zwei disjunkte Teilmengen aufteilen lässt, dass die daraus resultierenden Teilfunktionen jeweils konvex in einem Teil der Variablen sind, sobald der andere Teil der Variablen als fest angenommen wird. Eine Teilmenge des $\mathbb{R}^n$ wird in diesem Zusammenhang als bikonvex bezeichnet, falls sie eine entsprechende Eigenschaft für Mengen erfüllt.
Kapitel 11 gibt zunächst einen generellen Überblick über die wichtigsten Resultate für bikonvexe Mengen, Funktionen und Optimierungsprobleme. Dabei beweisen wir unter anderem, dass sich das Maximum einer bikonvexen Funktion über einer bikonvexen Menge stets auf dem Rand der Menge befindet. Weiterhin untersuchen wir Konvergenzeigenschaften der alternierenden konvexen Suche, die auf dem alternierenden Lösen der induzierten konvexen Teilprobleme basiert. Wir zeigen, dass die Folge der durch diese Methode generierten Punkte unter schwachen Voraussetzungen gegen einen stationären Punkt des Ausgangsproblems konvergiert.
Kapitel 12 beschäftigt sich mit einer erweiterte Suchstrategie für bikonvexe Optimierungsprobleme, die auf den bereits in Kapitel 3 präsentierten Ideen für die alternierende Block-Such-Methode beruht. Dabei wird die spezielle Struktur des bikonvexen Problems genutzt, um durch einen bikriteriellen Ansatz zusätzliche Informationen über das gegebene Problem zu erhalten. Neben den beiden Zielfunktionen der konvexen Unterprobleme verwenden wir dabei zusätzlich Gradienten-Informationen dieser Funktionen als weiteres Kriterium, um die Qualität der aus der alternierenden konvexen Suche (als Spezialfall der alternierende Block-Such-Methode) resultierenden lokalen Optima heuristisch zu verbessern.
Abschließend widmet sich Kapitel 13 dem Verbindungsstandort-Problem in der Ebene. Bei diesem speziellen Problem aus der Standort-Theorie werden Flüsse zwischen existierenden Standorten betrachtet. Jeder dieser Flüsse muss dabei einen Verbindungsknoten durchlaufen. Ziel ist es, einerseits günstige Standorte für die Verbindungsknoten zu wählen, und andererseits eine geeignete Aufteilung der Flüsse auf diese zu bestimmen, so dass entstehende Transportkosten minimiert werden. Wir zeigen zunächst, dass sich das gegebene Standort-Problem als bikonvexes Optimierungsproblem formulieren lässt und diskutieren im Anschluss, wie sich die in Kapitel 12 präsentierte erweiterte Suchstrategie auf dieses Problem anwenden lässt. Wir diskutieren unter anderem mehrere Varianten dieser erweiterten Strategie und vergleichen diese Varianten mit dem ursprünglichen Ansatz der alternierenden konvexen Suche anhand detaillierter, numerischer Tests der vorgeschlagenen Algorithmen.

Abstract

This thesis deals with problems from multiple objective optimization and describes how methods of this field of optimization can be used to solve single objective problems. While traditionally methods from single objective optimization are frequently used to determine efficient solutions of a given multiple objective problem, we want to take the reverse approach in this thesis. Thereby, two different concepts are in the main focus. On the one hand, we use solution concepts from multiple objective optimization to directly derive optimal solutions for single objective problems. On the other hand, we aim to improve existing solution concepts for single objective problems by exploiting additional information induced by a multiple objective description of the considered single objective problem.
In the preliminary Chapters 1 to 3 the theoretical background of the ideas presented in this thesis is discussed based on generalized single and multiple objective problem formulations. We concentrate on single objective problems that can be interpreted as an ε-constraint or a weighted sum version of a multiple objective optimization. In addition, we suggest an enhanced search strategy for the alternate block search method for the single objective problems, that is based on a multiple objective extension of the problem.
Starting from Chapter 4, the remainder of this thesis is partitioned into two parts. The first part (Chapters 4 to 10) deals with *combinatorial optimization problems.* Besides the above mentioned aspects, we additionally discuss the connectedness of the efficient set for this kind of problems. The second part (Chapters 11 to 13) is dedicated to *biconvex optimization.* In this context, we mainly focus on the connection location-allocation problem from location theory.

The first chapters of Part I (Chapters 4 to 6) deal with the question of how multiple objective solution approaches can be used to solve single objective combinatorial optimization problems. We mainly concentrate on constrained versions of single objective problems as well as on problems, where the objective function is given as the sum of different types of objectives like sum and bottleneck functions. The latter problems are frequently called algebraic sum problems in the literature. For this purpose, we recall the most important results for combinatorial optimization problems with sum and bottleneck objectives for the single as well as the multiple objective case in Chapter 4. In addition to a short survey of the existing literature, we present a generalized solution approach for multiple objective combinatorial problems with bottleneck objectives.
Chapter 5 is dedicated to generalized bottleneck problems. Instead of minimizing the largest cost coefficient of a feasible solution, the k^{th} largest coefficient has to be

minimized. This leads to the notion of a k-max optimization problem. We show that an optimal solution for the single objective problem can be determined by solving a sequence of single objective binary sum problems. For the multiple objective case, we use a modified ε-constraint approach to generate efficient solutions of the considered problem.

Chapter 6 is devoted to single objective combinatorial problems with one side constraint as well as problems with algebraic sum objectives. We show how algorithms for multiple objective problems with sum, bottleneck or k-max objective that have been presented in the Chapters 4 and 5 can be used to solve this kind of problems. Based on special classes of algebraic sum problems stated in the literature, we present solution approaches for generalized versions of these problems.

The remaining chapters of Part I (Chapters 7 to 10) are dedicated to the connectedness of the efficient set for combinatorial optimization problems. As combinatorial problems belong to the class of discrete optimization problems, we use concepts from graph theory to define the connectedness of the efficient set. When the efficient set of a given problem is known to be connected, the set itself can be determined by means of simple local search techniques. In Chapter 7 we discuss appropriate definitions for the adjacency of efficient solutions. We further show that the efficient set for most of the classical problems from combinatorial optimization is not connected in general. In addition, we perform numerical tests for different variants of the knapsack problem to analyze the likelihood that problems with non-connected efficient set occur in randomly generated problem instances.

The first part of Chapter 8 deals with the connectedness of the efficient set for combinatorial problems with bottleneck objectives. We give a general counter-example that shows that also in this case a non-connected efficient set can be expected for many classes of combinatorial problems. In the second part of this chapter, the biobjective binary knapsack problem with bottleneck objectives is discussed in more detail. We prove, amongst others, that the set of weakly-efficient solutions - in contrast to the efficient set - is always connected for this specific problem.

Chapters 9 and 10 are dedicated to special classes of combinatorial problems for which the connectedness of the efficient set can be proven. In Chapter 9 a class of knapsack problems with binary weights is discussed. Based on a knapsack problem with equality constraints on the binary weights, we develop a greedy algorithm that can be used to efficiently determine the non-dominated set of the associated triobjective unconstrained combinatorial optimization problem with two binary objectives. A numerical study shows that the algorithm can be used to solve instances with more than one million items and 180 billions non-dominated solutions within less than 30 minutes of CPU-time. In addition, we use this algorithm to prove the connectedness of the efficient set for this special type of optimization problem.

Chapter 10 deals with biobjective optimization problems on matroids where one of the two objectives is assumed to take binary values only. For this problem, the connectedness of the efficient set can be proven by means of a modified version of an algorithm that can be found in the literature for matroid intersection problems.

The second part of this thesis is dedicated to biconvex optimization. A function is called biconvex, if its set of variables can be partitioned into two disjoint blocks such that the resulting two subfunctions are convex with respect to one block if the other block is assumed to be fixed. In this context, a subset of $\mathbb{R}^n$ is called biconvex if it

satisfies an analogous property for sets.
At first, Chapter 11 surveys the most important results on biconvex sets, functions and optimization problems. Amongst others, we show that the maximum of a biconvex function on a biconvex set is always attained on the boundary of the given set. In addition, we discuss convergence properties of the alternate convex search method that is based on alternately solving the induced convex subproblems. We show that under mild assumptions on the problem the sequence of generated points converges to a stationary point.
An enhanced search strategy for biconvex optimization problems, that is based on the ideas for the alternate block search method already discussed in Chapter 3, is presented in Chapter 12. Thereby, the special structure of the biconvex problem is used to obtain additional information that is induced by a biobjective interpretation of the given problem. In addition to the two objectives of the respective convex subproblems we additionally use gradient information of these functions as further objectives to heuristically improve the quality of the local minima that result from applying the alternate convex search method that can be seen as a special version of the more general alternate block search method.
The connection location-allocation problem in the $\mathbb{R}^2$-plane is finally treated in Chapter 13. In this problem from location theory we are given flows between existing facilities where each flow must additionally pass a connection facility. The aim is to find favorable locations for these connection facilities and low-priced allocations of the given flows to these locations in order to minimize transportation costs. Initially, we show that this problem can be formulated as a biconvex optimization problem. We further discuss how the search strategies presented in Chapter 12 can be applied to this specific problem. Amongst others, we present several enhanced versions of this method and compare these variants to the original alternate convex search approach by means of detailed numerical tests of the proposed algorithms.

Chapter 1

Introduction

Taking the right decisions is one of the main aspects in everyday life. While most of the decisions we are faced with during the day, like which clothes to wear or when to have lunch, can be taken by routine and only affect the next moments of our life, others do not only have a deep impact on our own well-being, but also on the well-being of others. If a decision has to be made with respect to only a single criterion, it is often quite simple to find a satisfying solution for the problem.

However, only in rare cases important decisions are influenced by a single criterion. Often several independent and conflicting aspects have to be taken into account. Based on these different criteria, one is faced with the problem to find the ''best'' alternative among many possible decisions.

From a mathematical point of view, the notion of optimality for a problem that depends on a single criterion is straightforward. The situation changes if more than one objective is considered. Given two feasible solutions, it is not immediately clear which one we have to choose, when the first solution is better with respect to criterion A, while the second is more satisfactory for criterion B. However, if the first solution is at least as good as the second in all the considered criteria and strictly better in at least one, we rather choose the first solution but not the second.

This approach leads to the concept of Pareto optimality for multiple objective optimization problems, formally introduced by Vilfredo Pareto and Francis Edgeworth in the late 19^{th} century. In this context, a feasible solution is called (Edgeworth-)Pareto optimal or efficient if it cannot be improved with respect to one criterion without worsening at least one other criterion. While single objective optimization is in the main focus of research since a long time, multiple objective optimization was rarely discussed until the middle of the last century.

A common approach in multiple objective optimization is to solve a sequence of scalarizations to associated single objective optimization problems in order to generate different efficient solutions of the multiple objective problem. The ideas of these approaches go back to the middle of the last century.

In contrast, solution concepts from multiple objective optimization are scarcely used to solve single objective optimization problems, since multiple objective problems are generally as least as hard to solve as their single objective counterparts. However, multiple objective optimization yields a good basis to generalize and combine seemingly different approaches from single objective optimization, as we will show in this thesis.

Due to the growing interest in multiple objective optimization, the research on alternative solution concepts for these kind of problems has increased during the last decades. Nowadays, there exists a vast number of different solution concepts, both exact and heuristic, that can be applied to solve multiple objective problems.
In addition to solution approaches for continuous problems, also new solution concepts for multiple objective versions of discrete and combinatorial optimization problems are in the focus of current research. While many of the classical combinatorial problems can be treated efficiently, when a single objective problem is considered, their multiple objective counterparts can be solved exactly only for small to medium sized instances in a reasonable amount of time. Often, a deeper insight into the specific structure of the considered multiple objective combinatorial problem is necessary, to derive appropriate and satisfying solution concepts to efficiently derive non-dominated solutions for the considered problem. However, if such solution concepts exist, we show how they can be used to improve existing and to derive new solution concepts for single objective problems, not only limited to the combinatorial case.

Outline of this Thesis

The aim of this thesis is twofold. On the one hand, we investigate structural properties of the efficient set of multiple objective combinatorial optimization problems. These specific properties can be used to derive fast and efficient solution methods that no longer depend on any scalarizations of the considered multiple objective problem. On the other hand, we show how ideas and solution techniques from multiple objective optimization can be used to gain new insight into the structure and solution concepts for special classes of single objective problems. In this context, we do not restrict ourselves to combinatorial problems only, but we further show, how ideas from continuous multiple objective optimization can be used to improve existing solution approaches for special types of problems from location theory.
The content of this dissertation can be divided into two main parts focusing on the two topics stated above: Part I (Chapters 4 to 10) is dedicated to *combinatorial optimization*, while we consider *biconvex optimization problems* in Part II (Chapters 11 to 13) of this thesis.

After a short introduction to the most important concepts and definitions from single and multiple objective optimization in Chapter 2, we present the theoretical background of our approaches in Chapter 3. While we mainly concentrate on combinatorial and biconvex optimization problems in the remainder of this thesis, we already discuss the main concepts and ideas of how we use multiple objective optimization to solve single objective problems in this preliminary chapter. There, we do not focus on any special types of optimization problems, but we discuss how constrained single objective problems as well as problems, where the objective is given as the weighted sum of different types of functions can be solved by means of a multiple objective approach. In addition, we present an enhanced version of the alternate block search strategy that is frequently used to calculate local optima for general, non-linear optimization problems. Given a partitioning of the set of variables into several disjoint blocks, the optimization problem is iteratively solved in a single block of variables, while the remaining blocks are fixed. We introduce a multiple objective-based ap-

proach to improve this solution method. Besides the original objective function, we try to exploit additional information on the potential improvement of the objective value with respect to the variables that are kept fix during the solution process. We introduce the notion of the descent potential to measure this information.

Part I of this thesis deals with combinatorial optimization. Chapter 4 gives a short introduction to the most important definitions and general concepts from this field of optimization. We summarize the results on combinatorial optimization problems with sum objectives especially for the multiple objective case, and we additionally present a generalized solution approach for multiple objective combinatorial problems involving bottleneck objectives.
In the subsequent Chapter 5 we consider a generalized notion of a bottleneck objective. Given a combinatorial problem, we are no longer interested in minimizing the largest cost coefficient of a feasible solution, which corresponds to a bottleneck problem, but we rather want to optimize the k^{th} largest cost coefficient contained in a feasible solution. This leads to the notion of a k-max objective. We show that the resulting single objective optimization problem can be solved in polynomial time, whenever this holds true for an associated combinatorial problem with a binary sum objective. In addition, we present an efficient solution method for solving multiple objective k-max optimization problems with an additional sum objective. We further discuss the notion of a k-min objective and show that problems involving this objective can be solved by means of the same concepts as for the k-max case.
In Chapter 6 we first summarize some ideas from Chapter 3, and discuss how special types of single objective combinatorial problems can be solved by means of approaches from multiple objective optimization. In the first part of the chapter, we generalize multiple objective-based solution concepts from the literature that are frequently used to solve single objective combinatorial optimization problems with an additional side constraint. The second part of the chapter deals with combinatorial problems with algebraic sum objectives. We apply solution approaches from Chapters 4 and 5 to show, how these types of problems can be solved using solution concepts from multiple objective optimization. In addition, we summarize the literature for three special classes of algebraic sum problems and discuss generalized versions of these problems. We show that all these problems can be modeled and solved by means of associated multiple objective k-max optimization problems.
The remainder of the Part I is dedicated to the connectedness of the efficient set of multiple objective combinatorial optimization problems. When the efficient set of a given class of combinatorial problems is known to be connected independently from the considered instance, the complete efficient set of a given problem instance can be determined by applying a simple local search strategy that is based on the adjacency of the efficient solutions.
We investigate classical combinatorial optimization problems in Chapter 7. Based on two different definitions for the adjacency of efficient solutions, we show that the property of a connected efficient set does not hold for the considered classes in general. We further provide numerical studies on the occurrence of an unconnected efficient set for different variants of the knapsack problem.
While we focus on connectedness of the efficient set for combinatorial problems with sum objectives in Chapter 7, Chapter 8 addresses multiple objective problems with bottleneck objectives. While we have to treat each class of combinatorial problems

in Chapter 7 separately, we present a generalized counter-example for combinatorial problems with multiple bottleneck objectives in Chapter 8 that can be applied to several classes simultaneously. In addition, we develop an algorithm for the biobjective knapsack problem with bottleneck objectives. Based on this algorithm, we prove that the set of weakly efficient solutions is always connected for this specific type of knapsack problem. In addition, we derive a non-trivial sufficient condition for the connectedness of the efficient set for this special class of knapsack problems.
Chapter 9 deals with single objective knapsack problems with binary weights. We introduce a simple greedy strategy to solve the single objective problem with two equality constraints on the binary weights. We further make use of this approach to derive a greedy algorithm for the unconstrained triobjective optimization problem with two binary objectives, and show that our algorithm is optimal in terms of the expected worst case time complexity for the considered problem. Based on the correctness of our approach, we are able to prove the connectedness of the efficient set for this special type of triobjective optimization problem. In addition, we use an adapted variant of the algorithm for the triobjective problem to solve the knapsack problem with inequality constraints on the binary weights.
Matroid problems are considered in Chapter 10, where one of the two objectives is assumed to take binary values only. We present a modified version of an algorithm known from the literature that was originally developed to solve the problem with an equality constraint on the binary objective. We further use this algorithm to prove the connectedness of the efficient set of this specific problem. In more detail, we show that the set of efficient solutions consists of supported efficient solutions only.

Part II of this dissertation is devoted to biconvex optimization problems. A real-valued function is called biconvex if there exist two disjoint blocks of variables such that the given function is convex with respect to one block of variables while the other remains fixed. We summarize the most important properties of biconvex sets, functions and optimization problems in Chapter 11. Amongst others, we state a new and interesting result for biconvex maximization problems on biconvex sets. In more detail, we show that the maximum of a biconvex function on a biconvex set is always attained on the boundary of the given set. In addition, we provide a detailed analysis of existing solution concepts for biconvex minimization problems that exploit the biconvex structure of a given problem. We mainly concentrate on the alternate convex search that can be seen as a special case of the alternate block search strategy, already introduced in a more general framework in Chapter 3.
Chapter 12 relates the idea of the enhanced search technique for the alternate block search method presented in Chapter 3 to the alternate convex search method for biconvex optimization problems introduced in Chapter 11. We use gradient information to measure the descent potential with respect to the block of variables that remain fixed during consecutive iterations of the alternate convex search method. In addition, we discuss how the resulting augmented alternate convex search method can be used to solve biconvex optimization problems in practice.
Finally, Chapter 13 is devoted to a special biconvex optimization problem from location theory. In more detail, we focus on the connection location-allocation problem in the plane. Based on the biconvex structure of the problem, the alternate convex search method (also called location-allocation algorithm in this context) is frequently used to derive local minima for the given problem. We show that the performance of this

algorithm can be heuristically improved if the enhanced version of the alternate block search strategy that has been developed in Chapter 12 is used. In addition, we provide detailed numerical results on different versions of the augmented location-allocation algorithm presented in this chapter and compare these enhanced approaches to the original version of the algorithm.

Chapter 2

Basic Notation and Concepts

The focus of this thesis is multiple objective optimization and its implications to single objective optimization, especially for biconvex and combinatorial optimization problems. It is assumed that the reader is familiar with the basic ideas of linear and non-linear optimization, convex optimization, discrete and combinatorial optimization, as well as multiple objective optimization. For a deeper insight into these topics, we refer, for example, to the books of Bazaraa et al. [13], Boyd and Vandenberghe [25], Ehrgott [54], Hamacher and Klamroth [92], Miettinen [143], Nemhauser and Wolsey [150] and Steuer [197]. For the basic ideas on computational complexity issues, we refer to the book of Garey and Johnson [71]. In the following, we give a short introduction to the most important concepts and definitions from single as well as multiple objective optimization. Furthermore, we introduce the basic notation, that is used in the remainder of this work.

A *single objective optimization problem* (SOP) is a problem of the form

$$\begin{aligned} &\min\ f(x) \\ &\text{s.t.}\ \ x \in X, \end{aligned} \tag{SOP}$$

with non-empty *feasible set* $X \subseteq \mathbb{R}^n$ and real-valued *objective function* $f : X \to \mathbb{R}$. For SOP a point $x \in X$ is called *optimal* or *optimal solution* if $f(x) \leq f(x')$ for all $x' \in X$. Moreover, $x \in X$ is called *local optimal* or *local optimal solution* if there exists a neighborhood $U_x \subseteq X$ containing x in its interior such that $f(x) \leq f(x')$ for all $x' \in U_x$.

A set $X \subseteq \mathbb{R}^n$ is called *convex* if for $x^1, x^2 \in X$ it holds that $\lambda x^1 + (1-\lambda)x^2 \in X$ for all $\lambda \in [0,1]$. A function $f : X \to \mathbb{R}$ is called *convex* if $f(\lambda x^1 + (1-\lambda)x^2) \leq \lambda f(x^1) + (1-\lambda) f(x^2)$ for all $x^1, x^2 \in X$ and $\lambda \in [0,1]$. An SOP is called *convex* if both f and X are convex. For a detailed introduction to biconvex optimization, we refer to Chapter 11.

A *multiple objective optimization problem* (MOP) or *multiple criteria optimization problem* is given by

$$\begin{aligned} &\min\ f(x) = (f_1(x), \ldots, f_p(x))^\top \\ &\text{s.t.}\ \ x \in X, \end{aligned} \tag{MOP}$$

with vector-valued *objective function* $f : X \to \mathbb{R}^p$ and *feasible set* $X \subseteq \mathbb{R}^n$, where $\mathbb{R}^n$ is also called *decision space*. f consists of $p \in \mathbb{N}$ real valued objectives $f_i : X \to \mathbb{R}$ for

$i = 1, \ldots, p$. The set $Y := \{f(x) : x \in X\} = f(X) \subseteq \mathbb{R}^p$ denotes the image of the feasible set in the *objective space* $\mathbb{R}^p$ and is called *set of attainable outcomes*. For the case that $p = 2$, we refer to a *biobjective optimization problem*. If $p = 3$, we call MOP a *triobjective optimization problem*. Note that in the case that $p = 1$, MOP simplifies to SOP with objective function f.
Since there does not exist a canonical ordering on the Euclidian vector space $\mathbb{R}^p$ whenever $p \geq 2$, we use the following concepts of *componentwise ordering*:

$$\begin{aligned} y^1 \leqq y^2 &:\Leftrightarrow y_i^1 \leq y_i^2, \quad i = 1, \ldots, p, \\ y^1 \leq y^2 &:\Leftrightarrow y_i^1 \leq y_i^2, \quad i = 1, \ldots, p \quad \text{and} \quad y^1 \neq y^2, \\ y^1 < y^2 &:\Leftrightarrow y_i^1 < y_i^2, \quad i = 1, \ldots, p. \end{aligned}$$

Following the Pareto concept of optimality, we say that a feasible point $x^1 \in X$ *dominates* a point $x^2 \in X$ if $f(x^1) \leq f(x^2)$. If strict inequality holds for all p components, i.e. $f(x^1) < f(x^2)$, then x^1 *strongly dominates* x^2. If there does not exist any feasible point that dominates $x \in X$, we say that x is an *efficient solution* of MOP. For the case that there exists no feasible point that strongly dominates $x \in X$, we say that x is *weakly-efficient* for MOP. If there is no $x^2 \in X$, $x^2 \neq x^1$, such that $f(x^2) \leqq f(x^1)$, x^1 is called *strictly efficient*. Note that strict efficiency is the multiple objective analogon to unique optimal solutions for SOP. According to these definitions, the *efficient set* X_E and the *weakly efficient set* X_{wE} are defined by

$$\begin{aligned} X_E &:= \{x \in X : \text{there exists no } \bar{x} \in X \text{ with } f(\bar{x}) \leq f(x)\}, \\ X_{wE} &:= \{x \in X : \text{there exists no } \bar{x} \in X \text{ with } f(\bar{x}) < f(x)\}. \end{aligned}$$

Using the vector-valued mapping f, the images of these sets $Y_N := f(X_E)$ and $Y_{wN} := f(X_{wE})$ are called the *non-dominated set* and the *weakly non-dominated set*, respectively. In this context, a point $y^2 \in \mathbb{R}^p$ is called *dominated* by $y^1 \in \mathbb{R}^p$ if $y^1 \leq y^2$, and it is called *strongly dominated* by y^1 if $y^1 < y^2$ holds. The non-dominated set Y_N is called *externally stable* if for each dominated point $y \in Y$ there exists a non-dominated point $y' \in Y_N$ that dominates y. In continuous optimization Y_N is known to be externally stable if $Y \subseteq \mathbb{R}^p$ is non-empty and compact, i.e. closed and bounded (cf. Ehrgott [54]). For example, compactness of Y is given, when X is compact and the considered p objective functions are continuous (cf. Forster [65]). For the case that a discrete optimization problem is considered, Y_N is externally stable if, for example, Y consists of a finite set of singletons.
Let $y^1 = (y_1^1, \ldots, y_p^1)$ and $y^2 = (y_1^2, \ldots, y_p^2)$ be two vectors. The *lexicographical ordering* "$\leq_{\text{lex}}$" is defined as

$$y^1 \leq_{\text{lex}} y^2 :\Longleftrightarrow y^1 = y^2 \text{ or } y_i^1 < y_i^2 \text{ for } i = \min\{j : y_j^1 \neq y_j^2, j \in (1, \ldots, p)\},$$

where it is assumed that the index vector $(1, \ldots, p)$ corresponds to an ordered tuple of p integers. Let $\pi = (\pi(1), \ldots, \pi(p))$ be any permutation of $(1, \ldots, p)$. The vector $x^1 \in X$ is said to be *lexicographically optimal* with respect to π if there does not exist another vector $x^2 \in X$ such that $f_\pi(x^1) \leq_{\text{lex}} f_\pi(x^2)$, where $f_\pi(x^1) = (f_{\pi(1)}(x^1), \ldots, f_{\pi(p)}(x^1))$ and $f_\pi(x^2) = (f_{\pi(1)}(x^2), \ldots, f_{\pi(p)}(x^2))$. Note that if x^1 is lexicographically optimal for an arbitrary permutation π, x^1 is also efficient.

Let $y_i^I = \inf\{f_i(x) : x \in X\}$ for all $i \in \{1, \ldots, p\}$. Then the point $y^I := (y_1^I, \ldots, y_p^I)$ is called the *ideal point*. Given $y_i^N = \min\{f_i(x) : x \in X_E\}$ for all $i \in \{1, \ldots, p\}$, then the point $y^N := (y_1^N, \ldots, y_p^N)$ is said to be the *Nadir point* of the given problem. Note that while y^I can be computed by solving p independent single objective problems, the computation of y^N involves an optimization over the efficient set. No efficient methods for this problem are known for general MOPs, whenever $p > 2$ (cf. Ehrgott [54]).
For the case that f consists of p linear objective functions and X forms a polyhedral set in $\mathbb{R}^n$, MOP is called *multiple objective linear programming problem* (MLP). If further integrality conditions are added to the feasible set X, we refer to a *multiple objective integer linear programming problem* (MILP) or a *discrete* MOP in a more general setting. For a more detailed introduction to (multiple objective) combinatorial optimization, we refer to Chapter 4.
For $\lambda \in \mathbb{R}^p_{\geqq} := \{y \in \mathbb{R}^p : y_i \geq 0\}$, the single objective optimization problem

$$\begin{aligned} \min \ & \sum_{i=1}^{p} \lambda_i f_i(x) \\ \text{s.t.} \ & x \in X, \end{aligned} \tag{P_λ}$$

is called *weighted sum (scalarization) problem* of MOP. It is well-known that each optimal solution of this problem is efficient, whenever $\lambda \in \mathbb{R}^p_{>} := \{y \in \mathbb{R}^p : y_i > 0\}$ holds (cf. Ehrgott [54]). This being the situation, we call an efficient solution $x \in X_E$ *supported efficient (solution)* if there exists $\lambda \in \mathbb{R}^p_{>}$ such that x optimal for Problem (P_λ). Its corresponding image $y = f(x)$ in the objective space is called *supported non-dominated (solution)*. Otherwise, x and y are called *non-supported (solutions)*. Already Geoffrion [78] and Isermann [108] showed that every efficient solution of MLP corresponds to an optimal solution of Problem (P_λ) for an appropriate $\lambda \in \mathbb{R}^p_{>}$. Note that this is no longer the case when general non-convex optimization problems are considered. In this context it is also important to notice that for MILPs unsupported efficient solutions may exist, even for the case that the constraint matrix of the considered problem is known to be totally unimodular.
Let $j \in \{1, \ldots, p\}$ be arbitrary but fixed and let $\varepsilon \in \mathbb{R}^p$. Then the single objective optimization problem

$$\begin{aligned} \min \quad & f_j(x) \\ \text{s.t.} \quad & f_i(x) \leq \varepsilon_i \quad i = 1, \ldots, p, \, i \neq j, \\ & x \in X, \end{aligned} \tag{P_ε}$$

is called *ε-constraint problem* (with respect to the right hand side vector $\varepsilon \in \mathbb{R}^p$). Note that the component ε_j of this vector is not relevant for solving Problem (P_ε). A first extensive discussion of this problem can be found in Chankong and Haimes [36]. It is well-known that every optimal solution of Problem (P_ε) is at least a weakly efficient solution of the corresponding MOP, and that a feasible solution $x \in X$ is efficient for MOP if and only if there exists a vector $\varepsilon \in \mathbb{R}^p$ such that x is optimal solution of Problem (P_ε) for all $j = 1, \ldots, p$ (cf. Ehrgott [54]). In contrast to the weighted sum method, all efficient solutions can be generated by applying the ε-constraint method to MOP, even in the case of a general non-linear multiple objective optimization problem.

Chapter 3

From Single to Multiple Objective Optimization and Back - General Ideas of this Work

From a historical point of view, multiple objective optimization is mostly considered as a generalization of single objective optimization, since methods from single objective optimization are frequently used to develop new solution concepts for solving multiple objective optimization problems. However, we take the reverse approach in this thesis and show how concepts and solution concepts from multiple objective optimization can be used to gain a new insight into the structure of single objective optimization problems. In this context, we mainly focus on combinatorial optimization problems in the first part and on biconvex optimization problems in the second part of this work, respectively.

However, the theoretical background of our ideas is not limited to these two special classes of optimization problems only, but can also be applied to many problems from other fields of optimization. Hence, we do not restrict ourselves to special types of problems in the following, but we develop the theoretical background of our approaches that are presented in the remainder of this work based on general problem formulations. Hereby, we mainly focus on two different aspects in the following.

In the first two sections of this chapter, we show how multiple objective approaches can be used to solve constrained single objective optimization problems as well as optimization problems where the objective function corresponds to a weighted sum of different sub-objectives. Based on the single objective problem description, we formulate an *associated* multiple objective optimization problem and show that there exists at least one efficient solution of this multiple objective problem that is also optimal for the original single objective problem, as long as a weak assumption on the non-dominated set of the associated multiple objective problem is met. The general idea of our approach is also illustrated in Figure 3.1.

In a third section, we discuss a heuristic multiple objective approach for solving single objective problems. For these problems it is assumed that the set of variables can be partitioned into disjoint blocks of variables such that the resulting single objective subproblems can be solved efficiently, when only a single block of variables is optimized, while the remaining blocks are fixed. The main idea of our heuristic approach is trying

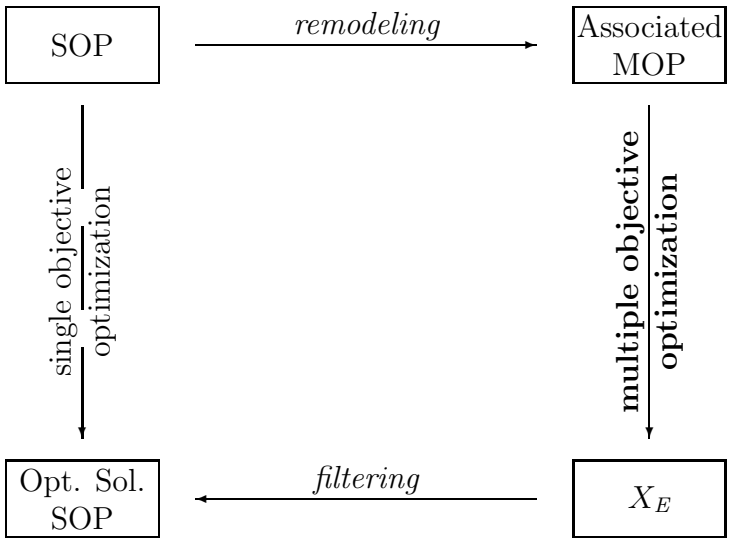


Figure 3.1: *Instead of solving the SOP directly, it is remodeled as an MOP. The optimal solution of the single objective problem is obtained by filtering the efficient set of the associated multiple objective problem for this solution.*

to exploit additional descent information that is implicitly contained in the blocks of fixed variables. This information is normally disregarded, when the given problem is solved with respect to the active variables.
While the impact of multiple objective optimization on single objective optimization problems was not in the focus of research until the end of the last millennium, the number of publications in this field of research has increased over the last decade. However, the literature mainly concentrates on constrained single objective optimization problems. In this context, especially in the field of evolutionary algorithms, multiple objective approaches are frequently used to handle these types of problems.
We omit a detailed review of the existing literature related to this topic, since a comprehensive survey was given by Mezura-Montes and Coello Coello [141] in the book of Knowles et al. [117]. We rather recall from this survey that the evolutionary methods stated in the literature can be partitioned into two different classes of approaches, based on the way how the given constrained problem is transformed into a multiple objective problem:
On the one hand, there exist methods that transform the given problem into an unconstrained *biobjective* optimization problem. Besides the original objective, the sum of constraint violations is used as an additional criterion. On the other hand, there exist solution techniques that treat each given constraint function as an additional objective. Hence, instead of the constrained problem, an unconstrained multiple objective optimization problem has to be solved. We also make use of this idea in Section 3.1 of this chapter.
From a more theoretical point of view, Klamroth and Tind [114] discussed how the most common solution techniques from multiple objective optimization are related to constrained single objective optimization problems and vice versa. Starting from a constrained problem, an associated unconstrained multiple objective optimization problem is formulated. Amongst others, the authors showed that many classical solu-

tion approaches have correspondences in the respective classes of optimization problems. Furthermore, an approximation scheme, based on the results in Klamroth et al. [115] was presented that can be used to approximate the optimal solutions as well as the Lagrange multipliers of convex constrained programming problems.

The close relationship between multiple objective optimization and constrained single objective optimization was also discussed for several other types of optimization problems. Amongst others, Fletcher and Leyffer [60] used the idea of accumulating the sum of constraint violations in an additional objective. Instead of solving the constrained problem, the authors considered the biobjective problem involving the original objective and the sum of constraint violations. The authors suggested a filter SQP method for non-linear programming problems to solve the biobjective problem. Amongst others, this method additionally makes use of dominance relations between the two given objective functions.

Carosi et al. [33] related vector optimization to semidefinite optimization, and finally the connections between Lagrangian relaxations and multiple objective optimization was discussed in Boyd and Vandenberghe [25]. For a further review on multiple objective approaches for solving constrained *combinatorial* optimization problems, we refer to Chapter 6.

In addition to constrained optimization problems also multiple objective approaches for solving classical single objective combinatorial optimization problems were presented in the literature. Neumann and Wegener [151] discussed an evolutionary multiple objective approach that solves the minimum spanning tree problem. Besides the original objective function, the authors used the number of connected components induced by a given set of edges as additional objective in the fitness function of their evolutionary algorithm. By a rigorous asymptotic analysis of the expected optimization time, they showed that their approach is superior to other "single objective" evolutionary algorithms in the case of randomly chosen dense graphs.

In Neumann and Wegener [152] the authors additionally investigated, whether evolutionary multiple objective approaches can be used to solve the single-source shortest path problem. The authors used a simple evolutionary algorithm to derive an optimal solution for the problem. The fitness function of their algorithm is based on a multiple objective representation of feasible paths from the start node to all other nodes contained in the graph. The authors concluded that their approach is an efficient heuristic that is able to solve the single-source shortest path problem within an expected optimization time of $\mathcal{O}(n^3)$.

In contrast to constrained optimization problems, multiple objective optimization solution approaches were rarely applied to single objective optimization problems, where the objective is given as a weighted sum of different sub-objectives. The literature mainly concentrates on the field of combinatorial optimization. For a detailed discussion of the related ideas, we refer to Chapter 6 of this work.

The remainder of this chapter is organized as follows. In Sections 3.1 and 3.2 we present the theoretical background for solving constrained as well as weighted sum type single objective optimization problems based on multiple objective solution approaches. In the subsequent section, the main ideas of an enhanced version of the alternate block search method that is based on a multiple objective approach are discussed. We finally summarize our results in Section 3.4 and give a further outlook on how the remainder of this work is organized with respect to the general ideas presented in this chapter.

3.1 Constrained Single Objective Optimization Problems

In order to generate efficient solutions of a multiple objective optimization problem, a common technique is to transform the given multiple objective problem into a related single objective optimization problem by means of an appropriate scalarization, like the ε-constraint method, presented in Chankong and Haimes [36] (cf. also Chapter 2). The aim of this section is to reverse the traditional relation between the multiple objective problem and its related single objective ε-constraint problem formulation. Given the constrained version of a single objective optimization problem, we show how ideas and approaches from multiple objective optimization can be used to derive an optimal solution for these types of optimization problems. From the introduction of this chapter we recall that the ideas presented in the section were already applied to several types of optimization problems in the literature. In addition, Klamroth and Tind [114] investigated interrelation between constrained optimization and multiple objective optimization in a more general framework. We mainly follow their ideas in the remainder of this section.
Let $f, g_1, \ldots, g_p : \mathbb{R}^n \to \mathbb{R}$ denote $p+1$ real-valued objective functions, where $p \geq 1$, and let $X \subseteq R^n$ be a non-empty and compact subset of $\mathbb{R}^n$. Then, a *constrained single objective optimization problem* (CSOP) is formally given by

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g_i(x) \leq \varepsilon_i \quad i = 1, \ldots, p, \\ & x \in X, \end{array} \tag{CSOP}$$

where $\varepsilon_i \in \mathbb{R}$ for $i = 1, \ldots, p$. In the reverse direction to the ε-constraint method, we transform the given constrained single objective problem into a multiple objective optimization problem that is formally given by

$$\begin{array}{ll} \min & G(x) = (f(x), g_1(x), \ldots, g_p(x))^\top \\ \text{s.t.} & x \in X. \end{array} \tag{AMOP$_\varepsilon$}$$

Based on the notation used in Klamroth and Tind [114], we refer to Problem (AMOP$_\varepsilon$) as the multiple objective optimization problem *associated* to Problem (CSOP). Considering the relation between the two above stated problems from the opposite point of view, Problem (CSOP) is nothing else than a constrained version of the unconstrained, multiple objective Problem (AMOP$_\varepsilon$), where individual constraints are set on the p different objectives $g_1, \ldots, g_p$.
To simplify the reasoning in the following, we assume that Problem (CSOP) is feasible and that at least one optimal solution exists. Furthermore, we assume that the non-dominated set of Problem (AMOP$_\varepsilon$) is externally stable, i.e. for each dominated vector $y \in G(X)$ there exists a non-dominated vector $\tilde{y} \in G(X)$ that dominates y. Since the feasible set X is assumed to be compact, both of the assumptions are met, if, for example, the involved objectives are assumed to be continuous (cf. Chapter 2).
The following theorem that can be found amongst others in the book of Steuer [197] relates the optimal solutions of Problem (CSOP) with the efficient solutions of Problem (AMOP$_\varepsilon$) and vice versa.

Theorem 3.1 *Let the non-dominated set of Problem* (AMOP_ε) *be externally stable. Then it holds:*

1. *Given an optimal solution of Problem* (CSOP)*, this solution is at least weakly efficient for Problem* (AMOP_ε)*. Furthermore, the set of optimal solutions of Problem* (CSOP) *contains at least one efficient solution of Problem* (AMOP_ε)*.*

2. *There exists an efficient solution of Problem* (AMOP_ε) *that is also optimal for Problem* (CSOP)*.*

Proof: The first part of the theorem is a classical result for the ε-constraint approach shown in Chankong and Haimes [36]. For the second part, let

$$x = \underset{x \in X_E}{\operatorname{argmin}} \{ f(x) : g_i(x) \leq \varepsilon_i, \, i = 1, \ldots, p \}. \tag{3.1}$$

We show that x is optimal for Problem (CSOP). Assume that this is not the case, i.e. there exists another feasible solution $\tilde{x} \in X \setminus X_E$ satisfying $g_i(x) \leq \varepsilon_i$ for $i = 1, \ldots, p$ such that $f(\tilde{x}) < f(x)$. Since the non-dominated set of Problem (AMOP_ε) is externally stable by assumption, there exists $x' \in X_E$ such that $\tilde{x}$ is dominated by x'. But this implies that x' is feasible for Problem (CSOP) and that $f(x') \leq f(\tilde{x}) < f(x)$, which contradicts the choice of x. □

Theorem 3.1 shows that we can find an optimal solution of Problem (CSOP) within the efficient set of the associated multiple objective problem. Hence, solution techniques from multiple objective optimization can be used to derive an optimal solution for the constrained single objective problem. More formally, this approach can be stated as follows:

(1.) Formulate the associated multiple objective optimization Problem (AMOP_ε).

(2.) Use solution techniques from multiple objective optimization to determine a complete set of efficient solutions for Problem (AMOP_ε).

(3.) Use Equation (3.1) to determine an optimal solution of Problem (CSOP) within the efficient set of Problem (AMOP_ε).

As we mainly focus on combinatorial optimization problems in the next part of this work, we remark that all the results presented in this section remain valid, if Problem (CSOP) corresponds to a discrete optimization problem. In this case, the set X normally corresponds to a finite set of distinct singletons.
From a practical point of view, one might argue whether it is a reasonable approach to solve a single objective problem by means of the above described procedure based on the associated multiple objective problem formulation. Besides the fact that multiple objective problems are in general much harder to solve compared to their single objective counterparts, a complete set of efficient solution of the associated problem has to be determined to derive the single solution we are only interested in. Hence, the practical application of the above stated approach may be limited to only a small number of optimization problems. However, as already suggested by the large number of publications related to the topic of this section, a multiple objective-based approach seems

to be one of the most promising ways to calculate an optimal solution for constrained optimization problems.
Especially for practical applications, the associated multiple objective problem that is given by Problem (AMOP_ε) may be of particular interest, whenever the structure of the feasible set X of Problem (CSOP) is well-known in advance. As the additional side constraints destroy the simple structure of the single objective problem, a multiple objective based approach seems to be favorable. This is especially the case, when there exist efficient algorithms to solve the unconstrained version of Problem (CSOP). Hence, multiple objective solution approaches that disregard the constraints of Problem (CSOP) can be used to determine parts of the non-dominated set of Problem (AMOP_ε), or even the complete set itself. This could be done, for example, by applying an appropriate weighted sum approach to Problem (AMOP_ε). As was also observed, for example, by Boyd and Vandenberghe [25] this approach corresponds to the Lagrangian relaxation of Problem (AMOP_ε) which is formally given by

$$\begin{array}{ll} \min & f(x) + \sum_{i=1}^{p} \lambda_i g_i(x) \\ \text{s.t.} & x \in X \end{array}$$

with Lagrangian multipliers $\lambda_i \geq 0$ for $i = 1, \ldots, p$.
Hence especially in this case, the associated multiple objective reformulation of the problem yields an efficient way to (approximately) solve the given constrained problem. For further details on this approach, we refer to Section 6.1. There, a general solution concept for solving combinatorial optimization problems with an additional side constraint is presented that makes use of the above described ideas.
We finally give another interesting application, why the associated multiple objective problem could be considered for solving constrained problems in practice. We focus on the discrete case in the following. However, the ideas presented in the remainder of this section are not only limited to this specific case, but can easily be extended to constrained versions of continuous optimization problems, too.
Let a constrained single objective optimization problem be given. It is assumed that the feasible set X consists of different solutions, also called *scenarios* in the following, that may correspond to different investments of an enterprise that can be realized. The objective function $f : X \to \mathbb{R}$ measures the costs that result from the realization of the scenario $x \in X$. In contrast, the constraint functions $g_1, \ldots, g_p : X \to \mathbb{R}$ describe p different budget functions that measure how much money $g_i(x)$ has to be spent from budget i to realize $x \in X$. In this case, the right hand side values $\varepsilon_1, \cdots, \varepsilon_p$ determine the maximum amount of money that is allowed to be used from budget i.
From the mathematical point of view, the right hand side values $\varepsilon_1, \ldots, \varepsilon_p$ impose strict upper bounds on the budget functions $g_1, \ldots, g_p$ that are not allowed to be exceeded by a feasible solution. But this normally implies that a lot of scenarios are excluded from the optimization process in advance, although they may only violate a small number of the given constraints. Furthermore, some of these scenarios may correspond to favorable solutions of the decision maker, when small violations of the bounds can be accepted.
To give a small example for the two-dimensional case, we refer to Figure 3.2. There, the optimal solution of the constrained problem, where the right hand side value is fixed to ε, is given by a representative of the point G_1. In contrast, the representatives of G_2

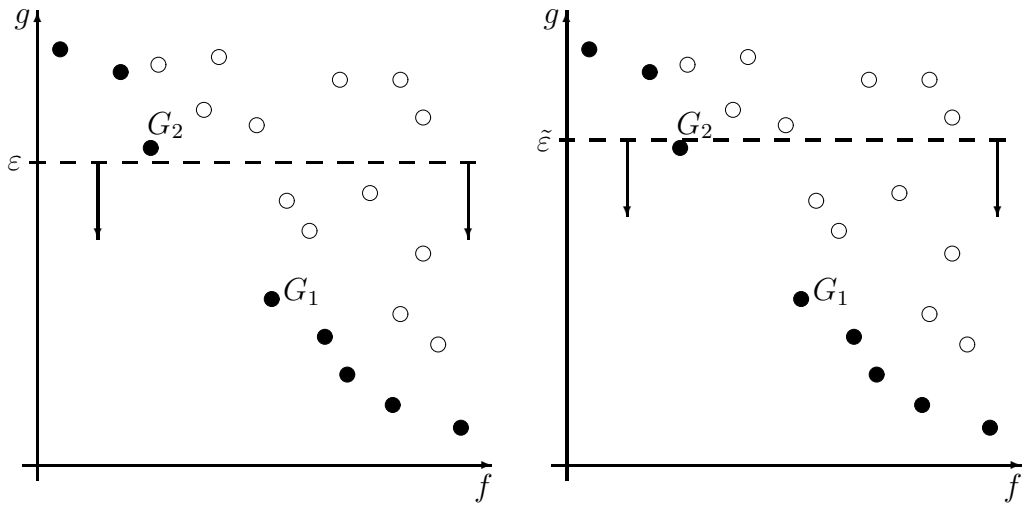


Figure 3.2: *In the left subfigure, the non-dominated point G_1 defines the optimal solution of the constrained problem, while the point G_2 is infeasible, although it is better with respect to f. When the bound on g is relaxed by only a small amount from ε to $\tilde{\varepsilon}$, G_2 becomes feasible and hence optimal for the constrained problem depicted in the right subfigure. In this case, the value of f can be improved significantly as compared to the original constrained problem only by means of a small relaxation of the bound on g.*

correspond to infeasible solutions of the constrained problem, although the constraint on g is only violated slightly. Hence, only a small increase of right hand side value ε to $\tilde{\varepsilon}$ leads to a strong improvement with respect to the objective function f, as the previously infeasible solution G_2 becomes feasible.
More generally, a too restrictive choice of the right hand side values ε_i may exclude solutions that occasionally could be preferred by a potential decision maker since only small violations of the given constraints have to be accepted to improve the given objective function f. However, it is not clear in advance which magnitude of violations of the side constraints has to be accepted that result in a significant improvement of the objective function f. Hence, instead of solving the original Problem (CSOP), a complete set X' of efficient solutions of the associated multiple objective problem (AMOP_ε) can be determined and presented to the final decision maker. Since the set X' contains alternative favorable solutions of the original problem, the decision maker can decide himself, whether a scenario that corresponds to an infeasible solution of the original problem is realized, or whether the optimal solution of the original problem is favorable.

3.2 Weighted Single Objective Sum Problems

In this section we discuss how multiple objective optimization can be used to solve optimization problems, where the objective is given as a weighted sum of different subobjectives involving strictly positive weights. We proceed similar to the last section.
Let $f_1, \ldots, f_p : \mathbb{R}^n \to \mathbb{R}$ be p real-valued objective functions, where $p \geq 2$, and let $X \subseteq R^n$ be a non-empty, compact subset of $\mathbb{R}^n$. Furthermore, let $\lambda_i \in \mathbb{R}^+ := \{x \in \mathbb{R} : x > 0\}$ for $i = 1, \ldots, p$. Then the *weighted single objective sum problem* (WSOP)

is formally given by

$$\begin{array}{ll} \min & \lambda_1 f_1(x) + \ldots + \lambda_p f_p(x) \\ \text{s.t.} & x \in X. \end{array} \tag{WSOP}$$

Similar to the idea used in the previous section for Problem (CSOP), we reinterpret the given single objective problem as a weighted sum scalarization of the more general multiple objective optimization problem

$$\begin{array}{ll} \min & F(x) = (f_1(x), \ldots, f_p(x))^\top \\ \text{s.t.} & x \in X. \end{array} \tag{AMOP$_\lambda$}$$

Using the same notation as in the case of Problem (AMOP$_\varepsilon$), we refer to Problem (AMOP$_\lambda$) as the multiple objective optimization problem *associated* to Problem (WSOP). From a multiple objective point of view, Problem (WSOP) is a weighted sum scalarization of the objective functions of Problem (AMOP$_\lambda$) with strictly positive weights.
Similar to Section 3.1, we assume in the following that there always exists an optimal solution for a given instance of Problem (WSOP), as well as that the non-dominated set of Problem (AMOP$_\lambda$) is externally stable. We make use of the results for the weighted sum approach originally proposed by Steuer [197] to prove:

Theorem 3.2 *Let the non-dominated set of Problem* (AMOP$_\lambda$) *be externally stable. Then it holds:*

1. *If x is an optimal solution of Problem* (WSOP)*, then x is also efficient for Problem* (AMOP$_\lambda$)*.*
2. *There exists an efficient solution of Problem* (AMOP$_\lambda$) *which is also optimal for Problem* (WSOP)*.*

Proof: Since $\lambda_i > 0$ for all $i = 1, \ldots, p$ by assumption, the first part follows directly from the corresponding property for the weighted sum scalarization for multiple objective problems (see Steuer [197]). For the second part, let

$$x = \operatorname*{argmin}_{x \in X_E} \{\lambda_1 f_1(x) + \ldots + \lambda_p f_p(x)\}. \tag{3.2}$$

We show that x is optimal for Problem (WSOP). Assume that this is not the case, i.e. there exists $\tilde{x} \in X \setminus X_E$ such that $\sum_{i=1}^p \lambda_i f_i(\tilde{x}) < \sum_{i=1}^p \lambda_i f_i(x)$. Since $F(X)$ is externally stable by assumption, there exists $x' \in X_E$ that dominates $\tilde{x}$, i.e. $f_i(x') \leq f_i(\tilde{x})$ for all $i = 1, \ldots, p$ where at least one given inequality is strict. But since $\lambda_i > 0$ for all $i = 1, \ldots, p$, this implies that

$$\sum_{i=1}^p \lambda_i f_i(x') < \sum_{i=1}^p \lambda_i f_i(\tilde{x}) < \sum_{i=1}^p \lambda_i f_i(x),$$

which contradicts the choice of $x \in X_E$. □

Theorem 3.2 implies that given an optimal solution to Problem (WSOP), it is automatically contained in the efficient set of the associated multiple objective problem.

Hence, a procedure similar to the one described in Section 3.1 for solving constrained single objective optimization problems, can be used to derive an optimal solution for Problem (WSOP). Note that in this case, Equation (3.2) instead of Equation (3.1) has to be used to calculate the optimal solution we are looking for.
Also in the case of Problem (WSOP) one might argue why a multiple objective-based approach should be used to solve the given single objective problem. Especially for the case, when the number of involved objectives is large, much redundant information may be calculated that is of no further use for the solution of the considered single objective problem. However, we will see in Section 6.2 that there exist several solution approaches for special classes of combinatorial optimization problems in the literature that implicitly make use of the ideas described in this section.
Since combinatorial problems belong to the class of discrete optimization problems, it is important to mention that all the results stated in this section remain valid, when the assumption that X is a compact subset of $\mathbb{R}^p$ is replaced by X being a finite set of elements. Note that in this case, the external stability of $F(X)$ is automatically ensured, due to the fact that $F(X)$ is of finite cardinality.

3.3 An Enhanced Alternate Block Search Strategy for Single Objective Optimization Problems

In this section we present the theoretical background of an enhanced block search strategy for single objective optimization problems that is based on a multiple objective approach. When a block search method is applied, the set of $n > 1$ variables of a single objective (non-linear) optimization problem is partitioned into $p \leq n$ different blocks of variables. Subsequently, the optimization problem is solved by consecutively optimizing the given objective function with respect to a specific block of variables, while the other blocks remain fixed. Under suitable convexity assumptions (see, e.g., Bertsekas and Tsitsiklis [18]), it can be shown that the sequence of generated points converge to a stationary point of the given problem. We refer to this solution approach as *alternate block search method* in the remainder of this work. Note that this method is also known as *block-relaxation method* (cf. de Leeuw [46]) in the literature. For the case that $p = n$, this given approach simplifies to the *cyclic coordinate method* (cf., e.g., Bazaraa et al. [13]).
While non-linear optimization problems are hard to solve in general, the alternate block search method may benefit from the fact that the resulting subproblems are much easier to handle as compared to the overall problem. This especially holds true, when a given problem can be decomposed into a sequence of convex subproblems, while the overall problem is known to be non-convex in general. For more details on the convergence of the alternate block search method for these special types of problems, we refer, amongst others, to Bazaraa et al. [13], de Leeuw [46] and Wendell and Hurter Jr. [215], as well as Section 11.3 of this work.
When a block search strategy is used to solve a single objective optimization problem, optimization is only performed on the active block of variables, while potential information contained in the other blocks is disregarded during the complete solution process. Hence, although the sequence of generated points may quickly converge to a stationary point of a given problem, no further information on the quality of the found

solution with respect to the global optimum can be given in general. If the considered problem is known to have a large number of local minima, numerous restarts of the alternate block search method have to be performed to improve the quality of the found local optimum with respect to the given objective function. However, even by applying a multi-start version of the above described method, it cannot be guaranteed that the global minimum of a given problem is found within a certain amount of time. In the remainder of this section, we present the theoretical background of a heuristic approach inspired by multiple objective optimization that can be used to potentially improve the performance of the alternate block search method. The approach is based on the idea of exploiting additional information that is contained in the blocks of fixed variables during the solution process. In more detail, we try to incorporate information on the expected improvement of the objective value with respect to the blocks of fixed variables, while optimization is performed on the active block. If the expected improvement with respect to the fixed blocks does not fall below a predefined threshold value, a suitable improvement in the next iteration of the method can be expected with respect to these blocks. Especially during the first iterations of the alternate block search method this information can be of interest, as using this additional information may result in a better local optimal solution at the end of the search process as compared to the original method.
However, as the application of the above described ideas neither implies a guaranteed convergence to an improved solution with respect to the given objective nor that the global minimum of a given problem can be found, the described approach must be seen as a heuristic method to improve the quality of a calculated solution with respect to the global optimum of the problem. In the following, we describe this enhanced approach in more detail.
Let $X_1 \subseteq \mathbb{R}^{n_1}, \ldots, X_p \subseteq \mathbb{R}^{n_p}$ denote $p \geq 2$ non-empty sets and let $B \subseteq X_1 \times \ldots \times X_p$. For $i \in I := \{1, \ldots, p\}$ we define the x_i-sections of B by

$$B_i = \{x \in X_i : (x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_p) \in B\}.$$

Furthermore, let $f : B \to \mathbb{R}$ be a given function on $X_1 \times \ldots \times X_p$. Then, the optimization problem considered in this section is formally given by

$$\min \{f(x_1, \ldots, x_p) : (x_1, \ldots, x_p) \in B\} \tag{BP}$$

To simplify the notation in the following, we define for arbitrary but fixed $i \in I$ the function $f_i : B_i \to \mathbb{R}$, $f_i(x) := f(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_p)$ where the variables $x_j \in X_j$, $j \in I \setminus \{i\}$, are assumed to be fixed. Applying the alternate block search method, we have to iteratively solve p optimization problems of the form

$$\min \{f_i(x) : x \in B_i\}, \tag{BP$_i$}$$

for $i \in I$. A short algorithmic description of the alternate block search strategy for Problem (BP) can be found in Algorithm 3.1. The algorithm stops when no significant improvement with respect to the given objective f is detected in two consecutive iterations, or if a prescribed number of iterations is reached.
Let $i \in I$ be arbitrary but fixed. For $j \in I \setminus \{i\}$ we assume that the function

$$\pi_{ij} : \left\{ \begin{array}{rcl} B_i & \to & \mathbb{R} \\ x & \mapsto & \pi_{ij}(x_1, \ldots, x_j, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_p) \end{array} \right. \tag{3.3}$$

Algorithm 3.1 Alternate Block Search Algorithm

Input: An instance of Problem (BP).
Output: A point $y \in B$.

```
1: Choose an arbitrary starting point y = (x_1, ..., x_p) ∈ B.
2: while no stopping criterion is satisfied do
3:    for i=1 to p do
4:       Find an optimal solution x* ∈ B_i such that x* is optimal for Problem (BP_i).
5:       Set x_i = x*.
6:    end for
7: end while
8: return y.
```

measures the potential improvement of the objective value with respect to the block of fixed variables $x_j \in X_j$, when the variables contained in the block X_i are considered as active. We call π_{ij} the *descent potential* for block x_j with respect to the active block x_i. It is assumed in the following that $\pi_{ij}(x) \geq 0$ for all $x \in X_i$, and that $\pi_{ij}(x) = 0$ if and only if f cannot be improved in the point $(x_1, \ldots, x_j, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_p)$ with respect to the fixed variables x_j. We do not go into further details, how the functions π_{ij} can be defined in practice, but we refer to Chapter 12 where the enhanced approach of the alternate block search strategy for biconvex optimization problems (cf. Chapter 11) is discussed in more detail. However, if a given point $(x_1, \ldots, x_p) \in B$ corresponds to a stationary point of f, this implies that $\pi_{ij} = 0$ for all $i \in I$ and $j \in I \setminus \{i\}$.
Instead of solving Problem (BP_i) for a fixed $i \in I$ directly, we additionally include the descent information provided by the $p-1$ functions π_{ij} in the blocks of fixed variables to the optimization process. This implies that while we aim to minimize f_i in the block of active variables, we additionally want to maximize the $p-1$ different descent potentials with respect to the blocks of fixed variables. This enhanced approach results in solving the multiple objective optimization problem

$$\begin{aligned} \min\ & F_i(x) = (f_i(x), -\pi_{i1}(x), \ldots, -\pi_{ij}(x), \ldots, -\pi_{ip}(x))^\top \\ & \text{s.t. } x \in B_i, \end{aligned} \tag{MOBP_i}$$

with p objectives, where $j \in I \setminus \{i\}$. The following theorem relates global optimal solutions of Problem (BP_i) with efficient solutions of Problem (MOBP_i) and vice versa under the assumption that the non-dominated set of Problem (MOBP_i) is externally stable.

Theorem 3.3 *Let the non-dominated set of Problem* (MOBP_i) *be externally stable. Then it holds:*

1. *All global optimal solutions of Problem* (BP_i) *are at least weakly-efficient for Problem* (MOBP_i). *Furthermore, the set of global optimal solutions of Problem* (BP_i) *contains at least one efficient solution of Problem* (MOBP_i).

2. *There exists an efficient solution of Problem* (MOBP_i) *that is global optimal for Problem* (BP_i).

Proof: For the first part of the theorem, let $x \in B_i$ denote a global optimum of Problem (BP_i), and we assume that x is not weakly-efficient for Problem (MOBP_i). Then, there exists $\bar{x} \in B_i$ that strongly dominates x. But this implies that $f_i(\bar{x}) < f_i(x)$, which contradicts the global optimality of x. Hence, x is at least weakly-efficient. If x is not contained in the efficient set of Problem (MOBP_i), the external stability of the non-dominated set of Problem (MOBP_i) guarantees the existence of an efficient solution $x^\star$ dominating x, i.e. $f_i(x^\star) \leq f_i(x)$. However, since x is global optimal for Problem (BP_i), this implies that $f_i(x) \leq f_i(x^\star)$. Hence, we conclude that $f_i(x^\star) = f_i(x)$ and $x^\star$ corresponds to a global optimal solution of Problem (BP_i) that is contained in the efficient set of Problem (MOBP_i).
For the second part, we set $x = \mathrm{argmin}_{x \in X_E}\{f_i(x)\}$, where X_E denotes the efficient set of Problem (MOBP_i). We claim that x is a global optimum of Problem (BP_i). Assume that this is not the case, i.e. there exists $\tilde{x} \in B_i \setminus X_E$ such that $f_i(\tilde{x}) < f_i(x)$. Since the non-dominated set of Problem (MOBP_i) is externally stable, there exists $x' \in X_E$ such that $\tilde{x}$ is dominated by x'. But this further implies and that $f(x') \leq f(\tilde{x}) < f(x)$, which contradicts the choice of x. □

Note that the results of the last theorem are similar to the results stated in Theorem 3.1, where the relation between a constrained single objective problem and its associated multiple objective counterpart is described. Theorem 3.3 ensures that the efficient set of the extended, multiple objective problem contains at least one global optimum of its single objective counterpart. However, if no further properties of the objective function f_i are known in advance, determining the global optimum of Problem (BP_i) by means of the extended Problem (MOBP_i) is at least as difficult as solving Problem (BP_i) itself.
In each step of the *enhanced alternate block search method* we determine a set of representative solutions for Problem (MOBP_i) instead of solving Problem (BP_i) directly. According to a given criterion defined by the decision maker, an efficient solution of Problem (MOBP_i) is chosen as the new value for the block containing the variables $x_i \in X_i$ in the next iterations. While in the first iterations, solutions with a larger expected improvement of the objective value with respect to several blocks of fixed variables may be favorable as compared to the descent potentials provided by the optimal solutions of the Problems (BP_i), the latter solutions can be of interest in later iterations of the enhanced version, since they ensure the convergence to a local minimum of the given problem.
However, since we have to solve a sequence of multiple objective instead of single objective problems in each iteration of the enhanced alternated block search method, the numerical complexity of finding a local minimum for Problem (BP) is significantly increased, compared to the original version of the method. Especially in the case when the p individual subproblems (BP_i) can be solved efficiently, the enhanced approach leads to a large increase of CPU time in general. One of the main reasons for this can be seen in the fact that we have to calculate a complete set of efficient solutions in each iteration, although only one of these solutions is selected at the end. However, the above described approach may lead to a significant improvement of the quality of the local minimum as compared to the solution obtained by the original method. Hence, especially when a given problem has many local optimal solutions, the multiple objective approach provides a reasonable alternative method to derive local optima

for the considered problem.
To additionally decrease the numerical complexity of the enhanced method several approaches seem reasonable. For example, one might think about limiting the number of considered blocks of fixed variables for the multiple objective problem formulation to a small number of blocks that will be optimized in the subsequent iterations. Alternatively, smaller blocks of variables could be combined into a single block to significantly decrease the number of additional objectives that have to be considered during the solution process. Moreover, as we are only interested in a single solution of Problem (BP_i) at the end, the number of calculated efficient solutions could be decreased to a fixed constant, and the updated value of the variables contained in the active block is chosen from this smaller set of alternatives.
As we are only interested in the theoretical background of the ideas of how multiple objective optimization can be used to solve single objective problems in this chapter, we neither go into further details on how the functions π_{ij} may be defined in practice, nor we discuss how the extended multiple objective problems (cf. Problem ($MOBP_i$)) should be solved, when the enhanced method is applied.
We refer to Chapter 12 where the enhanced alternate block search method is discussed in more detail for biconvex optimization problems. However, all the ideas and results that will be stated in Chapter 12 are not limited to the biconvex case, but they can easily be generalized and applied to any other (non-linear) single objective optimization problem. For an application of the ideas developed in this section, we further refer to Chapter 13 where an adapted version of the enhanced alternate block search method is applied to solve the connection location-allocation problem in the plane.

3.4 Conclusions and Organization of the Remainder of this Work

In this chapter we discussed how ideas from multiple objective optimization can be used to solve single objective optimization problems. Although the proposed approaches generally imply that instead of a single solution, a complete set of efficient solutions has to be calculated, multiple objective-based approaches seem suitable to gain a deeper insight into the specific structures of single objective problems.
In more detail, we showed how constrained as well as scalarized single objective optimization problems can be solved by means of associated multiple objective problem formulations. While traditionally, weighted sum as well as ε-constraint approaches are used to solve multiple objective problems, we took the reverse way and modeled the given single objective problems in a more general multiple objective framework. This implies that solution concepts for multiple objective optimization problems that are not directly related to single objective optimization can be used to solve the single objective problem. However, by using a multiple objective approach we have to accept that the complexity of the considered single objective problem is increased, as additional calculations have to be performed that may be of no further use during the solution process. However, we saw that multiple objective-based approaches were already frequently used in the literature to solve single objective problems.
We restricted ourselves to multiple objective reinterpretations of the ε-constraint and the weighted sum approach in this chapter. However, there exist further solution

concepts from multiple objective optimization that can be interpreted as an associated multiple objective problem formulation of a single objective problem. For example, the compromise programming method (cf., e.g., Ehrgott [54] and Miettinen [143]) with squared Euclidian distance can be interpreted as an associated multiple objective problem formulation of a least squares problem from adjustment theory (cf., e.g., Teunissen [201]). Generally, least squares methods are used to approximate solutions for overdetermined systems of linear equations. In this context, an optimal solution of the single objective optimization problem

$$\min \left\{ \|Ax - b\|_2^2 , x \in \mathbb{R}^n \right\}, \tag{3.4}$$

where $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, where $m \geq n$, and $\| \cdot \|$ denotes the squared Euclidian distance, has to be determined. Considering the right hand side vector $b = z^0$ as the ideal point of the problem, Problem (3.4) can be reformulated as

$$\begin{aligned} &\min \|z^0 - z\|_2^2 \\ &\text{s.t. } Ax = z \\ &x \in \mathbb{R}^n, z \in \mathbb{R}^m. \end{aligned} \tag{3.5}$$

From a different point of view, Problem (3.5) can be seen as a compromise programming scalarization of a multiple objective problem with m objectives, where the involved objectives correspond to the components of the vector $b - z$, and the feasible set of the problem is defined by the given side constraint. This alternative interpretation of the least-squares problem could be in the focus of further research.
Besides to the approaches discussed above, we further presented an enhanced version of the well-known alternate block search method for general optimization problems. Applying this method, a local optimal solution of a given single objective problem is derived by iteratively solving the problem in a block of active variables, while the variables contained in the other blocks remain unchanged. We introduced the idea of using additional information on the improvement of the objective value that is contained in the blocks of fixed variables. While this information is disregarded by the original method, we proposed a heuristic approach that is based on a multiple objective reformulation of the involved subproblem that additionally makes use of this information to improve the quality of local optimal solutions calculated by the original method. This enhanced approach is of special interest whenever the subproblems in the block of active variables can be solved efficiently, while the overall problem is hard to solve in general.

We finally relate the different topics discussed in this chapter to the remaining parts of this work. While the ideas presented in Sections 3.1 and 3.2 are applied to combinatorial optimization problems in Part I (Chapters 4 to 10) of this thesis, the enhanced block search approach is discussed for biconvex optimization problems in further details in Part II (Chapters 11 to 13).
In Part I, Chapter 4 serves to summarize the main ideas and concepts from the field of combinatorial optimization. We distinguish between single and multiple objective combinatorial problems with sum and bottleneck objectives, respectively. In Chapter 5, we generalize the concept of a bottleneck objective and present solution approaches for solving single as well as multiple objective k-max optimization problems, that we further exploit in Chapter 6.

In Chapter 6, we directly apply the ideas developed in Sections 3.1 and 3.2 to solve single constrained combinatorial optimization problems as well as combinatorial problems with an algebraic sum objective. For the first problem, we make use of the idea that the associated multiple objective optimization problem can be solved by a weighted sum approach to derive a coarse approximation of the optimal solution of the constrained problem, whenever the single objective combinatorial problem is efficiently solvable. The optimal solution for the constrained problem is finally obtained, for example, by the application of a local search method. For the second problem we show that many algorithms already stated in the literature for solving combinatorial problems with an algebraic sum objective are implicitly based on the ideas developed in Section 3.2. We show that single objective problems are often solved by means of methods that can be related to more general solution concepts that can also be used to solve the associated multiple objective optimization problem.
The remaining chapters of Part I deal with the connectedness of efficient solutions for multiple objective combinatorial optimization problems. A connected efficient set would, amongst others, imply that an optimal solution of a constrained single objective problem could be found by means of simple local search techniques. Unfortunately, the efficient set is not connected in general for most of the classical combinatorial optimization problems, independent from the objective functions that are considered (cf. Chapters 7 and 8).
Finally, two special classes of combinatorial optimization problems are discussed in Chapter 9 and Chapter 10 for which the connectedness of the efficient set can be proven. Especially for the triobjective unconstrained optimization problem with two binary objectives (cf. Chapter 9), we make use of the ideas presented in Section 3.1. Starting from an algorithm for the single objective problem with equality constraints on the binary objectives, we derive an efficient solution approach for the triobjective unconstrained combinatorial optimization problem. Based on this approach we turn to the original problem with two binary inequality constraints, and derive an efficient algorithm based on the results for the associated multiple objective problem.
Part II of the thesis is dedicated to biconvex optimization problems. Chapter 11 mainly summarizes the most important facts on biconvex sets and optimization problems with biconvex functions that can be found in the literature. In addition, we present some new results for biconvex maximization problems. Chapter 12 refers to the ideas developed in Section 3.3. We present an enhanced version of the alternate convex search method that is frequently used to derive local optima for biconvex optimization problems.
Finally, Chapter 13 deals with the connection location-allocation problem in the plane. We show that this specific problem from location theory can be formulated as a biconvex optimization problem. Based on the ideas presented in Chapter 12, we develop an enhanced search strategy that is based on the alternate convex search technique. Numerical experiments suggest that the presented enhanced versions of this technique can be used as an alternative solution approach to heuristically improve the quality of the resulting local minima compared to solutions that result from the original version of the method.

Part I

Combinatorial Optimization

Chapter 4

Theoretical Background of Combinatorial Optimization

In Chapter 3 we introduced the general idea of how solution techniques from multiple objective optimization can be used to derive optimal solutions for single objective problems. In the first part of this thesis we apply these ideas to appropriate optimization problems from the field of combinatorial optimization. In more detail, we show how the methods developed in Sections 3.1 and 3.2 can be transferred to single objective combinatorial optimization problems with additional constraints, as well as to problems where the objective is given as the algebraic sum of different types of objectives. These approaches are discussed in more detail especially in Chapter 6 and Chapter 9. Aiming at efficient solution methods for such multiple objective combinatorial optimization problems, we additionally investigate in Chapter 7 and Chapter 8 whether the idea of applying neighborhood search techniques is an appropriate method to determine the complete non-dominated set of a given combinatorial problem.
Since combinatorial optimization problems can be seen as special cases of general discrete optimization problems, not all solution concepts from continuous optimization can be directly applied to the combinatorial case. Hence, we formally introduce the most important definitions and concepts of this special class of optimization problems in the following. For this purpose we mainly focus on problems with more than one objective following the main ideas stated in Ehrgott [54] and Miettinen [143]. For a detailed overview on combinatorial optimization problems with refer to Nemhauser and Wolsey [150] and Schrijver [189]. Recent surveys on existing methods for solving combinatorial optimization problems with multiple objectives can be found in Ehrgott and Gandibleux [55, 56].
Let the set $\mathcal{E} = \{e_1, \ldots, e_n\}$ be a finite set of $n \in \mathbb{N}$ different elements, and let $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$, where $\mathcal{P}(\mathcal{E})$ denotes the power set of $\mathcal{E}$. Then, a *multiple objective combinatorial optimization problem* (MCOP) is given by

$$\begin{aligned} &\min\ f(S) = (f_1(S), \ldots, f_p(S))^\top \\ &\text{s.t.}\ \ S \in \mathcal{X}, \end{aligned} \tag{MCOP}$$

where $f : \mathcal{X} \to \mathbb{Z}^p$ consists of $p \in \mathbb{N}$ integer valued objective functions $f_i : \mathcal{X} \to \mathbb{Z}$ for $i = 1, \ldots, p$. In this context, the set $\mathcal{E}$ is called *ground set*, the set $\mathcal{X}$ *feasible set* and $\mathcal{Y} = f(\mathcal{X})$ denotes the *set of attainable outcomes*. Any $S \in \mathcal{X}$ is called

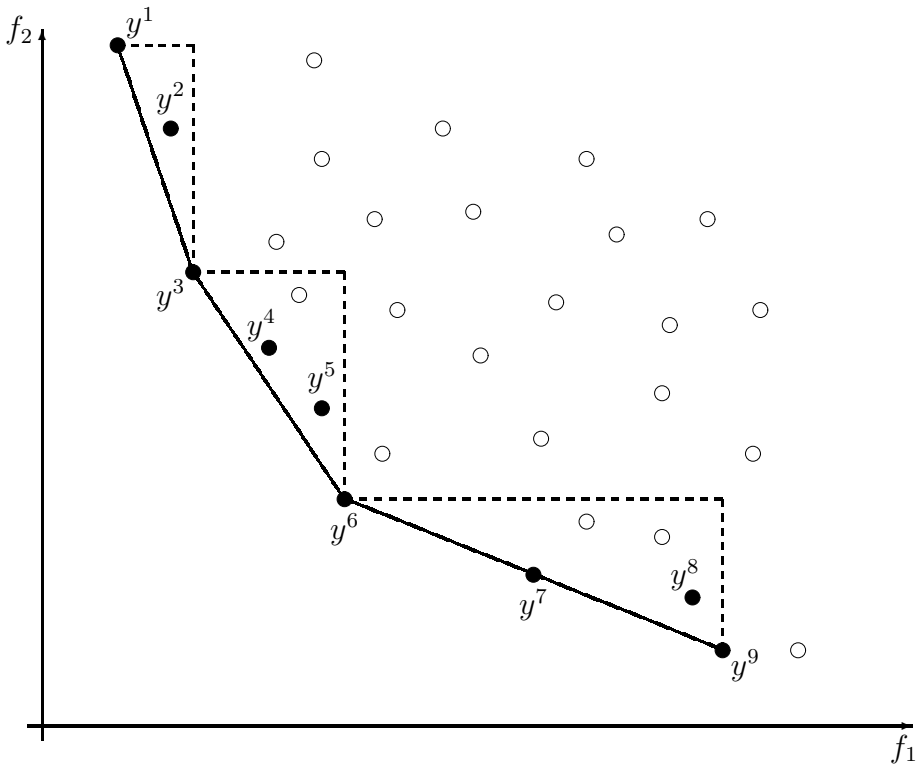


Figure 4.1: *Non-dominated frontier of a MCOP. Filled dots correspond to non-dominated points, while empty dots are dominated. The non-dominated points y^1, y^3, y^6 and y^9 correspond to extreme non-dominated points, while y^7, also located on the non-dominated frontier, is only supported non-dominated. The points y^2, y^4, y^5 and y^8 are non-supported.*

a *feasible solution*. In general, the function vector f consists of a special type of objective functions. Frequently, these functions correspond to sum- or bottleneck-type objectives. But of course, both types may also occur simultaneously in special MCOPs, and other types of objective functions are possible. For more details, we refer to the following sections and Chapter 5, respectively, where the different types of problems and appropriate solution approaches are discussed in more detail. Note that an instance of MCOP is completely described by the triple $(\mathcal{E}, \mathcal{X}, f)$.
Special types of combinatorial optimization problems which satisfy the above given definition are, amongst others, minimum spanning tree and shortest path problems on graphs, as well as knapsack problems and linear assignment problems. We refer to all these different types of problems as special *classes* of combinatorial optimization problems.
We recall that an algorithm is called *polynomial time algorithm* if there exists a polynomial p such that the running time of the algorithm is within $\mathcal{O}(p(n))$. Otherwise, the algorithm is said to be *exponential*. While for a large number of classes of combinatorial optimization problems with only one objective there exist polynomial time algorithms to solve these problems, this is in general no longer the case when more than one objective is considered.
Furthermore, the majority of all MCOPs is *intractable*, which means that there does not exist a polynomial p such that the cardinality of the set of non-dominated solutions $\mathcal{Y}_N$ is of order $\mathcal{O}(p(n))$, i.e. $\mathcal{Y}_N$ can be exponential in the size of an instance. This is even the case for unconstrained MCOPs (cf. Ehrgott [54]).

However, when the non-dominated set is known to be of polynomial size, the cardinality of the efficient set can still be exponential. In this case, one is usually interested in determining the complete set of non-dominated solutions $\mathcal{Y}_N$ in the objective space and a corresponding efficient representative from $\mathcal{X}_E$ in the decision space, rather than the complete set $\mathcal{X}_E$ itself. In this context, two (efficient) solutions S^1 and S^2 are called *equivalent* if $f(S^1) = f(S^2)$. Any subset $\mathcal{X}' \subseteq \mathcal{X}$ is called a *complete set* of efficient solutions whenever $f(\mathcal{X}') = \mathcal{Y}_N$ holds. Note that the binary relation $\mathcal{R}$, defined by

$$S^1 \,\mathcal{R}\, S^2 \iff f(S^1) = f(S^2) \qquad S^1, S^2 \in \mathcal{X}_E,$$

introduces an equivalence relation on the set of efficient solutions of a given MCOP. Each non-dominated vector in the objective space corresponds to an equivalence class of this relation and vice versa. Hence, an efficient solution $S \in \mathcal{X}_E$ can also be seen as a *representative* of the non-dominated vector $y = f(S) \in \mathcal{Y}_N$. In this context, determining a minimal complete set of efficient solutions means to find a representative for each non-dominated point located in the objective space.

Let $\mathrm{conv}(\cdot)$ denote the convex hull operator. The *non-dominated frontier* of an MCOP is the non-dominated set of $\mathrm{conv}(\mathcal{Y})$, i.e. $\{y \in \mathrm{conv}(\mathcal{Y}) : \mathrm{conv}(\mathcal{Y}) \cap (y - \mathbb{R}^p_{\geqq}) = \{y\}\}$ (cf. Figure 4.1). Note that, while $\mathcal{Y}$ is a discrete set of singletons in $\mathbb{R}^p$, the non-dominated frontier forms a continuous curve in the objective space.

For MCOPs it is well-known that for the case that $p = 2$, the non-dominated frontier is given as a piecewise linear curve. The extreme-points (also called *breakpoints*) of this curve are called *extreme non-dominated points* and its representatives in the decision space are said to be *extreme efficient solutions*. An efficient solution $S \in \mathcal{X}_E$ is supported if and only if its image $y = f(S)$ is an element of the non-dominated frontier (but not necessarily an extreme-point of it). This implies that non-supported non-dominated solutions are dominated by points located on the non-dominated frontier but that do not correspond to feasible outcomes of a given MCOP (cf. Figure 4.1).

In the early papers on multiple objective combinatorial optimization, most authors focused on supported efficient solutions, i.e. solutions that can be found by solving a sequence of weighted sum problems of the given objective functions, and ignored the existence of non-supported efficient solutions. However, Melamed [132] showed that both, the size of the set of supported as well as the set of non-supported efficient solutions can be of exponential size. Numerical studies on the knapsack problem provided by Visée et al. [208] show that the number of supported efficient solutions may grow linearly with the problem size, while the number of non-supported solutions grows exponentially.

Note that we focus on exact solution methods in the following, i.e. on methods that ensure that the complete set of non-dominated solutions can be determined. For a detailed overview on approximation methods as well as heuristics and metaheuristics for solving multiple objective combinatorial optimization problems we refer to the two surveys of Ehrgott and Gandibleux [55, 56].

In the following sections, we review existing and present some new solution concepts for (multiple objective) combinatorial optimization problems that do not depend on the special structure of a given class of combinatorial problems, but on the type of the objective function(s) involved. We focus on problems with sum- and bottleneck-objectives.

4.1 Combinatorial Optimization with Sum Objectives

In this section we consider combinatorial optimization problems with sum objective(s). For the single objective case, there exists a large number of methods and algorithms to solve problems for special classes of combinatorial sum problems. In this context, we refer, amongst others, to the books of Nemhauser and Wolsey [150] and Schrijver [189].
In the following, we primarily interested in general solution concepts which can be applied to any class of combinatorial optimization problems rather than comprehensively surveying the existing literature for a special class of problems. Hence, we mainly focus on basic solution concepts for general (multiple objective) combinatorial problems, and we only include special remarks and comments on the existing literature for special classes whenever it seems necessary.

4.1.1 Single Objective Optimization

Let $\mathcal{E}$ denote a finite ground set of n different elements and let $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$ denote the feasible set. Furthermore, let a cost function $c : \mathcal{E} \to \mathbb{Z}$ be given. Then, a *combinatorial sum problem* (CSP) is formally given by

$$\begin{aligned} \min \ & \sum_{e \in S} \{c(e)\} \\ \text{s.t.} \ & S \in \mathcal{X}. \end{aligned} \tag{CSP}$$

Because of the finite structure of CSP one may suggest to solve this problem by a total enumeration of all feasible solutions. Obviously, this solution concept does not depend on the given class of combinatorial problems, but the application of this approach is limited to instances with only a small number of feasible solutions.
In addition to this general solution concept, there exist carefully designed algorithms depending on the considered class of CSPs that explore special properties of this class in an efficient manner. In this context, we refer, for example, to the algorithms of Prim and Kruskal for the minimum spanning tree problem or the algorithm of Dijkstra for the shortest path problem (see, e.g., Hamacher and Klamroth [92]).
Besides these approaches, there also exist more general solution concepts that are mostly applied to $\mathcal{NP}$-hard combinatorial sum problems. For example, dynamic programming approaches and branch-&-bound procedures can be used to solve these kind of problems. For further details, we refer, amongst others, to the books of Nemhauser and Wolsey [150], Schrijver [189] and Kellerer et al. [113], where the latter is devoted to all kinds of knapsack problems.

4.1.2 Multiple Objective Optimization

After the short introduction to combinatorial optimization problems with a single sum objective, we deal with the multiple objective case in the following. Let $c_1, \ldots, c_p : \mathcal{E} \to \mathbb{Z}$ denote p different cost functions on the ground set $\mathcal{E}$. For $i = 1, \ldots, p$ we define p different sum objectives $f_i : \mathcal{X} \to \mathbb{Z}$, $f_i(S) = \sum_{e \in S} c_i(e)$ on the feasible set $\mathcal{X}$. Then,

the *multiple objective combinatorial sum problem* (MCSP) can be stated as follows:

$$\begin{aligned} &\min\ (f_1(S), \ldots, f_p(S))^\top = (\sum_{e \in S} c_1(e), \ldots, \sum_{e \in S} c_p(e))^\top \\ &\text{s.t.}\ \ S \in \mathcal{X}. \end{aligned} \tag{MCSP}$$

Note that also for these types of problems, dynamic programming approaches as well as branch-&-bound procedures are widely used in the literature, especially for the case when the underlying single objective problem is already $\mathcal{NP}$-hard. For a survey of such methods, we refer to Ehrgott and Gandibleux [55, 56].
In addition to these two approaches, the following solution concept is widely used in the literature, especially when there exist efficient algorithms to solve the corresponding single objective problem:

(1.) Based on an efficient algorithm for the single objective problem, calculate a complete subset of the set of supported efficient solutions. This is done by subsequently solving a sequence of weighted sum problems, where the weights for these problems are implicitly defined by the supported solutions found so far.

(2.) Use the supported non-dominated points from the first step to narrow down the search space, where potentially non-supported non-dominated points may be located, and determine a complete subset of the remaining non-supported efficient solutions of the given problem.

The above stated approach is also called *two-phase method* in the literature and is mostly applied to biobjective problems where the single objective counterpart can be solved in a polynomial amount of time. To the best of our knowledge, this method was used for the first time by Aggarwal et al. [2] to solve the minimum spanning tree with an additional side constraint by means of a biobjective approach (cf. also Section 3.1). Some years later, Hamacher and Ruhe [96] applied this method to approximate the set of non-dominated points for a biobjective minimum spanning tree problem. Moreover, Ulungu and Teghem [206] used this approach to solve general biobjective combinatorial problems, and after that it was applied to many different classes of combinatorial problems.
In Phase 1, the approach takes advantage of the fact that there exist efficient (polynomial time) algorithms to solve the single objective version of the considered combinatorial problem. Hence, a complete subset of extreme supported efficient solutions can be calculated in a straightforward way, as long as the cardinality of this set is polynomially bounded. Note that the set of supported solutions is of exponential size in general, since all feasible solutions may correspond to optimal solutions of a given weighted sum problem. Normally, solutions that have been found so far are used to determine a new weight vector λ for the weighted sum problem that has to be solved next. This approach is also referred to as *dichotomic search*.
In Phase 2, a complete set of the remaining non-supported solutions is determined. Information from supported solutions calculated in Phase 1 is used to considerably reduce the search space and the potential location of non-supported points in the objective space. In the biobjective case, only triangles defined by two consecutive supported non-dominated solutions have to be explored (cf. Figure 4.1). Normally, lower and upper bounds as well as reduced costs are additionally used to search for

non-supported points in an efficient manner. Unfortunately, $\mathcal{NP}$-hard problems have to be solved to find a complete set of non-supported efficient solutions in general. Instead of giving a complete survey of articles dealing with the two-phase method for special classes of combinatorial problems, we rather summarize general solution concepts in the following that can be used to determine (all) non-supported non-dominated solutions of a given MCOP. Note that such approaches can especially be used in Phase 2 of the above stated two-phase method.

In the biobjective case, one of the most common ideas to determine non-supported points is to enumerate all potentially non-dominated points which are located in the triangle between two consecutive supported points in the objective space. This can be done, for example, by considering not only the best, but also the second, third, ..., k-best solution of a weighted sum problem, where the weight λ is defined by two consecutive supported points in the objective space (cf., for example, Steiner and Radzik [196] for the biobjective minimum spanning tree problem or Przybylski et al. [169] for the biobjective assignment problem). In the latter article, also a combination of the two-phase method with a population based heuristic to improve computational performance was suggested. Note that the enumeration of all k-best solutions can also be started from the two lexicographically optimal solutions and the two-phase method does not have to be applied.

Another promising approach is to use neighborhood search techniques to determine non-supported efficient solutions. The success of such an approach crucially depends on the considered definition of a neighborhood. We discuss this topic in more detail in Chapter 7. In particular, we will show that the set of efficient solutions is not connected in general, where connectedness of a discrete set of points is defined based on the connectivity of a graph. Hence, when a neighborhood search techniques is applied, it is not guaranteed to find the complete set of non-dominated points in general. Nevertheless, using local search seems to be a promising approach for classes of combinatorial optimization problems where the single objective version is already $\mathcal{NP}$-hard.

For example, in Paquete and Stützle [160] the biobjective traveling salesman problem is considered. In Phase 1, only a single optimal solution to one of the two objective functions was calculated which was used as a starting solution in Phase 2 to search for non-dominated solutions exploiting a sequence of different formulations of the considered problem based on different aggregations of the objectives. The same problem was also considered by Lust and Teghem [126]. Applying the two-phase method, in Phase 1 a good approximation of the supported non-dominated points was generated, while in Phase 2, a local search strategy was used to approximate the set of non-supported solutions of the considered problem.

Moreover, an ε-constraint approach seems to be suitable to generate non-supported points located between the supported ones in the objective space. Hamacher et al. [94] proposed a so-called *Box-Algorithm* for the biobjective case which is based on solving a sequence of lexicographical ε-constraint problems. Since the lexicographical ordering is complete, it is ensured that an optimal solution of the lexicographical problem is always efficient for the considered biobjective problem and the drawback of generating weakly efficient solutions is avoided. For the biobjective case, in each step of the algorithm the rectangle spanned by subsequent non-dominated points is bisected by setting an ε-constraint on one of the two objectives and a lexicographical

ε-constraint problem is solved. This is repeated until either all non-dominated points are found or a certain predefined quality on the approximation of the non-dominated set is met. Numerical studies of this approach and numerical comparisons to other solution techniques can also be found in Ruzika [184]. Note that in these references the algorithm is directly applied to the considered problem without using the two-phase method.

Besides the already stated approaches, also reference point based methods can be used to determine non-supported points in the objective space. Given a (local) ideal point y^I of the problem defined by a certain number of already calculated non-dominated points, one tries to find the image of a feasible solution that minimizes the distance to the considered ideal point. The method is also referred to as *compromise programming method* in the literature (cf. Yu [220]). Frequently, the weighted Tchebycheff norm

$$\min_{S\in\mathcal{X}} \left\{ \max_{i=1,\dots,p} \{\omega_i | f_i(S) - y_i^I|\} \right\},$$

where $\omega_i \in (0,1)$, $\sum_{i=1}^p \omega_i = 1$, is used to measure the distance between y^I and the set of attainable outcomes. Since the solutions that are calculated by this approach only correspond to weakly efficient solutions for the overall problem in general, either a test for efficiency has to be performed (which normally means that a reoptimization of the found solution has to be performed), or an augmented weighted Tchebycheff norm

$$\min_{S\in\mathcal{X}} \left(\left\{ \max_{i=1,\dots,p} \{\omega_i | f_i(S) - y_i^I|\} \right\} + \rho \sum_{i=1}^{p} |f_i(S) - y_i^I| \right) \tag{4.1}$$

with $\rho > 0$ has to be used to guarantee the efficiency of the found solution (cf. Steuer and Choo [198]). Sayin and Kouvelis [186] presented an algorithm for solving general biobjective MCOPs using the compromise programming method in combination with the weighted Tchebycheff norm. Starting from the two lexicographical optima of the problem, new non-dominated solutions are generated by applying the compromise programming method to locally defined ideal points based on subsequent solutions that already have been determined. If a new non-dominated point is found, this point is inserted between the two old ones that have defined the local ideal point, and two new problems have to be solved. If no new solution is found, the complete rectangle between the considered two subsequent solutions is discarded since no further non-dominated solutions can be found inside. Obviously, this algorithm can also be applied in Phase 2 of the two-phase method to determine all non-supported points between subsequent supported ones.

We finally remark that also approaches based on polyhedral gauges can be used to determine non-supported non-dominated solutions. Klamroth et al. [115] presented an algorithm that generates piecewise linear approximations of the non-dominated set for continuous biobjective optimization problems. Their suggested algorithm is based on a polyhedral gauge that defines an oblique norm in $\mathbb{R}^2$ (cf. also Schandl et al. [187, 188]). The piecewise linear structure of the level sets of such norms is used to generate an approximation of the non-dominated set. During the course of the algorithm, the approximation is iteratively refined by points from the non-dominated set that minimize the distance to a (local) ideal or Nadir point. In this case, the distance function is induced by the oblique norm that additionally takes the already

determined solutions into account. In the discrete case, a similar approach can be used to generate non-supported solutions of a given combinatorial problem.
Of course, in addition to the methods stated above there also exist a large number of solution methods for special classes of MCOPs which can often be seen as generalizations of special methods developed to solve the single objective counterpart. For a survey of these methods we once more refer to Ehrgott and Gandibleux [55, 56].

4.2 Combinatorial Optimization with Bottleneck Objectives

Besides the sum objectives, also bottleneck objectives are one of the most common objective functions in combinatorial optimization. Given a finite set S of different elements from a ground set that are associated to some costs, a bottleneck function returns the maximum cost value of the elements contained in S. Since the set of different objective values of a bottleneck function is polynomially bounded by the cardinality of the underlying ground set, one might think that optimization problems with bottleneck objectives are in general much easier to handle compared to their sum objective counterparts. Unfortunately, there exist classes of combinatorial problems for which this is not the case. Only to mention one, the bottleneck traveling salesman problem is proven to be $\mathcal{NP}$-complete (cf. Garey and Johnson [71]). Nevertheless, we show in the following that the single as well as the multiple objective version of combinatorial bottleneck problems can be solved by means of a simple threshold approach that can independently be applied to any class of combinatorial optimization problems.

4.2.1 Single Objective Optimization

Let a ground set $\mathcal{E}$ containing n elements and a feasible set $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$ be given. Furthermore, let $c : \mathcal{E} \to \mathbb{Z}$ denote a scalar cost function on the elements of $\mathcal{E}$. Then, a *combinatorial bottleneck problem* (CBP) is a single objective optimization problem of the form

$$\begin{aligned} &\min \max_{e \in S} \{c(e)\} \\ &\text{s.t.} \quad S \in \mathcal{X}. \end{aligned} \tag{CBP}$$

Note that CBPs are also called *min-max problems* in the literature, since the objective is to minimize the maximum cost value of a feasible solution. In addition to these kind of problems, also *max-min problems* are frequently considered in the literature. Instead of minimizing the maximum cost value, the minimum cost value of a feasible solution has to be maximized. But since it holds that

$$\max_{e \in S} \{c(e)\} \; = \; - \min_{e \in S} \{-c(e)\},$$

results and algorithms for max-min problems can easily be transferred to the min-max case and vice versa.
In the following, we shortly survey existing general solution approaches for Problem (CBP) that can be found in the literature. We mainly focus on the so-called

threshold algorithm that further builds the basis for a solution approach derived for the multiple objective case in the next subsection.

Motivated by problems from location theory, Krarup and Pruzan [118] showed already in the early 80's that an optimal solution of CBP can be determined by solving an associated CSP when an appropriate transformation of the given cost function c is applied. Based on certain properties that have to be satisfied by such a transformation, a whole class of suitable transformations was derived in this article. Amongst others, all functions of the type $f(c) = c^\beta$ that raise the given cost coefficients to the power of $\beta \geq 1$ are contained in this class. In addition, the authors showed that there exists a certain threshold value $\bar{\beta} \geq 1$ such that the above stated result is valid for all $\beta \geq \bar{\beta}$.

An alternative, non-parametric transformation from CBP to CSP was suggested in Jorgensen and Powell [111] that only depends on the number of (different) cost coefficients of c. Based on a non-increasing ordering of the ground set $\mathcal{E}$ with respect to the cost function c, each cost coefficient $c_i := c(e_i)$ is identified with a positive integer. This integer ensures that if c_i is optimal for the given CBP, the optimal solution of CSP with respect to the transformed cost vector only contains the elements up to e_i. Other elements e_j, where $j > i$, whose modified cost coefficients are larger than the sum of all modified coefficients corresponding to e_k, $k < j$, are not contained in this solution. If the n cost coefficients of c are assumed to be pairwise different, the modified cost vector may correspond to the vector $(0, 1, 2, 4, \ldots, 2^{n-2})$. The authors showed that each solution of the sum problem is optimal for the bottleneck problem, but not necessarily vice versa. Due to the limitation of the approach to only small-sized problem instances in practice (the number of elements in $\mathcal{E}$ is not allowed to exceed the maximum word length of a computer), also modifications of the stated approach were discussed in this article.

Besides the ordinary bottleneck problem, also lexicographic versions of CBP were treated in the literature. Still, the largest cost coefficient has to be minimized in the first place. But among all optimal solutions to this CBP, one is interested in the solution that also optimizes the second largest cost coefficient, then the third largest and so on. We refer amongst others to the articles of Della Croce et al. [47] and Sokkalingam and Aneja [193] in this context.

In the remainder of this subsection, we discuss a slightly more general approach to solve CBP, the so-called *threshold algorithm*. Note that the idea of using a threshold approach was introduced for the first time by Edmonds and Fulkerson [51] in the context of bottleneck extrema.

To simplify the notation in the following, we set $b : \mathcal{X} \to \mathbb{Z}$, $b(S) = \max_{e \in S}\{c(e)\}$. While for combinatorial sum problems, there does not exist a general algorithm that works simultaneously for all classes of combinatorial optimization problems (except of the total enumeration of all feasible solutions), the following simple approach can be used to solve Problem (CBP) independently of the class of the combinatorial problems:

Given a feasible solution $S \in \mathcal{X}$, all elements $e \in \mathcal{E}$ satisfying $c(e) \geq b(S)$ are deleted from the ground set $\mathcal{E}$, and a new feasible solution is determined. If such a solution S' exists, it is automatically ensured that $b(S') < b(S)$ holds and once more, all elements $c(e) \geq b(S')$ can be removed from the ground set. This procedure is iteratively repeated until no further feasible solution can be found. Then, the last solution that was feasible for the modified problem is also optimal for Problem (CBP) and its cost

Algorithm 4.1 Threshold Algorithm for Combinatorial Bottleneck Problems

Input: An instance $(\mathcal{E}, \mathcal{X}, f)$ of CBP
Output: Optimal solution S^* and its corresponding objective value b^*.

1: **while** $\mathcal{E} \neq \emptyset$ **do**
2: Find a feasible solution $S^* \in \mathcal{X}$.
3: Determine threshold value $b^* := b(S^*) = \max_{e \in S^*}\{c(e)\}$.
4: Set $\mathcal{E} = \mathcal{E} \setminus \{e \in \mathcal{E} : c(e) \geq b^*\}$.
5: Update the feasible set $\mathcal{X} = \mathcal{X} \cap \mathcal{P}(E)$.
6: **end while**
7: **return** (S^*, f^*).

automatically determines the optimal objective value. A short outline of this approach can be found in Algorithm 4.1. To further improve this approach, it can be combined with a bisection of the ground set in each iteration of the algorithm. Assuming that the elements of $\mathcal{E}$ are sorted in non-decreasing order with respect to the cost function c, it can be concluded that:

Theorem 4.1 *Algorithm 4.1 is correct and solves CBP in $\mathcal{O}(T \log(n))$, where n is the number of elements contained in the ground set, and T denotes the time to solve the specific feasibility problem.*

Note that for combinatorial problems on graphs or networks it is normally quite simple to delete elements from the ground set, since these elements correspond to edges of the given graph or network, and the feasible set is updated automatically. However, besides these problems there also exist many combinatorial problems, where the feasible set $\mathcal{X}$ cannot be updated in a straightforward way, since it is defined by a set of constraints. Hence, discarding elements from the feasible set $\mathcal{X}$ may not be an easy task. In this case, an adapted version of the above described approach can be used that was suggested, amongst others, by Jorgensen and Powell [111].
Given a prescribed threshold value b^* , the costs of the elements that exceed b^* are set to ∞. Then, a sum problem in the modified cost function is solved. If the optimal objective value of this new problem is finite, there exists a feasible solution whose maximum cost value is less than the current threshold value b^* and the value can be updated. Otherwise the solution found in the last iteration must be optimal.
Combining the above described approach with a bisection of the ground set in each iteration, results in an algorithm that solves the given problem in $\mathcal{O}(T \log n)$, where T now denotes the time to solve the corresponding sum problem with respect to the modified cost function. Note that a similar approach will be used in Chapter 5 to derive an algorithm that does not minimize the largest, but the k^{th} largest cost coefficient contained in a feasible solution.
We finally remark that depending on the considered class of combinatorial optimization problems there exist algorithms that solve the specific CBP with lower time complexity than Algorithm 4.1 does, although these algorithms are also based on a threshold approach. For example, Camerini et al. [32] showed that the bottleneck spanning tree problem on an undirected graph $G = (V, A)$ can be solved within $\mathcal{O}(|A|)$. Gabow and Tarjan [68] extended this results to the case of a directed graph and presented similar results for the bottleneck maximum cardinality matching problem.

Furthermore, Punnen and Nair [175] presented an improved algorithm for the bottleneck biconnected subgraph problem, studied for the first time in the article of Parker [162], where, amongst others, the above stated threshold approach is applied to the considered problem. In addition to these two articles that also deal with the bottleneck traveling salesman problem, the article of Garfinkel and Gilbert [72], where the bottleneck TSP is discussed, has to be mentioned. Note that this problem is proven to be $\mathcal{NP}$-complete in general (cf. Garey and Johnson [71]). Finally, several variants for solving bottleneck assignment problems were discussed in the book of Burkard et al. [29].
Concerning the complexity of solving a given CBP, Punnen [171] stated an interesting result. Let $|\mathcal{E}| = n$. Given an algorithm that solves a CBP in $\mathcal{O}(\xi(n))$ assuming that the elements from $\mathcal{E}$ are already sorted in non-decreasing order, the author applied a threshold approach to show that in this case, the CBP with unsorted costs can be solved within $\mathcal{O}(\xi(n)\log^\star(n))$, where $\log^\star(n) = \min\{i : \log^{(i)}(n) \leq 1\}$, and $\log^{(i+1)}(x) = \log(\log^{(i)}(x))$ with $\log^{(0)}(x) = x$. The author concluded that, if the complexity of the algorithm that solves the bottleneck problem for sorted costs is less than that of sorting n numbers (which is of complexity $\mathcal{O}(n\log(n))$), this implies that the threshold approach presented in [171] solves the problem for unsorted costs faster than an approach that first sorts the elements according to their costs, and then solves the problem by applying the algorithm for the sorted costs.

4.2.2 Multiple Objective Optimization

In this subsection we deal with the multiple objective version of a CBP. After a short survey of the existing literature, we present a generalized solution approach that can be applied to any given class of combinatorial optimization problems.
In the following, let $c_1, \ldots, c_p : \mathcal{E} \to \mathbb{Z}$ denote p different cost functions on the ground set $\mathcal{E}$. For $i = 1, \ldots, p$ we define $b_i : \mathcal{X} \to \mathbb{Z}$, $b_i(S) = \max_{e\in S}\{c_i(e)\}$. Furthermore, let $f : \mathcal{X} \to \mathbb{Z}$ denote an additional objective function of arbitrary type. It is assumed in the following that there exists an algorithm that solves the single objective problem $(\mathcal{E}, \mathcal{X}, f)$ in a reasonable amount of time for each given instance of a special class of combinatorial optimization problems. We are interested in solving the *multiple objective combinatorial bottleneck problem* (MCBP)

$$\begin{aligned} &\min\ (f(S), b_1(S), \ldots, b_p(S))^\top \\ &\text{s.t.}\ \ S \in \mathcal{X}, \end{aligned} \tag{MCBP}$$

consisting of $p + 1$ objective functions.
MCBPs are mostly considered for special classes of combinatorial optimization problems in the literature. Only to mention a few, Melamed and Sigal [133, 134, 135, 136, 137, 138] and Melamed et al. [139] numerically analyzed the fraction of supported non-dominated solutions among the set of non-dominated solutions for the linear assignment, the minimum spanning tree, the asymmetric traveling salesman and the knapsack problem with either two or three bottleneck objectives or one sum and one or two bottleneck objectives, respectively. The main result of their numerical investigations is the fact that in their tests the relative number of non-supported solutions in the objective space only depended on the cardinality of the set of non-dominated

solutions itself, but not on any special features of the considered problem like the type of the objective functions, the range of the cost coefficients (which are chosen independently and uniformly distributed in certain intervals) or the type of the input data (integer or real). Interestingly, they further reported for the minimum spanning tree as well as the assignment problem that the ratio of supported non-dominated solutions is substantially larger for problems involving three objectives as compared to their biobjective counterparts. Furthermore, their investigations showed that the number of supported non-dominated solutions that do not correspond to extreme non-dominated points, i.e. to extreme-points of the non-dominated frontier of the problem, exceeds the number of extreme non-dominated solutions when passing from two to three objectives.

Besides the articles of Melamed and Sigal there exist a number of publications that deal with the multiple objective shortest path problem. Polynomial algorithms considering a sum- and a bottleneck-objective are presented by Hansen [98], Martins [129], Berman et al. [17] and Pelegrin and Fernandez [165]. Note that in the article of Martins [129] besides a sum objective also a minratio objective was considered. Gandibleux et al. [69] presented an algorithm for solving a multiple objective shortest-path problem involving q sum and an additional bottleneck objective based on the label setting algorithm developed by Martins [128]. More recently, shortest path problems involving two bottleneck- and a sum-objective were analyzed by Pinto et al. [166]. The authors solved the triobjective problem by a similar approach as the one that we will present in a more general framework at the end of this subsection.

Minoux [144] as well as Punnen and Nair [176] propose algorithms for solving the single objective problem involving the algebraic sum of a sum and a bottleneck objective based on a biobjective approach. We further discuss their algorithms in Section 6.2.

We only briefly mention that MCBPs are also considered for the combinatorial classes of transportation problems (cf. Srinivasan and Thompson [195]), spanning tree problems (cf. Punnen and Aneja [172]) and location problems (cf. Hamacher et al. [93]), generally involving a sum and several bottleneck objectives.

In the remainder of this subsection, we generalize the solution approaches that are presented, amongst others, in the articles of Melamed and Sigal [134, 137] for the minimum spanning tree and the assignment problem involving two or three objective functions, in Melamed et al. [139] for the biobjective knapsack problem, as well as in various articles on the shortest-path problems that have been stated above. In contrast to these articles, our approach is not limited to the case of only one or two given bottleneck objectives, but we present an algorithm that can be applied to any combinatorial optimization problem consisting of $p \in \mathbb{N}$ bottleneck objectives and an additional objective function f of arbitrary type. To the best of our knowledge, this approach has not yet been presented in a general framework, i.e. independent of any special class of combinatorial problems, and its correctness was not proven in the literature before.

For $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_p) \in \mathbb{R}^p$, we consider the following constrained formulation of Problem (MCBP):

$$\begin{aligned} &\min f(S) \\ &\text{s.t. } b_i(S) \leq \varepsilon_i, \quad i = 1, \ldots, p, \\ &S \in \mathcal{X}. \end{aligned} \tag{4.2}$$

From Chankong and Haimes [36] we recall that each optimal solution of Problem (4.2) is at least a weakly-efficient for Problem (MCBP). However, the non-dominated solutions that are obtained by solving Problem (4.2) crucially depend on the chosen right hand side components $\varepsilon_i \in \mathbb{R}$ $(i = 1, \ldots, p)$. Since in general, the exact values of these components are not known in advance (especially when the functions on the left hand side correspond to sum objectives), in the worst case the complete range of involved cost coefficients has to be scanned to determine a complete set of efficient solutions.
Fortunately, this drawback is no longer valid when MCBPs are considered. Since for each $i \in \{1, \ldots, p\}$ and $S \in \mathcal{X}$, $b_i(S)$ takes at most n distinct values contained in the set $c_i(\mathcal{E}) := \{c_i(e) : e \in \mathcal{E}\}$, all possible right hand side values are known in advance. Furthermore, their cardinality is polynomially bounded by $\mathcal{O}(n)$. Since for each combination of the right hand side components ε_i, at most a single non-dominated solution exists, this additionally implies that the cardinality of the non-dominated set is polynomially bounded by $\mathcal{O}(n^p)$. Based on this observation, we can derive a solution approach that determines a complete set of efficient solutions for a given MCBP: We iteratively solve Problem (4.2) for all combinations of right hand side values contained in the sets $c_i(\mathcal{E})$ $(i = 1, \ldots, p)$, followed by a filter step to exclude weakly efficient solutions that are not contained in the efficient set of the given problem.
We further discuss the special properties of Problem (4.2) in the following. Let $\varepsilon_i = c_i(e)$ for an arbitrary but fixed element $e \in \mathcal{E}$ and index $i \in \{1, \ldots, p\}$. We consider the i^{th} constraint of Problem (4.2) that is formally given by $b_i(S) \leq \varepsilon_i$. Obviously, this constraint implies that a solution $S \in \mathcal{X}$ is feasible if and only if its maximum cost coefficient with respect to the cost function c_i is at most ε_i. Hence, as for the single objective case, ε_i can be treated as a threshold value for the set $c_i(\mathcal{E})$. By eliminating all elements from the ground set satisfying $c_i(e) > \varepsilon_i$ for some $i \in \{1, \ldots, p\}$, it is automatically ensured that the constraint $b_i(S) \leq \varepsilon_i$ is satisfied by all solutions that are still feasible for the reduced problem. Hence, Problem (4.2) can be solved by considering an unconstrained single objective combinatorial optimization problem $(\mathcal{E}', \mathcal{X}', f)$ with objective function f, where $\mathcal{E}' \subseteq \mathcal{E}$ and $\mathcal{X}' = \{S \in \mathcal{X} : b_i(S) \leq \varepsilon_i,\ i = 1, \ldots, p\}$.
Since for many combinatorial problems, the feasible set $\mathcal{X}$ is not given explicitly, a slightly different approach can be used to solve Problem (4.2). This approach can be seen as a generalization of the one stated in the previous subsection for the single objective case. We assume in the following that the objective function f depends on a given cost function w on the elements of the ground set $\mathcal{E}$. To indicate this dependence, we replace f by f_w.
Let $\varepsilon \in \mathbb{R}^p$ such that $\varepsilon_i \in c_i(\mathcal{E})$ for all $i \in \{1, \ldots, p\}$. Instead of removing the elements from $\mathcal{E}$ that do not satisfy the constraint $b_i(S) \leq \varepsilon_i$, we define a modified cost function $\tilde{w} : \mathcal{E} \to w(\mathcal{E}) \cup \{+\infty\}$, where

$$\tilde{w}(e) = \begin{cases} w(e), & \text{if } c_i(e) \leq \varepsilon_i \quad \forall i \in \{1, \ldots, p\}, \\ +\infty, & \text{otherwise.} \end{cases}$$

Based on this modified cost function $\tilde{w}$, we solve the unconstrained single objective combinatorial optimization problem which is given by $(\mathcal{E}, \mathcal{X}, f_{\tilde{w}})$.

Algorithm 4.2 Threshold Algorithm for solving MCBP

Input: An instance $(\mathcal{E}, \mathcal{X}, (f_w, b_1, \ldots, b_p))$ of MCBP
Output: A complete subset $\mathcal{X}^\star \subseteq \mathcal{X}_E$.

```
Set X* = ∅ and C = c_1(E) × ··· × c_p(E) for i = 1, ..., m.
for All possible combinations of bounds ε ∈ C do
  for All e ∈ E do
    if c_i(e) ≤ ε_i for all i ∈ {1, ..., p} then
      Set w̃(e) = w(e).
    else
      Set w̃(e) = +∞.
    end if
  end for
  Solve the unconstrained problem (E, X, f_w̃) → S*.
  if f_w̃(S*) < ∞ then
    Update X* = X* ∪ {S*}.
  end if
end for
Filter X* for dominated solutions.
return X*.
```

Lemma 4.2 *If the optimal objective value of the instance $(\mathcal{E}, \mathcal{X}, f_{\tilde{w}})$ is finite, the solution obtained is both feasible as well as optimal for Problem (4.2) with respect to the considered right hand side vector $\varepsilon \in \mathbb{R}^p$. Otherwise, Problem (4.2) is infeasible.*

Proof: Let S denote the optimal solution of the instance $(\mathcal{E}, \mathcal{X}, f_{\tilde{w}})$, and let $f_{\tilde{w}}(S) < \infty$. The construction of $\tilde{w}$ implies that $c_i(e) \leq \varepsilon_i$ for all $e \in S$ and $i \in \{1, \ldots, p\}$, and hence, S is feasible for (4.2) and $f_{\tilde{w}}(S) = f_w(S)$ holds.
Now assume that S is not optimal for Problem (4.2). Then there exists $S^\star \in X$ with $f_w(S^\star) < f_w(S)$ satisfying $b_i(S^\star) \leq \varepsilon_i$ for all $i \in \{1, \ldots, p\}$ which means that $c_i(e) \leq \varepsilon_i$ for all $e \in S^\star$ and $i \in \{1, \ldots, p\}$. Hence, $f_{\tilde{w}}(S^\star) = f_w(S^\star) < f_w(S) = f_{\tilde{w}}(S)$ which contradicts the optimality of S for the instance $(\mathcal{E}, \mathcal{X}, f_{\tilde{w}})$.
On the other hand, if the optimal objective value is not finite, there do not exist feasible solutions satisfying $c_i(e) \leq \varepsilon_i$ for all $i \in \{1, \ldots, p\}$ simultaneously. Hence, Problem (4.2) is infeasible. □

As a result of Lemma 4.2, Problem (4.2) can be solved without explicitly deleting elements from the ground set $\mathcal{E}$. After a simple modification of the costs of the given cost function w, an unconstrained problem with objective $f_{\tilde{w}}$ has to be solved. We use this fact in Algorithm 4.2, where a complete set of the efficient solutions of Problem (MCBP) is determined by solving at most $\mathcal{O}(n^p)$ ε-constraint problems. Note that since an optimal solution of Problem (4.2) is only weakly efficient in general, an efficiency check has to be performed at the end of the algorithm.

Theorem 4.3 *Algorithm 4.2 is correct and solves MCBP in $O(n^p \cdot T)$, where n is the number of elements contained in the ground set, p is the number of bottleneck objectives and T denotes the time to solve the unconstrained combinatorial problem with respect to the modified objective function $f_{\tilde{w}}$.*

Theorem 4.3 implies that the considered MCBP is solvable in polynomial time, whenever this holds true for the single objective problem $(\mathcal{E}, \mathcal{X}, f_w)$ and p is assumed to be fixed. To reduce the number of constrained problems that have to be solved during the course of the algorithm, we suggest to fix the bounds on the bottleneck objectives in lexicographically decreasing order with respect to the index of the objective.
First, the unconstrained bottleneck problem is solved, to derive a first solution candidate S^1. In the next iteration, the bounds ε_i for b_i are fixed to $b_i(S^1)$ for $i = 1, \ldots, p-1$, while the bound on b_p is set to the next smaller cost coefficient that precedes $b_p(S^1)$ with respect to c_p. Then, the bottleneck problem is resolved to derive a new optimal solution S^2 and the next threshold value ε_p is given by the next smaller cost coefficient that precedes $b_p(S^2)$ with respect to c_p, while the remaining bounds on $b_1, \ldots, b_{p-1}$ remain fixed. This procedure is repeated, until the corresponding bottleneck problem is detected to be infeasible for the first time. Then, the constraint on b_p is relaxed, since decreasing the bound would automatically lead to further infeasible problems. In addition, the bound on b_{p-1} is now set to the next smaller cost coefficient with respect to c_{p-1}, to derive new threshold values for b_p for the subsequent iterations. Note that the bounds on $b_1, \ldots, b_{p-2}$ still remain fixed.
More generally, if for a fixed index $j \in \{2, \ldots, p\}$ an infeasible problem is detected, since the constraint $b_j(S) \leq \varepsilon_j$ cannot be satisfied by any feasible solution, it is useless to decrease the right hand side of this constraint, since the resulting problem would still be infeasible. Hence, instead of decreasing the cost bound for b_j, all the constraints on $b_j, b_{j+1}, \ldots, b_p$ are relaxed and the bound on b_{j-1} is moved to the next smaller cost coefficient with respect to c_{j-1}. In the next iteration, new bounds for $b_j, b_{j+1}, \ldots, b_p$ are derived for the subsequent iterations. This further implies that the bounds on $b_j, \ldots, b_{p-1}$ remain fixed, while the bound on b_p moves from large to small cost values with respect to c_p, until the infeasibility of the corresponding bottleneck problem is detected for the first time. Then, the constraint on b_p is relaxed again, and the bound on b_{p-1} is set to the next smaller cost coefficient with respect to c_{p-1} to derive a new threshold value for b_p.
This procedure has to be repeated, until the problem becomes infeasible for the first time, after the bound on b_1 has moved to the next smaller cost coefficient with respect to c_1 in the previous iteration. Then, it is automatically implied that the ε-constraint problem has been solved for all possible right hand side values that result in a feasible single objective bottleneck problem, and the procedure stops.
For further applications of Algorithm 4.2 we refer to Section 6.2 where modified versions of this algorithm are used to solve scalarized versions of multiple objective combinatorial problems involving the algebraic sum of a sum and several bottleneck objectives. In addition, a special version of Algorithm 4.2 is used in Section 8.2 to solve the biobjective binary knapsack problem with two bottleneck objectives.

4.3 Conclusions and Further Ideas

Since the research in the field of multiple objective combinatorial optimization has grown strongly over the last decades, it is impossible to give a comprehensive and detailed survey of the existing literature in only a few pages. In this chapter, we restricted ourselves to survey only those solution approaches that can be used inde-

pendently of any specific class of combinatorial problems. After a short review of the most important definitions and concepts of multiple objective combinatorial optimization, we summarized existing solution approaches in multiple objective combinatorial optimization for the two most common objective functions used in the literature. In addition, we presented a generalized solution approach for combinatorial problems with an arbitrary number of bottleneck objectives that is based on solving a sequence of single objective unconstrained combinatorial problems.

For further research directions, we remark that there already exist many solution approaches and algorithms based on the two-phase method applied to a variety of biobjective combinatorial optimization problems. In contrast, the literature on triobjective problems for this method is very scarce. Due to the fact that the scalarized problems in the first phase of this method can be solved efficiently by an appropriate algorithm for the associated single objective problem, it seems to be reasonable to further focus on generalized solution concepts that can be applied in the second phase of this method.

Unfortunately, not all approaches discussed in this chapter can be generalized to higher dimensions in a straightforward way. For example, considering the Box-Algorithm stated in Hamacher et al. [94] for the biobjective case, it is not clear in advance how the splitting procedure for the 2-dimensional boxes can be generalized to p-dimensions, where $p \geq 3$. In this regard, neighborhood search techniques seem to be a promising approach, since their performance is not directly related to the number of given objectives. However, the success of such an approach crucially depends on the considered definition of adjacency of efficient solutions and the value of the cost coefficients of the different objective functions involved, as we will show in Chapter 7.

Concerning the compromise programming method mentioned in Section 4.1, only a rather weak general performance of this method is reported in the literature, since the involved subproblems are usually $\mathcal{NP}$-complete (cf., e.g., Murthy and Her [147] for the shortest path problem). Nevertheless, improved solution concepts based on this method are of interest. For example, when an augmented weighted Tchebycheff norm is used to calculate the non-dominated solutions of the multiple objective problem, an appropriate weight parameter ρ (cf. Equation (4.1)) is normally chosen in advance and kept fixed throughout the solution process. However, the (numerical) performance of this specific solution approach crucially depends on the choice of this parameter that actually determines, whether all non-dominated solutions of the problem can be found or not (cf. Steuer [197] for further details). Hence, an adaptive choice of this parameter dependent on the coordinates of the involved (local) ideal and Nadir points can improve the general performance of this solution approach. For further details, we refer to Dächert et al. [44].

Finally, improvement versions of Algorithm 4.2 for special classes of combinatorial problems could be of interest. Since the individual cost coefficients of $\tilde{w}$ only differ slightly in consecutive iterations of the algorithm, it could be investigated whether optimal solutions from preceding iterations can be used to derive an optimal solution for the current subproblem. Such an approach may result in an improved time complexity dependent on the considered class of combinatorial problems.

Chapter 5

Combinatorial Optimization with k-max Objectives

Besides sum and bottleneck objectives that are discussed in Chapter 4, also other types of objectives may be of interest to model special problems in combinatorial optimization, where common types of objectives fail to describe the given problem appropriately. In this chapter we present a bottleneck-like objective function that does not take into account the largest, but the k^{th} largest cost coefficient of a given feasible solution. We present solution approaches for the single as well as for the multiple objective case involving this special type of objective in the context of minimization problems. Based on the presented results we derive similar approaches for the case when the k^{th} smallest cost coefficient of a feasible solution has to be minimized.
We assume throughout this chapter that a ground set $\mathcal{E}$ of n distinct elements is given. Furthermore, $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$ denotes a subset of the power set of $\mathcal{E}$, also called the *feasible set*. We restrict ourselves in this chapter to combinatorial optimization problems where $|S| \geq m$ is ensured for some $m \in \{1, \ldots, n\}$ and all $S \in \mathcal{X}$.
This chapter is organized as follows: In Section 5.1 we discuss the concept of a k-max objective function and present an algorithm that solves the k-max minimization problem. In Section 5.2 we deal with the multiple objective case, while we transfer the results for k-max objectives to the case that the k^{th} smallest cost coefficient of a feasible solution has to be minimized in Section 5.3. We conclude in Section 5.4.
We remark that the results presented in Section 5.1 are already published in Gorski and Ruzika [88]. In addition we note that the special types of objectives treated in this chapter where independently investigated by Turner [203]. For detailed results we refer to Turner [204].

5.1 Single Objective k-max Optimization Problems

In this first section we consider combinatorial optimization problems where the k^{th} largest cost coefficient of a feasible solution has to be minimized. We assume in the following that $|S| = m$ holds true for all $S \in \mathcal{X}$, since this simplifies the notation. Equivalent results for the case that $|S| \geq m$ for all $S \in \mathcal{X}$ can easily be derived from the results we present in the remainder of this section.
Let $c\colon \mathcal{E} \to \mathbb{Z}$ be a cost function on the elements of the ground set $\mathcal{E}$. We assume that

the elements of $\mathcal{E}$ are renumbered such that $c(e_1) \leq \ldots \leq c(e_n)$. Let $S = \{e_{i_1}, \ldots, e_{i_m}\}$ be a feasible solution with $1 \leq i_1 < \ldots < i_m \leq n$. We define an operator k-max which yields the k^{th} largest among the elements of S, i.e.,

$$k\text{-max}(S) \;=\; k\text{-}\max_{e \in S} c(e) \;=\; c(e_{i_{m-k+1}}).$$

The problem of minimizing the k^{th} largest cost coefficient can now be concisely formulated as

$$\begin{aligned} &\min\ k\text{-}\max_{e \in S}\{c(e)\} \\ &\text{s.t.}\ \ S \in \mathcal{X}. \end{aligned} \tag{kMAX}$$

We refer to this problem as *k-max optimization problem* (kMAX). The special case $k = 1$ is the well-known bottleneck or min-max problem, already treated in Section 4.2. We introduce the following convention concerning the notation of the problem: If we refer to Problem (kMAX) and its specific objective function in a general way, we speak of a k-max problem and of a k-max objective. If a special instance of the problem is considered for a fixed $k \in \{1, \ldots, m\}$, we write $k\text{-max}_{e \in S}\{c(e)\}$ or equivalently k-max(S).

Problems of type (kMAX) have not been intensely studied in the literature before. However, the notation of the k^{th} largest element of a feasible set S can be found in Punnen and Aneja [174]. There, the minimization of the maximum deviation of the cost coefficients of a feasible solution $S \in \mathcal{X}$ to its k^{th} largest coefficient is discussed for general combinatorial optimization problems.

Although Problem (kMAX) has not been considered, there exist several other interesting generalizations of CBPs. Some of them shall be mentioned in the following since they slightly resemble Problem (kMAX). In Punnen et al. [178] the ground set $\mathcal{E}$ is partitioned into p non-empty disjoint subsets. For feasible $S \in \mathcal{X}$ the objective is to minimize the sum of costs of those elements which have maximum cost in the subsets $S \cap \mathcal{E}_k$ for $k = 1, \ldots, p$. The considered problem generalizes combinatorial bottleneck problems (CBP) and combinatorial sum problems (CSP) simultaneously, since for $p = 1$ the problem simplifies to a CBP, while a CSP has to be solved for $p = |\mathcal{E}|$. A generalized CSP is discussed in Punnen and Aneja [173]. Instead of minimizing the complete sum of all cost coefficients of a feasible solution, only the sum of the k largest cost coefficients is considered (cf. also Section 6.2). Note that this approach also contains the bottleneck case, since for $k = 1$ the problem simplifies to a CBP, while for $k \geq \max\{|S| : S \in \mathcal{X}\}$ an ordinary CSP has to be solved. In lexicographic CBP (see Della Croce et al. [47], Sokkalingam and Aneja [193]), the largest cost coefficient has to be minimized in the first place. Among all optimal solutions to this CBP, the second largest cost coefficient has to be optimized, then the third largest and so on. Moreover, there exist several studies on CBPs with fixed cardinality (for a survey, we refer to Ehrgott et al. [57]), i.e. $|S| = m$ for all $S \in \mathcal{X}$ is required.

5.1.1 An Algorithm for Solving Single Objective k-max Problems

The algorithm for solving Problem (kMAX) presented in this subsection is applicable to general combinatorial optimization problems, since we only require the solution of

Algorithm 5.1 Bisection Algorithm for k-max Optimization Problems

Input: A ground set $\mathcal{E}$, a set of feasible solutions $\mathcal{X} \subset \mathcal{P}(\mathcal{E})$, and $k \in \mathbb{N}$.
Output: A feasible solution $S \in \mathcal{X}$ being optimal to Problem (kMAX).

1: Sort elements $e \in \mathcal{E}$ such that $c(e_i) \leq c(e_{i+1})$.
2: LB $\leftarrow 1$
3: UB $\leftarrow n$
4: $j \leftarrow \lceil \frac{n}{2} \rceil$
5: **while** $|\text{UB} - \text{LB}| > 0$ **do**
6: Assign costs d_j to each $e \in \mathcal{E}$.
7: Solve Problem (SP). Denote S the optimal solution and $d(S)$ the optimal value.
8: **if** $d_j(S) < k$ **then**
9: UB $\leftarrow j$
10: **else**
11: LB $\leftarrow j + 1$
12: **end if**
13: $j \leftarrow \text{LB} + \lfloor \frac{\text{UB}-\text{LB}}{2} \rfloor$
14: **end while**
15: Solve Problem (SP). Denote S the optimal solution and $d(S)$ the optimal value.
16: **return** S.

a sequence of CSPs. We utilize bisection search for the k^{th} largest cost coefficient in an optimal solution: In each iteration, we decide whether there exists a solution whose k^{th} largest cost coefficient is smaller than a given cost coefficient $c(e_i)$, for a fixed $i \in \{1, \ldots, n\}$. This decision is based on the solution of a sum problem having the same feasible set as Problem (kMAX).
Given $j \in \mathbb{N}$ with $1 \leq j \leq n$, we assign auxiliary costs to each element in the ground set by setting

$$d_j(e_i) := \begin{cases} 0, & \text{if} \quad i \leq j \\ 1, & \text{if} \quad i > j. \end{cases}$$

The sum problem which is iteratively solved during the algorithm is given by

$$\min_{S \in \mathcal{X}} d_j(S) := \sum_{e \in S} d_j(e). \tag{SP}$$

Since the costs are binary, Problem (SP) may be easier to solve than general CSPs over the same feasible set. Indeed, there exist combinatorial optimization problems where solving the sum objective problem is $\mathcal{NP}$-hard in general, while for only two distinct cost coefficients on $\mathcal{E}$ there exists a polynomial time algorithm to solve problems of type (SP) (see Ravi et al. [181] and Section 5.1.2).
An outline of our algorithm can be found in Algorithm 5.1. Note that for given $k \in \{1, \ldots, n\}$, the upper bound UB also could be initialized by $n - k + 1$ instead of n in Line 3 of Algorithm 5.1. Furthermore, the index $m - k + 1$ could alternatively be used as an initial value of the lower bound LB (cf. Line 2). In the following, we consider finiteness and correctness of our approach.

Theorem 5.1 *Algorithm 5.1 terminates in a finite number of steps. The solution S it returns is optimal for Problem (kMAX).*

Proof: In each iteration the value of LB or UB is increased or decreased by at least one unit, respectively. Therefore, termination is guaranteed. We prove the validity of the following loop-invariant which implies the correctness of the algorithm.

> In each iteration of the **while**-loop, the index of the k^{th} largest element of an optimal solution to Problem (kMAX) is contained in the set $\{t \in \mathbb{N} : \text{LB} \leq t \leq \text{UB}\}$.

Initialization: Lines 2 and 3 ensure that the loop invariant is valid at the start of the first iteration.

Maintenance: In each iteration, a cost value of zero is assigned to all elements having an index smaller or equal to j. All other elements is assigned a cost value of one. Note that Problem (kMAX) and Problem (SP) have the same feasible set. We denote an optimal solution of Problem (kMAX) by S^{opt}. The objective function value $d_j(S)$ of Problem (SP) counts the minimum number of elements in S that have an index larger than j.

Case 1: Let $d_j(S) < k$.
Then there exists a feasible solution whose k^{th} largest element of S is at most $c(e_j)$. Therefore, the index of the k^{th} largest element of S^{opt} is contained in $\{LB, \ldots, j\}$.

Case 2: Let $d_j(S) \geq k$.
By contradiction, suppose that the cost of the k^{th} largest element of S^{opt} is less or equal to $c(e_j)$. Consider the value $d_j(S^{opt})$. It is $d_j(S^{opt}) < k$ which is a contradiction to S being optimal for Problem (SP). Thus, the index of the k^{th} largest element of S^{opt} is contained in the set $\{j+1, \ldots, \text{UB}\}$.

Termination: We repeat the **while**-loop until UB = LB. In this case, $j = \text{UB} = \text{LB}$ and $c(e_j)$ is the optimal objective function value.

Due to the validity of the loop invariant, Line 15 ensures that an optimal solution S of Problem (kMAX) with $k\text{-max}(S) = c(e_j)$ is returned. □

However, it cannot be guaranteed that the optimal solution S of Problem (SP) generated in the last iteration of the **while**-loop is also optimal for Problem (kMAX) (cf. the example in Section 5.1.2). Therefore, Problem (SP) has to be solved once more in Line 15 after the optimal value for j has been found.

Lemma 5.2 *In the **while**-loop of Algorithm 5.1 at least one optimal solution S of Problem (kMAX) is computed when solving Problems* (SP) *in Line 7. More precisely, S is the optimal solution to the Problem* (SP) *for which the upper bound UB was updated the last time before the stopping criterion of the **while**-loop was satisfied.*

Proof: Let S denote the optimal solution to the Problem (SP), for which the upper bound was updated the last time before the stopping criterion of the **while**-loop was satisfied, and let $c(e_i) := k\text{-max}(S)$. By contradiction, suppose that S is not optimal for Problem (kMAX), i.e., there exists another feasible solution $S^\star \in \mathcal{X}$ such that $c(e_{i^\star}) := k\text{-max}(S^\star) < c(e_i)$, $i^\star < i$. Without loss of generality we assume that $\text{UB} = i$ holds true at the end of the iteration, when the solution S is obtained. Otherwise,

we have to perform at least one more subsequent iteration where the upper bound will be updated. Since the optimal $i^\star$ has to be contained in set $\{\text{LB}, \dots, \text{UB} - 1\}$ according to the loop-invariant, we have to perform at least one more iteration. Due to the definition of S, all subsequent iterations will never lead to an update of the upper bound UB until the stopping criterion is met. Hence, $i^\star \notin \{\text{LB}, \dots, j\}$ for all $\text{LB} \leq j \leq \text{UB} - 1$ and after the final iteration $i^\star = \text{LB} = \text{UB} = i$ holds true. This contradicts our assumption. □

Due to Lemma 5.2, Line 15 in Algorithm 5.1 can be omitted when storing the solution S which has led to an update of the upper bound UB at the end of the **while**-loop. At termination, this solution is returned by the algorithm.
Note that in the course of iterations, an optimal solution to Problem (kMAX) is not always necessarily contained in the current set of optimal solutions to Problem (SP). Especially in earlier iterations only a coarse estimation of the position of the optimal cost coefficient to Problem (kMAX) is available. Thus, there may exist many feasible solutions having a better value with respect to the current objective d_j of Problem (SP) than optimal solutions of Problem (kMAX).
Finally, we analyze the running time of our algorithm.

Theorem 5.3 *The running time of Algorithm 5.1 is in $\mathcal{O}(T \log n)$, where T denotes the time needed for solving Problem* (SP)*.*

Proof: First, we note that n elements can be sorted in $\mathcal{O}(n \log n)$ which can be assumed to be in $\mathcal{O}(T \log n)$. Line 13 guarantees that the **while**-loop is bisected in every iteration. Consequently, there are $\log n$ many iterations. Line 6 is in $\mathcal{O}(n)$, Line 7 is in $\mathcal{O}(T)$, all other operations in $\mathcal{O}(1)$ and, thus, the running time follows. □

5.1.2 Example

To demonstrate the properties of Algorithm 5.1 introduced in Section 5.1.1 we consider the k-max cardinality constrained knapsack problem

$$\min_{S \in \mathcal{X}} k\text{-}\max_{e \in S} \{c(e)\} \tag{5.1}$$

with ground set $\mathcal{E} = \{e_1, \dots, e_n\}$ and feasible set

$$\mathcal{X} = \{S \in \mathcal{P}(\mathcal{E}) : |S| = m \wedge \sum_{e \in S} p(e) \geq \text{const.}\}$$

for arbitrary but fixed $0 \leq m \leq n$ and $0 \leq k \leq m$. The vectors $c \in \mathbb{R}^n$ and $p \in \mathbb{R}^n$ are called *cost vector* and *profit vector,* respectively. It is well known that the cardinality constrained knapsack problem with sum objective is $\mathcal{NP}$-hard in general (see Mazzola and Schantz [131]). However, if the cost vector c is binary, i.e. $c \in \{0, 1\}^n$, as it is the case for Problem (SP), we propose the following approach:

1. Partition the ground set $\mathcal{E} = \mathcal{E}_0 \cup \mathcal{E}_1$ into two disjoint subsets $\mathcal{E}_i := \{e \in \mathcal{E} : c(e) = i\}$, $i \in \{0, 1\}$.
2. Sort the elements of $\mathcal{E}_0$ and $\mathcal{E}_1$ in non-increasing order according to their profit $p(e)$.

$c(e)$	1	2	3	4	5	6	7	8	9	10	11	$\sum_{e\in S} p(e)$
$p(e)$	45	54	21	10	14	58	15	56	17	43	27	
S_1	1	1	0	0	0	**1**	0	1	0	1	1	283
S_2	1	1	**1**	0	0	1	0	1	0	1	0	277
S_3	1	1	0	0	0	**1**	0	1	1	1	0	273
S_4	1	1	0	0	0	**1**	1	1	0	1	0	271
S_5	1	1	0	0	**1**	1	0	1	0	1	0	270
S_6	1	1	0	**1**	0	1	0	1	0	1	0	266
S_7	1	1	**1**	0	0	1	0	1	0	0	1	261

Table 5.1: *In the first two rows the assigned values to c and p are listed, while in the subsequent rows the feasible solutions of* (5.2) *are shown.*

3. Construct a first solution candidate S that contains the first m most profitable elements of $\mathcal{E}_0$ if $|\mathcal{E}_0| \geq m$, or all elements of $\mathcal{E}_0$ and the $m - |\mathcal{E}_0|$ most profitable elements of $\mathcal{E}_1$ if $|\mathcal{E}_0| < m$, respectively.

4. If the solution S satisfies the constraint

$$\sum_{e\in S} p(e) \geq \text{const.},$$

it must be optimal by construction.

5. Otherwise iteratively replace the most unprofitable element $e_{\text{out}} := \text{argmin}\{p(e) : e \in (S \cap \mathcal{E}_0)\}$ of $S \cap \mathcal{E}_0$ by the most profitable element $e_{\text{in}} := \text{argmax}\{p(e) : e \in (\mathcal{E}_1 \setminus S)\}$ of $\mathcal{E}_1 \setminus S$ until the sum constraint (4) is met for the first time. If this is never the case, the given problem is infeasible.

We state:

Lemma 5.4 *The presented approach solves the binary sum-version of the cardinality constrained knapsack problem within a polynomial amount of time.*

As the proof of Lemma 5.4 is straightforward, we leave the details to the reader. As a consequence of Lemma 5.4, Problem (5.1) can also be solved in polynomial time although the same problem with sum objective is $\mathcal{NP}$-hard in general.
In the following we consider the *4*-max cardinality constrained knapsack problem with the ground set $\mathcal{E} = \{e_1, \ldots, e_{11}\}$ and the feasible set

$$\mathcal{X} := \left\{ S \in \mathcal{P}(\mathcal{E}) : |S| = 6 \wedge \sum_{e\in S} p(e) \geq 261 \right\}, \tag{5.2}$$

where the profits p and the costs c are specified in the first two rows of Table 5.1. This problem has seven feasible solutions $S_1, \ldots, S_7$. They are also listed in Table 5.1.

Iteration	S^{opt}	$d_j(S^{\text{opt}})$	LB	UB	j	$\lvert\text{UB} - \text{LB}\rvert$
prior to 1st	–	–	1	11	6	10
1st	$\{S_2, S_5, S_6, S_7\}$	2	1	6	3	5
2nd	$\{S_2, S_7\}$	3	1	3	2	2
3rd	$\{S_1, \ldots, S_7\}$	4	3	3	3	0

Table 5.2: *The rows give the values of the algorithm before and after each iteration.*

Note that $c(e_i) = i$ for all $i \in \{1, \ldots, 11\}$. Obviously, the solutions S_2 and S_7 are both optimal for the given 4-max-problem with $4\text{-max}(S_2) = 4\text{-max}(S_7) = c(e_3) = 3$.
The values assigned to the variables LB, UB, and j during the course of Algorithm 5.1 as well as the set of optimal solutions for each subproblem (SP) and the optimal objective values with respect to d_j can be found in Table 5.2.
The algorithm stops after three iterations. In the first two iterations, the optimal objective value of d_j is less than $k = 4$. In these cases the upper bound UB is updated using the current value of j. In the last iteration, the optimal objective value of d_j equals 4, and the lower bound LB is updated to $j+1$. This leads to $\text{UB} = \text{LB} = j = 3$ and the stopping criterion of the while-loop is satisfied.
In the first iteration the set of optimal solutions for the subproblem (SP) consists of the four feasible solutions S_2, S_5, S_6, and S_7, while in the second iteration S_2 and S_7 are optimal for Problem (5.1). In the third iteration, all feasible solutions of Problem (5.1) are also optimal for Problem (SP). According to the loop-invariant used in the proof of Theorem 5.1, $j = 3$ corresponds to the index of the optimal objective value of Problem (5.1). Due to Lemma 5.2, S_2 and S_7 are both optimal for Problem (5.1), since they are optimal for Problem (SP) in the iteration in which the upper bound is updated the last time before the stopping criterion of the **while**-loop is met. Note that the optimal solution S obtained when solving Problem (SP) the last time during the **while**-loop does not have to correspond to an optimal solution of Problem (5.1) since, for example, $S = S_1$ is possible.

5.1.3 Further Applications

Despite the fact that Problem (kMAX) generalizes CBPs, it may enable the modeling of many real world problems which could not have been easily formulated so far. We want to demonstrate briefly its potential in image registration (cf. also Section 11.3). In general registration problems, two given data sets have to be rendered in a joint coordinate system such that corresponding features of these data sets are aligned. Such data sets normally correspond to 2- or 3-dimensional images as it is the case, for example, in medical image registration (see, e.g., Zitová and Flusser [223] for more details). For a given model set A of n distinct points in the plane (which may correspond for example to characteristic points of a given reference image) and an image set B of the same cardinality (one may think of characteristic points in a template image), it is assumed that the points in B correspond to points in A but are afflicted with some data errors. The goal is to find the best possible assignment

between the points of the two sets such that some distance measure between each pair of aligned points is as small as possible (see, e.g., Stiglmayr et al. [199]). Neither minimizing the average deviation of the points (which would be modeled by a CSP) nor the worst case assignment (this corresponds to a CBP) adequately deals with the noise-induced errors in the set B. Instead, it seems more suitable to disregard those $k-1$ pairs of points which are furthest apart from each other. These $k-1$ assignments are considered outliers due to noise, and the task in the optimization problem is to minimize the k^{th} largest distance in the assignment of set B to set A. However, the optimal choice of the accepted number of outliers (i.e. the parameter k) is critical and crucially depends on the given problem instance and the applied point extraction method.

5.2 Multiple Objective k-max Optimization Problems

Analogous to Section 4.2, where multiple objective problems with several bottleneck objectives are considered, we are interested in multiple objective problems with $p+1$ objective functions in the following, where the first objective $f : \mathcal{X} \to \mathbb{Z}$ is of arbitrary type, while the p other objectives $g_1, \ldots, g_p : \mathcal{X} \to \mathbb{Z}$ correspond to k-max objective functions.
Let $k_1, \ldots, k_p \in \{1, \ldots, m\}$ be arbitrary but fixed integers and let $c_1, \ldots, c_p : \mathcal{E} \to \mathbb{Z}$ be p different cost functions on the ground set $\mathcal{E}$. For $i \in \{1, \ldots, p\}$ the p k-max objective functions are given by

$$g_i(S) \;=\; k_i\text{-max}(S) \;=\; \underset{e \in S}{k_i\text{-max}}\{c_i(e)\}.$$

The considered *multiple objective k-max optimization problem* (MkMAX) is given by

$$\begin{aligned} &\min\ (f(S), g_1(S), \ldots, g_p(S))^\top \;=\; (f(S), k_1\text{-max}(S), \ldots, k_p\text{-max}(S))^\top \\ &\text{s.t.}\ \ S \in \mathcal{X}. \end{aligned} \tag{MkMAX}$$

Note that each function g_i can take at most $n-k_i+1$ different values which are known in advance. Hence, we state:

Lemma 5.5 *The cardinality of the non-dominated set $\mathcal{Y}_N$ of Problem (MkMAX) is within $\mathcal{O}(n^p)$.*

Proof: Let an instance of Problem (MkMAX) be given. Since for each combination of the $(n-k_i+1)^p$ different values of the vector $(g_1(S), \ldots, g_p(S))^\top$ there exists at most one value of $f(S)$ which leads to a non-dominated solution, $(n-k_i+1)^p$ is a strict upper bound on the set of non-dominated points in the objective space. □

As in the bottleneck case (cf. Section 4.2), an ε-constraint approach is suitable to solve Problem (MkMAX). The corresponding problem formulation is given by

$$\begin{aligned} \min\ & f(S) \\ \text{s.t. } & g_i(S) \le \varepsilon_i, \quad i = 1, \ldots, p, \\ & S \in \mathcal{X}, \end{aligned} \tag{5.3}$$

where $\varepsilon_i \in \{c_i(e_1), \ldots, c_i(e_n)\}$ for $i = 1, \ldots, p$. Since there is no obvious way how to handle the side constraints $g_i(S) \leq \varepsilon_i$ compared to the case of the bottleneck objectives, where, for example, all items $e \in \mathcal{E}$ satisfying $c_i(e) > \varepsilon_i$ for at least one $i \in \{1, \ldots, p\}$ can be simply deleted from the ground set $\mathcal{E}$, we use the following auxiliary construction adopted from the single objective case:
Let $i \in \{1, \ldots, p\}$. For fixed $j_i \in \{1, \ldots, n\}$, we define a cost indicator for g_i by setting

$$\tau_{j_i} := c_i(e_{j_i}).$$

We assign auxiliary weights to each element of the ground set by

$$w_{j_i}(e_t) := \begin{cases} 0, & \text{if} \quad c_i(e_t) \leq \tau_{j_i}, \\ 1, & \text{if} \quad c_i(e_t) > \tau_{j_i}, \end{cases}$$

where $t \in \{1, \ldots, n\}$. The new i^{th} auxiliary function $f_{j_i} : \mathcal{X} \to \mathbb{N}$ is now given by

$$f_{j_i}(S) := \sum_{e \in S} w_{j_i}(e).$$

Let $J := (j_1, \ldots, j_p)$. Instead of solving Problem (5.3) directly, we consider the auxiliary problem

$$\begin{aligned} \min \; & f(S) \\ \text{s.t. } & f_{j_i}(S) \leq k_i - 1, \quad i = 1, \ldots, p, \\ & S \in \mathcal{X}. \end{aligned} \tag{A_J}$$

where $j_i \in \{1, \ldots, n\}$ is uniquely determined by the right hand side values ε_i of Problem (5.3) for $i = 1, \ldots, p$. In more detail, j_i is given by the index $t \in \{1, \ldots, n\}$ such that $c_i(e_t) = \varepsilon_i$ holds true.
Problem (A_J) can be interpreted as follows: For each $i \in \{1, \ldots, p\}$, the side constraint $f_{j_i}(S) \leq k - 1$ introduces an additional knapsack constraint to the single objective combinatorial optimization problem with objective function f. These side constraints can be seen as counters on the cardinality of the most expensive elements from $\mathcal{E}$ which are contained in a feasible solution with respect to the given cost functions c_i. For fixed j_i only up to $k - 1$ of these most expensive elements with respect to the cost function c_i are allowed to be included in a feasible solution, where the threshold value τ_{j_i} indicates which elements have to be considered as "expensive" and which not. So the problem can also be seen as a multiple choice version of the considered combinatorial optimization problem.
In the following, we relate optimal solutions of Problem (A_J) to efficient solutions of Problem (MkMAX) and vice versa.

Theorem 5.6 *Let S_J denote an optimal solution of Problem (A_J) for some $J = (j_1, \ldots, j_p)$. Then S_J is a weakly efficient solution of Problem (MkMAX).*

Proof: Assume that S_J is not a weakly efficient solution of Problem (MkMAX), i.e. there exists $S^\star \in \mathcal{X}$ such that $f(S^\star) < f(S_J)$ and

$$k_i\text{-}\max_{e \in S^\star}\{c_i(e)\} < k_i\text{-}\max_{e \in S_J}\{c_i(e)\} \quad \forall i \in \{1, \ldots, p\}. \tag{5.4}$$

Since S_J is feasible for Problem (A_J),

$$\underset{e\in S_J}{k_i\text{-max}}\{c_i(e)\} \leq \tau_{j_i}$$

for $i = 1, \ldots, p$ by the construction of f_{j_i}. Using (5.4), we obtain that $S^\star$ is also feasible for Problem (A_J) and $f(S^\star) < f(S_J)$. This contradicts the optimality of S_J for Problem (A_J). □

In order to strengthen the result of Theorem 5.6, i.e. to obtain efficiency, uniqueness of the optimal solution of Problem (A_J) yields a sufficient condition.

Theorem 5.7 *Let S_J be a unique optimal solution of Problem (A_J) for some $J = (j_1, \ldots, j_p)$. Then S_J is a strictly efficient solution of Problem (MkMAX).*

Proof: Assume that S_J is not a strictly efficient solution of Problem (MkMAX), i.e. there exists $S^\star \in \mathcal{X}$, $S^\star \neq S_J$, such that

$$f(S^\star) < f(S_J) \text{ and } \underset{e\in S^\star}{k_i\text{-max}}\{c_i(e)\} \leq \underset{e\in S_J}{k_i\text{-max}}\{c_i(e)\} \quad \forall i \in \{1, \ldots, p\},$$

or

$$f(S^\star) \leq f(S_J) \text{ and } \underset{e\in S^\star}{k_i\text{-max}}\{c_i(e)\} \leq \underset{e\in S_J}{k_i\text{-max}}\{c_i(e)\} \quad \forall i \in \{1, \ldots, p\},$$

and there exists at least one index $t \in \{1, \ldots, p\}$ such that

$$\underset{e\in S^\star}{k_t\text{-max}}\{c_t(e)\} < \underset{e\in S_J}{k_t\text{-max}}\{c_t(e)\}.$$

Applying the same reasoning as in the proof of Theorem 5.6 shows that $S^\star$ is feasible for Problem (A_J) in both cases. But since S_J is the unique optimal solution of Problem (A_J), this implies that $f(S_J) < f(S)$ for all feasible S. Hence, it holds that $f(S_J) < f(S^\star)$ which leads to a contradiction. □

For efficient solutions of Problem (MkMAX), we state:

Theorem 5.8 *If $S^\star$ is an efficient solution of Problem (MkMAX), then there exists $J = (j_1, \ldots, j_p) \in \{1, \ldots, n\}^p$ such that $S^\star$ is optimal for Problem (A_J).*

Proof: Let $S^\star$ be an efficient solution of Problem (MkMAX). For each $i \in \{1, \ldots, p\}$ let $j_i \in \{1, \ldots, n\}$ denote the index of that element $e \in \mathcal{E}$ which corresponds to the k_i-max of $S^\star$. We set $\tau_{j_i} = c_i(e_{j_i})$, and we claim that $S^\star$ is an optimal solution to Problem (A_J) for $J = (j_1, \ldots, j_p)$.
Assume that this is not the case. Then there exists a feasible $\hat{S} \in \mathcal{X}$ with $f(\hat{S}) < f(S^\star)$ and $f_{j_i}(\hat{S}) \leq k_i - 1$ for $i = 1, \ldots, p$. By the choice of j_i, $f_{j_i}(S^\star) = k_i - 1$ and hence $f_{j_i}(\hat{S}) \leq f_{j_i}(S^\star)$. But this means by the definition of Problem (A_J) and w_{j_i} that $k_i\text{-max}(\hat{S}) \leq k_i\text{-max}(S^\star)$ for all $i \in \{1, \ldots, p\}$. Hence, $\hat{S}$ dominates $S^\star$ which contradicts the efficiency of $S^\star$. □

Note that the above stated results on the relation between optimal solutions of Problem (A_J) and efficient solutions of Problem (MkMAX) are equivalent to the results for the general ε-constraint approach for solving MCOPs (cf. Chankong and Haimes

Algorithm 5.2 Algorithm for Problem (MkMAX)

Input: An instance $(\mathcal{E}, \mathcal{X}, (f, g_1, \ldots, g_p))$ of Problem (MkMAX).
Output: A complete subset $\mathcal{X}^\star \subseteq \mathcal{X}_E$.

1: Set $X^\star = \emptyset$ and $\mathcal{J} = \{1, \ldots, n\}^p$.
2: **for** All $J \in \mathcal{J}$ **do**
3: Determine the optimal solution $S^\star$ of the constrained optimization Problem (A_J), whenever it is feasible.
4: Update $\mathcal{X}^\star = \mathcal{X}^\star \cup \{S^\star\}$.
5: **end for**
6: Filter $\mathcal{X}^\star$ for dominated solutions.
7: **return** $\mathcal{X}^\star$.

[36]). But since Problem (A_J) is just a remodeling of the ε-constraint Problem (5.3), this is not surprising.
The algorithmic consequence of Theorem 5.6 and Theorem 5.8 can be found in Algorithm 5.2. The algorithm determines a complete set of the efficient solutions for any instance of Problem (MkMAX). As the solution concept of Algorithm 5.2 is similar to the concept used in Section 4.2 for multiple objective bottleneck problems, we suggest to handle the bounds on the k-max objectives similar to the approach described for Algorithm 4.2 in the above mentioned section. To shorten the algorithmic presentation, the case that Problem (A_J) is infeasible is not treated separately in Algorithm 5.2. Since an ε-constraint approach is used to solve an MCOP, a filtering for dominated solutions has to be performed at the end of the algorithm.

Theorem 5.9 *Algorithm 5.2 correctly determines the set of non-dominated solutions $\mathcal{Y}_N$ in $\mathcal{O}(n^p \cdot T)$, where T denotes the maximum time for solving the subproblem (A_J) in each step of the algorithm.*

Proof: Applying Theorem 5.6 and Theorem 5.8 we construct a subset of the weakly efficient set that contains all efficient solutions of Problem (MkMAX). Since Problem (A_J) is solved for all possible combinations of indices, the set $\mathcal{X}^\star$ which is returned after filtering the dominated solutions corresponds to a complete set of efficient solutions. Hence, the complete non-dominated set $\mathcal{Y}_N$ is calculated by Algorithm 5.2. The stated time bound follows directly from Lemma 5.5 and from the fact that we have to solve at most n^p modified ε-constraint problems given by Problem (A_J). □

Theorem 5.9 states a theoretical time bound that is independent from any special class of combinatorial problem. Since constrained versions of single objective combinatorial optimization problems that can be solved in polynomial time are $\mathcal{NP}$-hard problems in general, even if the minimum spanning tree (cf. Aggarwal et al. [2]) or the shortest-path problem (cf. Garey and Johnson [71]) is considered, Problem (A_J) is in general $\mathcal{NP}$-hard to solve. Nevertheless, due to the simple binary structure of the auxiliary sum objective function f_{j_i}, it may be possible to derive algorithms that are able to solve this special type of problem in a polynomial amount of time depending on the considered class of combinatorial problems. In this case, also Problem (MkMAX) is solvable in polynomial time due to Theorem 5.9.
We close this section with a short discussion of such a special type of combinatorial problem that is also treated in a generalized framework in Chapter 10. In more

detail, we consider the biobjective minimum spanning tree problem, where one of the objective functions is given as a k-max objective. Let a connected graph $G = (V, A)$ be given, where V and A denote the set of nodes and the set of edges of G, respectively. Furthermore, let f and $g_1 = k_1\text{-max}(S)$ denote an arbitrary sum objective and a k-max objective, respectively, where $k_1 \in \{2, \ldots, |V|-1\}$. We use the results from Chapter 10, where biobjective matroid problems with a binary sum objective are discussed. For the minimum spanning tree problem under consideration, Problem (A_J) is equivalent to Problem $(BMP_{\leq})$ defined in Section 10.2, where the right hand side value for this specific problem is given by $k_1 - 1$. In Section 10.3 we show that an algorithm proposed by Gabow and Tarjan [67] can be used to solve Problem (A_J) efficiently. The authors stated a time bound of $\mathcal{O}(m + n \cdot \log(n))$ for their approach, where $|V| = n$ and $|A| = m$. Hence, we conclude:

Corollary 5.10 *The biobjective minimum spanning tree problem on a connected graph $G = (V, A)$ involving an arbitrary sum objective and a k-max objective can be solved in $\mathcal{O}(m^2 + mn\log(n))$, where $|V| = n$ and $|A| = m$.*

5.3 Optimization with k-min Objectives

We close this section with a short discussion of optimization problems with k-min objective(s). Since this type of function is closely related to a k-max objective, we will not go into all details, but we briefly state the main results that can be derived from the previous two sections.
Let $c\colon \mathcal{E} \to \mathbb{Z}$ denote a cost function on the elements of $\mathcal{E}$. We assume that the elements of $\mathcal{E}$ are given numbered in non-decreasing order with respect to their cost values. Let $S = \{e_{i_1}, \ldots, e_{i_m}\}$ be a feasible solution, where $1 \leq i_1 < \ldots < i_m \leq n$. We define the operator k-min which yields the k^{th} smallest among the elements of S, i.e.,

$$k\text{-min}(S) \;=\; k\text{-}\min_{e\in S} c(e) \;=\; c(e_{i_k}).$$

To establish a connection between a k-min and k-max objective, we recall the well-known fact that for all $S \subseteq \mathcal{E}$ it holds true that

$$\max\{c(e) : e \in S\} = -\min\{-c(e) : e \in S\}.$$

A similar relation can be proven for the two new types of objectives.

Lemma 5.11 *Let $S \subseteq \mathcal{E}$ such that $|S| = m$ with $1 \leq m \leq n$. Furthermore, let $k \in \{1, \ldots, m\}$ and $k^\star = m - k + 1$. Then,*

$$k\text{-}\max_{e\in S}\{c(e)\} \;=\; -\,k\text{-}\min_{e\in S}\{-c(e)\} \;=\; -\,k^\star\text{-}\max_{e\in S}\{-c(e)\},$$

or equivalently,

$$-\,k\text{-}\max_{e\in S}\{c(e)\} \;=\; k\text{-}\min_{e\in S}\{-c(e)\} \;=\; k^\star\text{-}\max_{e\in S}\{-c(e)\}.$$

Proof: Note that we will mainly use the second equation in the following. Hence, we omit to give a proof for the first and focus on the second. Since the transformation of the cost values from $c(e)$ to $-c(e)$ for all $e \in \mathcal{E}$ implies a reversion of the ordering relation on the elements in $\mathcal{E}$, the first equality is obviously true. Furthermore, since the cardinality of S is a fixed constant, the k^{th} smallest element of S is automatically the $(m-k+1)^{\text{th}}$ largest element of S, too. Hence, the second equality is also valid. □

Similar to Problem (kMAX), the problem of minimizing the k^{th} smallest among the cost coefficients of a feasible solution is given by

$$\begin{aligned} &\min\ k\text{-}\min_{e\in S}\{c(e)\} \\ &\text{s.t.}\ \ S \in \mathcal{X}, \end{aligned} \tag{kMIN}$$

Lemma 5.11 yields an approach how Problem (kMIN) can be solved, when all feasible sets contained in $\mathcal{X}$ are of the same cardinality, i.e. $|S| = m$ for all $S \in \mathcal{X}$ where $m \in \{1, \ldots, n\}$ is a fixed constant. Instead of minimizing the k^{th} smallest element of S, we minimize the $(m-k+1)^{\text{th}}$ largest element of S for the same cost function c. Since there are no further restrictions for the cost coefficients used in Algorithm 5.1, this algorithm can be used to solve Problem (kMIN).
If the cardinality of feasible solutions varies, i.e. if there exist $S_1, S_2 \in \mathcal{X}$ such that $|S_1| \neq |S_2|$, only the first equality of Lemma 5.11 is valid, while the second can no longer be applied simultaneously to all feasible solutions, since now m depends on S. For this case, we propose the following approach whose main idea is similar to the idea presented in Section 5.1:
For a fixed index $j \in \mathbb{N}$, where $1 \leq j \leq n$, we define auxiliary costs for each element in the ground set by

$$d_j(e_i) := \begin{cases} 1, & \text{if} \quad i \leq j, \\ 0, & \text{if} \quad i > j, \end{cases}$$

i.e. exactly in the reverse order as it was the case in Section 5.1.1 for a given k-max objective. Instead of solving Problem (SP), we now iteratively solve

$$\max_{S\in\mathcal{X}} d_j(S) := \sum_{e\in S} d_j(e). \tag{5.5}$$

If the maximum value of Problem (5.5) is at least k, there exists a feasible solution such that its k^{th} smallest element has an index that is not larger than j. Hence, an index that is at most j has to be the one that corresponds to the minimum cost coefficient that has to be determined. If the maximum value of Problem (5.5) is strictly smaller than k, there does not exist a feasible solution such that its k^{th} smallest element has an index that is at most j. We conclude that the index of the optimal cost coefficient is at least $j+1$. Hence, we can apply a bisection algorithm which is similar to Algorithm 5.1 to solve Problem (kMIN). According to Theorem 5.3 we state:

Corollary 5.12 *Problem* (kMIN) *can be solved within* $\mathcal{O}(T \log(n))$ *where* T *denotes the time needed for solving Problem* (5.5).

We further remark that

$$\max_{S\in\mathcal{X}} \left\{ k\text{-}\min_{e\in S}\{c(e)\} \right\} = -\min_{S\in\mathcal{X}} \left\{ -\,k\text{-}\min_{e\in S}\{c(e)\} \right\} = -\min_{S\in\mathcal{X}} \left\{ k\text{-}\max_{e\in S}\{-c(e)\} \right\}$$

holds true, according to Lemma 5.11. Hence, the problem of maximizing the k^{th} smallest among the cost coefficients of a feasible solution can be solved by applying Algorithm 5.1 to a corresponding k-max optimization problem using the transformed cost function $\tilde{c}$, where $\tilde{c}(e) = -c(e)$ for all $e \in \mathcal{E}$.
In the remainder of the section, we focus on the multiple objective case. First, we show how a constraint on a k-min objective can be transformed into a constraint similar to the one derived in Section 5.2 for a k-max objective to solve multiple objective optimization problems involving this type of objective. To keep the notation as simple as possible, we just describe how a single constraint of the form "$g(S) \leq c(e)$", where $g(S) = k\text{-min}(S)$ and $e \in \mathcal{E}$, can be transformed into an equivalent constraint based on a similar binary transformation as for the case of a k-max objective.
Let $e_j \in \mathcal{E}$ arbitrary but fixed. To ensure that the k^{th} smallest cost coefficient of a feasible solution $S \in \mathcal{X}$ is at most $\tau_j := c(e_j)$, we once more define auxiliary costs by setting

$$w_j(e) := \begin{cases} 0, & \text{if} \quad c(e) > \tau_j \\ 1, & \text{if} \quad c(e) \leq \tau_j. \end{cases}$$

Let $f_j(S) = \sum_{e \in S} w_j(e)$. Then, the side constraint "$g(S) \leq c(e_j)$" is equivalent to the constraint "$f_j(S) \geq k$": If the k^{th} smallest cost coefficient of S is at most $c(e_j)$, then S must contain at least k elements that have a cost value that does not exceed the threshold value τ_j. On the other hand, if the k^{th} smallest element of S is larger than $c(e_j)$, S cannot contain more than $k-1$ elements of costs less or equal to $c(e_j)$.
We conclude that the same approach as the one presented in Section 5.2 can be used to solve an MCOP involving an objective function f of arbitrary type and p k-min objectives. Furthermore, a similar time bound as the one stated in Theorem 5.9 is valid for this type of optimization problem.
We finally combine the two different objectives into one common multiple objective problem. Let a feasible set $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$, $p+q$ cost functions $c_i : \mathcal{E} \to \mathbb{Z}$, as well as a function f of arbitrary type, p k-max objectives $g_i(S) = k_i\text{-max}(S)$ $(i = 1, \ldots, p)$ and q k-min objectives $g_t(S) = k_t\text{-min}(S)$ $(t = p+1, \ldots, p+q)$ be given. We assume that $|S| \geq m$ for all $S \in \mathcal{X}$. Furthermore, let the cost function c_j be associated with the corresponding objective g_j for all $j \in \{1, \ldots, p+q\}$. Given g_j, we further assume that $k_j \in \{1, \ldots, m\}$ for all $j \in \{1, \ldots, p+q\}$. Note that several of the given cost functions may coincide. If this is the case, we further assume that the corresponding objectives do not describe the same element of S for all feasible S, i.e. if $c_i(e) = c_j(e)$ for all $e \in \mathcal{E}$ and some indices $i, j \in \{1, \ldots, p+q\}$, $i \neq j$, then there exist $S_1, S_2 \in \mathcal{X}$ such that $g_i(S_1) \neq g_j(S_2)$. We define the *multiple objective mixed k-min/k-max optimization problem* (MIXED) by

$$\begin{aligned} &\min \ (f(S), g_1(S), \ldots, g_p(S), g_{p+1}(S), \ldots, g_{p+q}(S))^\top \\ &\text{s.t.} \ \ S \in \mathcal{X}. \end{aligned} \tag{MIXED}$$

Applying the two different transformations for the k-max objectives (cf. Section 5.2) and the k-min objectives given in this section, we deduce the transformed ε-constraint problem that has to be solved at most $\mathcal{O}(n^{p+q})$ times to generate the complete non-dominated set of Problem (MIXED). Let $J := (j_1, \ldots, j_{p+q})$, where $j_i \in \{1, \ldots, n\}$ and let f_{j_i} and $f_{j_{p+t}}$ denote the transformed objective functions for the given p k-max

and q k-min objectives, respectively, where $i \in \{1, \ldots, p\}$ and $t \in \{1, \ldots, q\}$. Then, the associated ε-constraint problem is given by

$$\begin{aligned} \min\ & f(S) \\ \text{s.t.}\quad & f_{j_i}(S) \leq k_i - 1, \quad i = 1, \ldots, p, \\ & f_{j_{p+t}}(S) \geq k_{p+t}, \quad t = 1, \ldots, q, \\ & S \in \mathcal{X}. \end{aligned} \tag{M_J}$$

Using similar proofs as in the previous section, it is easy to show that given a vector J, each optimal solution of Problem (M_J) is at least weakly efficient for Problem (MIXED). Furthermore, for each efficient solution S of Problem (MIXED) there exists a vector J such that S is optimal for Problem (M_J). Hence, an algorithm similar to Algorithm 5.2 can be deduced that determines a complete set of efficient solutions of Problem (MIXED).

Corollary 5.13 *The set of all non-dominated solutions $\mathcal{Y}_N$ of Problem* (MIXED) *can be determined in $\mathcal{O}(T \cdot n^{p+q})$, where T denotes the time that is needed to solve Problem (M_J).*

Once more, if Problem (M_J) can be solved in polynomial time, also the non-dominated set of Problem (MIXED) can be determined in a polynomial amount of time.

5.4 Conclusions and Further Ideas

In this chapter we discussed two different types of objective functions for combinatorial problems which can be seen as a generalization of the bottleneck objective. The task is to find a feasible solution $S \in \mathcal{X}$ either minimizing the k^{th} largest or the k^{th} smallest cost coefficient of S. Algorithms to solve these two types of problems were proposed. Both algorithms are based on a successive reformulation of the considered problem as a simple sum problem with binary costs over the same feasible set. Based on this reformulation technique of the single objective problem, we derived an ε-constraint approach for solving multiple objective problems with an arbitrary number of k-max and/or k-min objectives.
Further research on combinatorial optimization problems involving k-max objectives could be manifold. For the single objective case, one might focus on the development of efficient algorithms that solve special classes of combinatorial optimization problems with a binary sum objective, since these problems correspond to the subproblems that have to be solved to find the optimal solution of a single objective k-max problem. Furthermore, one can focus on multiple objective problems with several k-max objectives. In general, the presented solution approach implies that a sequence of (multiply) constrained single objective combinatorial problems has to be considered that are $\mathcal{NP}$-hard to solve. However, the derived polynomial time algorithm for the biobjective minimum spanning tree problem with a sum and a k-max objective (cf. Corollary 5.10) encourages the hope that due to the simple structure of the involved subproblems, the overall problem is still solvable in a polynomial amount of time for special classes of combinatorial problems.

Furthermore, the complexity of such an algorithm could be improved, depending on the considered class of combinatorial problems. For instance, one can investigate whether it is possible that an optimal solution obtained in a previous iteration of Algorithm 5.2 presented in Section 5.2 can be used as a "warm start solution" for the next ε-constraint problem. Since only a few values of the involved cost functions change, the former optimal solution is either feasible for the new problem, or it could at least provide a good lower or upper bound on the optimal objective value of the new subproblem that has to be solved.
Following the ideas presented in Chapter 3, we finally remark that instead of solving the constraint Problem (A_J) in Algorithm 5.2 directly, an alternative approach based on the associated multiple objective optimization problem formulation could be favorable. Due to the simple structure of Problem (A_J), neighborhood search techniques (based on a simple exchange of items from the ground set) could be used to determine efficient solutions for the associated multiple objective problem with binary objectives that correspond to optimal solutions of Problem (A_J). Of course, if such an approach is applied, it must be guaranteed that the set of efficient solutions is connected with respect to the considered swap operation. For more details on this topic, we refer to Chapter 7. To motivate this approach we briefly mention that, for example in the case of biobjective matroid problems for an arbitrary and a binary sum objective, the important property of the connectedness of the efficient set can be proven (cf. Chapter 10). In this case, it is automatically guaranteed that the underlying k-max problem can be solved in a polynomial amount of time (cf. Corollary 5.10).

Chapter 6

Solving Single Objective Combinatorial Problems with Multiple Objective Approaches

While it is common to use single objective optimization approaches to solve multiple objective problems, we want to take the reverse approach in this chapter. Given a single objective combinatorial optimization problem, we model the problem in a more general multiple objective framework based on the ideas developed in Chapter 3 of this work. We focus on constrained versions of single objective combinatorial problems as well as on combinatorial problems with an algebraic sum objective in the following (cf. Sections 3.1 and 3.2 for more details). Based on associated multiple objective problem formulations of such problems, we show that many solution concepts stated in the literature can be interpreted as special versions of the more general approaches for multiple objective combinatorial problems presented in the Chapters 4 and 5. We further make use of this idea to present new solution concepts for special types of problems with algebraic sum objective. Amongst others, we show that these types of problems are solvable in a polynomial amount of time, whenever this holds true for the appropriate single objective subproblems that have to be solved in each iteration of the presented solution approach.
This chapter is organized as follows: In the first section we consider single objective combinatorial problems with an additional side constraint. In Section 6.2 we discuss combinatorial problems with an algebraic sum objective. We distinguish the two cases whether the involved objectives are defined on a single or on different cost functions on the ground set, followed by a recapitulation of our results in Section 6.3.

6.1 Constrained Single Objective Problems

In this section we deal with constrained versions of single objective combinatorial optimization problems and show how multiple objective solution approaches can be used to solve such problems. However, since up to now efficient algorithms that solve multiple objective combinatorial problems with more than two objectives are scarce, we restrict ourselves in the following to combinatorial sum problems with only a single side constraint. We further discuss this assumption at the end of this section. In the

following, we present a generalized version of a solution approach that has already been successfully applied to solve constrained problems for individual classes of combinatorial optimization problems. Starting from a short summary of the results found in the literature, we show how the general ideas developed in Section 3.1 can be used to derive a generalized solution concept based on an associated biobjective reformulation of the problem. Before we go into more details, we give a short description of the problem under consideration.
Let $\mathcal{E}$ denote a ground set of n distinct elements, and let $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$ be a subset of the powerset of $\mathcal{E}$. Furthermore, let $f, g : \mathcal{X} \to \mathbb{Z}$ denote two sum objectives. Then, a *combinatorial optimization problem with a single constraint* is formally defined by

$$\begin{aligned} \min\ & f(S) \\ \text{s.t. } & g(S) \leq \varepsilon \\ & S \in \mathcal{X}, \end{aligned} \tag{6.1}$$

where $\varepsilon \in \mathbb{R}$. According to the notation from Section 3.1, the associated biobjective combinatorial problem is given by

$$\begin{aligned} \min\ & F(S) = (f(S), g(S))^\top \\ \text{s.t. } & S \in \mathcal{X}. \end{aligned} \tag{6.2}$$

From Theorem 3.1 in Section 3.1 we recall:

Corollary 6.1 *It holds:*

1. *Every optimal solution of Problem* (6.1) *is at least weakly efficient for Problem* (6.2)*, and the set of optimal solutions of Problem* (6.1) *contains at least one efficient solution of Problem* (6.2)*.*

2. *There exists an efficient solution of Problem* (6.2) *that is also optimal for Problem* (6.1)*.*

Corollary 6.1 is of particular interest, whenever there exist polynomial time algorithms to solve Problem (6.2). However, it is beyond the focus of this section to give a complete survey of special classes of multiple objective combinatorial optimization problems that satisfy this property. Nevertheless, we refer to Chapter 10 of this work, where, amongst others, the biobjective spanning tree problem with a binary sum objective is discussed. We show that for this special case, the given biobjective problem can be solved in a polynomial amount of time.
Unfortunately, such results appear to be exceptional. In general, Problem (6.1) as well as Problem (6.2) are $\mathcal{NP}$-hard to solve for most of the classical combinatorial optimization problems, when no further information on at least one of the involved objectives is given. In particular, this holds true for many classes of combinatorial problems, where the unconstrained counterpart can be solved efficiently. For more details on this topic, we refer to Garey and Johnson [71] for the shortest-path problem, to Aggarwal et al. [2] for the minimum spanning tree and to Lieshout and Volgenant [124] for the linear assignment problem.
However, the general solution approach we discuss in the following mainly addresses classes of combinatorial problems, where at least the unconstrained version of the

problem can be solved in a polynomial amount of time. We show that in this case, an approach similar to the two-phase method for multiple objective combinatorial problems (cf. Section 4.1.2) is suitable to construct an optimal solution for Problem (6.1). Similar to the first phase of the original method, we solve a sequence of weighted sum scalarizations of the multiple objective problem. As we are only interested in a single non-dominated solution of Problem (6.2) rather than in the complete set of non-dominated solutions itself, we use the weighted sum scalarization of the biobjective problem to derive a first approximation of the optimal solution of Problem (6.1). This can be done efficiently, whenever the unconstrained counterpart of the considered combinatorial problem can be solved in a polynomial amount of time. As in the original two-phase method, a local search technique is finally used to derive (or at least to further approximate) the optimal solution of Problem (6.1).

It is important to remark that the approach we present in the remainder of this section can be seen as a generalization of analogous solution concepts that already have been successfully applied to many specific classes of combinatorial optimization problems. We summarize the most important literature in the following.

We start with the literature on the single constrained minimum spanning tree problem. Aggarwal et al. [2] as well as Hamacher and Ruhe [96] were the first who used a two phase approach, similar to the one described above, to solve the single constrained version of the minimum spanning tree problem. While in [2], a branch and discard technique was used to calculate the optimal solution of the problem in the second phase, Hamacher and Ruhe applied local search to derive the optimal solution. Later on, also Ziegelmann [222] made use of a two phase approach. The author applied a ranking approach in the second phase of the algorithm. Finally, Henn [102] discussed the weight-constrained minimum spanning tree problem in his diploma thesis. Besides a detailed overview on further approaches for solving this special type of problem, the author presented an alternative two phase approach mainly based on the connectedness of the supported efficient trees of the biobjective problem, also proven in Ruzika [184] (cf. Chapter 7 for further details on the connectedness of the efficient set). Taking advantage of this special property of the supported efficient spanning trees, an efficient branch and bound scheme was presented to derive an optimal solution for the constrained problem.

In addition to exact algorithms, also approximation schemes for the single constrained minimum spanning tree problem were presented in the literature. For example, Xue [219] derived a polynomial time approximation scheme for the problem that is mainly based on applying the first phase of the above described procedure only. For a detailed overview on further approximation schemes that are, amongst others, based on using the Lagrangian dual of the considered problem (cf.,e.g., Ravi and Goesmans [180]), we refer to Henn [102].

The constrained shortest path problem was considered by Aneja and Nair [7]. The authors applied a parametric approach in the first phase of the algorithm to calculate a sequence of supported efficient shortest paths. They wrongly claimed to calculate the optimal solution for the constrained problem with their method, ignoring the potential existence of non-supported efficient shortest paths that may be optimal for the constrained problem. In addition to [7], Handler and Zang [97] and Beasley and Christofides [15] presented two phase approaches to solve the single constrained shortest path problem. While both methods solve the Lagrangian dual of the prob-

lem in the first phase, Handler and Zang applied a k-best approach, while Beasley and Christofides used a specially designed branch and bound technique to derive an optimal solution of the constrained problem in the second phase. Ziegelmann [222] presented label setting as well as label correction methods that can be applied in the second phase of the above described procedure. Furthermore, numerical experiments were performed to compare the different two phase approaches. Finally, Ruzika [184] tackled the constrained shortest path problem in this dissertation. The author suggested an efficient branch & bound approach that is based on the connectedness of the supported efficient solutions of the associated biobjective shortest path problem (cf. also Chapter 7). For an overview on further solution concepts that can be applied to the constrained shortest path problem we refer to [222].
Finally, Aggarwal [1] presented a two phase approach for the constrained linear assignment problem. While in the first phase, the Lagrangian dual of the constrained problem is solved, the author applied a ranking method that had been developed by Murty [148] to derive an optimal solution for the constrained problem in the second phase of the algorithm.
Besides these approaches developed for special classes of combinatorial problems, Klamroth and Tind [114] investigated the relation between multiple objective optimization and constrained optimization problems. Amongst others, the authors discussed how an optimal solution to a single constrained continuous problem can be approximated only by means of an approach presented in Klamroth et al. [115]. This specific approach uses polyhedral distance functions to construct inner and outer approximations of the non-dominated set for the associated biobjective optimization problem.
In the following, we combine the main ideas of the solution concepts presented in the above stated references from the literature, to derive a generalized solution approach that can be applied to any combinatorial optimization problem with a single constraint. Let f and g denote two different integer valued sum objectives, defined on the same feasible set $\mathcal{X}$. For $\lambda \in [0,1]$ let

$$\begin{array}{ll} \min & w_\lambda(S) := \lambda f(S) + (1-\lambda) g(S) \\ \text{s.t.} & S \in \mathcal{X} \end{array} \qquad (\text{WS}(\lambda))$$

denote the weighted sum scalarization of f and g. Furthermore, let S_f and S_g denote the lexicographically optimal solutions with respect to (f,g) and (g,f), respectively. We recall from Ehrgott [54] that an optimal solution of Problem (WS(λ)) is also an efficient solution of Problem (6.2), whenever $\lambda \in (0,1)$. From the first part of Chapter 4 we recall that such solutions are also called supported efficient solutions of the problem. If in addition, such a solution corresponds to a breakpoint of the non-dominated frontier of the biobjective problem, it is called extreme efficient solution. In the following, let $\mathcal{X}_{sE}$ denote the set of all supported efficient solutions. To exclude trivial cases, we assume that Problem (6.1) is always feasible and that $g(S_f) > \varepsilon$, since otherwise S_f is feasible and hence, optimal for Problem (6.1).
Similar to the two-phase method, our general approach also decomposes into two different phases. In the first phase, we use an adaption of the dichotomic search technique to find two supported efficient solutions S^1 and S^2 of Problem (6.2) that approximate the optimal objective value of Problem (6.1) as well as possible (cf. Figure 6.1).

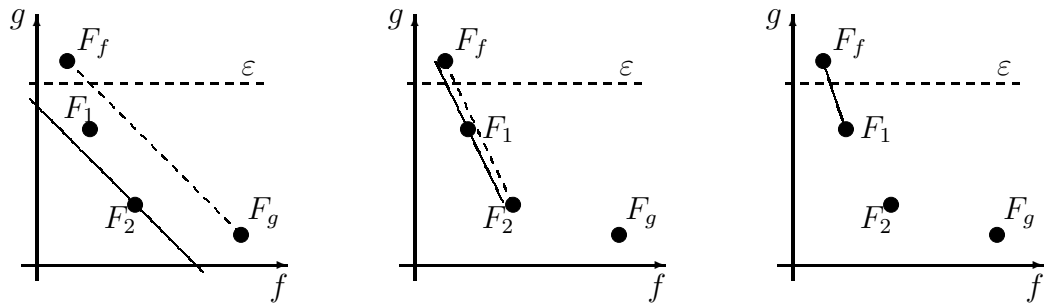


Figure 6.1: *Phase I: Starting from the lexicographically optimal solutions F_f and F_g, a sequence of weighted sum problems is solved to generate representatives (S_f, S_1) of the points F_f and F_1 that encloses the optimal objective value. Note that dominated as well as non-supported non-dominated solutions of the problem are neglected in the figure.*

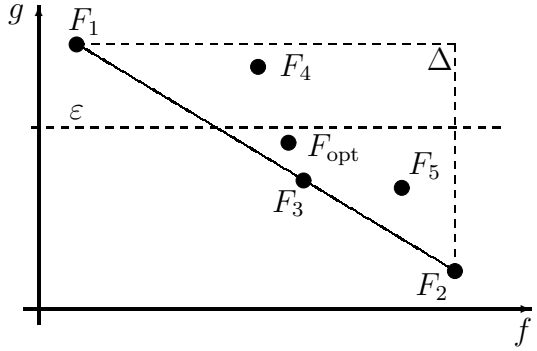


Figure 6.2: *F_4, F_{opt} and F_5 correspond to non-supported non-dominated points within the triangle Δ. While the corresponding representative S_4 is infeasible, S_{opt} is the optimal solution of the problem.*

In the second phase, we apply a local search for non-supported non-dominated solutions that are contained in the triangle $\Delta := \text{conv}(\{F(S^1), F(S^2), (f(S^2), g(S^1))\})$, where $\text{conv}(A)$ denotes the convex hull of a set $A \subseteq \mathbb{R}^2$ (cf. Figure 6.2). Then, an optimal solution of Problem (6.1) is either given by S^2, if all representatives of the non-supported non-dominated solutions contained in Δ correspond to infeasible solutions of Problem (6.1). Otherwise, Equation (3.1) can be used to derive the optimal solution for Problem (6.1). To simplify the notation in the following, we define:

Definition 6.2 *Let S_{opt} denote an optimal solution of Problem* (6.1)*. Furthermore, let $S^1, S^2 \in \mathcal{X}_{sE}$ such that $f(S^1) < f(S_{opt}) \leq f(S^2)$ and $g(S^2) \leq g(S_{opt}) < g(S^1)$ holds true. We say that the ordered tuple $(S^1, S^2) \in \mathcal{X}_{sE}^2$* encloses *the optimal objective value of Problem* (6.1)*, if for all $S \in \mathcal{X}_{sE}$ we have that if S is feasible for Problem* (6.2)*, then $f(S) \geq f(S^2)$, and if S is infeasible for Problem* (6.2)*, then $f(S) \leq f(S^1)$.*

We mainly focus on the calculation of an enclosing tuple $(S^1, S^2) \in \mathcal{X}_{sE}^2$ in the following and propose Algorithm 6.1 for its generation. The algorithm works as follows: We start from the two lexicographically optimal solutions $S^1 = S_f$ and $S^2 = S_g$ of the given problem. Then, we solve a sequence of weighted sum problems $\text{WS}(\lambda)$ to generate new supported efficient solutions of Problem (6.2) that approximate the optimal objective value of Problem (6.1) within $F(\mathcal{X}_{sE})$. In each iteration of the algorithm, the new parameter $\lambda \in (0, 1)$ for Problem $\text{WS}(\lambda)$ is chosen in such a way that the two supported efficient solutions S^1 and S^2 lie on the same level curve with respect to the scalarized objective function of Problem $\text{WS}(\lambda)$ (cf. Line 3 of Algorithm 6.1 and Figure 6.1,

Algorithm 6.1 PHASE *I*

Input: Objective function $F = (f, g)$, right hand side value ε.
Output: A Tuple $(S^1, S^2) \in \mathcal{X}_{sE}^2$ that encloses the optimal solution of Problem (6.1).

```
1: Determine the two lexicographically optimal solutions S^1 and S^2 with respect to
   (f, g) and (g, f), respectively.
2: loop
3:    Set λ = (g(S^1) − g(S^2))/(f(S^2) − f(S^1) + g(S^1) − g(S^2)).
4:    Solve Problem (WS(λ)) ⟶ S^⋆.
5:    if w_λ(S^⋆) = w_λ(S^1) then
5:       break
6:    end if
7:    if g(S^⋆) ≤ ε then
8:       Set S^1 = S^⋆.
9:    else
10:      Set S^2 = S^⋆.
11:   end if
12: end loop
13: Determine among all the alternative solutions of Problem (WS(λ)) the tuple
    (S^1, S^2) that encloses the optimal solution of Problem (6.1).
14: return (S^1, S^2)
```

respectively). In this context, the specific value of λ given by

$$\lambda = \frac{g(S^1) - g(S^2)}{f(S^2) - f(S^1) + g(S^1) - g(S^2)}.$$

Note that for $(x_1, y_1) := (f(S_1), g(S_1))$ and $(x_2, y_2) := (f(S_2), g(S_2))$ we have that

$$\begin{aligned}
\lambda f(S^1) + (1-\lambda) g(S^1) &= \frac{y_1 - y_2}{x_2 - x_1 + y_1 - y_2} \cdot x_1 + \frac{x_2 - x_1}{x_2 - x_1 + y_1 - y_2} \cdot y_1 \\
&= \frac{x_2 \cdot y_1 - x_1 \cdot y_2}{x_2 - x_1 + y_1 - y_2} \\
&= \frac{y_1 - y_2}{x_2 - x_1 + y_1 - y_2} \cdot x_2 + \frac{x_2 - x_1}{x_2 - x_1 + y_1 - y_2} \cdot y_2 \\
&= \lambda f(S^2) + (1-\lambda) g(S^2).
\end{aligned}$$

If the optimal solution $S^\star \in \mathcal{X}_{sE}$ of Problem WS(λ) is infeasible for Problem (6.1), we update the upper bound S^1 of our approximation. Otherwise, the lower bound S^2 is updated. Then, the next iteration starts.
The algorithm finally terminates, when the objective vector $F(S^\star)$ as well as $F(S^1)$ and $F(S^2)$ lie on the same level curve with respect to the objective function of the weighted sum problem solved before. In this case, no further *extreme* efficient solution exists that represents a component of a tuple that encloses the optimal objective value better than the calculated (S^1, S^2) does. Nevertheless, for the last weighted sum problem solved, there might exist alternative optimal solutions that do not correspond to extreme efficient solutions, but that lie on the facet of the non-dominated frontier defined by $F(S^1)$ and $F(S^2)$ (cf., e.g., the non-dominated solution F_3 in Figure 6.2). Hence, all

alternative optima of this weighted sum problem have to be checked whether there exist better alternative optima that enclose the optimal objective value of Problem (6.1) better than the tuple (S^1, S^2) returned by the loop.
We further remark that the case $g(S^\star) = \varepsilon$ is not treated separately in Algorithm 6.1. In this case, $S^\star$ obviously corresponds to an optimal solution of Problem (6.1). We conclude:

Lemma 6.3 *Algorithm 6.1 is correct and determines a tuple* $(S^1, S^2) \in \mathcal{X}_{sE}^2$ *that encloses the optimal solution of Problem* (6.1).

Proof: Let (S^1, S^2) denote the tuple that is calculated during the final cycle of the loop between the Lines 2 and 12 of Algorithm 6.1. According to Line 13, the determination of the best tuple among the optimal solutions of the final weighted sum problem WS(λ) that encloses the optimal objective value is given as a black box algorithm. Hence, it suffices to show that there does not exist another tuple $(\bar{S}^1, \bar{S}^2) \in \mathcal{X}_{sE}^2$, such that at least one of the two solutions $\bar{S}^1$ or $\bar{S}^2$ corresponds to an extreme efficient solution of Problem (6.2) that is not optimal for the final weighted sum problem WS(λ) such that $(\bar{S}^1, \bar{S}^2)$ encloses the optimal objective value better than the tuple (S^1, S^2) does. Assume that this would be the case. Let $S^\star$ denote the optimal solution of the last weighted sum problem WS(λ) that has been solved before the tuple (S^1, S^2) is returned by the loop. By Line 3 of Algorithm 6.1, the solutions S^1 and S^2 have been used to define the parameter λ for this weighted sum problem. But since $(\bar{S}^1, \bar{S}^2)$ encloses the optimal objective value by assumption, Definition 6.2 automatically implies that

$$\min\{w_\lambda(\bar{S}^1), w_\lambda(\bar{S}^2)\} < w_\lambda(S^1) = w_\lambda(S^2).$$

Hence, S^1 or S^2 is updated by $\bar{S}^1$ or $\bar{S}^2$, respectively, and another iteration is performed. But this contradicts the assumption that the tuple (S^1, S^2) is returned at the end of the final cycle of the loop. □

Note that the complexity of Algorithm 6.1 strongly depends on the cardinality of the set $F(\mathcal{X}_{sE})$, as well as the cardinality of the set of alternative optimal solutions for the single objective problems. While the cardinality of the first set may be polynomially bounded for special classes of combinatorial problems (cf., e.g., Ruzika [184] for the biobjective minimum spanning tree problem), the latter is of exponential size in general. Hence, Line 13 of Algorithm 6.1 can be skipped, depending on the method used in the second phase.
A brief outline of our complete algorithm also including the second phase can be found in Algorithm 6.2. We omit a detailed discussion of the potential methods that can be used in the second phase of Algorithm 6.2 to determine the non-supported non-dominated points contained in the triangle Δ (cf. Figure 6.2). We rather refer to the above stated literature, as well as to Section 4.1.2 where potential solution concepts have been presented. In general, the applied method crucially depends on the class of combinatorial problems under consideration. Either a local search technique is applied that exploits special properties of the given combinatorial problem, or special methods are used to enumerate all non-supported non-dominated points contained in the triangle Δ, as described for example in Section 4.1.2.
However, if the connectedness of the efficient set can be proven for the associated multiple objective Problem (6.2) (cf. Chapter 7 for more details), a local search technique

Algorithm 6.2 Algorithm for solving Single Constrained Sum Problems

Input: Objective function $F = (f, g)$, right hand side value ε.
Output: An optimal solution of Problem (6.1).

1: {**PHASE** I}
Determine the tuple $(S^1, S^2) \in \mathcal{X}_{sE}^2$ that encloses the optimal objective value of Problem (6.1) by applying Algorithm 6.1.
2: {**PHASE** II}
Apply a local search technique to determine a complete set $\mathcal{X}^\star$ of efficient solutions that represent the set of non-dominated solutions contained in the triangle Δ defined by $F(S^1)$, $F(S^2)$ and $(f(S^2), g(S^1))$.
3: **if** $g(S) > \varepsilon$ for all $S \in \mathcal{X}^\star$ **then**
4: **return** S^2.
5: **else**
6: **return** $S = \text{argmin}_{S \in \mathcal{X}^\star}\{f(S) : g(S) \leq \varepsilon\}$.
7: **end if**

that exploit the adjacency of efficient solutions seems to be favorable in the second phase of Algorithm 6.2. In this case, the enclosing tuple calculated in the first phase of the algorithm can be used as initial solutions for the local search. Unfortunately, we show in Chapter 7 that the connectedness of the efficient set fails for most of the classical problems from multiple objective combinatorial optimization. We summarize our results in a final theorem.

Theorem 6.4 *Algorithm 6.2 is correct and determines an optimal solution of Problem* (6.1).

Proof: The theorem immediately follows from Lemma 6.3 and Corollary 6.1. □

We restricted ourselves in this section to single constrained combinatorial optimization problems only. Nevertheless, the above described two phase approach can theoretically be applied to higher-dimensional problems (i.e. to combinatorial optimization problems with several side constraints), too. However, the algorithms for the first as well as for the second phase are no longer that easy to handle as in the biobjective case. Considering the first phase of Algorithm 6.2, it is not clear in advance, which facet of the approximated non-dominated frontier has to be chosen for the next iteration of Algorithm 6.1. It may happen that although all representatives of the supported non-dominated points that define a facet of the final non-dominated frontier correspond to infeasible solutions of the constrained problem, while the facet itself contains points whose representatives correspond to feasible solutions of the overall problem (cf. also Klamroth and Tind [114] for an example for the continuous case). Hence, unless no further information on the optimal solutions of Problem (6.2) is given, the approximation in the first phase must be refined for all facets of the approximated non-dominated frontier calculated so far.
Furthermore, it is also not obvious how the second phase of the algorithm can be handled efficiently. In general, there exist more than one facet of the non-dominated frontier defining a simplex that has to be scanned for potential non-supported non-dominated solutions that may correspond to optimal solutions of the constrained problem. Hence, in the worst case all facets of the non-dominated frontier have to be

checked for potential optimal solutions.
We finally remark that although Algorithm 6.2 is clearly designed for combinatorial problems where the single objective problem can be solved efficiently, it also can be applied to any other combinatorial problem that is already $\mathcal{NP}$-hard for the unconstrained case.

6.2 Combinatorial Problems with Algebraic Sum Objectives

In this section we consider single objective combinatorial problems where the objective function is given as the sum of different types of objectives defined on a common feasible set $\mathcal{X}$. Note that such a sum is also called *algebraic* in the literature (cf., e.g., Minoux [144]). We discuss two different types of objectives in the following: Given the algebraic sum of p (different) functions, we consider the case that either these p functions are defined on p pairwise different cost functions on the ground set, or that only a single cost function for all p objectives is given. Since the first case is already almost completely covered by the results from Chapter 4 and Chapter 5, we mainly focus on the latter in the following.
Based on problems from the literature, we show how these special types of problems can be modeled and solved using the associated multiple objective reformulation of the problem (cf. Section 3.2). Amongst others, we discuss existing solution approaches for problems that aim to minimize the deviation between the cost coefficients of a feasible solution. We show that most solution approaches that can be found in the literature are closely related to the algorithms presented in Chapter 4 of this work. Based on this observation, we present generalized versions of these problems and show that they can be solved based on algorithms developed in Chapter 5. Note that we do not focus on any special class of combinatorial problems in the following. We rather show how the general ideas and algorithms of the multiple objective problems discussed in the previous chapters can be used to solve combinatorial optimization problems with an algebraic sum objective.
We start with a formal introduction of the considered problems in this section. Let $\mathcal{E}$ denote a ground set of n distinct elements, and let $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$ be a subset of the powerset of $\mathcal{E}$. Furthermore, let $f_1, \ldots, f_p : \mathcal{X} \to \mathbb{Z}$ denote $p \geq 2$ different objective functions of arbitrary type. Then, a *combinatorial optimization problem with algebraic sum objective* is formally given by

$$\begin{array}{ll} \min & \sum_{i=1}^{p} f_i(S) \\ \text{s.t.} & S \in \mathcal{X}. \end{array} \tag{6.3}$$

Considering the p different summands of Problem (6.3) as independent objectives, this problem can be seen as a scalarized version of the associated multiple objective combinatorial problem

$$\begin{array}{ll} \min & F^p(S) = (f_1(S), \ldots, f_p(S))^\top \\ \text{s.t.} & S \in \mathcal{X}. \end{array} \tag{6.4}$$

From Theorem 3.2 in Section 3.2 we recall:

Corollary 6.5 *It holds:*

1. *Every optimal solution of Problem* (6.3) *is also efficient for Problem* (6.4).
2. *There exists an efficient solution of Problem* (6.4) *that is also optimal for Problem* (6.3).

Especially the second part of Corollary 6.5 is important for the problems we consider in this section: Given the efficient set of Problem (6.4), we can derive an optimal solution for Problem (6.3). According to the results from Section 3.2, this optimal solution $S^\star \in \mathcal{X}_E$ is given by

$$S^\star = \operatorname*{argmin}_{S \in \mathcal{X}_E} \{f_1(S) + \ldots + f_p(S)\} \tag{6.5}$$

and corresponds to a supported efficient solution of Problem (6.4). However, applying Equation (6.5) to derive the optimal solution of Problem (6.3) is not very useful in practice. A complete set of efficient solutions has to be generated and filtered to derive the solution $S^\star$ we are only interested in. Hence, the application of an algorithm that originally was designed for solving a general multiple objective combinatorial problem normally generates much redundant information that cannot further be used to derive an optimal solution for Problem (6.3). However, we will see in the following that many approaches in the literature implicitly make use of the above described multiple objective reinterpretation of the problem. This is especially the case, when the involved subproblems that result from this reinterpretation are solvable in an efficient manner. To further reduce the storage of redundant information, the following modified version of an algorithm that solves Problem (6.4) is commonly used in the literature: Let an algorithm that determines a complete set $\mathcal{X}'$ of efficient solutions for the associated multiple objective problem be given. Since only a single efficient solution among the sets contained in $\mathcal{X}'$ is also optimal for Problem (6.3), only the best solution with respect to the given objective function of Problem (6.3) is stored during the course of the algorithm. If a new solution candidate is generated by an appropriate subroutine of the algorithm, its objective value is compared to the value of the best solution found so far. If the generated solution improves the objective value of Problem (6.3), it replaces the former best solution. Otherwise, the solution is discarded.
If it is ensured that the given algorithm for the associated multiple objective problem generates a complete set of efficient solutions, it obviously suffices to return the best solution as well as its corresponding objective value at the end of the algorithm. Hence, we assume in the following:

Remark 6.6 *Whenever we refer to an algorithm from Chapter 4 and Chapter 5 in the following subsections, it is assumed that only the best solution with respect to the objective function of Problem* (6.3) *found so far is stored at the end of an iteration. After the final iteration, only this stored solution as well as its corresponding objective value is returned by the algorithm.*

In the following subsections, we discuss complexity results of solution approaches that are based on solving a sequence of single objective sum, bottleneck and k-max problems, respectively. To simplify the notation, let $T^{\mathcal{S}}$, $T^{\mathcal{B}}$ and $T^{\mathcal{K}}$ denote the time to

solve an unconstrained sum, bottleneck and k-max optimization problem, respectively. If an additional subscript is added, this subscript represent the number of *binary* constraints that additionally have to be considered when the specific problem is solved. We will see in the following that these additional side constraints result from k-max objectives that are used to model specific types of combinatorial optimization problems. For example, $\mathcal{O}(T_k^{\mathcal{S}})$ denotes the complexity of solving a single objective sum problem with k binary side constraints, where $k \geq 1$.

6.2.1 Algebraic Sums based on Different Cost Functions

In this subsection we focus on the case that the p different summands of the algebraic sum objective are given based on p different cost functions $c_i : \mathcal{E} \to \mathbb{Z}$ $(i = 1, \ldots, p)$ on the ground set $\mathcal{E}$.

We start with a brief review of the existing literature on this special type of combinatorial optimization problem. Note that most articles focus on the algebraic sum of a sum and a bottleneck objective. Hence, we mainly summarize the results of this specific problem in the following that was introduced by Minoux [144] in 1989. In this specific article, the author showed that based on a modified version of the biobjective threshold algorithm (cf. Algorithm 4.2) presented in Section 4.2.2 it is possible to derive an optimal solution for the algebraic sum problem within $\mathcal{O}(nT^{\mathcal{S}})$ amount of time. Note that this bound coincides with the bound stated in Theorem 4.3 for the general biobjective threshold algorithm.

To reduce the number of subproblems that have to be solved in the course of Minoux's algorithm, Punnen [170] proposed an improved version of Algorithm 4.2 that decreases the average number of involved subproblems. Although the general performance of this improved version is reported to be much better, the proposed algorithm still has a worst case time complexity of $\mathcal{O}(nT^{\mathcal{S}})$.

Finally, Punnen and Nair [176] presented an approach that minimizes the algebraic sum of a sum and a bottleneck objective for the minimum spanning tree problem. Based on a specially designed data structure for the spanning trees of a graph, the authors were able to solve this specific algebraic sum problem within $\mathcal{O}(m \log(n))$ time, where n denotes the number of nodes and m the number of edges of the given graph.

In addition to the combinatorial versions of the problem, Punnen [170] also discussed the LP-version of the algebraic sum problem under consideration. Different from many other solution approaches that use n additional inequality constraints to handle the bottleneck objective, a parametric approach is presented. Based on an LP-reformulation of the problem, the n additional constraints on the bottleneck objective are treated implicitly and not as additional constraints during the solution process of the involved LPs.

Given an objective function f_i, $i \in \{1, \ldots, p\}$, it is assumed in the remainder of this subsection that f_i either corresponds to a sum, a bottleneck or a k-max objective (cf. Chapters 4 and 5). Furthermore, given an instance of Problem (6.3), we briefly show that the involved sum objectives can be recombined into a single objective of the same type.

Let $c_1, \ldots, c_q$ denote q different cost functions on $\mathcal{E}$, where $2 \leq q \leq p$. Furthermore, let $f_1, \ldots, f_q$ correspond to q different sum objectives, i.e. $f_i(S) = \sum_{e \in S} c_i(e)$ for all

$i \in \{1, \ldots, q\}$. Then, the partial sum of these q objectives can be simplified to

$$\sum_{i=1}^{q} f_i(S) \;=\; \sum_{i=1}^{q} \left(\sum_{e \in S} c_i(e) \right) \;=\; \sum_{e \in S} \left(\sum_{i=1}^{q} c_i(e) \right) \;=\; \sum_{e \in S} \tilde{c}(e),$$

where $\tilde{c}(e) = \sum_{i=1}^{q} c_i(e)$ for fixed $e \in \mathcal{E}$. Hence, the q sum objectives can be combined to a single objective of the same type without affecting the overall objective value. Unfortunately, this is no longer the case, when q different bottleneck or k-max objectives are considered. In general, only

$$\sum_{i=1}^{q} k\text{-}\max_{e \in S} \{c_i(e)\} \;\leq\; k\text{-}\max_{e \in S} \left\{ \sum_{i=1}^{q} c_i(e) \right\}$$

holds true for these objectives. Hence, neither the bottleneck nor the k-max objectives can be recombined to a single objective for the overall problem.
As the number of sum objectives can be reduced to one, a given instance of Problem (6.3) either simplifies to an optimization problem with a single sum objective, or the associated multiple objective problem of the form (6.5) is equivalent to Problem (MCBP) and Problem (MkMAX) from Section 4.2.2 and Section 5.2, respectively. This implies that the algorithms stated in these sections can be used to solve any combinatorial problem where the corresponding objective function is given as an algebraic sum of a sum and several bottleneck, k-max and even k-min objectives. Hence, we just summarize the results obtained for the different combinations of these objectives from the last chapters and refer to the above stated sections for further information on solution approaches and algorithms.

Corollary 6.7 *Consider an instance of Problem* (6.3) *consisting of p bottleneck and an additional sum-objective f. Then, Algorithm 4.2 can be applied to solve the problem. In this case, the running time of this approach is in $O(n^p\, T^{\mathcal{S}})$.*

In particular, Corollary 6.7 implies that the algorithms proposed by Minoux [144] and Punnen [170] (cf. the literature review given above) for minimizing the algebraic sum of a sum and a bottleneck objective can both be interpreted as specially designed versions of the more general Algorithm 4.2.

Corollary 6.8 *Consider an instance of Problem* (6.3) *consisting of p k-max and an additional sum-objective f. Then, Algorithm 5.2 can be used to solve this problem within $O(n^p\, T_p^{\mathcal{K}})$ of time.*

Note that similar results hold for the case that the summands of the algebraic sum objective consist of k-min or combined k-max and k-min objectives (cf. Section 5.3).

6.2.2 Algebraic Sums based on a Single Cost Function

In this subsection we consider optimization problems where all functions that represent summands of the given algebraic sum objective are defined based on a single cost function $c : \mathcal{E} \to \mathbb{Z}$. Starting from combinatorial problems already treated in the literature, we present generalized versions of these problems, and we show that all of these problems can be modeled by means of the three types of objectives discussed in

Chapter 4 and Chapter 5: the sum, the bottleneck and the k-max/k-min objective. Moreover, we discuss how multiple objective approaches can be used to solve these types of problems.
We further remark that several problems, solution approaches and results stated in this subsection were independently developed by Turner [203] and can be found in Turner [204].
The following discussions are all based on the ideas and the results developed at the beginning of this section and Section 3.2, respectively. Given a combinatorial optimization problem with an algebraic sum objective, we use the associated multiple objective optimization problem to derive an optimal solution for the single objective problem based on the results of Corollary 6.5. Taking into account that a single objective problem has to be solved, modified versions of the algorithmic schemes developed for the multiple objective bottleneck (cf. Section 4.2.2), the k-max (cf. Section 5.2) and the combined k-max/k-min problem (cf. Section 5.3) can be use to construct appropriate algorithms for all the problems treated in the following (cf. Remark 6.6). We further show that most algorithms proposed for the considered problems in the literature correspond to carefully designed variants of the algorithms described in the previous two chapters.
Note that it is not in the main focus of this section to derive new types of algorithms that solve special types of problems already mentioned in the literature. We rather want to show how these problems can be seen and modeled in an alternative multiple objective framework which may lead to a deeper insight into the general structure of such problems.
The remainder of this section is organized as follows: Each type of optimization problem is presented in an individual subsection. First, we introduce the problem and give a short review of the existing literature. Then, we describe how the given problem can be solved based on the associated multiple objective problem formulation. This is mainly done by referring to the specific algorithms developed in Chapter 4 and Chapter 5. Furthermore, we relate our suggested approaches to the approaches already presented in the literature. At the end of each subsection, we present solution approaches for generalized versions of the combinatorial problem that has been treated in the specific subsection. We finally summarize our results in a last paragraph.
For all the problems stated in the following, we assume that each feasible solution satisfies a minimal cardinality constraint, i.e. there exists an integer $m \geq 2$ such that $|S| \geq m$ holds true for all $S \in \mathcal{X}$. If a fixed cardinality for all feasible solutions is needed, it is mentioned in the specific subsection. For a survey on combinatorial problems with fixed cardinality, we refer to Ehrgott et al. [57].

Balanced Combinatorial Optimization Problems

The class of *balanced combinatorial optimization problems* is a special type of an algebraic sum of two bottleneck objectives. Given a ground set $\mathcal{E}$ of cardinality n, a set of feasible solutions $\mathcal{X}$ and a scalar cost function $c : \mathcal{E} \to \mathcal{X}$, the aim of the problem is to minimize the range of the values contained in a feasible solution. Formally, the problem is given by

$$\min_{S \in \mathcal{X}} \left[\max_{e \in S} \{c(e)\} - \min_{e \in S} \{c(e)\} \right]. \tag{6.6}$$

Balanced combinatorial optimization problems were introduced by Martello et al. [127]. Given a subroutine that solves a simple feasibility problem, the overall problem is solved in this reference by iteratively scanning the range of the cost values contained in a predefined interval. If the distinct values of c are sorted in increasing order, the initial interval only contains the minimum cost coefficient. During the course of the algorithm, the bounds on the range of the coefficients contained in the interval that is scanned for feasible solutions is consecutively modified. If a feasible solution is found, the lower bound on the range is increased by one unit, which results in a reduced search interval in the next iteration. Otherwise, the upper bound is increased by one unit which leads to an enlarged search range for the next iteration. The best solution is recorded and the algorithm stops either when the complete range of the cost values is scanned, or a solution with optimal cost of zero is found. The authors state a time bound of $\mathcal{O}(n\,T)$, where T denotes the time to solve the feasibility problem.

Duin and Volgenant [50] presented a similar approach, based on solving a sequence of bottleneck optimization problems instead of numerous feasibility problems. At the beginning of the algorithm, the ordinary bottleneck problem is solved. Then, the minimum cost coefficient of the optimal solution is determined and used as a threshold value. All cost coefficients smaller than the given threshold value are set to infinity and the problem is resolved to derive the next larger threshold value. This procedure is repeated until no further finite solution exists. In addition to this algorithm, the authors presented an approach that solves the problem in a reverse order: Starting from a maximal threshold value with respect to the involved cost coefficients, all coefficients smaller than the threshold value are set to infinity and the bottleneck problem is solved for the modified costs. If the optimal value of this problem is finite, the resulting optimal solution is treated as a potential solution candidate for the overall problem. In each step of the algorithm the threshold value is relaxed to the next smaller value with respect to the original costs, and its original value is restored. The procedure stops, when all cost coefficients are updated with their original values.

We further note that a modified version of the algorithm presented in Martello et al. [127] was used in the book of Burkard et al. [29] to solve the balanced linear assignment problem.

In addition to the standard problem, Punnen and Nair [177] discussed balanced optimization problems with an additional sum constraint. The authors developed an solution approach that is mainly based on the algorithm presented in [127]. Given the current search interval, a sum problem is solved instead of a feasibility problem. If the considered problem is feasible, the resulting optimal solution is discarded if it does not satisfy the given side constraint. Otherwise it is accepted as potential optimal solution of the overall problem. Besides the constrained problem, also the biobjective case as well as a scalarization of both objectives is discussed. Based on the algorithm described above, a complexity of $\mathcal{O}(n^2T)$ is reported for both problems, where T denotes the time to solve the sum problem restricted to the current search interval for the cost coefficients.

Finally, the lexicographic version of Problem (6.6) is considered in Punnen and Aneja [174]. The authors transform the lexicographic problem into n independent lexicographic bottleneck problems that have to be solved to find the optimal solution of the overall problem.

We further remark that a continuous version of Problem (6.6) was discussed in Ahuja

[3]. The author presented a solution approach for the balanced linear programming problem that uses the relationship between the given balanced optimization problem and a parametric bottleneck linear programming problem. Different from other solution approaches that transform the given optimization problem into a standard LP by means of additional variables and side constraints, an optimal solution for the balanced optimization problem is derived by solving the associated bottleneck linear programming problem parametrically.

In the following, we solve Problem (6.6) based on the multiple objective approach presented at the beginning of Section 6.2. We will see that this approach can be considered as a generalization of the afore mentioned approaches from the literature.

Let an instance of Problem (6.6) be given. Since $-\min\{c(e) : e \in S\} = \max\{-c(e) : e \in S\}$ holds true for all $S \in \mathcal{X}$, the associated biobjective combinatorial problem is given by

$$\min_{S\in\mathcal{X}} \left(\max_{e\in S}\{c(e)\}, \max_{e\in S}\{-c(e)\} \right)^{\top}.$$

According to Section 4.2.2, this problem is a biobjective bottleneck problem, and Algorithm 4.2 can be used to find an optimal solution for this problem within $O(n\,T^{\mathcal{B}})$ time (cf. Theorem 4.3). If a fixed cardinality for all feasible solutions is ensured, i.e. $|S| = m$ for all $S \in \mathcal{X}$, we further deduce from Lemma 5.11 that

$$\max_{e\in S}\{c(e)\} - \min_{e\in S}\{c(e)\} \;=\; \max_{e\in S}\{c(e)\} + m\text{-}\max_{e\in S}\{-c(e)\}$$

also holds true. Hence, in this particular case, also Algorithm 5.2 can be used to solve Problem (6.6).

Comparing Algorithm 4.2 with the algorithms stated in Martello et al. [127] and Duin and Volgenant [50], we conclude that both algorithms from the literature are special versions of the more general solution approaches presented in Section 4.2. From this section we recall that a complete set of efficient solutions of a biobjective bottleneck problem can be generated by solving a sequence of ε-constraint problems. Given two bottleneck objectives this approach implies that each of the involved ε-constraint problems can be reduced to a single objective unconstrained bottleneck problem based on an appropriate modification of the involved cost coefficients. While the algorithm stated in [50] directly exploits this solution approach, the algorithm presented in [127] rather uses the idea that an optimal solution of a bottleneck problem can alternatively be calculated by solving a sequence of feasibility problems (cf. also Section 4.2.1). Hence, especially the algorithm of Duin and Volgenant [50] can be seen as an adapted version of Algorithm 4.2 that takes advantage of the special structure of the given combinatorial optimization problem.

We finally discuss a natural generalization of Problem (6.6) and its solution independently suggested by Turner [203], where the two bottleneck objectives are replaced by the corresponding k-max and k-min objectives, not yet treated in the literature. We assume that $|S| \geq m$ for all $S \in \mathcal{X}$. Furthermore, let $k, l \in \{1, \ldots, m\}$, such that $k\text{-max}(S) \geq l\text{-min}(S)$ holds true for all $S \in \mathcal{X}$. Then, we define the *(k,l)-balanced optimization problem* by

$$\min_{S\in\mathcal{X}} \left[k\text{-}\max_{e\in S}\{c(e)\} - l\text{-}\min_{e\in S}\{c(e)\} \right]. \tag{6.7}$$

Obviously, Problem (6.7) generalizes the balanced optimization problem that results from Problem (6.7) when $k = l = 1$ is chosen. As a special case of Problem (6.7), the $(k, 1)$-balanced spanning tree problem was already discussed Duin and Volgenant [50]. Since $-\,l\text{-min}\{c(e) : e \in S\} = l\text{-max}\{-c(e) : e \in S\}$ holds true, Problem (6.7) is a scalarized version of the associated biobjective k-max optimization problem

$$\min_{S \in \mathcal{X}} \left(k\text{-}\max_{e \in S}\{c(e)\}, l\text{-}\max_{e \in S}\{-c(e)\} \right)^{\top}.$$

Hence, Algorithm 5.2 can be used to derive an optimal solution for Problem (6.7). In this case, one of the two k-max objectives has to be chosen as additional objective f. Since additional side constraints do not influence the validity of Algorithm 5.1 for the single objective k-max problem, applying the solution approach stated in Section 5.2 results in solving a sequence of binary sum problems with an additional binary constraint in each iteration of the algorithm. We summarize our results in the following theorem.

Theorem 6.9 *Let a (k, l)-balanced optimization problem be given. Algorithm 5.2 can be used to derive an optimal solution for this problem within $\mathcal{O}(n\, T_1^{\mathcal{K}})$ of time.*

Note that besides Algorithm 5.2, also Algorithm 4.2 can be used to solve Problem (6.7), for the special case that $k = 1$. This further holds true, whenever $l = 1$ and a fixed cardinality for all feasible solutions is ensured for problem instance under consideration.

Minimum Deviation Problems

Let a ground set $\mathcal{E}$ of n different elements, a feasible set $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$ and a scalar cost function $c : \mathcal{E} \to \mathbb{Z}$ be given. We assume that all sets $S \in \mathcal{X}$ satisfy a prescribed fixed cardinality constraint, i.e. $|S| = m$ holds true for all $S \in \mathcal{X}$. Then, the *minimum deviation problem* is given by

$$\min_{S \in \mathcal{X}} \sum_{e \in S} \left[\max_{e \in S}\{c(e)\} - c(e) \right] = |S| \cdot \max_{e \in S}\{c(e)\} - \sum_{e \in S} c(e). \tag{6.8}$$

Solving Problem (6.8) means to find a feasible solution that minimizes the sum of deviations of its maximum cost component to all other components of the feasible solution. Note that Problem (6.8) is closely related to balanced optimization problems. This can be verified, when the sum of deviations is replaced by the maximum deviation in the description of Problem (6.8). In this case we have that

$$\max_{e \in S} \left\{ \max_{e \in S}\{c(e)\} - c(e) \right\} = \max_{e \in S}\{c(e)\} - \min_{e \in S}\{c(e)\}.$$

Thus, for balanced optimization problems only the absolute deviation of the maximum cost component from the minimum cost component of a feasible solution is relevant, whereas for minimum deviation problems the complete sum of deviations has to be optimized.
Problem (6.8) was firstly treated by Gupta and Punnen [89] in the literature. The authors presented an threshold approach that solves the given problem within $\mathcal{O}(n\, T^{\mathcal{S}})$,

where the related sum problem has to be solved for negative costs. Starting from a feasible solution S that minimizes the sum of the negative costs of the given instance, a first threshold value, defined as the maximum cost component of S, is calculated. Then, all cost coefficients equal or larger than the given threshold value are set to $-\infty$. In the following, the sum problem is resolved for the modified costs to find a new solution candidate. Based on this candidate a new threshold value is derived, the cost coefficients are updated and the sum problem is resolved once more. This procedure is repeated until the optimal objective value of the sum problem equals $-\infty$. When this is the case, the procedure stops and the best solution with respect to the objective function of Problem (6.8) is returned by the algorithm. In addition to Problem (6.8), also minimum deviation linear programs were considered by the authors.

Another version of the above described solution approach was given by Duin and Volgenant [50]. While the algorithm described in [89] solves the problem by setting more and more cost coefficients equal to infinity, Duin and Volgenant solve the problem in the reverse order: Given a fixed threshold value, all coefficients *smaller* than this value are set to infinity and the minimizer of the sum problem with negative costs is calculated. In the next iteration, the threshold value is set to the next smaller cost value and the original values of all coefficients equal to or larger than the new threshold value are restored. Then, the sum problem is resolved. The procedure stops when the smallest cost coefficient is reached, and the optimal solution of the minimum deviation problem is returned by the algorithm. Note that although this procedure implies that the sum problem has to be solved for all different values of the cost coefficients involved, the authors reported an improved complexity for the minimum spanning tree problem as compared to the algorithm stated in [89]. Since only a small reoptimization of the last optimal spanning tree has to be performed to derive the optimal tree for the current iteration, the reverse algorithm of Duin and Volgenant solves the problem within $\mathcal{O}(mn)$ time, while the algorithm stated in [89] takes $\mathcal{O}(mn^2)$ in general, where n denotes the number of nodes and m the number of edges of the given graph.

In the following we concentrate on the multiple objective interpretation of Problem (6.8). By rearranging the summands of the given sum of deviations from the maximum cost coefficient, we conclude that the given problem is equivalent to

$$\min_{S\in\mathcal{X}} \left[|S| \cdot \max_{e\in S}\{c(e)\} + \sum_{e\in S}\{-c(e)\} \right].$$

Hence, an optimal solution of Problem (6.8) must be contained in the efficient set of the associated biobjective combinatorial problem

$$\min_{S\in\mathcal{X}} \left(\max_{e\in S}\{c(e)\}, \sum_{e\in S}\{-c(e)\} \right)^{\top}. \tag{6.9}$$

Due to the results from Section 4.2.2, we conclude that Algorithm 4.2 can be used to solve the minimum deviation problem within $O(nT^{\mathcal{S}})$ time, where $T^{\mathcal{S}}$ denotes the time to find the minimizer of the sum problem with negative weights (cf. Theorem 4.3). Contrary to balanced optimization problems, the two given objectives of

Problem (6.9) have to be multiplied by different factors to derive an optimal solution for Problem (6.8). While the value of the sum objective remains unchanged, the bottleneck objective has to be multiplied by a factor $|S|$ (cf. also the general problem formulation in Section 4.2.2).
Furthermore, comparing Algorithm 4.2 with the algorithms of Gupta and Punnen [89] and Duin and Volgenant [50], we see that also these two algorithms can be interpreted as special versions of the approach presented in Section 4.2.2.
We finally remark that the fixed cardinality constraint is essential for the generation of the optimal solution for Problem (6.8) using the biobjective approach. When the fixed cardinality constraint is dropped, it may happen that dominated solutions of Problem (6.9) exist that are optimal for Problem (6.8), as the following example shows.

Example 6.10 Let a minimum deviation problem be given, where $\mathcal{E} = \{e_1, \ldots, e_5\}$ with $c(e_1) = 3$, $c(e_2) = 2$ and $c(e_3) = c(e_4) = c(e_5) = 1$. Furthermore, let the only two feasible solutions be defined by $S_1 = \{e_2, e_3, e_4, e_5\}$ and $S_2 := \{e_1, e_5\}$. Then, S_1 dominates S_2 since

$$\max_{e\in S_1}\{c(e)\} = 2 < 3 = \max_{e\in S_2}\{c(e)\} \text{ and } \sum_{e\in S_1}\{-c(e)\} = -5 < -4 = \sum_{e\in S_2}\{-c(e)\}.$$

However, S_2 is optimal for Problem (6.8), since

$$|S_2|\cdot\max_{e\in S_2}\{c(e)\} - \sum_{e\in S_2} c(e) \;=\; 2\cdot 3 - 4 \;=\; 2 \;<\; 3 \;=\; 4\cdot 2 - 5 \;=\; |S_1|\cdot\max_{e\in S_1}\{c(e)\} - \sum_{e\in S_1} c(e).$$

Nevertheless, a modified version of Algorithm 4.2 can be used to solve the minimum deviation problem for arbitrary feasible sets $\mathcal{X}$, as long as it is ensured that the constrained problem is solved for *all* possible right hand side values contained in $c(\mathcal{E})$. In addition, it has to be guaranteed that the optimal solution of the sum problem that is solved in each iteration of the algorithm, is of minimal cardinality with respect to the alternative optima of this problem.
Besides to the sum of deviations from the maximum cost component of a feasible solution, also the sum of deviations from the minimal component can be of interest. The corresponding optimization problem is formally given by

$$\min_{S\in\mathcal{X}} \sum_{e\in S} \left[c(e) - \min_{e\in S}\{c(e)\}\right] \;=\; \sum_{e\in S} c(e) - |S|\cdot\min_{e\in S}\{c(e)\}. \tag{6.10}$$

Since

$$\sum_{e\in S} c(e) - |S|\cdot\min_{e\in S}\{c(e)\} \;=\; \sum_{e\in S} c(e) + |S|\cdot\max_{e\in S}\{-c(e)\},$$

a biobjective approach, similar to the one stated above can be used to solve the problem. In this case, Algorithm 4.2 has to be applied to the associated biobjective bottleneck problem

$$\min_{S\in\mathcal{X}} \left(\max_{e\in S}\{-c(e)\}, \sum_{e\in S}\{c(e)\}\right)^{\top}.$$

We finally discuss a generalized version of the deviation problem, where the sum of deviations to k^{th} largest cost coefficient has to be minimized. We call this problem

k-deviation problem which is formally given by

$$\min_{S\in\mathcal{X}} \sum_{e\in S} |\, k\text{-}\max_{e\in S}\{c(e)\} - c(e)|. \tag{6.11}$$

Note that the bottleneck-version of this problem was already treated in Punnen and Aneja [174]. The authors solved the problem based on a simple cost modification scheme. A time bound of $\mathcal{O}(n\,T^{\mathcal{B}})$ is reported for their algorithm, where $T^{\mathcal{B}}$ denotes the time to solve the bottleneck problem using the modified cost coefficients.
In contrast to Punnen and Aneja [174], we use additional k-max objectives to reformulate Problem (6.11), and distinguish the two cases that $2 \le k \le \lfloor \frac{m}{2} \rfloor$ and $\lfloor \frac{m}{2} \rfloor < k \le m-1$, where $\lfloor x \rfloor = \max\{n \in \mathbb{N} : n \le x\}$. Let $2 \le k \le \lfloor \frac{m}{2} \rfloor$ first. We have that:

$$\begin{aligned}
&\sum_{e\in S} |\, k\text{-}\max_{e\in S}\{c(e)\} - c(e)| = \\
&\quad = \sum_{t=1}^{k-1} \left(t\text{-}\max_{e\in S}\{c(e)\} - k\text{-}\max_{e\in S}\{c(e)\} \right) + \sum_{t=k}^{m} \left(k\text{-}\max_{e\in S}\{c(e)\} - t\text{-}\max_{e\in S}\{c(e)\} \right) \\
&\quad = \sum_{t=1}^{k-1} t\text{-}\max_{e\in S}\{c(e)\} - \sum_{t=k}^{m} t\text{-}\max_{e\in S}\{c(e)\} + (m-2k+2)\cdot k\text{-}\max_{e\in S}\{c(e)\} \\
&\quad = \sum_{e\in S}\{-c(e)\} + 2\cdot \sum_{t=1}^{k-1} t\text{-}\max_{e\in S}\{c(e)\} + (m-2k+2)\cdot k\text{-}\max_{e\in S}\{c(e)\}.
\end{aligned}$$

Hence, Problem (6.11) can be seen as a scalarized version of the associated $(k+1)$-objective optimization problem

$$\min_{S\in\mathcal{X}} \left(\sum_{e\in S}\{-c(e)\}, \max_{e\in S}\{c(e)\}, 2\text{-}\max_{e\in S}\{c(e)\}, \ldots, k\text{-}\max_{e\in S}\{c(e)\} \right)^{\top}.$$

According to the results from Section 5.2, Algorithm 5.2 can be used to derive an optimal solution for this problem within $\mathcal{O}(n^k\, T_k^{\mathcal{S}})$, where the sum problem has to be solved for negative costs coefficients. As in the case of the ordinary deviation problem, the value of the sum objective remains unchanged, while the k-max objectives have to be multiplied by a factor of 2 and $m-2k+2$, respectively, to determine an optimal solution for Problem (6.11).
For the second case, let $\lfloor \frac{m}{2} \rfloor < k \le m-1$. We use Lemma 5.11 to deduce that

$$\begin{aligned}
&\sum_{e\in S} |\, k\text{-}\max_{e\in S}\{c(e)\} - c(e)| = \\
&\quad = \sum_{t=1}^{k} \left(t\text{-}\max_{e\in S}\{c(e)\} - k\text{-}\max_{e\in S}\{c(e)\} \right) + \sum_{t=k+1}^{m} \left(k\text{-}\max_{e\in S}\{c(e)\} - t\text{-}\max_{e\in S}\{c(e)\} \right) \\
&\quad = \sum_{t=1}^{k} t\text{-}\max_{e\in S}\{c(e)\} - \sum_{t=k+1}^{m} t\text{-}\max_{e\in S}\{c(e)\} + (2k-m)\cdot k^{\star}\text{-}\max_{e\in S}\{-c(e)\} \\
&\quad = \sum_{e\in S}\{c(e)\} + 2\cdot \sum_{t=1}^{m-k} t\text{-}\max_{e\in S}\{-c(e)\} + (2k-m)\cdot k^{\star}\text{-}\max_{e\in S}\{-c(e)\},
\end{aligned}$$

where $k^\star = m - k + 1$. This is a scalarized version of the associated $(k^\star + 1)$-objective optimization problem

$$\min_{S \in \mathcal{X}} \left(\sum_{e \in S} \{c(e)\}, \max_{e \in S} \{-c(e)\}, 2\text{-}\max_{e \in S} \{-c(e)\}, \ldots, k^\star\text{-}\max_{e \in S} \{-c(e)\} \right)^\top,$$

where the optimal solution of this problem can be calculated by applying Algorithm 5.2 within $\mathcal{O}(n^{k^\star} T^{\mathcal{S}}_{k^\star})$. In this case, the k-max objectives now have to be multiplied by a factor of 2 and $2k - m$, respectively. We summarize our results in the following theorem.

Theorem 6.11 *Let a k-deviation problem be given and let $k^\star = m - k + 1$. Then, this problem can be solved by applying Algorithm 5.2 within $\mathcal{O}(n^k T^{\mathcal{S}}_k)$, if $2 \leq k \leq \lfloor \frac{m}{2} \rfloor$, and within $\mathcal{O}(n^{k^\star} T^{\mathcal{S}}_{k^\star})$, if $\lfloor \frac{m}{2} \rfloor < k \leq m - 1$, respectively.*

k-sum Optimization Problems

Let a ground set $\mathcal{E}$ of n different elements, a feasible set $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$ and a scalar cost function $c : \mathcal{E} \to \mathbb{Z}$ be given. For each $S = \{e_{t_1}, \ldots, e_{t_{|S|}}\} \in \mathcal{X}$ it is assumed that $c(e_{t_1}) \geq \ldots \geq c(e_{t_{|S|}})$ holds true. Furthermore, let $k \in \{1, \ldots, n\}$ arbitrary but fixed and set $p = \min\{|S|, k\}$. Then, the k-sum optimization problem is to minimize the sum of the p largest cost coefficients contained in a feasible solution and is formally given by

$$\min_{S \in \mathcal{X}} \sum_{i=1}^{p} c(e_{t_i}). \tag{6.12}$$

Note that Problem (6.12) simultaneously generalizes the sum as well the bottleneck optimization problem. While for $k = n$ an ordinary sum problem has to be solved, a bottleneck problem is obtained for the choice $k = 1$.
Gupta and Punnen [90] showed that Problem (6.12) can be solved by a simple cost modification scheme for the case that all feasible solutions have the same cardinality. Given the $d \leq n$ different cost coefficients contained in $c(\mathcal{E})$ in ascending order, the cost coefficients are replaced by $\tilde{c}(e) = \max\{c(e_i), c(e)\}$ in the i^{th} iteration of the algorithm, where $i \in \{1, \ldots, d\}$. Then, the ordinary sum problem is solved for the modified costs and the optimal solution is stored. After d iterations the best of these stored solutions with respect to the given objective function is returned by the algorithm, since it can be proven that this solution represents an optimal solution of Problem (6.12). The algorithm takes $\mathcal{O}(nT + n^2)$ time to solve the problem, where T denotes the time to solve the sum problem for the modified costs. In addition to the k-sum problem, the authors also considered the k-sum deviation problem. They presented an algorithm that is based on solving at most n k-sum optimization problems. An overall time bound of $\mathcal{O}(n^2 T)$ is reported for this approach, where T is defined as stated above.
In Punnen and Aneja [173], the authors presented an improved version of their algorithm from [90] that can be applied to an arbitrary feasible set $\mathcal{X}$. For their new approach, the authors assumed that only non-negative cost coefficients are given. Compared to the original version of the algorithm stated in [90], the cost coefficients in the i^{th} iteration of the algorithm are now replaced by $\tilde{c}(e) = \max\{c(e) - c(e_i), 0\}$, while the rest of the algorithm remains unchanged. Hence, the time bound of $\mathcal{O}(nT + n^2)$ is also

valid for the new algorithm. In addition to the improved algorithm, the authors further presented a polynomial time ε-approximation scheme for Problem (6.12), based on the assumption that there exists such an approximation scheme for the ordinary sum problem of the considered combinatorial optimization problem.

In the following, we discuss the multiple objective interpretation of this problem that can be used to solve the k-sum optimization problem for an arbitrary cost function $c : \mathcal{E} \to \mathbb{Z}$. We assume a minimal cardinality for all feasible solutions, i.e. there exists $m \geq 1$ such that $|S| \geq m$ for all $S \in \mathcal{X}$. Furthermore, let $k \in \{1, \ldots, m\}$. Instead of sorting the cost coefficients in non-increasing order, we use the k-max objective to model the problem. Thus, Problem (6.12) is equivalent to

$$\min_{S \in \mathcal{X}} \sum_{t=1}^{k} t\text{-}\max_{e \in S}\{c(e)\}. \tag{6.13}$$

Hence, Problem (6.13) can be seen as a scalarized version of the associated multiple objective combinatorial problem

$$\min_{S \in \mathcal{X}} \left(\max_{e \in S}\{c(e)\}, 2\text{-}\max_{e \in S}\{c(e)\}, \cdots, k\text{-}\max_{e \in S}\{c(e)\} \right)^{\top}. \tag{6.14}$$

We distinguish the cases $k = 2$ and $k > 2$ in the following. If $k = 2$, Problem (6.14) corresponds to a biobjective optimization problem of a bottleneck and a k-max objective. Hence, Algorithm 4.2 can be used to derive the optimal solution for the k-sum problem in this case. According to Theorem 4.3 and Theorem 5.3, the resulting algorithm solves the problem within $\mathcal{O}(n \log(n) T^{\star})$ time. Here, $T^{\star}$ denotes the time to find the optimal solution of the binary sum problem that results from the k-max optimization problem that has to be solved in each iteration of the algorithm. For the case that all cost coefficients are given non-negative, we can compare the multiple objective approach with the algorithm of Punnen and Aneja stated in [173]. According to the results stated above, we conclude that under the condition that the binary version (i.e., all costs are in $\{0, 1\}$) of the considered combinatorial problem with sum objective is easier to solve by a factor of $\mathcal{O}(\log(n))$ as compared to the general case, Algorithm 4.2 is as least as good as the algorithm stated in [173] with respect to the worst case time complexity. If this condition is not satisfied, the algorithm of Punnen and Aneja should be rather used to solve the given problem for $k = 2$.

For the case that $k > 2$, Algorithm 4.2 no longer applies, since more than one k-max objective has to be considered. However, Algorithm 5.2 can be used to solve the problem instead. From Theorem 5.9 we recall that this can be done within $\mathcal{O}(n^{k-1} T^{\mathcal{K}}_{k-1})$ time. Since exactly one bottleneck and $k - 1$ k-max objectives are given, either the bottleneck or one of the k-max objectives can be chosen as objective function for the ε-constraint problem that has to be solved in each iteration of the algorithm. However, in both cases Algorithm 4.2 results in a higher order of complexity, compared to the Algorithm stated in [173]. Hence, the multiple objective approach for $k \geq 2$ should be seen as a generalized approach to model the k-sum problem rather than an efficient approach to solve the given problem.

For the case that all feasible solutions have the same cardinality m, we further remark

that we can use Lemma 5.11 to remodel the k-sum objective as

$$\sum_{i=1}^{p} c(e_{t_i}) = \sum_{e\in S} c(e) + \sum_{t=1}^{m-k} t\text{-}\max_{e\in S}\{-c(e)\}.$$

For the case that $k = m-1$ this problem simplifies to the minimum deviation problem (cf. Problem (6.10)) stated in Section 6.2.2. Hence, once more Algorithm 4.2 can be used to solve the problem. If $k < m-1$, Algorithm 5.2 has to be used instead. Note that in this case, the k-max objectives could also be replaced by the corresponding k-min objectives, using Lemma 5.11.
For the generalized version of the k-sum optimization problem, let $|S| \geq m$ for some $m \in \mathbb{N}$ and all $S \in \mathcal{X}$. Furthermore, let $1 \leq l \leq k \leq m$. Applying the notation from the beginning of this subsection, we can define the *(l,k)-sum optimization problem* by

$$\min_{S\in\mathcal{X}} \sum_{i=l}^{k} c(e_{t_i}) = \sum_{t=l}^{k} t\text{-}\max_{e\in S}\{c(e)\}. \tag{6.15}$$

Note that Problem (6.15) have been investigated in the context of location theory. In this context, Problem (6.15) is also known as *(k,l)-trimmed mean problem* (cf., e.g., Nickel and Puerto [153]). Independently, these special types of problems are discussed by Turner [204] who introduced the concept of *universal combinatorial optimization problems.*
Having a closer look at Problem (6.15), we see that it generalizes several other problems, already treated in this section. For the choice $l = 1$, the original k-sum optimization problem is obtained. If a fixed cardinality for all feasible solutions is assumed, the given problem simplifies to the minimum deviation problem if $l = 2$ and $k = m$, while Problem (6.10) is recovered by the choice $l = 1$ and $k = m-1$. Hence, Problem (6.15) simultaneously generalizes both, the k-sum as well as the minimum deviation problem. Since Problem (6.15) is the algebraic sum of $(k-l+1)$ k-max objectives, it can be interpreted as scalarized version of the associated multiple objective problem

$$\min_{S\in\mathcal{X}} \left(l\text{-}\max_{e\in S}\{c(e)\}, \cdots, k\text{-}\max_{e\in S}\{c(e)\} \right)^{\top},$$

and once more, Algorithm 5.2 can be used to derive an optimal solution of Problem (6.15). Since $(k-l+1)$ k-max objectives are given, the algorithm takes $\mathcal{O}(n^{k-l}T^{\mathcal{K}}_{k-l})$ time to solve the problem.

Theorem 6.12 *Let an (l,k)-sum optimization problem be given. Then, Algorithm 5.2 can be used to determine an optimal solution for the given problem within $\mathcal{O}(n^{k-l}T^{\mathcal{K}}_{k-l})$ of time.*

If a fixed cardinality for all feasible solutions is assumed, we can use Lemma 5.11 to remodel the (l,k)-sum optimization problem as

$$\begin{aligned}\sum_{t=l}^{k} t\text{-}\max_{e\in S}\{c(e)\} &= \sum_{e\in S} c(e) - \sum_{t=1}^{l-1} t\text{-}\max_{e\in S}\{c(e)\} - \sum_{t=k+1}^{m} t\text{-}\max_{e\in S}\{c(e)\} \\ &= \sum_{e\in S} c(e) + \sum_{t=m-l+2}^{m} t\text{-}\max_{e\in S}\{-c(e)\} + \sum_{t=1}^{m-k} t\text{-}\max_{e\in S}\{-c(e)\}. \end{aligned} \tag{6.16}$$

Optimization Problem	**Fixed Cardinality**	**Algorithm**	**Complexity**
Balanced Opt. Prob.	no	Algorithm 4.2	$\mathcal{O}(nT^{\mathcal{B}})$
(k,l)-Balanced Opt. Prob.	no	Algorithm 5.2	$\mathcal{O}(nT_1^{\mathcal{K}})$
Minimum Deviation Prob.	yes	Algorithm 4.2	$\mathcal{O}(nT^{\mathcal{S}})$
k-Deviation Prob.	yes	Algorithm 5.2	$\mathcal{O}(n^k T_k^{\mathcal{S}})$
k-Sum Opt. Prob.	no	Algorithm 5.2	$\mathcal{O}(n^{k-1} T_{k-1}^{\mathcal{K}})$
(l,k)-Sum Opt. Prob.	no	Algorithm 5.2	$\mathcal{O}(n^{k-l} T_{k-l}^{\mathcal{K}})$

Table 6.1: *Summary of the treated optimization problems, where $T^{\mathcal{S}}$, $T^{\mathcal{B}}$ and $T^{\mathcal{K}}$ denote the time to solve the corresponding sum, bottleneck and k-max optimization problem, respectively. Subscripts represent the number of additional binary constraints that have to be considered due to the additional k-max objectives involved.*

Also in this case, Algorithm 5.2 solves the problem within $\mathcal{O}(n^s T_s^{\mathcal{S}})$ time, where $s = m-k+l-1$ and $T_s^{\mathcal{S}}$ denotes the time to solve the sum problem for the original costs and s additional binary constraints. Note that this alternative formulation can especially be used to reduce the number of objectives. If $k - l \geq \lceil \frac{m}{2} \rceil$ holds true, where $\lceil \frac{m}{2} \rceil = \min\{n \in \mathbb{N} : n \geq x\}$, the number of objectives for the alternative formulation (6.16) is at most $\lfloor \frac{m}{2} \rfloor$, and hence smaller than for the original formulation (6.15).
We finally remark that due to Lemma 5.11, the k-max objectives used in Problem (6.16) can be replaced by the corresponding k-min objectives resulting in an alternative formulation.

Summary of the Results

In this subsection, we summarize and discuss the results that we derived for the combinatorial problems treated in the previous subsections. Table 6.1 gives a short overview of the problems and the results we obtained for the respective multiple objective approach. Note that specific reformulations of the considered problems that are based on a fixed cardinality assumption are omitted in the table.
As already discussed in the last subsections, we have seen that many approaches from the literature that are used to solve the problems listed in Table 6.1 are based on a similar idea: The given problem is implicitly understood as a scalarized version of a general multiple objective combinatorial problem. A carefully designed version of an algorithm that solves the multiple objective problem is used to derive a representative of a (supported) non-dominated solution that corresponds to an optimal solution of the scalarized problem. Hence, multiple objective optimization builds a good framework to understand and solve most of these problems.
However, in general the multiple objective approach does not yield the most efficient approach to solve Problem (6.3), when the involved objectives are defined based on a single cost function. As in the case of the k-sum optimization problem, the multiple objective approach takes in general much more time to solve the problem as compared to the algorithm presented in Punnen and Aneja [173]. Nevertheless, it is a theoretically interesting result illuminating the relation between seemingly different kind of problems: It is possible to reformulate all the considered problems and their gen-

eralized versions as a scalarized version of an associated multiple objective problem, simply using sum, bottleneck and k-max objective functions.
Concerning the complexity of the presented algorithms, we conclude from Table 6.1 that the given combinatorial problems are solvable in a polynomial amount of time, whenever this holds true for the corresponding sum, bottleneck and k-max subproblems that have to be solved in each iteration of the specific algorithm. Since for some of the considered problems additional binary constraints have to be taken into account, it may happen that the resulting constrained problems are harder to solve than their unconstrained counterparts. Nevertheless, the multiple objective solution approach has two important advantages: First, the involved side constraints are all of binary type. We will show in Chapters 9 and 10 that there exist classes of combinatorial problems that still can be solved in polynomial time although a binary constrained problem has to be solved. Second, the structure of the involved side constraints does not change that much during the course of the algorithm. Hence, as especially mentioned in Duin and Volgenant [50] for the minimum deviation spanning tree problem, it may be possible to derive an optimal solution for the current iteration based on previously calculated solutions. If this is the case, this may result in a decrease of the complexity for the used solution approach. Thus, the overall complexity may depend, amongst others, on the decision, whether the sequence of constrained problems is solved from small to large costs or vice versa (cf. also [50]).

6.3 Conclusions and Further Ideas

Based on the ideas developed in Chapter 3, we investigated how multiple objective solution approaches can be used to solve single objective combinatorial optimization problems. Especially when a combinatorial problem with a single constraint is given, we saw that the more general biobjective version of the problem can be used to calculate a good approximation of the optimal solution within the set of supported efficient solutions. If the unconstrained version of the problem is solvable in polynomial time and the number of extreme supported efficient solution is polynomially bounded, such a first approximation can be determined within a polynomial amount of time, too. However, the quality of this approximation crucially depends on the given problem instance.
In the second part of this chapter we dealt with combinatorial optimization problems where the objective is given as the algebraic sum of different types of objective functions. As such problems can be seen as scalarized versions of more general multiple objective problems, the ideas developed in Section 3.2 especially apply to these special types of combinatorial problems. Based on the associated multiple objective problem formulations we showed that many algorithms presented in the literature can be interpreted as specially designed versions of more general algorithms that we discussed in Chapters 4 and 5 for multiple objective optimization problems involving sum, bottleneck and k-max objectives. Based on this observation, we further made use of this idea to derive solution concepts for generalized balanced, minimum deviation and k-sum optimization problems. We showed that these generalized versions can be solved in a polynomial amount of time, whenever this holds true for the constrained subproblems that have to be solved to derive (weakly) efficient solutions of the associated multiple

objective problem.
We finally remark that we did not focus on any special class of combinatorial optimization problems in this chapter. Hence, especially for the optimization problems discussed in Section 6.2.2 there may exist improved solution concepts that can be used to develop appropriate algorithms for special classes of combinatorial problems, like the assignment, the spanning tree or the shortest path problem. Given a specific class, future research could focus on specially designed solution concepts for (generalized) balanced, minimum deviation and k-sum optimization problems that use special properties of the given class of combinatorial problems. For more details on this topic, we especially refer to Turner [204] where, amongst others, many of the above stated problems are discussed in a generalized approach to combinatorial optimization problems. In addition, also the literature cited in the different sections could be of interest, as improved versions of the more general algorithms applied to a specific class of combinatorial problems were already discussed in several articles.
If a new solution concept is based on a general algorithm presented in Section 6.2.2, it could be further investigated whether solutions calculated in earlier iterations of the algorithm can be used to derive an optimal solution for the current iteration. Since the involved subproblems only change slightly during the course of the algorithm, it may be possible to decrease the overall time complexity of the algorithm if special properties of a class of combinatorial problems are used. However, this has to be tested at the example of well-studied classes of combinatorial problems.

Chapter 7

Connectedness of Efficient Solutions in Multiple Objective Combinatorial Optimization

Structural properties of the efficient set of multiple objective combinatorial optimization problems (MCOP) play a crucial role for the development of efficient solution methods. A central question relates to the connectedness of the efficient set with respect to combinatorially or topologically motivated neighborhood structures. A positive answer to this question would provide a theoretical justification for the application of fast neighborhood search techniques, not only for multiple objective but also for appropriate formulations of single objective problems (cf. Chapters 3 and 6).

Given a multiple objective combinatorial problem where the single objective counterpart can be solved in polynomial time, the connectedness of the efficient set would imply that non-supported non-dominated solutions of the given problem could be found in a reasonable amount of time, when the two-phase method is applied (cf. Chapter 4 for further details).

In addition, neighborhood structures are often used in evolutionary approaches for solving MCOPs (see, e.g., Ghosh and Dehuri [79] for a general survey on evolutionary solution techniques for MCOPs). In these approaches, new solution candidates are frequently generated by a mutation of already existing potentially efficient solutions based on simple exchange operations of involved elements from the ground set. Hence, the connectedness of the efficient set additionally plays an important role for the development of efficient heuristics to solve MCOPs.

After reviewing the existing results on the connectedness of the efficient set, we present two different concepts for defining adjacency of efficient solutions in multiple objective combinatorial optimization in the following. Based on these two definitions and on already existing negative results for the connectedness of the efficient set for the multiple objective minimum spanning tree problem (MSTP) and the multiple objective shortest path problem (MSPP), we extend these results to most of the classical problems in combinatorial optimization like the linear assignment problem, the knapsack problem and the traveling salesman problem, amongst others. We further show that the connectedness property also fails for the weakly efficient set of such problems in general. In addition, we provide numerical investigations on the frequency of a non-connected

efficient set for special classes of cardinality constrained knapsack problems.
We recall that in continuous optimization connectedness is defined in a topological sense. A set S is called *topologically connected* if there do not exist nonempty open sets S_1 and S_2 such that $S \subseteq S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$.
For multiple objective linear programming problems (MLP) the efficient set and the non-dominated set are topologically connected as shown by Naccache [149] and Warburton [211], respectively. This definition is not directly applicable in combinatorial optimization due to the discrete structure of the efficient set. Instead of the topological connectedness, a graph theoretical definition can be used for MCOPs as described, for example, in Ehrgott and Klamroth [58] and Paquete and Stützle [161].

Definition 7.1 *For a given MCOP the* adjacency graph of efficient solutions $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ *of the MCOP is defined as follows: $\mathcal{V}$ consists of all efficient solutions of the given MCOP. An (undirected) edge is introduced between all pairs of nodes which are adjacent with respect to the considered definition of adjacency for the given MCOP. These edges form the set $\mathcal{A}$.*

The connectedness of the efficient set $\mathcal{X}_E$ is now defined via the connectedness of an undirected graph. We recall that an undirected graph $\mathcal{G}$ is said to be connected if every pair of nodes is connected by a path.

Definition 7.2 *The set $\mathcal{X}_E$ of all efficient solutions of a given MCOP is said to be* connected *if its corresponding adjacency graph $\mathcal{G}$ is connected.*

Since for MCOPs the adjacency of two efficient solutions x and x' can usually be expressed by an application of some *elementary move* (i.e., x can be obtained from x' by applying exactly one move), a neighborhood concept is introduced to the problem. An efficient solution x' is contained in the k-neighborhood of an efficient solution x if x' is reachable from x by applying at most k elementary moves. The minimum number of elementary moves needed to get from x to x' is called the *distance* between these two solutions. Using this concept, the definition of the adjacency graph can be further extended.

Definition 7.3 *The* weighted adjacency graph $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$ *of efficient solutions is defined as follows: $\mathcal{G}'$ is a complete and undirected graph. Its set of nodes $\mathcal{V}'$ consists of all efficient solutions of the given MCOP. The weight w_{ij} of an edge between two nodes $v_i, v_j \in \mathcal{V}'$ is given by the distance between these two nodes with respect to the considered neighborhood.*

For each $k \in \mathbb{N}$ a subgraph $\mathcal{G}'_k$ can be extracted from $\mathcal{G}'$ that contains all the nodes of $\mathcal{G}'$ but only those edges which have a weight less or equal to k. Since $\mathcal{G} = \mathcal{G}'_1$, $\mathcal{X}_E$ is connected if and only if $\mathcal{G}'_1$ is connected. If $\mathcal{X}_E$ is not connected, the graph $\mathcal{G}'_1$ decomposes into at least two connected subgraphs which build a partition of $\mathcal{G}'_1$. More generally we define:

Definition 7.4 *A* component *or a* cluster *of efficient solutions at distance k is a maximally connected subgraph of $\mathcal{G}'_k$.*

If $\mathcal{G}'_k$ is a connected graph, there exists exactly one component which is equal to $\mathcal{G}'_k$. Otherwise, the set of all components build a partition of $\mathcal{G}'_k$.

The literature on the connectedness of the set of efficient solutions in multiple objective optimization is scarce. The first publications appeared in the seventies together with the development of the multiple objective simplex method. In his fundamental work, Isermann [109] showed that the set of basic feasible and fundamental solutions of MLPs are connected and, thus, established the correctness of multiple objective simplex methods. Two solutions of an MLP are said to be adjacent in the sense of Isermann [109] if they have $m-1$ basic variables in common, where m denotes the length of the basis. Naccache [149] established connectivity for more general problems with closed, convex and K-compact objective spaces where K is a closed, convex and pointed cone. Helbig [101] generalized this to locally convex spaces.

Lately, research on the connectedness of efficient solutions of MCOPs was coined by assertions and falsifications. Martins [128] claimed that there always exists a sequence of adjacent efficient paths connecting two arbitrary efficient paths for MSPP. However, Ehrgott and Klamroth [58] demonstrated the incorrectness of the connectedness conjecture for MSPP and MSTP by a counter-example and, thus, disproved Martins [128]. Ehrgott and Klamroth [58] showed that any graph can be extended in such a way that the adjacency graph (of MSPP and MSTP) for the problem on the extended graph is not connected. They conjectured that in practice, it is rather unlikely that the adjacency graph of a specific MSTP is not connected. However, their numerical tests included only 50 randomly generated graphs with a rather small number of nodes.

In Przybylski et al. [168], the example of Ehrgott and Klamroth [58] was used to show the incorrectness of the algorithm of Sedeño-Noda and González-Martín [190]. The latter tried to find all efficient flows of a biobjective integer flow problem by a method based on simplex pivots.

O'Sullivan and Walker [155] proposed two algorithms for the equally-weighted biobjective knapsack problem to determine the complete set of efficient solutions. Whether the complete set can be generated, depends on "unproven characteristics of efficient knapsacks" - the connectedness of the set of efficient solutions for this problem.

In da Silva et al. [43], the geometrical configuration of the non-dominated set for three different models of the biobjective binary knapsack problem was discussed. Under a cardinality constraint and the supplementary assumption that the sum of each pair of the objective coefficients is constant, it was shown that the set of all efficient solutions is connected. In this case, the non-dominated set consists of a line segment with slope -1. An LP-based approach is used to define adjacency of two efficient solutions.

Gorski [83] recognized that the definition of adjacency is not canonical. One could think of structural, problem-dependent definitions or of LP-based, problem independent definitions. Based on ideas mentioned in Ehrgott and Klamroth [58], he aimed at a formal definition of adjacency.

The numerical study of Paquete et al. [158] investigates the number of clusters of near efficient solutions obtained with some local search algorithms for the multiple objective traveling salesman problem. In Paquete and Stützle [161] statistics on the clusters of near efficient solutions for the biobjective travelling salesman problem and the biobjective quadratic assignment problem are reported. A stochastic local search method was employed to retrieve the near optimal solutions. It should be pointed out that neither the solutions obtained are guaranteed to be efficient, nor that all efficient solutions are found by the local search method. Thus, the focus of this study is on the performance assessment of local search for the two MCOPs mentioned.

Some comments on the connectedness of efficient solutions for biobjective multimodal assignment problems are also contained but not further perused in Pedersen [163].
Lust and Teghem [126] presented a two-phase Pareto local search method (cf. also Paquete et al. [158]) that is based on a neighborhood search technique to find a good approximation of the efficient set of the biobjective traveling salesman problem.
Finally, Ruzika [184] discussed a branch & bound approach for solving weight constrained shortest path and spanning tree problems, respectively. In the presented approach, the author made use of the connectedness of the supported efficient solutions of the associated biobjective combinatorial problems to derive an efficient branching rule for the specific problems.
While we focus on MCOPs with sum objectives in this chapter, the connectedness of the efficient set for general MCOPs with bottleneck objectives is discussed in Chapter 8.
The remainder of this chapter is organized as follows. In Section 7.1, we discuss different definitions of adjacency of feasible solutions of a MCOP. We show that adjacency may be defined based on appropriate IP-formulations of a given problem and using the natural neighborhood of basic feasible solutions of linear programming. For many concrete problems, however, it appears to be more convenient to consider a combinatorial neighborhood. In Section 7.2 we discuss and extend existing results for the MSPP and the MSTP and present new connectedness results for other major classes of MCOPs like the multiple objective knapsack problem (MOKP), the multiple objective assignment problem (MOAP) and the multiple objective traveling salesman problem (MTSP), amongst others. We report numerical tests on adjacency of efficient solutions for the binary MOKP with bounded cardinalities and the binary multiple choice MOKP in Section 7.3. Finally, we conclude in Section 7.4 with current and future research ideas.
We further remark that the main results of this chapter are published as a technical report (cf. Gorski et al. [84, 85]) and in Ruzika [184].

7.1 Categorizing Different Concepts of Adjacency

We distinguish between two essentially different concepts of adjacency of efficient solutions:

- The adjacency of two efficient solutions is defined via the adjacency of basic feasible solutions of an appropriate model of the MCOP as a multiple objective integer linear programming problem (MILP), and its LP relaxation.

- The definition of adjacency is adapted to the special combinatorial structure of the given MCOP.

While the latter concept has received some attention in the recent literature, for example, in the context of neighborhood search algorithms (see, for example, Paquete and Stützle [161]), the former has only been used so far for special types of MCOPs (cf. Ehrgott and Klamroth [58], for the MSPP and the MSTP and da Silva et al. [43] for binary knapsack problems). Subsequently, we formalize these two different concepts of adjacency.

7.1.1 MILP-based Definition of Adjacency and Appropriate MILP Models

For the MILP-based definition, we assume that the MCOP can be formulated as a combinatorial optimization problem with sum type objective functions as specified in Definition 7.5 below.

Definition 7.5 *Let $\mathcal{E} := \{e_1, \ldots, e_n\}$ be a nonempty finite set, let $c = (c_1, \ldots, c_p) : \mathcal{E} \to \mathbb{Z}^p$, $p > 1$, consists of p integer-valued weighting functions on the elements of $\mathcal{E}$, and let $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$ be a subset of the powerset of $\mathcal{E}$. A* multiple objective combinatorial sum problem *is a problem of the form*

$$\begin{array}{ll} \min & \left(\sum_{e\in S} c_1(e), \ldots, \sum_{e\in S} c_p(e)\right)^\top \\ s.t. & S \in \mathcal{X}. \end{array} \tag{7.1}$$

An instance of this problem is denoted by $(\mathcal{E}, \mathcal{X}, c)$.

In most cases, the feasible set $\mathcal{X}$ is not arbitrary but introduces a certain structure to the problem. As it is well-known, every feasible solution $S \in \mathcal{X}$ of a MCOP (7.1) can be identified with a binary vector $x \in \{0,1\}^n$ by setting

$$x_i := \begin{cases} 1 & \text{if } e_i \in S \\ 0 & \text{otherwise.} \end{cases} \tag{7.2}$$

The weight $\sum_{e\in S} c(e)$ of a solution S and of its binary representation x can be computed as Cx using an appropriate objective matrix $C \in \mathbb{Z}^{p\times n}$ with components $c_{ij} := c_i(e_j)$ for $i = 1, \ldots, p$ and $j = 1 \ldots, n$ (cf., e.g., Ehrgott [54]).

Theorem 7.6 *Every multiple objective combinatorial sum problem* (7.1) *can be modeled as an MILP of the form*

$$\min \{Cx : Ax \leqq b, x \in \{0,1\}^n\},$$

using variables x as defined in (7.2) *and with an appropriate constraint matrix $A \in \mathbb{Z}^{m\times n}$ and right hand side vector $b \in \mathbb{Z}^m$, such that there is a one-to-one correspondence between all feasible (and particularly all efficient) solutions of the two problems.*

A proof of this observation can be found, for example, in Ehrgott [54]. We refer to a formulation of a MCOP according to Theorem 7.6 as a *canonical MILP formulation* of the MCOP. Note that a canonical MILP formulation is in general not unique. For the sake of simplicity we assume in the following that a canonical MILP formulation is given. This is, however, not a necessary assumption for our analysis. Namely, instead of a canonical MILP formulation we may consider any MILP formulation of the problem that satisfies the conditions of Definition 7.8 below, i.e., there is a one-to-one correspondence between the feasible solutions of the MCOP and the extreme points of the LP relaxation of the MILP.
Denote by $U := \{x \in \{0,1\}^n : Ax \leqq b\}$, $P := \{x \in [0,1]^n : Ax \leqq b\}$ and $P^* := \{x \in \mathbb{R}^n : x \in \text{conv}(U)\}$ the feasible set of a canonical MILP formulation, the feasible set of the LP relaxation of the MILP and the convex hull of all feasible solutions of the MILP, respectively.

Definition 7.7 *If the polytope P of the LP relaxation of a canonical MILP formulation coincides with the polytope P^*, we say that* $\min\{Cx : Ax \leqq b, x \in \{0,1\}^n\}$ *is an* exact *MILP formulation of the given MCOP.*

In order to use an LP-based definition of adjacency, a one-to-one correspondence between feasible solutions of the MCOP and basic feasible solutions of the LP relaxation of the corresponding MILP formulation is indispensable. Therefore, we restrict our analysis to exact MILP formulations of a given MCOP. Note that otherwise there may exist basic feasible solutions of the LP relaxation of the (non-exact) MILP formulation that are not integer and that do not correspond to feasible solutions of the MCOP. On the other hand, every feasible solution of the MCOP must correspond not only to a feasible solution of its (exact) MILP formulation, but to a basic feasible (or extreme point) solution of the LP relaxation of the MILP.

Definition 7.8 *An MILP formulation of a given MCOP is called* appropriate *if it satisfies the following two conditions:*

- *The MILP formulation is exact, i.e., $P = P^*$.*
- *Every feasible solution of the MILP is an extreme point of P^*.*

Polyhedral theory can be used to show that for every MCOP at least one appropriate MILP formulation exists.

Lemma 7.9 *There exists at least one appropriate MILP formulation for every instance $(\mathcal{E}, \mathcal{X}, c)$ of a MCOP.*

Proof: Suppose that an arbitrary instance $(\mathcal{E}, \mathcal{X}, c)$ of a MCOP is given, let U denote the set of all feasible binary vectors for the canonical formulation of the MCOP (cf. Theorem 7.6), and let P^* denote the convex hull of U. Then an exact formulation of the problem is given by

$$\min\{Cx : x \in U\}. \tag{7.3}$$

Since all vectors $x \in U$ are binary, they are essential for the generation of the convex hull P^* of U. Hence, problem (7.3) is equivalent to

$$\min\{Cx : x \in P^*,\, x \in \{0,1\}^n\}, \tag{7.4}$$

and every feasible solution of this problem is an extreme point of P^*. Moreover, since all feasible vectors $x \in U$ are essential for the generation of P^*, there exists a description of P^* by means of a finite set of linear inequalities of the form $P^* = \{x \in \mathbb{R}^n : \tilde{A}x \leqq \tilde{b}\}$ with appropriate rational $\tilde{A} \in \mathbb{Q}^{\tilde{m}\times n}$ and $b \in \mathbb{Z}^{\tilde{m}}$ (see, for example, Nemhauser and Wolsey [150]), yielding an appropriate MILP formulation for problem (7.4) and hence for the MCOP. □

Note that the polytope P^* of an appropriate MILP formulation of MCOP does not contain any integer points in its interior nor in the interior of any of its faces.
The following two properties can be derived for an appropriate MILP formulation of a MCOP which will be used later for the LP-based definition of adjacency.

Lemma 7.10 *If an MILP formulation of a MCOP is appropriate, then its LP relaxation, after transformation into standard form, has the following two properties:*

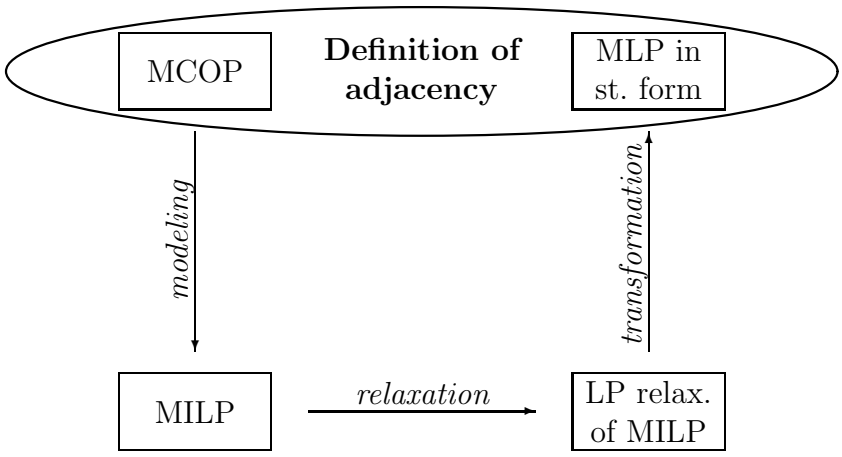


Figure 7.1: *Definition of adjacency via an appropriate MILP formulation.*

(M1) Every basic feasible solution corresponds to a feasible solution of the MCOP.

(M2) For every feasible solution of the MCOP there exists at least one basis such that the solution of the MCOP is equal to the corresponding basic feasible solution of the above LP relaxation of the MILP in standard form.

Proof: Follows immediately from the analysis above and from polyhedral theory. □

Lemma 7.11 *Suppose that a MCOP is given by an (arbitrary binary) MILP formulation. If the LP relaxation of the MILP, after transformation into standard form, satisfies (M1) and (M2), then the MILP is an appropriate formulation of the MCOP.*

Proof: Let the MILP formulation of the MCOP be given as $\min\{Cx : Ax \leqq b, x \in \{0,1\}^n\}$ and let U, P and P^* be defined as above. Furthermore, let U_p denote the finite set of all extreme points of P. Since we assumed that the MCOP is finite and $\mathcal{E} \neq \emptyset$, P is bounded and hence $P = \text{conv}\{u_p \in U_p\}$.
First, we show that $P = P^*$. By construction we have that $P^* \subseteq P$. To show that also $P \subseteq P^*$ suppose that, to the contrary, there exists an extreme point $x^0 \in U_p \subseteq P$ that is not contained in P^*. Since $x^0 \in P$, x^0 satisfies $Ax^0 \leqq b$ and $x^0 \in [0,1]^n$. Furthermore, since x^0 is an extreme point of P there exists a basic feasible solution of the LP relaxation of the MILP (after transformation into standard form) corresponding to x^0. Hence, $x^0 \in \{0,1\}^n$ by (M1) and therefore $x^0 \in U$, contradicting the assumption that $x^0 \notin P^*$.
Since (M2) and Theorem 7.6 imply that there is a one-to-one correspondence between feasible solutions u of the MILP and feasible solution $S \in \mathcal{X}$ of the MCOP, the MILP is indeed an appropriate formulation of the MCOP. □

Since the two properties (M1) and (M2) characterize appropriate MILP formulations of MCOPs, they can be used for the definition of an LP-based concept of adjacency for these problems. Figure 7.1 illustrates this idea. In this context, two bases of an LP are called adjacent if they can be obtained from each other by one single pivot operation.

Definition 7.12 *Let an appropriate MILP formulation of a MCOP be given. Two feasible solutions x^1 and x^2 of the MCOP are called* adjacent with respect to the given MILP formulation *if there exist two adjacent bases of the LP relaxation of the MILP (after transformation into standard form) corresponding to x^1 and x^2, respectively.*

Since by MLP theory all bases that represent efficient solutions of the LP relaxation of the MILP in standard form are connected (see, for example, Ehrgott [54]), the resulting adjacency graph always contains a connected subgraph representing these solutions. Note that these solutions are always supported efficient solutions of the MCOP.
The above definition of adjacency (and hence the resulting adjacency graph) depends on the chosen appropriate MILP formulation of the given MCOP, which is in general not unique. If different appropriate MILP formulations are used to model the same MCOP, we can expect different results concerning the connectedness of efficient solutions of the problem. In this context, Definitions 7.1 and 7.2 must always be understood with respect to the chosen appropriate MILP formulation of a MCOP.
Note also that, using the above definitions of adjacency and connectedness, polyhedral theory implies that the set of optimal solutions of a single objective combinatorial optimization problem is always connected (or even unique). Therefore, the question whether the corresponding multiple objective optimization problems have a connected adjacency graph is in general non-trivial.
The following well-known fact from polyhedral theory shows that the last step in Figure 7.1, i.e., the transformation of the LP relaxation of the MILP into standard form, can as well be omitted in the definition of adjacency (Definition 7.12) since the considered MILPs are always bounded problems.

Theorem 7.13 *Let $P = \{x \in [0,1]^n : Ax \leqq b\}$ be the feasible set of an LP and let P_{st} denote the polyhedron obtained from P after transformation into standard from. Then two extreme points of P are connected by an edge in P if and only if the corresponding extreme points of P_{st} are connected by an edge in P_{st}.*

7.1.2 Combinatorial Definitions of Adjacency

Combinatorial definitions of adjacency are usually based on simple operations that transform one feasible solution of a specific problem class into another, "adjacent" feasible solution. We call such operations *(elementary) moves*. An elementary move is called *efficient* if it leads from one efficient solution of the problem to another efficient solution. Two efficient solutions are called *adjacent* if one can be obtained from the other by one efficient move.
Examples for elementary moves for specific problem classes are the insertion and deletion of edges in a spanning tree, the modification of a matching along an alternating cycle, or simply the swap of two bits in a binary solution vector. In single objective optimization such elementary moves are frequently used in exact algorithms (e.g., the negative dicycle algorithm for the minimum cost flow problem) as well as in heuristic algorithms (e.g., the two-exchange heuristic for the TSP). Note that for specific problem classes, a combinatorial definition of adjacency may in fact coincide with an MILP-based definition of adjacency as discussed in Section 7.1.1.

While for the MILP-based definition of adjacency the set of optimal solutions of the single objective problem corresponding to a given MCOP is always connected in the sense of Definition 7.1, this is not necessarily true for combinatorial definitions of adjacency.
We call an elementary move for a given problem class *canonical* if the set of optimal solutions of the corresponding single objective problem is connected for all problem instances. Although non-canonical moves immediately imply non-connectedness results also in the multiple objective case, such extensions may be used for the development of heuristic methods based on neighborhood search (see, for example, Paquete and Stützle [161]).
For some classes of combinatorial problems, an elementary move corresponds to a move from one extreme point to another adjacent extreme point along an edge of the polytope which is obtained by the LP relaxation of an MILP formulation of the given combinatorial problem (cf. Definition 7.12). If the given MILP formulation is appropriate in the sense of Definition 7.8, the corresponding elementary move is always canonical. As an immediate consequence of Theorem 7.13, we finally state:

Theorem 7.14 *Let a move-operation introduce a combinatorial definition of adjacency for a class of MCOPs for which also an appropriate MILP formulation exists, and let P denote the polytope of its LP relaxation. If there is a one-to-one correspondence between the set of all possible elementary moves between feasible solutions of the MCOP and the edge-structure of P, i.e., solution x_1 can be obtained from x_2 by an elementary move if and only if the x_1, x_2-corresponding extreme points of P are connected by an edge in P, then the resulting adjacency graphs for the combinatorial definition and the MILP-based definition of adjacency coincide.*

7.2 Connectedness Results for Specific Combinatorial Optimization Problems

In this section, adjacency of efficient solutions is comprehensively investigated for various combinatorial optimization problems. Due to intended clarity and legibility, each of these fundamental problems is treated in a separate paragraph. Each paragraph contains - to the best of our knowledge - all results available in the literature. The more significant part of this section yet contains two major components.
First, we investigate the question of adjacency of the graph of efficient solutions for problems which have not been treated in the literature so far. Second, the concept of adjacency of the graph of efficient solutions is extended and structural properties of this graph, and its extensions are investigated. The latter is done exemplarily for MSPP. Treating all of the combinatorial optimization problems in this section likewise certainly goes beyond the scope of this thesis. Nevertheless, it should be emphasized that analogous results can be achieved for other problems utilizing similar techniques.

7.2.1 Shortest Path Problems

Let $G = (V, A)$ be a directed graph with source node s and sink node t. The multiple objective shortest path problem (MSPP) can be formulated as

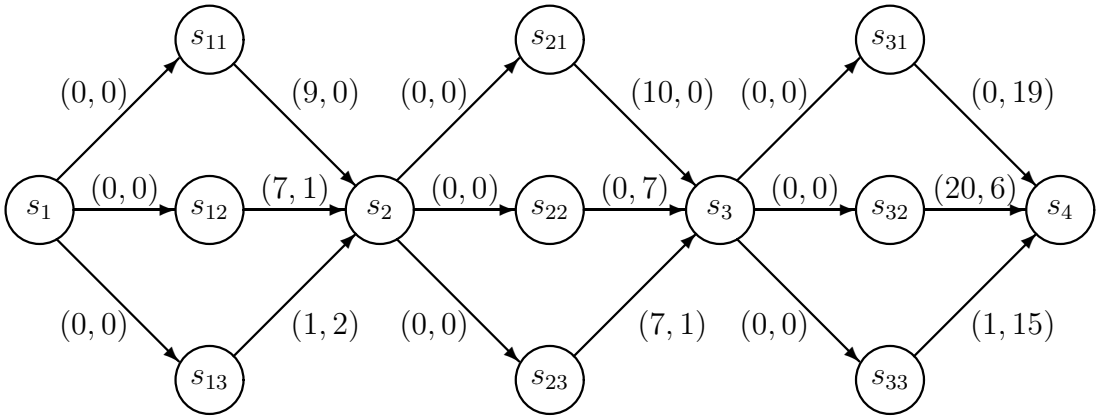


Figure 7.2: *Digraph from Ehrgott and Klamroth [58].*

$$\begin{aligned} &\min(c^1x, \ldots, c^px)^T \\ \text{s.t.} \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} &= \begin{cases} 1, & \text{if } i = s \\ 0, & \text{if } i \in \{1, \ldots, n\} \setminus \{s, t\} \\ -1, & \text{if } i = t. \end{cases} \\ &x_{ij} \in \{0, 1\}, \quad \forall a = (i, j) \in A, \end{aligned} \tag{7.5}$$

where the vectors $c^1, \ldots, c^p$ are assumed to be non-negative. Ehrgott and Klamroth [58] called two efficient paths adjacent if they correspond to two adjacent basic feasible solutions of the linear program (7.5). Gorski [83] showed that this LP formulation is appropriate in the sense of Definition 7.8. Furthermore, in Ehrgott and Klamroth [58] it is shown that every given graph $G = (V, A)$ with cost vectors $c^1, \ldots, c^p : A \to \mathbb{R}_+$ can be extended in such a way that the adjacency graph of the MSPP on the extended graph is not connected.

For this problem, a combinatorial definition of adjacency can be derived which is equivalent to the MILP-based definition. Paths are associated with flows and the residual flow of two paths is used to decide whether they are adjacent. A shortest path P_1 is adjacent to a shortest path P_2 if the symmetric difference of their edge set in the residual graph corresponds to a single cycle. Note that these definitions are canonical extensions of the single objective case in the sense of Section 7.1.2.

In Figure 7.2 the digraph used in Ehrgott and Klamroth [58] is depicted. All efficient paths are listed in Table 7.1 together with their cost values. It is easy to verify that P_8 is not connected to any other efficient path. The adjacency graph has two connected components, $\{P_8\}$ being a singleton and $\{P_i : 1 \leq i \leq 12, i \neq 8\}$. This implies the following result.

Theorem 7.15 (Ehrgott and Klamroth [58]) *The adjacency graphs of efficient shortest paths are non-connected in general.*

In the example of Ehrgott and Klamroth [58] the set of weakly efficient solutions is connected. Consequently, an extension of the adjacency graph to *weakly* efficient

Efficient Path	Interm. Nodes			Objective Vector
P_1	s_{13}	$\boxed{s_{22}}$	s_{31}	$(1, 28)$
P_2	s_{13}	$\boxed{s_{22}}$	s_{33}	$(2, 24)$
P_3	s_{13}	s_{23}	s_{31}	$(8, 22)$
P_4	s_{13}	s_{23}	s_{33}	$(9, 18)$
P_5	s_{13}	s_{21}	s_{33}	$(12, 17)$
P_6	s_{11}	s_{23}	s_{33}	$(17, 16)$
P_7	s_{11}	s_{21}	s_{33}	$(20, 15)$
P_8	s_{12}	s_{22}	s_{32}	$(27, 14)$
P_9	s_{13}	s_{23}	$\boxed{s_{32}}$	$(28, 9)$
P_{10}	s_{13}	s_{21}	$\boxed{s_{32}}$	$(31, 8)$
P_{11}	s_{11}	s_{23}	$\boxed{s_{32}}$	$(36, 7)$
P_{12}	s_{11}	s_{21}	$\boxed{s_{32}}$	$(39, 6)$

Table 7.1: *All efficient paths of the graph depicted in Figure 7.2 from Ehrgott and Klamroth [58]. Edges of paths which are common with the edges of P_8 are marked with a box.*

solutions may lead to a universally valid positive result concerning the connectedness of the weakly efficient set for these special type of combinatorial problem. However, a slight modification of the previous example, depicted in Figure 7.3, proves that this extension also does not result in a connected adjacency graph in general. The resulting efficient spanning trees are depicted in Figure 7.4. We state:

Theorem 7.16 *The adjacency graphs of weakly efficient shortest paths are non-connected in general.*

In all examples so far, only two connected components of the adjacency graphs exist. One of them consists of a single element, while the second comprises all other (weakly) efficient solutions. Yet in general, we can derive the following structural property.

Theorem 7.17 *In general, the number of connected components and the cardinality of the components are exponentially large in the size of the input data.*

Proof: Suppose we have k copies of the graph shown in Figure 7.3. The cost vectors of copy k are multiplied by the factor 100^k. These k copies are connected sequentially by connecting node s_4 of copy i, $i = 1, \ldots, k-1$, with node s_1 of copy $i+1$ using an edge with costs $(0, 0)$. The resulting adjacency graph has $(19 \cdot k - 1)$ edges and 2^k different connected components. The largest component subsumes 11^k efficient solutions, the second largest 11^{k-1} efficient solutions, and so on. □

Note that Ruzika [184] developed an efficient branch & bound algorithm for the weight constrained shortest path problem that is based on the connectedness of the supported efficient solutions of the associated biobjective problem formulation (cf. also Chapter 6 for this concept).

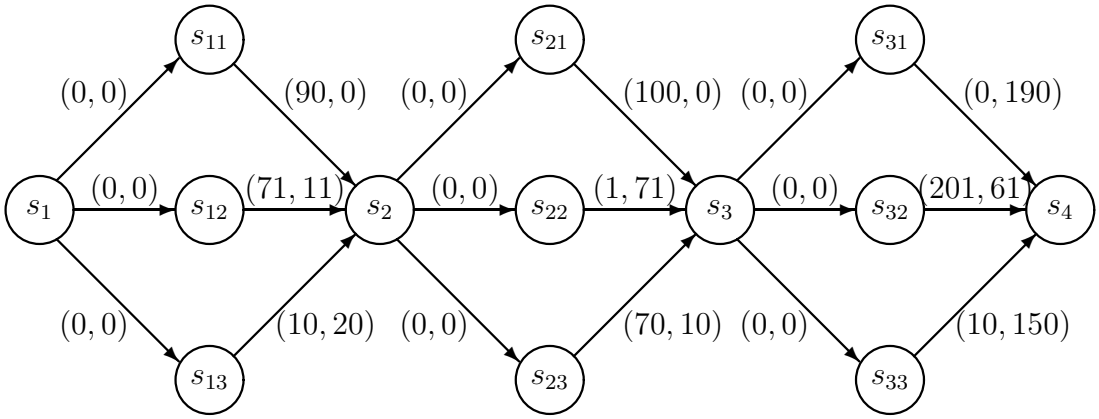


Figure 7.3: *Modified digraph from Ehrgott and Klamroth [58].*

7.2.2 Minimum Spanning Tree Problems

Let $G = (V, A)$ be an undirected graph with $|V| = n$ nodes, and denote by $A(S) := \{a = [i, j] \in A : i, j \in S\}$ the subset of edges in the subgraph of G induced by $S \subseteq V$. The multiple objective spanning tree problem (MSTP) can be formulated as

$$\begin{aligned} & \min(c^1 x, \ldots, c^p x)^T \\ \text{s.t.} \quad & \sum_{a \in A} x_a = n - 1 \\ & \sum_{a \in A(S)} x_a \leq |S| - 1 \quad \forall\, S \subseteq V \\ & x_a \in \{0, 1\}. \end{aligned} \tag{7.6}$$

In Ehrgott and Klamroth [58] a combinatorial definition of adjacency for efficient spanning trees was considered. Two spanning trees are said to be adjacent if they have $n-2$ edges in common. Non-connectivity of the adjacency graph was proven once more by means of the graph from Figure 7.2, as there exists a one-to-one correspondence between efficient shortest paths and efficient spanning trees of this specific problem. It was also shown that every given graph can be extended in such a way that the adjacency graph for the multiple objective spanning tree problem in the new graph is non-connected. Gorski [83] showed that the MILP formulation (7.6) is appropriate in the sense of Definition 7.8. The counter-example of Ehrgott and Klamroth [58] was used to prove that the adjacency graph of MSTP is also non-connected in this case. Since there is a one-to-one correspondence between the efficient shortest paths and efficient spanning trees for the example given in Theorem 7.16 (see Figure 7.3), the above results can be generalized similar to Theorem 7.17, using the same extensions of the original example:

Corollary 7.18 *The adjacency graph of (weakly) efficient spanning trees is in general non-connected. Its number of connected components and the number of nodes in these components can be exponentially large in the size of the input data.*

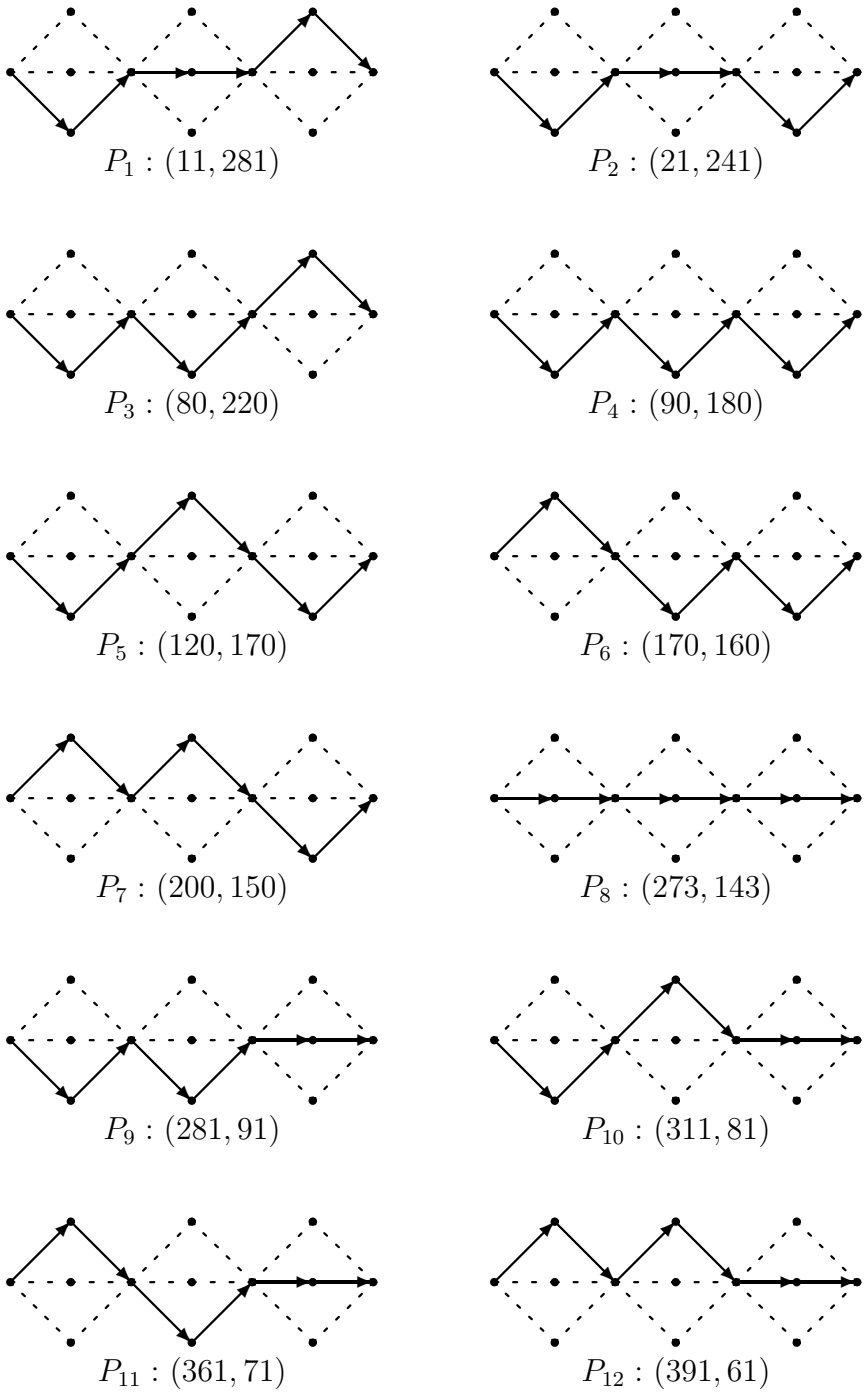


Figure 7.4: *All efficient shortest paths for the example shown in Figure 7.3 and their objective vectors.*

Note that for the spanning tree problem there exists a subclass of problems where the adjacency graph for both the combinatorial and the MILP-based definition of adjacency is always connected. This subclass is the set of all graphs which contain exactly one cycle. In addition, Ruzika [184] used the connectedness of the supported efficient spanning trees to derive an efficient branch & bound scheme for solving the weight constrained minimum spanning tree problem based on the associated biobjective problem formulation (cf. also Chapter 6 for this concept).
Moreover, we refer to Chapter 10, where, amongst others, the biobjective spanning tree problem with a binary cost objective is discussed. We prove that the set of efficient solutions for this specific problem is always connected with respect to the above stated combinatorial definition of adjacency (cf. Corollary 10.15).

7.2.3 Minimum Cost Flow Problems

Let $G = (V, A)$ be a directed graph with capacities $u_{ij} \geq 0$ for every edge $(i, j) \in A$ and supply / demand values b_i for every node $i \in V$. The multiple objective minimum cost flow problem (MCFP) can be formulated as

$$\begin{aligned} & \min(c^1 x, \ldots, c^p x)^T & & \\ \text{s.t.} \quad & \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b_i & & \forall\, i \in N \\ & 0 \leq \; x_{ij} \leq u_{ij} & & \forall\, (i,j) \in A. \end{aligned} \tag{7.7}$$

For the MCFP two efficient solutions are said to be adjacent if there exists a pivot operation between two bases corresponding to these solutions or, equivalently, if two spanning trees representing the solutions exist which differ by one edge only. This definition of adjacency is an extension of the definition for the shortest path problem and the spanning tree problem. Using the counter-example of Ehrgott and Klamroth [58] and arguing that the shortest path problem is a particular case of the minimum cost flow problem, Przybylski et al. [168] conclude that the adjacency graph of the minimum cost flow problem is not connected in general.

7.2.4 Optimization Problems on Matroids

A natural, combinatorial definition of adjacency for matroids is to call two solutions (consisting of n elements each) adjacent if they have $n - 1$ elements in common. Since the MSTP is an example for a multiple objective minimization problem on a matroid for which we have shown non-connectedness with respect to this definition of adjacency in Section 7.2.1, we can conclude that the adjacency graph of such problems is in general non-connected.
However, we prove in Chapter 10 that the efficient set of the biobjective matroid problem with an arbitrary and a binary sum objective is always connected, based on a combinatorial definition of adjacency (cf. Corollary 10.15).

7.2.5 Binary Knapsack Problems

For the binary knapsack problem some results concerning the connectedness of the set of efficient solutions can be found in the recent literature. In da Silva et al. [43], three

different models of binary knapsack problems were studied and some connectedness results using an MILP-based definition of adjacency were presented for very specific problem classes. O'Sullivan and Walker [155] proposed two algorithms for the equally-weighted biobjective knapsack problem using a combinatorial definition of adjacency. These algorithms are only guaranteed to find the set of all efficient solutions under the assumption that this set is connected. We review the ideas of these two papers and show that the set of efficient solutions is in general non-connected neither in the sense of adjacency in da Silva et al. [43] nor in the sense of adjacency in O'Sullivan and Walker [155].
We consider a special class of binary knapsack problems with equal weights and bounded cardinality, i.e.,

$$\begin{aligned} &\max\,(c^1x, c^2x)^T \\ \text{s.t.}\quad &\sum_{i=1}^{n} x_i = k \\ &x_i \in \{0,1\} \qquad i = 1\ldots,n, \end{aligned} \tag{7.8}$$

where $c_i^j \geq 0$ represents the value of item i on criterion j, $k \in \mathbb{N}$ with $k \leq n$ denotes the number of items that can be selected, and variables $x_i = 1$ if and only if item i is included in the knapsack. Let $KP(n,k)$ denote an instance of Problem (7.8). Obviously, this problem has $\binom{n}{k}$ feasible solutions. As mentioned in da Silva et al. [43],Problem (7.8) can be relaxed to the case that at most k items have to be chosen. Since all item values are non-negative, every efficient solution will have maximum cardinality.
We start our analysis with a combinatorial definition of adjacency which is also used in O'Sullivan and Walker [155].

Definition 7.19 *Two efficient knapsacks $x = (x_1,\ldots,x_n)^T$ and $x' = (x'_1,\ldots,x'_n)^T$ of $KP(n,k)$ are called* adjacent *if x' can be obtained from x by replacing one item in x with one item of x' which is not contained in x.*

Note that this elementary move is canonical. Two efficient knapsacks x and x' are adjacent if and only if $\sum_{i=1}^{n} |x_i - x'_i| = 2$, i.e. if their Hamming distance is 2. For $n \in \{1,2,3,4\}$ or $k \in \{0,1,n-1,n\}$ it is easy to see that $KP(n,k)$ has a connected adjacency graph.

Lemma 7.20 *The adjacency graph of $KP(n,k)$ is connected for $n \in \{1,2,3,4\}$ or $k \in \{0,1,n-1,n\}$.*

In da Silva et al. [43] another sufficient condition yielding a connected adjacency graph is specified.

Theorem 7.21 (da Silva et al. [43]) *Let an instance $KP(n,k)$ be given such that $c_i^1 + c_i^2 = \alpha$ for all $i = 1,\ldots,n$ and for some $\alpha \in \mathbb{N}$. Then all $\binom{n}{k}$ feasible solutions are efficient solutions of (7.8) and hence, the adjacency graph of the problem is connected.*

Unfortunately, this connectedness result is no longer valid for the general case.

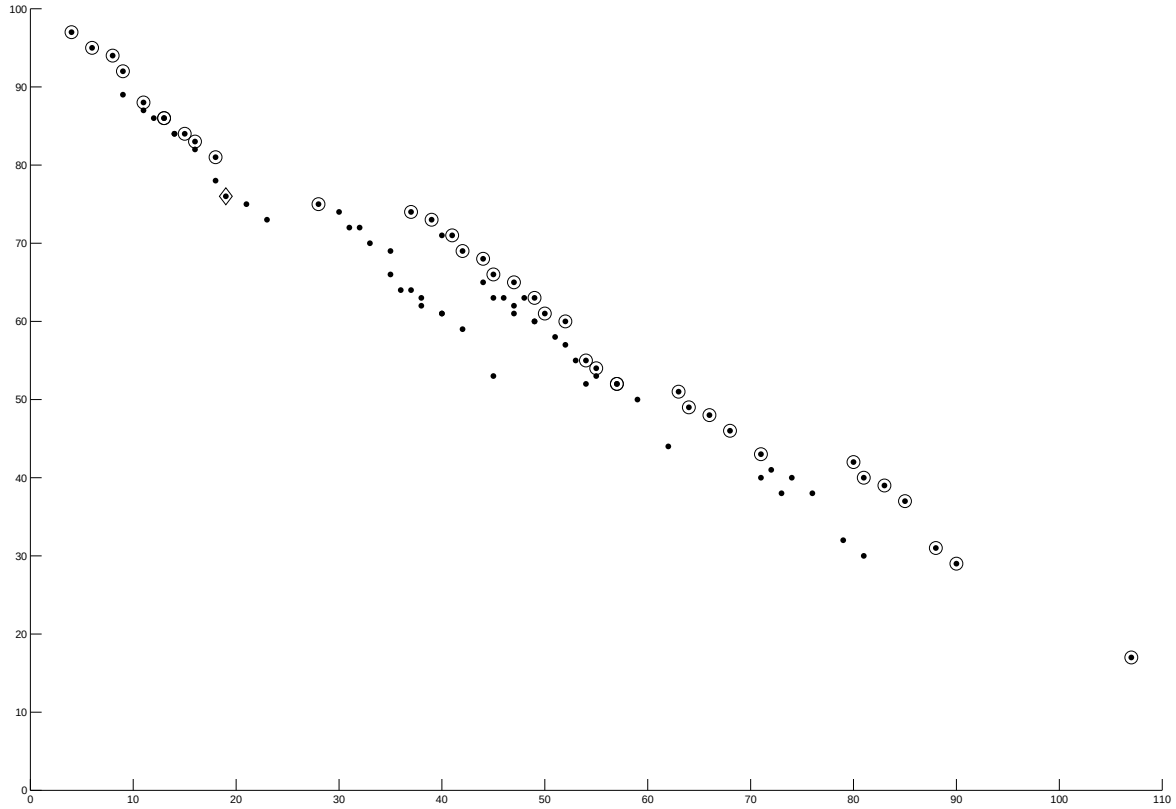


Figure 7.5: *Image of the feasible set of the counter-example used in the proof of Theorem 7.22. The non-dominated set consists of two connected components, one indicated by circles, the other - a singleton - indicated by a diamond.*

Theorem 7.22 *The adjacency graph of a binary knapsack problem of the form* (7.8) *with adjacency defined as in Definition 7.19 is non-connected in general.*

Proof: Consider $KP(9,3)$ with the objective function vectors

$$\begin{pmatrix} c^1 \\ c^2 \end{pmatrix} = \begin{pmatrix} 44 & 36 & 27 & 10 & 8 & 5 & 3 & 1 & 0 \\ 0 & 8 & 9 & 21 & 23 & 29 & 31 & 32 & 34 \end{pmatrix}.$$

The problem has 84 feasible and 38 efficient solutions (cf. Figure 7.5). All efficient solutions $S_1, \ldots, S_{38}$ and their corresponding objective function vectors are listed in Table 7.2. Using the plotted boxes it is easy to verify that the efficient solution S_{11} is not adjacent to any other solution in the sense of Definition 7.19. Consequently, the adjacency graph of the given problem which can be seen in Figure 7.6 is non-connected. □

Note that the given counter-example in Theorem 7.22 is minimal in the sense that deleting any combination of profit vectors from the problem always leads to a connected adjacency graph, assuming that $k = 3$. We conclude:

Corollary 7.23 *The algorithms proposed by O'Sullivan and Walker [155] for solving the binary knapsack problem with equal weights and bounded cardinality fail to compute the set of efficient solutions in general.*

C	x	x									S
4	97	0	0	0	0	0	0	1	1	1	S_1
6	95	0	0	0	0	0	1	0	1	1	S_2
8	94	0	0	0	0	0	1	1	0	1	S_3
9	92	0	0	0	0	0	1	1	1	0	S_4
11	88	0	0	0	0	1	0	1	0	1	S_5
13	86	0	0	0	1	0	0	1	0	1	S_6
13	86	0	0	0	0	1	1	0	0	1	S_7
15	84	0	0	0	1	0	1	0	0	1	S_8
16	83	0	0	0	0	1	1	1	0	0	S_9
18	81	0	0	0	1	0	1	1	0	0	S_{10}
19	76	0	0	0	1	1	0	0	1	0	S_{11}
28	75	0	0	1	0	0	0	0	1	1	S_{12}
37	74	0	1	0	0	0	0	0	1	1	S_{13}
39	73	0	1	0	0	0	0	1	0	1	S_{14}
41	71	0	1	0	0	0	1	0	0	1	S_{15}
42	69	0	1	0	0	0	1	0	1	0	S_{16}
44	68	0	1	0	0	0	1	1	0	0	S_{17}
45	66	1	0	0	0	0	0	0	1	1	S_{18}
47	65	1	0	0	0	0	0	1	0	1	S_{19}
49	63	1	0	0	0	0	1	0	0	1	S_{20}
50	61	1	0	0	0	0	1	0	1	0	S_{21}
52	60	1	0	0	0	0	1	1	0	0	S_{22}
54	55	1	0	0	1	0	0	0	0	1	S_{23}
55	54	1	0	0	0	1	0	1	0	0	S_{24}
57	52	1	0	0	1	0	0	1	0	0	S_{25}
57	52	1	0	0	0	1	1	0	0	0	S_{26}
63	51	0	1	1	0	0	0	0	0	1	S_{27}
64	49	0	1	1	0	0	0	0	1	0	S_{28}
66	48	0	1	1	0	0	0	1	0	0	S_{29}
68	46	0	1	1	0	0	1	0	0	0	S_{30}
71	43	1	0	1	0	0	0	0	0	1	S_{31}
80	42	1	1	0	0	0	0	0	0	1	S_{32}
81	40	1	1	0	0	0	0	0	1	0	S_{33}
83	39	1	1	0	0	0	0	1	0	0	S_{34}
85	37	1	1	0	0	0	1	0	0	0	S_{35}
88	31	1	1	0	0	1	0	0	0	0	S_{36}
90	29	1	1	0	1	0	0	0	0	0	S_{37}
107	17	1	1	1	0	0	0	0	0	0	S_{38}

Table 7.2: *All efficient solutions of the example used in the proof of Theorem 7.22.*

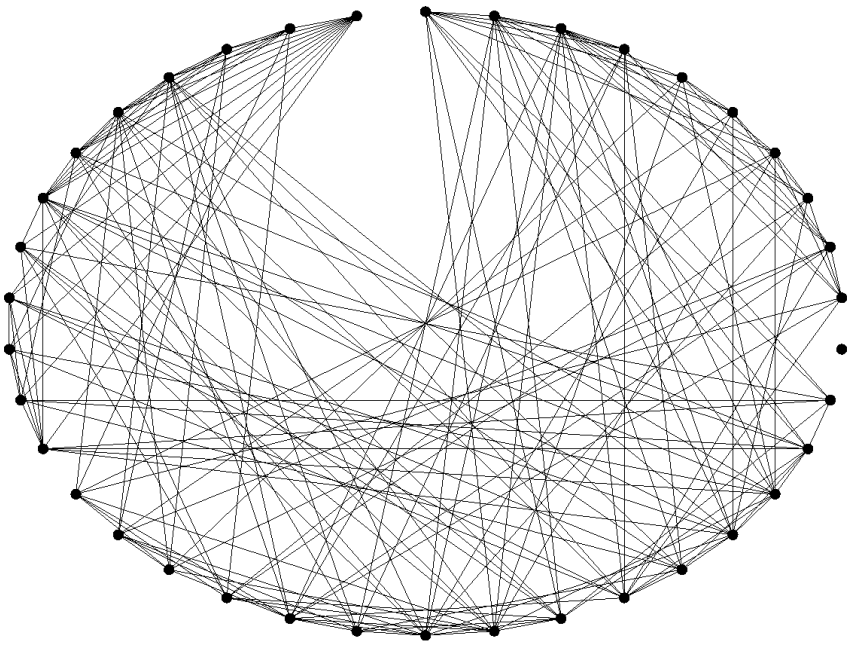


Figure 7.6: *Adjacency graph of the non-connected example problem used in the proof of Theorem 7.22.*

In Section 7.3, we report about numerical results indicating the likelihood that a non-connected adjacency graph of Problem (7.8) appears in randomly generated instances. Note that for these investigations problems $KP(n,k)$ with $k > \frac{n}{2}$ are not of interest as they can be transformed into an equivalent knapsack problem where the decision is which objects to leave out of a knapsack. The resulting problem can be interpreted as $KP(n,\tilde{k})$ for $\tilde{k} \leq \frac{n}{2}$.
Next, we concentrate on the MILP-based definition of adjacency which is considered in da Silva et al. [43]. Since the MILP formulation (7.8) is canonical it can be extended to an appropriate MILP formulation using the proof of Lemma 7.9. Let $P := \{x \in [0,1]^n : \sum_{i=1}^n x_i = k\}$ denote the feasible set of the LP relaxation of (7.8).

Lemma 7.24 *Let $n \geq 5$. Two extreme points u and v of the binary knapsack polytope $P = \{x \in [0,1]^n : \sum_{i=1}^n x_i = k\}$ are connected by an edge if and only if u and v are adjacent in the sense of Definition 7.19.*

Proof: According to Geist and Rodin [73] it suffices to show that two extreme points u and v of P are connected by an edge if and only if there does not exist two other extreme points w^1 and w^2 of P, i.e., other feasible solutions of $KP(n,k)$, such that

$$\frac{1}{2}(w^1 + w^2) = \frac{1}{2}(u + v). \tag{7.9}$$

First, let two feasible solutions u and v of $KP(n,k)$ be given that are adjacent according to Definition 7.19. By definition they differ in exactly one item in the knapsack. Without loss of generality we assume that $u_1 = v_2 = 1$, $u_2 = v_1 = 0$ and $u_i = v_i$ for all $i = 3, \ldots, n$. Suppose u and v are not connected by an edge in P, i.e., there exist two other feasible solutions w^1 and w^2 satisfying equation (7.9). Since u and v are equal starting from the third component and thus $u_i = v_i = 0$ or $u_i = v_i = 1$

for $i = 3, \ldots, n$, $\frac{1}{2}(u_i + v_i)$ equals either 0 or 1 and hence $w_i^1 = w_i^2 = u_i = v_i$ for all $i = 3, \ldots, n$ must hold to satisfy (7.9). So, w^1 and w^2 can differ from u and v only in the first two components which means that either $w_1^j = w_2^j = 0$ or $w_1^j = w_2^j = 1$ ($j \in \{1, 2\}$), which is impossible due to the constraint $\sum_{i=1}^{n} w_i^j = k$. Hence, u and v are connected by an edge in P.
Now, let u and v be not adjacent solutions in the sense of Definition 7.19. Then u and v differ in at least two different items in each knapsack. Without loss of generality we assume that the first and the second item is contained in u but not in v and the third and the fourth item is contained v but not in u. We define

$$w_i^1 = \begin{cases} 1\,, \text{ if } i \in \{1,3\} \\ 0\,, \text{ if } i \in \{2,4\} \\ u_i, \text{ if } i \geq 5 \end{cases} \quad \text{and } w_i^2 = \begin{cases} 1\,, \text{ if } i \in \{2,4\} \\ 0\,, \text{ if } i \in \{1,3\} \\ v_i, \text{ if } i \geq 5. \end{cases}$$

Then, w^1 and w^2 are feasible and both different from u and v. Equation (7.9) is satisfied and hence u and v are not connected by an edge in P. □

According to Lemma 7.24, the adjacency structure of the efficient extreme points of P coincides with the adjacency structure induced by Definition 7.19. Hence, the adjacency graph with respect to the appropriate MILP formulation based on Problem (7.8) and the adjacency graph resulting from Definition 7.19 are the same (cf. Theorem 7.14). Thus, Theorem 7.22 immediately implies the following result.

Corollary 7.25 *In general, the set of efficient solutions of $KP(n, k)$ is non-connected with respect to the appropriate MILP formulation based on Problem* (7.8).

Finally, we investigate a combinatorial definition of adjacency for another variant of the knapsack problem, the so-called *binary multiple choice knapsack problem with equal weights.*

$$\begin{aligned} & \max \left(\sum_{i=1}^{n} \sum_{j=1}^{k_i} c_{ij}^1 x_{ij}, \sum_{i=1}^{n} \sum_{j=1}^{k_i} c_{ij}^2 x_{ij} \right)^T \\ \text{s.t.} \quad & \sum_{j=1}^{k_i} x_{ij} = 1, \qquad i = 1, \ldots, n, \\ & x_{ij} \in \{0,1\}, \quad i = 1 \ldots, n,\ j = 1, \ldots, k_i. \end{aligned} \tag{7.10}$$

The given problem can be interpreted as follows: Given n disjoint baskets $B_1, \ldots, B_n$ each having exactly k_i items, the objective is to maximize the overall profit with the restriction that exactly one item is chosen from each basket. Problem (7.10) is a more structured knapsack problem compared to Problem (7.8) since items cannot be combined arbitrarily. We consider the following combinatorial definition of adjacency.

Definition 7.26 *Two efficient knapsacks x and x' of the binary multiple choice knapsack problem with equal weights are called* adjacent *if x' and x differ in one item in exactly one basket B_i for an $i \in \{1, \ldots, n\}$.*

This definition of adjacency is again canonical since for single objective problems, any maximal knapsack must contain an item with maximal profit from each basket.

Alternative optimal solutions may exist if at least one basket contains more than one item with maximal profit. All these optimal solutions are adjacent in the sense of Definition 7.26.
In the multiple objective case the situation is, however, different. The counter-example from Ehrgott and Klamroth [58] and its modification in Subsection 7.2.1 can be used to establish the following result.

Theorem 7.27 *The adjacency graph of (weakly) efficient solutions for the binary multiple choice knapsack problem with equal weights, where adjacency of two efficient solutions is defined according to Definition 7.26, is non-connected in general.*

Proof: In the counter-example for the MSPP given in the proof of Theorem 7.16 we redefine the cost vectors c_{ij} of the three paths from node s_i to node s_{i+1}, $i = 1, 2, 3$, via s_{ij}, $j = 1, 2, 3$, by setting

$$\tilde{c}_{ij}^q = \max\{c_{ij}^q : i, j = 1, 2, 3;\, q = 1, 2\} - c_{ij}^q$$

for $i, j = 1, 2, 3$ and $q = 1, 2$ and interpret the resulting vectors of the three paths from the node s_i to node s_{i+1} as profit vectors for basket B_i, $i = 1, 2, 3$. This results in the three baskets

$$B_1 = \left\{ \begin{pmatrix} 111 \\ 201 \end{pmatrix}, \begin{pmatrix} 130 \\ 190 \end{pmatrix}, \begin{pmatrix} 191 \\ 181 \end{pmatrix} \right\}, \quad B_2 = \left\{ \begin{pmatrix} 101 \\ 201 \end{pmatrix}, \begin{pmatrix} 200 \\ 130 \end{pmatrix}, \begin{pmatrix} 131 \\ 191 \end{pmatrix} \right\},$$
$$B_3 = \left\{ \begin{pmatrix} 201 \\ 11 \end{pmatrix}, \begin{pmatrix} 0 \\ 140 \end{pmatrix}, \begin{pmatrix} 191 \\ 51 \end{pmatrix} \right\}.$$

Since we have transformed the minimization problem into a maximization problem by taking the negative value of each cost vector followed by a shift of these vectors by an amount of $\max\{c_{ij}^q\} = 201$, there is a one-to-one correspondence between the efficient solutions of the modified problem and the efficient solutions of the counter-example considered in Theorem 7.16. The profit vectors of the resulting solutions $K_1, \ldots, K_{12}$ are given by $(603, 603)^T - c(P_i)$ where $c(P_i)$ corresponds to the cost vector of P_i in Figure 7.4 for $i = 1, \ldots, 12$. Items in at least two baskets have to be exchanged when transforming K_8 into K_j, $j \neq 8$ by elementary moves. Hence, K_8 is not adjacent to any other (weakly) efficient solution in the sense of Definition 7.26. □

Note that, since there is a one-to-one correspondence between the example used in the proof of Theorem 7.27 and the example given in Theorem 7.16 (see Figure 7.3), the above result can be generalized similar to Theorem 7.17, using the same extension of the original example:

Corollary 7.28 *In general, the number of connected components and the cardinality of the components in the adjacency graph of a binary multiple choice knapsack problem with equal weights, where adjacency of two efficient solutions is defined according to Definition 7.26, are exponentially large in the size of the input data.*

In Section 7.3.2 we additionally investigate the frequency with which a non-connected adjacency graph for problem (7.10) occurs empirically in randomly generated instances.

7.2.6 General Knapsack Problems

Since the general knapsack problem subsumes the binary knapsack problem with bounded cardinality discussed above as a special case, the general knapsack problem is in general non-connected as well if connectedness is defined, for example, based on elementary moves similar to Section 7.2.5 above.

7.2.7 Integer Programming Problems with Fixed or Bounded Cardinalities

The same reasoning as in Section 7.2.6 applies.

7.2.8 Unconstrained Binary Optimization Problems

Since, in general, the adjacency graph for the binary knapsack problem with equal weights and bounded cardinality is non-connected for well-established definitions of adjacency of efficient solutions (see Section 7.2.5), we focus on unconstrained binary problems in this subsection since these problems possess even less structure. Formally, an unconstrained binary problem is defined as follows:

$$\begin{aligned} & \max \left(c^1 x, c^2 x\right)^T \\ \text{s.t.} \quad & x_i \in \{0,1\}, \quad i = 1, \ldots, n. \end{aligned} \tag{7.11}$$

We assume without loss of generality that $c_i^1 \cdot c_i^2 < 0$ (but not necessarily $c_i^1 < 0$ and $c_i^2 > 0$) for all $i = 1, \ldots, n$. Otherwise either $x_i = 0$ or $x_i = 1$ in every efficient solution. For Problem (7.11), the number of non-zero variables is not fixed, and hence, also not known in advance. Consequently, an appropriate notion of adjacency is not evident. Nevertheless, Definition 7.26 can be transferred to this problem, considering the following modified version of Problem (7.11).

$$\begin{aligned} & \max \left(c^1 x,\, c^2 x\right)^T \\ \text{s.t.} \quad & x_i + y_i = 1, \qquad i = 1, \ldots, n \\ & x_i, y_i \in \{0,1\}, \quad i = 1, \ldots, n. \end{aligned} \tag{7.12}$$

Clearly, either $x_i = 1$ or $y_i = 1$ holds, and $\sum_{i=1}^{n}(x_i + y_i) = n$. As a consequence, the number of non-zero components of the feasible solution vector $(x, y)^T$ is exactly n. This implies that solutions of Problem (7.12) are of the same cardinality. However, the notion of adjacency for binary knapsack problems with fixed cardinality does not apply directly to Problem (7.12), since the values of x_i and y_i, $i = 1, \ldots, n$, cannot be chosen independently as they are coupled by a side constraint. By introducing additional zero cost vectors for each y_i, $i = 1, \ldots, n$, Problem (7.12) can be interpreted as a binary multiple choice knapsack problem with equal weights where either x_i or y_i has to be included in the knapsack, $i = 1, \ldots, n$. Hence, Definition 7.26 can be applied to Problem (7.12). Since this definition of adjacency for the extended problem results in single '1-to-0' or '0-to-1' swaps in exactly one x_i for Problem (7.11), we define:

Definition 7.29 *Two efficient solutions x and x' of the unconstrained binary problem are called* adjacent *if they differ in exactly one component, i.e. if* $\sum_{i=1}^{n} |x_i - x'_i| = 1$.

If we extend the last definition to all 2^n feasible solutions of the problem which can be identified with the set of all extreme points of the n-dimensional unit cube $W := [0,1]^n$, two feasible (efficient) solutions are adjacent if and only if they are connected by an edge in W. But since W in combination with Problem (7.11) can be easily modeled by an appropriate MILP formulation, the adjacency graph which results from Definition 7.29 coincides with the adjacency graph of this appropriate MILP formulation by Theorem 7.14. We state.

Theorem 7.30 *The adjacency graph of an unconstrained binary problem of given by Problem* (7.11)*, where adjacency of two efficient solutions is defined according to Definition 7.29, is non-connected in general.*

Proof: Consider the following unconstrained binary problem with objective matrix

$$C = \begin{pmatrix} -126 & -121 & -120 & -103 & -100 & -97 & -17 & -13 \\ 100 & 94 & 90 & 74 & 73 & 68 & 23 & 7 \end{pmatrix}.$$

The set of all efficient solutions of this problem consists of 110 vectors. It can be shown that the efficient solution $x = (0,1,0,1,1,1,0,1)^T$ with objective value $Cx = (-434, 316)^T$ is not adjacent to any other efficient solution in the sense of Definition 7.29. □

Note that the given counter-example in Theorem 7.30 is minimal in the sense that deleting any combination of profit vectors from the problem always leads to a connected adjacency graph.

Corollary 7.31 *Let an appropriate MILP formulation of problem* (7.11) *be given where the polytope of the LP relaxation describes the n-dimensional unit cube $[0,1]^n$. Then, the adjacency graph of the unconstrained binary problem with respect to the given MILP formulation is non-connected in general.*

While the efficient set for Problem (7.11) is not connected in general, we refer to Chapter 9, where the triobjective unconstrained optimization problem with two binary objectives is discussed. In Section 9.5 we prove that the efficient set of this specific type of problem is always connected, based on a combinatorial definition of adjacency (cf. Theorem 9.33).

7.2.9 Linear Assignment Problems

We consider two definitions of adjacency for the linear assignment problem: A combinatorial definition based on swapping rows in the assignment matrix, and an MILP-based definition of adjacency. The biobjective linear assignment problem can be for-

mulated as

$$\begin{aligned} \min \quad & \left(\sum_{i,j=1}^{n} c_{ij}^1 x_{ij}, \ \sum_{i,j=1}^{n} c_{ij}^2 x_{ij} \right)^T \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, \ldots, n, \\ & \sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots, n, \\ & x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n, \end{aligned} \tag{7.13}$$

with objective coefficients $c_{ij}^1, c_{ij}^2 \geq 0$ for all $i, j = 1, \ldots, n$.
First, we consider an intuitive combinatorial definition of adjacency based on a simple swap of two rows of the assignment matrix. This definition is not canonical, i.e., it does not yield a connected graph of optimal solutions for the single objective version of the problem:

Theorem 7.32 *Swapping two rows of the assignment matrix of a single objective linear assignment problem without changing the objective value does in general not permit to construct the complete set of optimal solutions starting from an arbitrary optimal solution.*

Proof: Consider a single objective linear assignment problem with $n = 4$ and cost matrix

$$C = (c_{ij}^1)_{i,j=1,\ldots,n} = \begin{pmatrix} 1 & \infty & 1 & \infty \\ 1 & 1 & \infty & \infty \\ \infty & \infty & 1 & 1 \\ \infty & 1 & \infty & 1 \end{pmatrix}.$$

This problem has two optimal assignments with value 4:
Assignment 1: $x_{11} = x_{22} = x_{33} = x_{44} = 1$ and $x_{ij} = 0$ otherwise.
Assignment 2: $x_{13} = x_{21} = x_{34} = x_{42} = 1$ and $x_{ij} = 0$ otherwise.
Clearly, these assignments cannot be obtained from each other by a single row swap. □

Turning our attention to an MILP-based definition of adjacency using Problem (7.13), we observe that the biobjective linear assignment problem is a special case of the minimum cost flow problem (cf. Section 7.2.3). Thus, we can expect the existence of a canonical definition of adjacency in this case. The matrix describing the assignment polytope is totally unimodular (see, e.g., Nemhauser and Wolsey [150]) and, hence formulation (7.13) is an appropriate MILP formulation. To simplify the discussion, we use the following combinatorial interpretation of the resulting concept of adjacency.

Definition 7.33 *Let $G = (V_1 \cup V_2, A)$ with $|V_1| = |V_2| = n$ be a bipartite graph with edge costs $c^1, c^2 : A \to \mathbb{R}$ that models a given instance of the biobjective linear assignment problem, and let A_1 and A_2 be the edges selected in two different assignments. We call the solutions corresponding to A_1 and A_2* adjacent *if the graph induced by $A_1 \cup A_2$ contains exactly one cycle.*

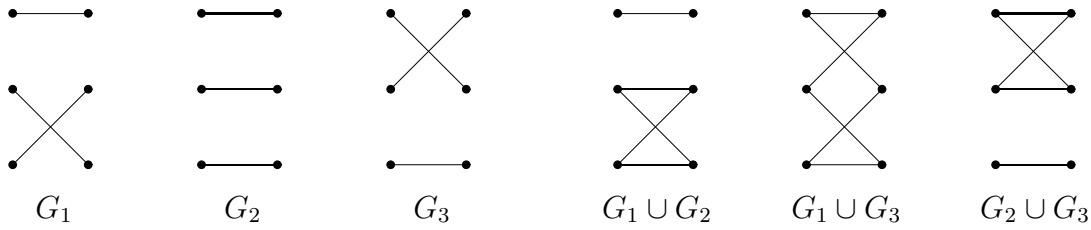


Figure 7.7: *All feasible assignments with finite costs for the subproblems S_i in the proof of Theorem 7.34 and their pairwise union.*

According to Balinski and Russakoff [10], this combinatorial definition of adjacency corresponds to the MILP-based definition of adjacency induced by the assignment polytope P. Equivalently, two assignments A_1 and A_2 are adjacent if and only if their symmetric difference $A_1 \triangle A_2 = (A_1 \cup A_2) \setminus (A_1 \cap A_2)$ consists of exactly one cycle the edges of which alternately belong to A_1 and A_2, respectively (cf.Hausmann [100]). Since any pair of vertices of the assignment polytope is connected by a path of length less than or equal to 2 (see, for example, Balinski and Russakoff [10], Hausmann [100]), the generalized adjacency graph $\mathcal{G}'_2$ of Problem (7.13) is always connected. However, if we restrict ourselves to direct adjacency according to Definition 7.33, the adjacency graph $\mathcal{G} = \mathcal{G}'_1$ of the biobjective linear assignment problem is non-connected in general.

Theorem 7.34 *The adjacency graph $\mathcal{G}$ of (weakly) efficient solutions for the biobjective linear assignment problem using Definition 7.33 for characterizing adjacent assignments is not connected in general.*

Proof: We restructure the counter-example for the MSPP given in the proof of Theorem 7.16. Consider the six cost-submatrices of a (9×9) biobjective linear assignment problem given by

$$C^1_{(1:3,1:3)} = \begin{pmatrix} 0 & 0 & \infty \\ 10 & 71 & 0 \\ \infty & 90 & 0 \end{pmatrix}, \quad C^1_{(4:6,4:6)} = \begin{pmatrix} 0 & 0 & \infty \\ 70 & 1 & 0 \\ \infty & 100 & 0 \end{pmatrix}, \quad C^1_{(7:9,7:9)} = \begin{pmatrix} 0 & 0 & \infty \\ 10 & 201 & 0 \\ \infty & 0 & 0 \end{pmatrix},$$

$$C^2_{(1:3,1:3)} = \begin{pmatrix} 0 & 0 & \infty \\ 20 & 11 & 0 \\ \infty & 0 & 0 \end{pmatrix}, \quad C^2_{(4:6,4:6)} = \begin{pmatrix} 0 & 0 & \infty \\ 10 & 71 & 0 \\ \infty & 0 & 0 \end{pmatrix}, \quad C^2_{(7:9,7:9)} = \begin{pmatrix} 0 & 0 & \infty \\ 150 & 61 & 0 \\ \infty & 190 & 0 \end{pmatrix},$$

and let all remaining cost coefficients be set to infinity. This problem decomposes into three (3×3)-subproblems denoted by S_1, S_2 and S_3, where each subproblem S_i has three solutions G_1, G_2 and G_3 that have finite costs in both objectives. These three solutions have the same structure for all three subproblems and are depicted in Figure 7.7. Note that the cost vector of each solution G_j of subproblem S_i is chosen such that it corresponds to the cost vector of the path connecting node s_i with node s_{i+1} via node s_{ij} in Figure 7.3. Consequently, there is a one-to-one correspondence between the efficient solutions of this instance of the biobjective linear assignment problem and the efficient solutions of the biobjective shortest path problem shown in Figure 7.4.

From Figure 7.7 it can be seen that the pairwise union of two subgraphs G_i and G_j, $i \neq j$, contains exactly one cycle. According to Definition 7.33, two efficient assignments of the overall problem are thus adjacent if and only if they differ in exactly one subproblem S_i, $i \in \{1, 2, 3\}$. Since the efficient path P_8 in Figure 7.4 differs from all other efficient paths in at least two connections, the corresponding assignment (consisting of G_2 in all three subproblems) differs from all other efficient assignments in at least two subproblems and is thus not adjacent to any other efficient assignment of the overall problem. □

Since, similar to the case of the binary multiple choice knapsack problem with equal weights, there is a one-to-one correspondence between the example used in the proof of Theorem 7.34 and the example given in Theorem 7.16, this example can be used to generalize the above stated result similar to Theorem 7.17, using again the same extension of the original example.

Corollary 7.35 *In general, the number of connected components and the cardinality of the components in the adjacency graph of the biobjective linear assignment problem, where adjacency of two efficient solutions is defined according to Definition 7.33, are exponentially large in the size of the input data.*

7.2.10 Transportation and Transshipment Problem

Since this problem can be interpreted as a special case of the linear assignment problem (see, e.g., Ehrgott and Gandibleux [55]), the non-connectedness result of Section 7.2.9 can be transferred.

7.2.11 The Traveling Salesman Problem

Paquete et al. [158] and Paquete and Stützle [161] introduced a combinatorial definition of adjacency for the MTSP: Two feasible tours of the MTSP are called adjacent if they differ in exactly four edges. Note that this definition corresponds to the 2-edge-exchange neighborhood of the TSP. However, the 2-edge-exchange neighborhood does not induce a canonical definition of adjacency.

Theorem 7.36 *The set of optimal solutions of the single objective TSP is in general non-connected with respect to the 2-edge-exchange neighborhood.*

Proof: Consider the following instance of a symmetric TSP with five nodes and distance matrix D given by

$$D = \begin{pmatrix} 0 & 7 & 10 & 3 & 1001 \\ 7 & 0 & 3 & 10 & 1000 \\ 10 & 3 & 0 & 7 & 1000 \\ 3 & 10 & 7 & 0 & 1001 \\ 1001 & 1000 & 1000 & 1001 & 0 \end{pmatrix}.$$

The problem has two optimal tours $T_1 = (1, 4, 3, 2, 5, 1)$ and $T_2 = (1, 4, 5, 3, 2, 1)$ with cost $c(T_1) = c(T_2) = 2014$. However, T_1 differs from T_2 in six edges, i.e., T_1 is only

contained in the 3-edge-exchange neighborhood of T_2 and not in its 2-edge-exchange neighborhood. □

Note that the formulation of an alternative, MILP-based definition of adjacency is not immediate in the case of the biobjective TSP as long as no appropriate MILP formulation of the problem is available. Moreover, it is $\mathcal{NP}$-complete to decide whether two given vertices of the TSP-polytope are adjacent [157].

7.3 Numerical Results

All results in Section 7.2 are obtained from a worst-case analysis. Therefore, it is an interesting question how frequently the phenomenon 'non-connected adjacency graph' occurs in practice. To learn more about the practical relevance of adjacency considerations, we exemplarily conduct numerical studies for the biobjective binary knapsack problem with bounded cardinality and the biobjective binary multiple choice knapsack problem (cf. Section 7.2.5). All in all, more than six million randomly generated problem instances have been analyzed.
Before describing the design of the numerical experiments in more detail, we discuss special properties of the problems under consideration. First, it is not sufficient to compute just a single efficient solution for each non-dominated point. Instead, all efficient solutions have to be found. Consequently, an algorithm enumerating all alternative solutions for the same non-dominated outcome is required. Second, after having found all efficient solutions, their adjacency relationships have to be explored. The set of efficient solutions has to be ordered or traversed several times, efficient solutions have to be compared pairwise, and clusters of efficient solutions have to be calculated. This post-solution analysis requires a substantial amount of computation time. Third, non-connected adjacency graphs cannot be expected to occur with a high frequency in randomly generated problem instances. Conclusions can therefore not be drawn on the basis of just a few dozen instances. For each set-up thousands of instances have to be generated and tested to yield representative results. Fourth, computation power as well as computation time are limited.
Recapitulating, one can conclude that under these circumstances the instances treated in our study have to be rather small and do not nearly match the sizes of state-of-the-art benchmark problems.

7.3.1 Biobjective Binary Knapsack Problems with Bounded Cardinality

For the computation of the efficient and non-dominated set of the generated instances of the biobjective binary knapsack problem with bounded cardinality $KP(n,k)$ we used a dynamic programming approach for general multiple objective knapsack problems developed in Klamroth and Wiecek [116]. Recent numerical tests of Bazgan et al. [14] on a slightly extended version of this approach proved its efficiency for solving even large scale bi- and multiple objective binary knapsack problems.
From Klamroth and Wiecek [116], we implemented Model III for the binary case. Using this approach, we can solve k knapsack instances $KP(n,1),\ldots,KP(n,k)$ simultaneously, i.e., without any additional computational cost. For each efficient solution of

Meth.	10/9/10	20/10/10	30/15/10	40/20/10	60/30/10	80/40/20	100/50/20
1	50000	20000	1000	1000	-	-	-
2	50000	20000	1000	1000	-	-	-
3	50000	20000	1000	1000	-	-	-
4	50000	20000	1000	1000	-	-	-
5	50000	20000	1000	1000	-	-	-
6	50000	50000	50000	10000	10000	10000	1000

Table 7.3: *Setup of computational experiments for the biobjective binary knapsack problem with bounded cardinality.*

one of the problem instances, we store a binary vector representing this solution in a list. Recall from Section 7.2.5 that two efficient solutions are defined to be adjacent if their Hamming distance is equal to two. Starting with the first efficient solution in the list, we find all its neighbors (by pairwise computations of Hamming distances with all the remaining solutions) in the list and mark them with a certain label. Different labels signalize different adjacency clusters. We proceed likewise with the second solution in the list. Eventually, adjacency clusters have to be merged, i.e., markers have to be re-assigned. After having processed all efficient solutions, the number of different markers indicate the number of clusters.
The aim of the numerical study is to report the number of adjacency clusters of randomly generated instances of $KP(n,k)$ when

a) n increases,

b) k increases for fixed n, and

c) the objective coefficients are generated according to different methods.

We generated seven problem setups For each setup, we used six different methods to generate the objective coefficients. In the first row of Table 7.3, we use a scheme of the form $Pos1/Pos2/Pos3$ to code these seven setups. $Pos1$ specifies the total number n of items. The upper bound k for the right hand side parameter of the knapsack constraint is specified under $Pos2$. We determined the adjacency graph for all possible right hand sides $i \in \{1,\ldots,Pos2\}$. Finally, the coefficients of the first objective c^1 were chosen in the interval $[0,r]$, where $r = Pos1 \cdot Pos3$. The coefficients of the second objective c^2 were chosen according to six different methods which were motivated by the study of Pedersen et al. [164] and are described in the following.

Method 1: c^1 was sorted in decreasing, c^2 in increasing order to obtain pairwise non-dominated profit vectors. Weakly-dominated vectors were omitted.

Method 2: The profit vectors $p_1 := (c_1^1, c_1^2)^T = (r,0)^T$ and $p_n := (c_n^1, c_n^2)^T = (0,r)^T$ were fixed at the beginning. The remaining vectors were chosen within the triangle $(0,0)^T$, p_1 and p_n. Preferably pairwise non-dominated profit vectors were generated. Only profit matrices with a few dominated profit vectors were accepted.

Meth.	10/9/10	20/10/10	30/15/10	40/20/10	60/30/10	80/40/20	100/50/20
1	0	0	0	2	-	-	-
2	0	2	1	1	-	-	-
3	0	2	0	0	-	-	-
4	0	1	0	0	-	-	-
5	0	1	1	0	-	-	-
6	0	0	0	0	0	1	0

Table 7.4: *Number of instances with adjacency graph having more than one connected component.*

Method 3: The profit vectors were generated as in Method 2, but now within the triangle $(r, r)^T$, p_1 and p_n.

Method 4: The profit vectors p_1 and p_n were fixed like in Method 2. The remaining vectors were generated spread around the concave part of the half circle with midpoint $(0, 0)^T$ connecting the points p_1 and p_n. Preferably pairwise non-dominated profit vectors were generated and profit matrices with only a few dominated profit vectors were accepted.

Method 5: The profit vectors were generated as in Method 4, but now spread around the convex half circle with midpoint $(r, r)^T$ connecting p_1 and p_n.

Method 6: The entries of the profit matrix were generated uniformly at random.

The entries of Table 7.3 correspond to the number of instances that were processed for each of the setups. Note that these numbers are not always the same. Some setups result in more difficult instances and thus, we could only process fewer instances in a reasonable amount of time. A dash indicates that this setup was not tested due to its numerical difficulty.
Table 7.4 presents the number of non-connected adjacency graphs that were found for each setup. For the generated instances, only very few non-connected adjacency graphs were found. Nevertheless, for each of the six data generation methods at least one instance possessing a non-connected adjacency graph could be found. Based on the small number of components, there do not seem to exist significant trends - neither with respect to increasing k or n nor with respect to some particular generation method for the data.
For the case of randomly generated profit matrices (Method 6), non-connected adjacency graphs seem to occur extremely rarely. As mentioned in da Silva et al. [43], an item x_i corresponding to a dominated profit vector p_i can only be contained in an efficient knapsack if at least one of the items x_j corresponding to profit vectors p_j dominating p_i is also contained in the knapsack. The set of all efficient solutions of such a problem consists of a few number of elements and is more structured than in the case when only pairwise non-dominated profit vectors are considered. For the problem size (40/20/10), the maximum number of elements of an efficient set for a problem instance generated by Method 6 is given by 290 while for the other methods

Setup of test instances	20/5/10	20/10/10	20/15/10
Number of instances generated	10000	5000	1000
Instances having a non-connected adjacency graph	118	295	111

Table 7.5: *Setup of computational experiments for the biobjective binary multiple choice knapsack problem with uniform weights.*

the maximum number does not fall below 1392 and has a maximum value of over 5300 elements for a problem instance generated by Method 2. Unfortunately, Method 6 seems to be the "standard" way to generate data when testing an algorithm numerically. Yet, for algorithms based on neighborhood search, this problem class seems to be quite uninteresting.

7.3.2 Biobjective Binary Multiple Choice Knapsack Problems

The second part of the numerical study is devoted to the biobjective binary multiple choice knapsack problem also introduced in Section 7.2.5. Recall that this problem is closely related to the biobjective binary knapsack problem with bounded cardinality. Yet, this problem behaves quite differently with respect to the adjacency issue.
Suppose that a biobjective binary multiple choice knapsack problem is given with n baskets and k possible items per basket. To obtain the set of efficient solutions $\mathcal{X}_E$, we use a simple dynamic programming scheme. In the i-th step, $i = 1, \ldots, n$, we combine every solution being efficient for the problem with baskets $B_1, \ldots, B_{i-1}$, with the items in basket i. Dominated solutions are deleted. The remaining solutions form the set of efficient solutions for the problem with baskets $B_1, \ldots, B_{i-1}$. Note that the items in each basket should be pairwise non-dominated since dominated items are never included in an efficient solution. It should be pointed out that applying this scheme, we solve in fact not only the problem for n baskets, but n different problems for i, $i = 1, \ldots, n$, baskets. Similar to the previous study, a post-optimality procedure is applied to $\mathcal{X}_E$ to retrieve the adjacency information.
We study the frequency of problems with non-connected adjacency graphs when

a) the number of baskets increases from 1 to n, and

b) the (fixed) number of items per basket increases.

As in Section 7.3.1, we use a scheme $Pos1/Pos2/Pos3$ coding the setup of the instances. $Pos1$ indicates the number of baskets while the number of items per basket is given in $Pos2$. The integer cost coefficients are taken from the interval $[1, Pos1 \cdot Pos3]$ according to Method 1 in Section 7.3.1. Table 7.5 reports the setups and the number of instances we have tested.
For each of the setups, Table 7.5 contains the number of instances possessing a non-connected adjacency graph cumulated over all $i = 1, \ldots, 20$ baskets. One obvious difference to the results obtained in Section 7.3.1 is that non-connected adjacency graphs occur far more often for this special type of knapsack problems.

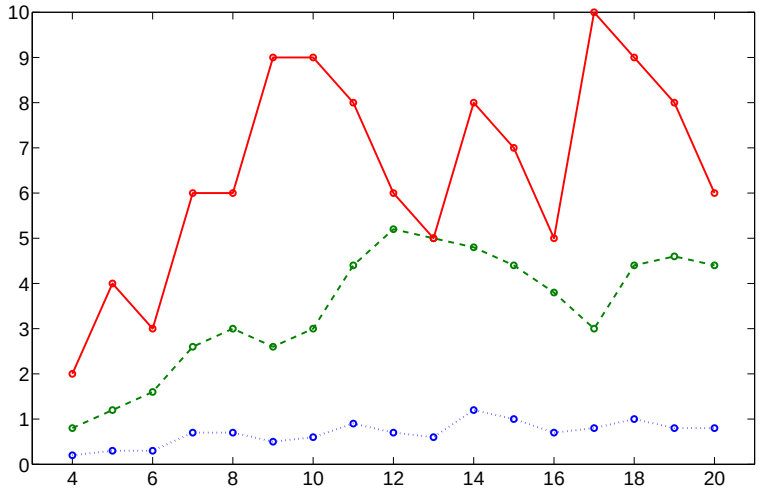


Figure 7.8: *Number of instances per thousand (y-axis) with non-connected adjacency graph for $i = 1, \ldots, 20$ baskets (x-axis) for 5 (dotted line), 10 (dashed line), and 15 (solid line) items per basket.*

Figure 7.8 shows the (normalized) number of instances with non-connected adjacency graph per thousand instances tested. Detailed results for each problem with i baskets, $i = 1, \ldots, 20$, are reported. The dotted line, the dashed line, and the solid line correspond to the setups with 5, 10, and 15 items per basket, respectively. All curves are (slightly) increasing, i.e., non-connectedness is detected more often when the number of baskets is increased. Furthermore, the more items per basket, the higher the likelihood for having a non-connected adjacency graph.
Table 7.6 provides more details about the character of the clusters. Among those instances with non-connected adjacency graph, two clusters appear more often than three clusters. However, with increasing number of items per basket three clusters are getting more likely. Interestingly, the maximal distance between clusters of instances with two clusters only is never greater than 2. The maximal (pairwise) distance between three clusters, however, can be as much as 8.
To summarize the discussion above, the two problems treated in this section behave very differently with respect to adjacency although their combinatorial structure seems quite similar. This shows how careful one has to be with statements about adjacency

Setup of test instances	20/5/10	20/10/10	20/15/10
Instances with one cluster	199882	99705	19889
Instances with two clusters	115	282	100
Instances with three clusters	3	13	11
Maximal distance between two clusters	2	2	2
Maximal distance between three clusters	3	4	8

Table 7.6: *Number of connected components (clusters) in the adjacency graph and maximal distance between two components for the biobjective binary multiple choice knapsack problem.*

and it also shows the limitations of our study: No results about larger instances, about problems with more than two objective functions, and about other problems are available so far.

7.4 Conclusions and Further Ideas

As in the case of single objective combinatorial optimization, the question of adjacency of solutions is one of the core aspects in multiple objective combinatorial optimization. The concept of adjacency of optimal solutions in multiple objective problems certainly exceeds its single objective analogon in terms of complexity because of a more involved optimality concept. Maybe it is due to this increased complexity that research on this subject has widely been neglected. To the best of our knowledge there does not exist (correct) exact algorithms for computing the set of efficient solutions based on neighborhood structures apart from the algorithms that are presented in this work (cf. Chapter 9 for specially structured unconstrained optimization problems and Chapter 10 for a special class of matroid problems), nor does the literature formalize different notions of adjacency. Beyond, adjacency of MCOPs has not been investigated numerically.

The aim of this chapter was threefold. First, we formally introduced two different concepts of adjacency. One class of adjacency concepts relies on problem-dependent combinatorial structures, while the other one is based on appropriate models for the problem and, ultimately, goes back to the definition of adjacency for multiple objective linear programs. Second, we surveyed the current state of the art and supplement it with our own findings. As a result, we listed eleven combinatorial optimization problems and discussed their adjacency properties. Third, we conducted numerical experiments to analyze the adjacency structure of two special types of biobjective knapsack problems. Although being structurally related and possessing a non-connected adjacency graph in general, these knapsack problems differ significantly in the practical occurrence of adjacency.

The presented results should be understood as a first step towards an in-depth investigation of adjacency in MCOPs. In addition, several research directions seem to be promising and are currently under investigation:

Although we proved the non-connectedness of many fundamental MCOPs in this chapter, special variants of these problems might possess a connected adjacency graph (cf. da Silva et al. [43] for knapsack problems and Chapter 9 and Chapter 10 for special classes of unconstrained triobjective and biobjective matroid problems, respectively). It seems that the choice and the range of the involved cost coefficients of the given objectives play a crucial role for proving connectedness. Hence, further explorations of the structure, the size and the geometry of the input data and the resulting connected components should be carried out.

Another interesting stream of research is the development of new definitions of adjacency possibly yielding connected adjacency graphs for a wider class of MCOPs. Based on this research, new ideas to prove the connectedness of the efficient set have to be developed. Up to now, only two promising concepts exist: Either one suggests an algorithm that is based on the connectedness of the efficient set and prove its correctness, or one shows that all efficient solutions correspond to supported efficient solutions

when an MILP-based definition of adjacency is considered. In this case, these solutions always form a connected subgraph of the adjacency graph (cf. Section 7.1). Note that we apply both ideas independently of each other in Chapter 9 and Chapter 10, respectively, to prove connectedness of the efficient set for the problems considered in these chapters.

Furthermore, we note that the connectedness of the efficient set is indeed a nice property, but that it is not absolutely necessary to determine the non-dominated set of a given problem. Since most of the classical multiple objective combinatorial optimization problems are proven to be intractable, a complete enumeration of the efficient set is not preferable. Hence, given a definition of adjacency for solutions in the decision space, one could use this definition to map the idea of connectedness into the objective space. In this case, two non-dominated solutions of a given problem are said to be adjacent if there exist two representatives of these solutions that are adjacent in the decision space. Obviously, the connectedness of the efficient set immediately implies the connectedness of the non-dominated set, but not necessarily vice versa. Nevertheless, the existence of a complete set of connected efficient solutions in the decision space, would suffice to determine the non-dominated set based on local search.

We finally remark that the structural results presented in Section 7.2 are based on a worst-case analysis. Studying theoretically the average case gives detailed information about the expected occurrence of adjacency in practical problems and might justify the application of adjacency-based algorithms even for problems having non-connected adjacency graphs in general. If an average-case analysis is not possible for some problems, intensive numerical investigations about the adjacency behavior might provide empirical evidence for the effectiveness of adjacency-based algorithms.

Chapter 8

Connectedness Results for Combinatorial Problems with Bottleneck Objectives

While we concentrated on combinatorial optimization problems with sum objectives in Chapter 7, the natural question arises whether better results in terms of connectedness of the efficient set can be achieved when other types of objectives are considered. In this chapter we present some new results for multiple objective combinatorial optimization problems with bottleneck objectives (MCBP).

From Section 4.2 we recall that these problems can be solved by a sequential reduction to single objective problems of the same class of combinatorial problems (cf. Algorithm 4.2). Hence, the application of neighborhood search techniques is mainly interesting for classes of combinatorial problems where the single objective version is already $\mathcal{NP}$-hard to solve. Nevertheless, it seems to be worth discussing connectedness properties of the (weakly) efficient set for general MCBPs.

In terms of connectedness of the efficient set $\mathcal{X}_E$, we recall from Definition 7.1 that, given a definition for the adjacency of efficient solutions, $\mathcal{X}_E$ is said to be connected if and only if the corresponding adjacency graph $\mathcal{G}$ is connected. In this context, a node of $\mathcal{G}$ represents an efficient solution of the given multiple objective problem. Furthermore, two nodes are joined by an edge if and only they are adjacent with respect to the considered definition of adjacency.

For a short review of the literature for MCBPs, we refer to Section 4.2.2. To the best of our knowledge, results on the connectedness of the (weakly) efficient set for MCBPs have not been presented in the literature before.

We remark that the focus of this chapter is twofold. In the first part, we state a counter-example that shows that the set of efficient solutions for MCBPs is not connected in general. While we had to treat each class of combinatorial optimization problems separately in Chapter 7 when sum objectives are considered, we give a counter-example that can be applied to most classes of combinatorial optimization problems simultaneously here. This will be done in Section 8.1.

In Section 8.2, we present an algorithm that solves the fixed cardinality biobjective binary knapsack problem with two bottleneck objectives in polynomial time, based on the ideas developed in Section 4.2. We prove that the set of weakly efficient solutions is always connected for this problem, while this is not the case for the efficient set by the results from Section 8.1. We finally conclude in Section 8.3.

8.1 Connectedness for General Combinatorial Bottleneck Problems

In Chapter 7 we have shown that the set of efficient solutions of most of the classical combinatorial optimization problems is not connected in general when sum objectives are considered. We prove in this section that this result remains valid, when problems with bottleneck objectives have to be solved instead. Different to the sum objective case, we state a counter-example that can be applied to many classes of combinatorial problems simultaneously. In more detail, given a definition for the adjacency of efficient solutions, we can construct a general problem instance with only two efficient solutions such that these two solutions differ in a maximum number of elements.

Definition 8.1 *Let an instance of a multiple objective combinatorial optimization problem be given, and let $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$ denote the weighted adjacency graph (cf. Definition 7.3) of the problem. Two efficient solutions are said to be* at maximum distance *if the weight of the edge connecting the two nodes that correspond to these two solution is maximal with respect to the considered definition of adjacency.*

In general, the maximum distance between two efficient solutions crucially depends on the considered class of combinatorial problems. For example, the maximum distance between two efficient assignments equals two, assuming that the combinatorial definition of adjacency stated in Definition 7.33 is applied (cf. Section 7.2.9). In contrast, using Definition 7.19 in the case of a binary knapsack problem with equal weights and fixed cardinality treated in Section 7.2.5, two efficient solutions $S_1, S_2 \in \mathcal{X}$ are at maximum distance, whenever we have that $|S_1 \cap S_2| = \min\{|S \cap T|, S, T \in \mathcal{X}\}$. So, the maximum distance between two efficient solutions may further depend on the cardinality of the involved feasible solutions.

Remark 8.2 *We assume in the following that all feasible solutions of an instance $(\mathcal{E}, \mathcal{X}, b)$ of an MCBP have the same cardinality, i.e. there exists $k \in \mathbb{N}$, such that $|S_1| = |S_2| = k$ for all $S_1, S_2 \in \mathcal{X}$.*

Having a look back to Chapter 7, we see that most of the classical combinatorial optimization problems satisfy the above stated condition. Moreover, Remark 8.2 implicitly implies that the given definition of adjacency of efficient solutions has to be based on an exchange of a prescribed number of elements contained in two different efficient solutions. However, there exist classes of combinatorial optimization problems where this definition is exclusively based on removing (or adding) a fixed number of elements from (or to) an efficient solution to construct new efficient solutions (cf., e.g., Subsection 7.2.8 and Chapter 9 for unconstrained multiple objective combinatorial optimization problems). Hence, the main theorem of this section, which is stated next, can be applied to many but not all classes of combinatorial optimization problems simultaneously.

Theorem 8.3 *Let a class of combinatorial optimization problems be given, where each instance of this class satisfies the condition stated in Remark 8.2. Then there exist instances $(\mathcal{E}, \mathcal{X}, b)$, where the objective function vector $b = (b_1, \ldots, b_p)$, $p > 1$, consists of p bottleneck objectives involving p cost functions $w_1, \ldots, w_p$ on the elements of $\mathcal{E}$, such that each pair of efficient solutions is at maximum distance.*

Proof: We start with a proof for $p = 2$. Let a class of combinatorial optimization problems be given that satisfies Remark 8.2. Furthermore, let $S_1, S_2 \in \mathcal{X}$, $S_1 \neq S_2$, denote two feasible solutions that are at maximum distance, assuming that Definition 8.1 is extended to the complete set of feasible solutions. We construct a problem instance of the considered class, such that S_1 and S_2 correspond to the only two efficient solutions of this problem. This shows the theorem for $p = 2$.
We partition the ground set $\mathcal{E}$ into the subsets $E_1 = S_1 \setminus S_2$, $E_2 = S_2 \setminus S_1$, $E_{12} = S_1 \cap S_2$ and $E_0 = \mathcal{E} \setminus (S_1 \cup S_2)$. As all feasible solutions have the same cardinality $k \in \mathbb{N}$ by assumption, we have that $E_1 \neq \emptyset \neq E_2$, since otherwise this would imply that $S_1 = S_2$. Furthermore, $|S_1 \cap S_2| < k$ has to be valid. For $e \in \mathcal{E}$ we define the following cost function vector $(w_1(e), w_2(e)) \in \mathbb{R}^2$:

$$\begin{pmatrix} w_1(e) \\ w_2(e) \end{pmatrix} = \begin{cases} (3,3)^\top, \text{ if } e \in E_0, \\ (1,2)^\top, \text{ if } e \in E_1, \\ (2,1)^\top, \text{ if } e \in E_2, \\ (1,1)^\top, \text{ if } e \in E_{12}. \end{cases} \tag{8.1}$$

Let $b_i(S) = \max_{e \in S}\{w_i(e)\}$, $i \in \{1,2\}$, denote the two bottleneck objectives. Then for $S \in \mathcal{X}$ we have that:

$$\begin{pmatrix} b_1(S) \\ b_2(S) \end{pmatrix} = \begin{cases} (1,2)^\top, \text{ if } S = S_1, \\ (2,1)^\top, \text{ if } S = S_2, \\ (2,2)^\top, \text{ if } S \subsetneq S_1 \cup S_2,\ S_1 \neq S \neq S_2, \\ (3,3)^\top, \text{ if } S \cap E_0 \neq \emptyset. \end{cases}$$

By construction, the considered instance $(\mathcal{E}, \mathcal{X}, (b_1, b_2))$ has two efficient solutions S_1 and S_2 that are at maximum distance, and $\mathcal{Y}_N = \{(1,2), (2,1)\}$. This completes the proof for $p = 2$.
To extend the result to the case $p > 2$, we simply keep w_1 and w_2 as defined in (8.1) and define $w_i(e) = 1$ for all $e \in \mathcal{E}$ and $i \in \{3, \ldots, p\}$. Obviously, S_1 and S_2 are still the only two efficient solutions of the problem and are at maximum distance. This completes the proof for the general case. □

Note that Remark 8.2 is essential for the validity of Theorem 8.3. If we drop the assumption that all feasible solutions have the same cardinality, there may exist a feasible solution S that is completely contained in $E_{12} = S_1 \cap S_2$. But this would imply that $(b_1(S), b_2(S)) = (1,1)$, and S_1 and S_2 would be dominated by S.
Moreover, the counter-example in the proof of Theorem 8.3 does not automatically imply that the set of weakly efficient solutions of the considered instance is non-connected in general. As in the case of combinatorial problems with several sum objectives, the potential connectedness of the weakly efficient set for MCBPs strongly depends on the considered class of combinatorial problems, i.e. its feasible set $\mathcal{X}$, the structure of the cost functions involved and the definition of adjacency used for the problem.
For example, while Theorem 8.3 shows that the efficient set of the biobjective binary knapsack problem with equal weights and fixed cardinality involving bottleneck objectives is not connected in general, we prove in the next section that the weakly efficient

set of this problem is always connected whenever Definition 7.19 is used to define the adjacency of weakly efficient solutions. Furthermore, a simple sufficient condition for the connectedness of $\mathcal{X}_E$ based on the structure of the cost coefficients can be proven for this problem (cf. Theorem 8.8).

8.2 Biobjective Binary Knapsack Problems with Bottleneck Objectives

As already mentioned in the last section, we consider a special class of combinatorial knapsack problems in the following. In more detail, we present an algorithm to solve the biobjective binary knapsack problem with a fixed cardinality constraint involving two bottleneck objectives that is based on the ideas stated in Section 4.2.2. Furthermore, we prove that, while the set of efficient solutions of this problem is not connected in general, this property always holds for the set of weakly efficient solutions.
The single objective version

$$\min \max_{1\leq i\leq n} \left\{ c_i x_i : \sum_{i=1}^{n} x_i = k,\, x_i \in \{0,1\},\, i = 1,\dots,n \right\}, \tag{BKP}$$

of the considered problem, where $c_i \in \mathbb{Z}$ for $i = 1,\dots,n$ and $k \in \{1,\dots,n\}$ fixed, was already studied by Xu and Liu [218]. An optimal solution of this problem can be obtained by choosing the k lowest cost coefficients of the vector $c = (c_1,\dots,c_n)$. Starting from the set of optimal solutions for the problem, the authors defined an *optimal solution transformation graph* $\mathcal{G}$ similar to the adjacency graph of efficient solutions for a multiple objective combinatorial problem (cf. Definition 7.1), where the node set of $\mathcal{G}$ corresponds to the optimal solutions of the problem. Two nodes are joint by an edge if two optimal solutions have exactly two different components. The authors proved that the resulting graph is edge-Hamiltonian, i.e. each edge is contained in a Hamiltonian cycle of $\mathcal{G}$. We recall that a Hamiltonian cycle of a graph is a cycle that visits each node exactly once. Furthermore, an algorithm for generating a Hamiltonian cycle in $\mathcal{G}$ is presented that enumerates all optimal solutions of the given problem.
The remainder of this section is organized as follows: In the next subsection we formally introduce the problem and our notation. In the subsequent subsection we present our algorithm, followed by a short example. Finally, connectedness properties of the (weakly) efficient set are discussed in Section 8.2.4.

8.2.1 Problem Formulation and Notation

Let a finite set $\mathcal{E} = \{e_1,\dots,e_n\}$ of n different items be given, and let $k \in \{1,\dots,n\}$ be fixed. Furthermore, let $\mathcal{X} \subseteq \mathcal{P}(\mathcal{E})$ denote the set of all subsets of $\mathcal{E}$ containing exactly k items. Let $w_1, w_2 : \mathcal{E} \to \mathbb{Z}$ denote two weight functions on the elements of $\mathcal{E}$, and let $b_i : \mathcal{X} \to \mathbb{Z}$ denote the bottleneck objective involving the weight w_i for $i = 1, 2$. Then, the *biobjective binary knapsack problem with fixed cardinality* (BBKP) with two bottleneck objectives is given by

$$\min \left\{ (b_1(S), b_2(S))^\top : S \in \mathcal{X} \right\}. \tag{8.2}$$

The set of (weakly) efficient solutions of this problem is denoted by $\mathcal{X}_E$ (and $\mathcal{X}_{wE}$, respectively), while its image in the objective space is given by $\mathcal{Y}_N$ (and $\mathcal{Y}_{wN}$). For $S \in \mathcal{X}$, let $b(S) = (b_1(S), b_2(S))$. Furthermore, let χ denote the bijective *characteristic mapping* between the power set $\mathcal{P}(\mathcal{E})$ of $\mathcal{E}$ and the set of binary vectors $\{0,1\}^n$, where for $S \in \mathcal{X}$

$$\chi(S) := (x_1, \ldots, x_n) = x \in \{0,1\}^n, \text{ with } x_i = \begin{cases} 1, & e_i \in S, \\ 0, & e_i \notin S, \end{cases}$$

(cf. also Section 7.1). By setting $c_i^j = w_j(e_i)$ for $i = 1, \ldots, n$ and $j = 1, 2$, and defining $f = (f^1, f^2)$ where

$$f^j : \begin{cases} \{0,1\}^n & \to & \mathbb{Z}, \\ X & \mapsto & \max\{c_i^j x_i,\ i = 1, \ldots, n\}, \end{cases}$$

for $j \in \{1, 2\}$, we can transform Problem (8.2) into the binary biobjective optimization problem

$$\begin{aligned} &\min\ (f^1(X), f^2(X))^\top \\ &\text{s.t.} \sum_{i=1}^{n} x_i = k, \\ &\qquad x_i \in \{0,1\} \quad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{BBKP}$$

By construction, we have that $f(X) = CX$ where the matrix $C \in \mathbb{Z}^{2\times n}$ corresponds to the transformed weights of the items from the ground set. In the following, we mostly refer to the binary problem formulation and its corresponding notation (cf. Problem (BBKP)), but change to the combinatorial one (cf. Problem (8.2)), whenever it is necessary and simplifies the reasoning.

In the remainder of this section it is further assumed that the cost coefficients of the first row of C are ordered in non-decreasing order, i.e. for the first objective vector c^1 it holds true that $c_1^1 \leq c_2^1 \leq \ldots \leq c_n^1$. If two or more subsequent elements of c^1 coincide, we do not assume any a priori ordering in the second component c^2. Furthermore, two different columns of C may be the same.

Since two subsequent elements of the vector c^1 may coincide, let $\hat{c}_{i_1}^1, \ldots, \hat{c}_{i_m}^1$, where $1 \leq m \leq n$, denote the distinct cost coefficients of items from $\mathcal{E}$ with respect to c^1. Using these m pairwise different coefficients, we can partition $\mathcal{E}$ into m layers $L_j := \{e_i \in \mathcal{E} : w_1(e_i) = \hat{c}_{i_j}^1\}$ where $j \in \{1, \ldots, m\}$. We call L_j the j^{th} layer of $\mathcal{E}$. Obviously, the set $\{L_1, \ldots, L_m\}$ forms a partition of $\mathcal{E}$. We further define

$$j_0 := \min\left\{ t \in \{1, \ldots, m\} : \sum_{j=1}^{t} |L_j| \geq k \right\}$$

and call L_{j_0} the *marginal layer of $\mathcal{E}$ relative to k (with respect to the first objective).* For $\tau \in \{0, \ldots, m - j_0\}$ we set $s_\tau = \sum_{j=1}^{j_0+\tau} |L_j|$. Note that for $\tau \in \{1, \ldots, m - j_0\}$ it holds that

$$k \leq s_{\tau-1} < s_{\tau-1} + 1 \overset{(A)}{\leq} s_\tau \overset{(B)}{\leq} n,$$

with equality in (A) if $|L_{j_0+\tau}| = 1$, and with equality in (B) if $\tau = m - j_0$.

Finally, let

$$D_{s_\tau} = \left(d^1_{s_\tau}, d^2_{s_\tau}\right)^\top := \begin{pmatrix} c^1_1 & \dots & c^1_{s_\tau} \\ c^2_1 & \dots & c^2_{s_\tau} \end{pmatrix}$$

denote the restriction of the matrix C to the first s_τ columns for $\tau \in \{0, \dots, m - j_0\}$.

8.2.2 Algorithm for Solving the Biobjective Binary Knapsack Problem

In the following, we present an algorithm that aims to generate the complete non-dominated set $\mathcal{Y}_N$ of an instance of Problem (BBKP) only by means of implicitly solving at most n different ε-constraint problems by a simple enumeration approach. Any further scalarization of the two objectives is not needed. Using the algorithm presented in Xu and Liu [218] for solving the single objective problem in each iteration of the algorithm, even the complete set of efficient solutions $\mathcal{X}_E$ can be determined. Note that the cardinality of this set can be of exponential size, while the cardinality of $\mathcal{Y}_N$ is bounded by n (cf. Section 4.2.2).
To simplify the notation in the remainder of this section, let $BBKP(C, k)$ denote an instance of Problem (BBKP), where $C \in \mathbb{Z}^{2\times n}$ and $k \in \{1, \dots, n\}$. In addition, let $BKP(c, k)$ correspond an instance of Problem (BKP), where $c_i \in \mathbb{Z}$ for $i = 1, \dots, n$. Based on the notation introduced in Subsection 8.2.1 we prove:

Lemma 8.4 *Let $X_{s_0} \in \{0,1\}^{s_0}$ be an optimal solution of* BKP($d^2_{s_0}$,k) *with objective value $f^2_{s_0}$. Then*

$$\hat{X}_{s_0} := (X_{s_0}, \underbrace{0, \dots, 0}_{s_0+1,\dots,n}) \in \{0,1\}^n$$

is an efficient solution of $BBKP(C, k)$, where the corresponding non-dominated objective vector is given by $f(\hat{X}_{s_0}) = (c^1_{s_0}, f^2_{s_0})$.

Proof: Let X_{s_0} be an optimal solution of *BKP($d^2_{s_0}$,k)*. Since by construction $c^1_{s_0}$ is the k^{th} largest coefficient of c^1, there cannot exist any other feasible solution $\hat{X}$ such that $f^1(\hat{X}) < f^1(\hat{X}_{s_0}) = c^1_{s_0}$. Since $f^2_{s_0}$ is the optimal objective value in the second component with respect to all elements $e \in \mathcal{E}$ satisfying $w_1(e) \le c^1_{s_0}$, we cannot improve the objective value in the second component of Y_{s_0} without worsening the first. Hence, $f(\hat{X}_{s_0}) = (c^1_{s_0}, f^2_{s_0})$ is a non-dominated solution of $BBKP(C, k)$ and $\hat{X}_{s_0}$ is efficient. □

Note that if s_0 is equal to k, then $f^2_{s_0}$ is just the maximum entry of the components of $d^2_{s_0}$, i.e. $f^2_{s_0} = \max\{c^2_1, \dots, c^2_{s_0}\}$. For completeness, we state the obvious fact that:

Corollary 8.5 *If $j_0 = m$, there exists exactly one non-dominated solution that can be obtained by applying Lemma 8.4, i.e. $|\mathcal{Y}_N| = 1$.*

In the following we suppose that $j_0 < m$. Let $\chi^{-1} : \{0,1\}^n \to \mathcal{P}(\mathcal{E})$ denote the inverse function of the characteristic mapping χ defined in Subsection 8.2.1.

Lemma 8.6 *Let $j_0 < m$, let $\tau \in \{1, \dots, m - j_0\}$, and let $X_{s_\tau} \in \{0,1\}^{s_\tau}$ be an optimal solution of* BKP($d^2_{s_\tau}$,k) *with objective value $f^2_{s_\tau}$. Furthermore, let $f^2_{s_{\tau-1}}$ denote the optimal objective value of* BKP($d^2_{s_{\tau-1}}$,k). *We set*

$$\hat{X}_{s_\tau} = (X_{s_\tau}, \underbrace{0, \dots, 0}_{s_\tau+1,\dots,n}) \in \{0,1\}^n,$$

and, we define $S := \chi^{-1}(\hat{X}_{s_\tau})$.

1. *If $f^2_{s_\tau} < f^2_{s_{\tau-1}}$, S contains at least one element from $L_{j_0+\tau}$ and $\hat{X}_{s_\tau}$ is efficient for BBKP(C,k). The corresponding non-dominated vector $f(\hat{X}_{s_\tau})$ is given by $(c^1_{s_\tau}, f^2_{s_\tau})$.*
2. *If $f^2_{s_\tau} = f^2_{s_{\tau-1}}$ and S contains at least one element of $L_{j_0+\tau}$, then there exists another feasible solution $\tilde{X} \in \{0,1\}^n$ whose restriction to the first $s_{\tau-1}$ components is optimal for both* BKP($d^2_{s_{\tau-1}}$,k) *and* BKP($d^2_{s_\tau}$,k), *and hence dominates $f(\hat{X}_{s_\tau}) = (c^1_{s_\tau}, f^2_{s_\tau})$ for BBKP(C,k).*

Proof: Let $\tau \in \{1, \dots, m - j_0\}$ be arbitrary but fixed. First, we suppose that $f^2_{s_\tau} < f^2_{s_{\tau-1}}$. Then S must contain at least one element which is contained in $L_{j_0+\tau}$, otherwise the vector $\tilde{X} := (\hat{x}_1, \dots, \hat{x}_{s_{\tau-1}})$ would be feasible for *BKP($d^2_{s_{\tau-1}}$,k)* with objective value

$$f^2(\tilde{X}) = f^2_{s_\tau} < f^2_{s_{\tau-1}}.$$

This is a contradiction to the optimality of $f^2_{s_{\tau-1}}$ for *BKP($d^2_{s_{\tau-1}}$,k)*. Hence, $f(\hat{X}_{s_\tau}) = (c^1_{s_\tau}, f^2_{s_\tau})$, with $f^2_{s_\tau}$ optimal for *BKP($d^2_{s_\tau}$,k)*.
If we want to improve $f(\hat{X}_{s_\tau})$ in the second component, we have to consider another feasible solution $\bar{X}$ such that $\chi^{-1}(\bar{X})$ contains at least one element of the set $\bigcup_{j=j_0+\tau+1}^{m} L_j$ if the label $L_{j_0+\tau+1}$ exists. But then $f^1(\bar{X}) > c^1_{s_\tau}$ by construction. Hence, $f(\hat{X}_{s_\tau})$ is a non-dominated solution of BBKP(C,k) and $\hat{X}_{s_\tau}$ is efficient. This shows (1.).
Now suppose that $f^2_{s_\tau} = f^2_{s_{\tau-1}}$ and that S contains at least one element of $L_{j_0+\tau}$. Let $\hat{X}_{s_{\tau-1}}$ be an optimal solution of *BKP($d^2_{s_{\tau-1}}$,k)*. Obviously,

$$\tilde{X} = (\hat{X}_{s_{\tau-1}}, \underbrace{0, \dots, 0}_{s_{\tau-1},\dots,s_\tau})$$

is also feasible for *BKP($d^2_{s_\tau}$,k)* with objective value $f^2_{s_{\tau-1}} = f^2_{s_\tau}$. Hence, $\tilde{X}$ is optimal for *BKP($d^2_{s_\tau}$,k)*, and $f(\tilde{X})$ dominates $f(\hat{X}_{s_\tau}) = (c^1_{s_\tau}, f^2_{s_\tau})$, since $c^1_{s_{\tau-1}} < c^1_{s_\tau}$ by construction. □

If we consider the single objective problem *BKP($d^2_{s_\tau}$,k)* from Lemma 8.6 in more detail, we recognize that solving this problem is equivalent to finding the optimal solution of

$$\begin{aligned} \min\ & f^2(X) \\ \text{s.t. } & f^1(X) \le \hat{c}^1_{j_0+\tau}, \\ & \sum_{i=1}^{n} x_i = k, \\ & x_i \in \{0,1\}\ \forall i \in \{1, \dots, n\}. \end{aligned} \tag{8.3}$$

Algorithm 8.1 Algorithm for solving the Biobjective Binary Knapsack Problem

Input: An instance (C,k) of Problem (BBKP).
Output: The set of non-dominated solutions $\mathcal{Y}_N$.

```
1: Set Y_N = ∅.
2: Sort the first row of C in non-decreasing order.
3: Determine the layers L_1, ..., L_m and specify the marginal layer L_{j_0} and s_0.
4: Solve the initial problem BKP(d^2_{s_0},k) → (c^1_{s_0}, f^2_{s_0}).
5: Y_N ← (c^1_{s_0}, f^2_{s_0}).
6: if j_0 < m then
7:    for τ = 1 to m − j_0 do
8:       Solve the problem BKP(d^2_{s_τ},k) → (c^1_{s_τ}, f^2_{s_τ}).
9:       if f^2_{s_τ} < f^2_{s_{τ−1}} then
10:         Y_N ← (c^1_{s_τ}, f^2_{s_τ}).
11:      end if
12:   end for
13: end if
14: return Y_N.
```

Hence, solving *BKP($d^2_{s_\tau}$,k)* is nothing else than solving an ε-constraint problem for BBKP(C,k) with f^2 as objective function and a constraint on f^1. From Chankong and Haimes [36] we recall that if $\hat{X}$ is an optimal solution of Problem (8.3), then $\hat{X}$ is at least weakly efficient. If $\hat{X}$ is the unique optimal solution of Problem (8.3), i.e. of *BKP($d^2_{s_\tau}$,k)*, then $\hat{X}$ is efficient. So, Lemma 8.6 can be seen as a strengthened version of this result adopted to the special structure of BBKP(C,k).
A short outline of the algorithm that solves Problem (BBKP) is given in Algorithm 8.1. We restrict ourselves to the calculation of the non-dominated set $\mathcal{Y}_N$ only. Note that in this case, each subproblem BKP$(d^2_{s_\tau},k)$ can be solved by an algorithm that calculates the k-largest element in an unsorted list. This can be done in $\mathcal{O}(n)$ (cf. Cormen et al. [42]). If, in addition, a complete set $\mathcal{X}^\star \subseteq \mathcal{X}$ of efficient knapsacks is of interest, an algorithm that returns the k largest elements of an unsorted list is needed. For fixed k this is still possible in $\mathcal{O}(n)$ (cf. Cormen et al. [42]). If the complete efficient set has to be calculated, the algorithm proposed in Xu and Liu [218] can be used. Note that in this case no polynomial bound on the time complexity can be given, since the size of $\mathcal{X}_E$ can be exponential in general.

Theorem 8.7 *Given an instance of Problem* (BBKP)*, Algorithm 8.1 determines the set of all non-dominated solutions $\mathcal{Y}_N$ in $\mathcal{O}(n^2)$ time.*

Proof: Since all solutions generated by Algorithm 8.1 correspond to non-dominated solutions of BBKP(C,k) by Lemma 8.4 and Lemma 8.6, it suffices to show that given a non-dominated solution of the problem, this solution will be generated during the course of the algorithm.
So, let a non-dominated vector $f(X)$ of BBKP(C,k) with corresponding efficient representative $X \in \{0,1\}^n$ be given, and define $S := \chi^{-1}(X)$. Then, there exists at least one element $e \in (\mathcal{E} \cap S)$ such that $e \in \bigcup_{j=j_0}^{m} L_j$. Otherwise, X would not be feasible for BBKP(C,k) by the definition of L_{j_0}.
We set $\alpha = \max\left\{t \in \{0,\dots,m-j_0\} : \bigcup_{j=j_0}^{j_0+t} L_j \cap S \neq \emptyset\right\}$. If $\alpha = 0$ or $j_0 = m$, X

and $f(X)$ are generated by Lemma 8.4 and Corollary 8.5, respectively. Otherwise, we consider Line 7 of Algorithm 8.1 for $\tau = \alpha > 0$. Since X is an optimal solution of Problem (8.3) for the right hand side $\hat{c}^1_{j_0+\tau} = f^1(X)$, $f(X)$ is generated when the subproblem BKP($d^2_{s_\alpha}$,k) is solved. Obviously, $f(X)$ satisfies $f^2(X) < f^2_{s_{\tau-1}}$, since otherwise there would exist another feasible solution $\bar{X}$ of BBKP(C,k) that dominates X by Lemma 8.6. Hence, $f(X)$ is added to $\mathcal{Y}_N$ which shows that $f(X)$ is generated during the course of the algorithm.
To prove the stated time complexity of $\mathcal{O}(n^2)$, we note, that the first row of the matrix C can be sorted in $\mathcal{O}(n \log(n))$. Determining the k^{th} largest element in an unsorted list can be done in $\mathcal{O}(n)$ time (cf. Cormen et al. [42]). This step has to be performed at most $\mathcal{O}(n)$ times. So, the overall complexity of the algorithm is given by $\mathcal{O}(n^2)$. □

Note that the time-complexity of Algorithm 8.1, stated in Theorem 8.7, coincides with the time-complexity given in Theorem 4.3 for the general case. Indeed, Algorithm 8.1 can be seen as a special case of the general Algorithm 4.2, where the unconstrained problem BKP($d^2_{s_\tau}$,k), $\tau = 0, \ldots, m$, is solved in the Lines 4 and 8, respectively. Elements contained in layers with index higher than τ are ignored when solving this subproblem, i.e. their weights are implicitly set to infinity.

8.2.3 Example

In this subsection, we give a short example for solving an instance of Problem (BBKP), using the notation given in the previous subsections. So, let in the following $\mathcal{E} = \{e_1, \ldots, e_{10}\}$, $k = 3$ and

$$C = \begin{pmatrix} 1 & 2 & 3 & 3 & 3 & 4 & 5 & 6 & 6 & 6 \\ 6 & 4 & 5 & 6 & 8 & 5 & 2 & 2 & 4 & 1 \end{pmatrix}.$$

The given problem consists of the $m = 6$ layers $L_1 = \{e_1\}$, $L_2 = \{e_2\}$, $L_3 = \{e_3, e_4, e_5\}$, $L_4 = \{e_6\}$, $L_5 = \{e_7\}$, and $L_6 = \{e_8, e_9, e_{10}\}$. The marginal layer with respect to w_1 is L_3, i.e. $j_0 = 3$, $s_0 = 5$ and

$$D_{s_0} = \begin{pmatrix} 1 & 2 & 3 & 3 & 3 \\ 6 & 4 & 5 & 6 & 8 \end{pmatrix}.$$

In the first step, we have to solve BKP($d^2_{s_0}$,3), where $d^2_{s_0} = (6, 4, 5, 6, 8)$. The optimal objective value is given by $f^2_{s_0} = 6$ and can be obtained by the feasible solutions $S_1 = \{e_1, e_2, e_3\}$, $S_2 = \{e_1, e_2, e_4\}$, $S_3 = \{e_1, e_3, e_4\}$ and $S_4 = \{e_2, e_3, e_4\}$. Obviously, these four solutions are also feasible for BBKP(C,k) and lead to the same objective vector $f(\chi(S_i)) = (c^1_{s_0}, f^2_{s_0}) = (3, 6)$, $i = 1, \ldots, 4$, which is non-dominated according to Lemma 8.4.
For $\tau = 1$ we obtain that $s_1 = s_0 + |L_4| = 6$ and $d^2_{s_1} = (6, 4, 5, 6, 8, 5)$. The optimal objective value of BKP($d^2_{s_1}$,3) is given by $f^2_{s_1} = 5$ and can be obtained only by the feasible solution $S_5 = \{e_2, e_3, e_6\}$. Since $f^2_{s_1} = 5 < 6 = f^2_{s_0}$, $= f(\chi(S_5)) = (c^1_{s_1}, f^2_{s_1}) = (4, 5)$ is a non-dominated vector of *BBKP(C,k)* with efficient solution B_5, according to Lemma 8.6.
For $\tau = 2$ we get that $s_2 = s_1 + |L_5| = 7$ and $d^2_{s_2} = (6, 4, 5, 6, 8, 5, 2)$. The optimal objective value of BKP($d^2_{s_2}$,3) is given by $f^2_{s_2} = 5$ and can be obtained by the four

feasible solutions $S_5 = \{e_2, e_3, e_6\}$, $\tilde{S}_1 = \{e_2, e_3, e_7\}$, $\tilde{S}_2 = \{e_2, e_6, e_7\}$, $\tilde{S}_3 = \{e_3, e_6, e_7\}$. The objective vector $f(\chi(\tilde{S}_i)) = (c^1_{s_2}, f^2_{s_2}) = (5, 5)$, $i = 1, \ldots, 3$, is a dominated solution of BBKP(C,3), since it is dominated by $f(\chi(S_5)) = (4, 5)$. Note that S_5 is an optimal solution for both BKP($d^2_{s_1}$,3) and BKP($d^2_{s_2}$,3), dominating $\tilde{S}_1, \tilde{S}_2, \tilde{S}_3$, as stated in Lemma 8.6.
Finally for $\tau = 3$, we have that $s_3 = s_2 + |L_6| = 10$ and $d^2_{s_3} = (6, 4, 5, 6, 8, 5, 2, 2, 4, 1)$. The optimal objective value of BKP($d^2_{s_3}$,3) is given by $f^2_{s_3} = 2$ and can be only obtained by the feasible solution $S_6 = \{e_2, e_7, e_8\}$. Since $f^2_{s_3} = 2 < 5 = f^2_{s_2}$, $f(\chi(S_6)) = (c^1_{s_3}, f^2_{s_3}) = (6, 2)$ is a non-dominated solution of BBKP(C,3) with corresponding efficient solution S_6.
We conclude that the set of efficient solutions $\mathcal{X}_E$ and the set of non-dominated solutions $\mathcal{Y}_N$ of the considered instance BBKP(C,3), are given by $\mathcal{X}_E = \{S_1, \ldots, S_6\}$ and $\mathcal{Y}_N = \{(3, 6), (4, 5), (6, 2)\}$, respectively.

8.2.4 Connectedness of the (Weakly-)Efficient Set

In this final subsection we discuss connectedness properties of the (weakly) efficient set for the biobjective binary knapsack problem with fixed cardinality based on the results from Section 8.1. From Section 7.2.5 we recall that two efficient solutions X^1, X^2 of Problem (BBKP) are called adjacent if and only if X^2 can be obtained from X^1 by replacing one item in X^1 with one item of X^2 that is not contained in X^1, i.e. if their corresponding Hemming-distance $d_H(X^1, X^2) := \sum_{i=1}^n |x^1_i - x^2_i| = 2$ (cf. Definition 7.19). Since all feasible solutions of Problem (BBKP) have the same cardinality by the definition of the problem, the condition stated in Remark 8.2 is satisfied. Hence, the set of efficient solutions is not connected in general for these types of problems, due to Theorem 8.3.
Despite this negative result, we will state a sufficient condition for the connectedness of $\mathcal{X}_E$ for Problem (BBKP), only using Algorithm 8.1 in the following. Furthermore, we prove that the set of weakly efficient solutions is always connected for this type of biobjective knapsack problem.

Theorem 8.8 *The set $\mathcal{X}_E$ of all efficient solutions of Problem* (BBKP) *is connected if the two objective vectors c^1 and c^2 consist of pairwise different coefficients, respectively.*

Proof: We use Algorithm 8.1 to prove this theorem. We assume that in the Lines 4 and 8 of the algorithm, not only the optimal solution of the subproblem is returned, but also the complete set of optimal knapsacks, generating this solution.
Since the elements of $\mathcal{E}$ are sorted in non-decreasing order with respect to their weight in the first component, it holds that $c^1_i < c^1_j$ if and only of $i < j$ $(i, j \in \{1, \ldots, n\})$. Since c^1 consists of pairwise different elements, the labels L_j are singletons containing exactly the element $e_j \in \mathcal{E}$, additionally $m = n$ and $s_0 = j_0 = k$. For this reason, $d^2_{s_{\tau-1}}$ and $d^2_{s_\tau}$ differ exactly in the cost vector $c^2_{k+\tau}$ in each iteration of the algorithm, where $\tau \in \{1, \ldots, n-k\}$.
Since also c^2 consists of pairwise different elements, all subproblems BKP($d^2_{s_\tau}$,k) have a unique optimal solution for $\tau \in \{0, 1, \ldots, n-k\}$. Since these not necessarily pairwise different solutions are unique solutions of Problem (8.3) with appropriately chosen right hand side coefficient, these solutions correspond to efficient solutions of the instance BBKP(C,k) (cf. Chankong and Haimes [36]). More precisely, from Lemma 8.6 it

follows that a new efficient solution is generated by solving BKP($d^2_{s_\tau}$,k), where $\tau \in \{1,\dots,n-k\}$, which is not yet contained in $\mathcal{X}_E$, if and only if $c^2_{k+\tau} < f^2_{s_{\tau-1}}$. Note that if $c^2_{k+\tau} > f^2_{s_{\tau-1}}$, the uniquely determined optimal solution for problem BKP($d^2_{s_{\tau-1}}$,k) is also optimal for BKP($d^2_{s_\tau}$,k).
Let $\tau \in \{1,\dots,n-k\}$, and let $X_{s_{\tau-1}}$ be the solution which is optimal for BKP($d^2_{s_{\tau-1}}$,k), and let X_{s_τ} be optimal for BKP($d^2_{s_\tau}$,k), supposing that

$$c^2_{k+\tau} \leq f^2(X_{s_\tau}) < f^2(X_{s_{\tau-1}}) = f^2_{s_{\tau-1}}.$$

Since $L_{k+\tau} = \{e_{k+\tau}\}$, $\chi^{-1}(X_{s_\tau})$ is obtained from $\chi^{-1}(X_{s_{\tau-1}})$ by exchanging exactly the element $e_{k+\tau} \in \chi^{-1}(X_{s_\tau})$ with the uniquely determined element $e_t \in \chi^{-1}(X_{s_{\tau-1}}) \setminus \chi^{-1}(X_{s_\tau})$ satisfying $w_2(e_t) = f^2(X_{s_{\tau-1}})$. Hence, $d_H(X_{s_{\tau-1}}, X_{s_\tau}) = 2$, and the two efficient solutions $X_{s_{\tau-1}}$ and X_{s_τ} are adjacent. □

Since the first $k-1$ smallest entries of c^1 and c^2, respectively, are not "essential" for the value of the non-dominated objective vectors for an instance of BBKP(C,k) we can still strengthen the last theorem:

Corollary 8.9 *The set $\mathcal{X}_E$ of all efficient solutions of Problem* (BBKP) *is connected if the first $n-k+1$ largest components of c^1 and c^2 are pairwise different, respectively.*

Note that the sufficient condition for the connectedness of $\mathcal{X}_E$ in Corollary 8.9 cannot be weakened to the case that only the $n-k+1$ largest coefficients of one row of the cost matrix C have to be pairwise different. If only c^1 but not c^2 satisfies this property, $\mathcal{X}_E$ does not have to be connected in general, as the following example shows.

Example 8.10 Let the matrix C be given by

$$C = \begin{pmatrix} 2 & 2 & 3 & 4 \\ 2 & 2 & 1 & 1 \end{pmatrix},$$

and consider the instance BBKP(C,2). The efficient set of this problem consists of the two efficient solutions $X^1 = (1,1,0,0)$ and $X^2 = (0,0,1,1)$, where $f(X^1) = (2,2)$ and $f(X^2) = (4,1)$, respectively. Obviously, these two solutions are not adjacent, since $d_H(X^1, X^2) = 4$, and $\mathcal{X}_E$ is not connected.

For completeness, we recall from Theorem 8.3 in Section 8.1:

Corollary 8.11 *Let $N = 2n$ for $n \in \mathbb{N}$. Then there exist matrices $C \in \mathbb{Z}^{2\times N}$ such that for the resulting problem instance BBKP(C,n) the distance between all pairs of efficient solutions of Problem* (BBKP) *is exactly n, i.e. at maximum.*

Proof: Consider the problem instance given by the matrix $C \in \mathbb{R}^{2\times N}$, where $c^1_i = 1 = c^2_{n+i}$ and $c^2_i = 2 = c^1_{n+i}$ for all $i \in \{1,\dots,n\}$. Then the instance BBKP(C,n) has exactly two efficient solutions

$$X^1 = (\underbrace{1,\dots,1}_{i=1,\dots,n}, 0,\dots,0) \text{ and } X^2 = (0,\dots,0, \underbrace{1,\dots,1}_{i=n+1,\dots,N})$$

with corresponding objective vectors $(1,2)$ and $(2,1)$, respectively. Since $d_H(X^1, X^2) = N$, exactly n exchanges are needed to generate X^1 from X^2 (and vice versa). □

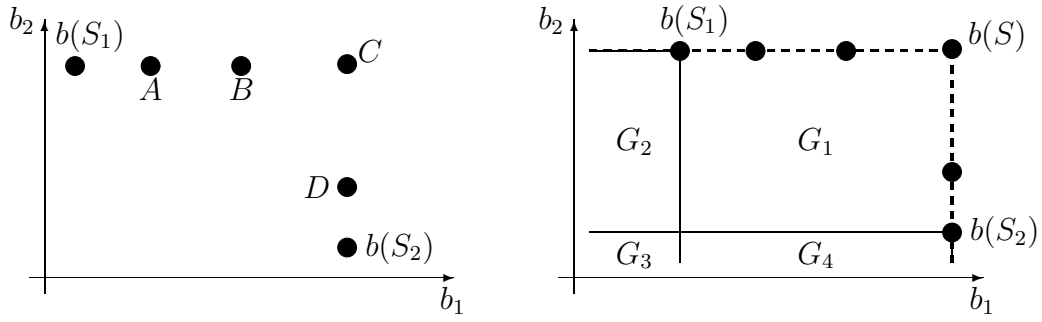


Figure 8.1: *Left: For efficient S_1 and S_2 the weakly non-dominated points A, B and D can be generated by an appropriate ε-constraint approach, while the point C is always dominated by $b(S_1)$ or $b(S_2)$. Right: For subsequent non-dominated solutions $b(S_1)$ and $b(S_2)$ the point $b(S)$ must be weakly non-dominated for Problem* (BBKP).

While the efficient set of the counter-example presented in the proof of Theorem 8.3 is not connected, one can easily verify that the set of weakly efficient solutions satisfies this property. In fact, all feasible solutions of the given problem instance are also weakly efficient. We prove in the following that this result does not only hold for this example problem, but that the set of weakly efficient solutions for Problem (BBKP) is always connected, where the adjacency of weakly-efficient solutions is based on a relaxed version of Definition 7.19.
To simplify the notation in the following, we use the combinatorial definition of the biobjective binary knapsack problem with fixed cardinality given by Problem (8.2) and its corresponding notation, rather than the notation of the problem description induced by Problem (BBKP) (cf. Subsection 8.2.1 for further details). Furthermore, we assume that $k \geq 2$, since the weakly efficient set of Problem (BBKP) is automatically connected for the case that $k = 1$.
Having a closer look at the given problem, we recognize that we have to consider two different types of weakly efficient solutions: those that can be generated by an ε-constraint approach, i.e. as a solution of the subproblem BKP($d^2_{s_\tau}$,k) for an appropriate $\tau \in \{0, \ldots, m-j_0\}$ and the others that are not optimal to any ε-constraint problem (cf. point C in the left subfigure of Figure 8.1). In the biobjective case these solutions are representatives of the weakly-efficient point $(b_1(S_2), b_2(S_1))$, where S_1 and S_2 denote representatives of two subsequent non-dominated solutions with $b_1(S_1) < b_1(S_2)$, i.e. there does not exist $S \in \mathcal{X}_E$ such that $b_1(S_1) < b_1(S) < b_1(S_2)$ and $b_2(S_2) < b_2(S) < b_2(S_1)$ holds true, whenever $(b_1(S_2), b_2(S_1))$ is a feasible point in the objective space. Obviously, these two types of weakly efficient solutions form a partition of $\mathcal{X}_{wE}$.

Lemma 8.12 *Let S_1 and S_2 denote representatives of two subsequent non-dominated solutions of Problem* (BBKP). *Then there exist feasible solutions $\bar{S}_1, \bar{S}_2 \in \mathcal{X}$ such that $b(\bar{S}_i) = (b_1(S_2), b_2(S_1))$ and $\bar{S}_i$ is adjacent to S_i for $i = 1, 2$.*

Proof: We show the lemma for $i = 1$, since the result for $i = 2$ follows by a simple exchange of the indices. So, let S_1 and S_2 be given as defined above, i.e. it holds that $b_1(S_1) < b_1(S_2)$ and $b_2(S_2) < b_2(S_1)$. For $j \in \{1, 2\}$ and $S \in \mathcal{X}$ we define

$$M_j(S) \; = \; \{e \in S : w_j(e) = b_j(S)\}.$$

Since $k \geq 2$ by assumption, we can choose $e_2 \in M_1(S_2)$ arbitrary, but fixed. Obviously,

$e_2 \notin S_1$, since otherwise this would imply that $b_1(S_1) < b_1(S_2) = w_1(e_2) \leq b_1(S_1)$. Now consider $S := (S_1 \cup \{e_2\}) \setminus \{e_1\}$, where $e_1 \in S_1 \setminus M_2(S_1)$ if $|M_2(S_1)| = 1$ and $e_1 \in M_2(S_1)$, whenever $|M_2(S_1)| > 1$ holds. By construction, we have that $|S| = k$, and $b_1(S) = w_1(e_2) = b_1(S_2)$ since $w_1(e) \leq b_1(S_1) < b_1(S_2)$ holds true for all $e \in S_1$. Furthermore, $b_2(S) = b_2(S_1)$, since $w_2(e_2) \leq b_2(S_2) < b_2(S_1)$ and, by the choice of e_1, there exists at least one item $e \in S \cap (S_1 \setminus \{e_1\})$ such that $w_2(e) = b_2(S_1)$. This completes the proof. □

Since S_1 and S_2 are defined to be representatives of two subsequent non-dominated solutions, $\bar{S}_1, \bar{S}_2$ are weakly efficient for Problem (BBKP), since if they would be strongly dominated, this would imply that either $b(S_1)$ and $b(S_2)$ do not correspond to two subsequent non-dominated solutions of the problem, or at least one of the two points is dominated (cf. also the right subfigure of Figure 8.1).

Lemma 8.13 *Let S_1 and S_2 denote representatives of two subsequent non-dominated solutions of Problem* (BBKP)*. Then there exists a sequence $\{S_i\}$ of weakly efficient solutions, starting from S_1 and ending in S_2, such that subsequent elements of this sequence are adjacent.*

Proof: Let $S \subseteq (S_1 \cup S_2)$ be arbitrary but fixed, such that $|S| = k$ and $S_1 \neq S \neq S_2$. We prove that S is a weakly efficient solution of Problem (BBKP). Since k different items from E always form a feasible knapsack of the problem, this automatically implies the existence of the desired sequence.
Assume that S is strongly dominated by $\tilde{S} \in \mathcal{X}$. Since by construction $w_1(e) \leq b_1(S_2)$ and $w_2(e) \leq b_2(S_1)$ holds for all $e \in S$, we may assume that S corresponds to a representative of the point $(b_1(S_2), b_2(S_1))$ in the objective space (for the construction of this solution, we refer to the proof of Lemma 8.12) that is dominated by $\tilde{S}$. Using the right subfigure of Figure 8.1, we conclude that $\tilde{S}$ must be contained in G_i for an $i \in \{1, \ldots, 4\}$. This yields a contradiction, since $\tilde{S}$ cannot be contained neither in G_1 (since S_1 and S_2 are defined to be representatives of two subsequent non-dominated solutions of problem) nor in G_i $(i = 2, 3, 4)$, since this would imply that either S_1, S_2 or even both efficient solutions would be dominated by $\tilde{S}$. □

We finally relate the weakly efficient solutions to the efficient solutions of the problem.

Lemma 8.14 *Let S denote a weakly efficient solution of Problem* (BBKP)*. Then there exist an efficient solution $\bar{S} \in \mathcal{X}$ and a sequence $\{S_i\}$ of weakly efficient solutions, starting from S and ending in $\bar{S}$, such that subsequent elements of this sequence are adjacent.*

Proof: If $S \in \mathcal{X}_E$, there is nothing to show. So, let $S \in (\mathcal{X}_{wE} \setminus \mathcal{X}_E)$. First, we assume that S is optimal for the single objective knapsack problem with objective b_1 and b_2, respectively. Hence, S corresponds to a node in the optimal solution transformation graph defined in the introduction of Section 8.2 for this problem, and by a result of Xu and Liu [218] there must exist a sequence of adjacent optimal solutions connecting S with any other optimal solution of this specific problem. Since all these solutions are weakly efficient for Problem (BBKP) (cf. Chankong and Haimes [36]) and at least one of these solutions is contained in $\mathcal{X}_E$, as it corresponds to a lexicographic optimum of Problem (BBKP), the lemma is proven for this case.

It remains the case that S is not optimal for none of the two single objective knapsack problems with objective b_1 and b_2, respectively. Hence, there must exist $S_1, S_2 \in \mathcal{X}_E$ such that $b_1(S_1) \leq b_1(S) \leq b_1(S_2)$ and $b_2(S_2) \leq b_2(S) \leq b_2(S_1)$, with at least one strict inequality in each case, where $b(S_1)$ and $b(S_2)$ denote subsequent non-dominated solutions of Problem (BBKP). First, consider the case that $b(S) = (b_1(S_2), b_2(S_1))$. Following the same line of argument as in the proof of Lemma 8.13, it is easy to verify that all knapsacks $\bar{S} \subseteq (S_1 \cup S)$ with $|\bar{S}| = k$ correspond to weakly efficient solutions of Problem (BBKP), and hence, S_1 and S can be connected by a sequence of weakly-efficient solutions.
For the remaining case that S is an optimal solution of an ε-constraint problem with objective b_2 (or b_1) and an appropriately chosen constraint on the other objective, we assume that $b_2(S) = b_2(S_1)$ and $b_1(S_1) < b_1(S) < b_1(S_2)$ holds in the following. For the opposite case, just change the order of b_1 and b_2. By construction, S as well as S_1 are both optimal for the ε-constraint problem given by Problem (8.3) with right hand side $b_1(S)$. Hence, both solutions correspond to nodes of the optimal solution transformation graph of the single objective problem BKP($d^2_{s_\tau}$,k) for an appropriately chosen $\tau \in \{1, \ldots, m - j_0\}$. By the results of Chankong and Haimes [36] and Xu and Liu [218], there must exist a sequence of optimal solutions to this ε-constraint problem, i.e. weakly efficient solutions of Problem (BBKP), connecting S with S_1 such that subsequent elements of this sequence are adjacent. □

Note once more that whenever we have that $b(S) = (b_1(S_2), b_2(S_1))$ in the proof of Lemma 8.14, S is no longer an optimal solution of Problem (8.3) with right hand side $b_1(S)$, since in this case also S_2 is a feasible solution satisfying $b_2(S_2) < b_2(S)$ (cf. Figure 8.1). We conclude:

Theorem 8.15 *The set $\mathcal{X}_{wE}$ of weakly efficient solutions for Problem* (BBKP) *is always connected.*

Proof: By Lemma 8.14, each weakly efficient solution can be connected to an efficient solution by a sequence that consists of weakly efficient solutions only. Furthermore, by Lemma 8.13, two efficient solutions that correspond to representatives of two subsequent non-dominated solutions of Problem (BBKP) can be connected by a sequence of adjacent weakly efficient solutions. Hence, it suffices to show that two different efficient representatives of the same non-dominated solution can also be connected by a sequence of weakly efficient solutions. But this is once more implied by the results of Chankong and Haimes [36] and Xu and Liu [218]. □

8.3 Conclusions and Further Ideas

In this chapter we investigated the connectedness of the efficient set for general multiple objective combinatorial optimization problems with several bottleneck objectives. Based on the main concepts and definitions for analyzing the connectedness of the efficient set developed in Chapter 7, we showed that the efficient set is not connected for most of the classical combinatorial problems, when bottleneck objectives are involved. Actually, the presented counter-example does not depend on a special class of combinatorial problems but rather on the cardinality of the feasible solutions. If all

feasible solutions have the same cardinality, non-connectedness can be proven for the general case.
Despite of this negative result, we further presented a solution approach for solving the fixed cardinality constraint biobjective binary knapsack problem with two bottleneck objectives. The developed algorithm can be seen as a special case of Algorithm 4.2 stated in Section 4.2.2. Based on this algorithm, we derived a sufficient condition for the connectedness of the efficient set of this problem. Furthermore, also the connectedness of the weakly efficient set for arbitrary instances was proven. To the best of our knowledge, this is the first non-trivial problem found so far, where the set of efficient solutions is not connected in general, but where the weakly efficient set satisfies this property independently from any special input data.
Concerning future streams of research on this topic, one might suggest to investigate, whether the property of a connected weakly efficient set for the considered knapsack problem still holds, when more than two bottleneck objectives are considered.
We already mentioned at the beginning of this chapter that neighborhood search based solution concepts for multiple objective problems with bottleneck objectives are mainly relevant for problems where the single objective problem is already $\mathcal{NP}$-hard to solve (like it is the case for the bottleneck traveling salesman problem). Hence, one could focus on these problems and try to investigate, whether the weakly efficient set of these problems is connected or not. Maybe it is also possible to derive conditions under which the set of efficient solutions of such problems is ensured to be connected, like it is possible for the considered knapsack problem in Section 8.2.4.

Chapter 9

Greedy Algorithms for a Class of Knapsack Problems with Binary Weights

The binary multidimensional knapsack problem is a classical $\mathcal{NP}$-hard problem with many applications and for which several theoretical results are known (see, e.g., Weingartner and Ness [213] and Kellerer et al. [113]). In general, the problem consists in selecting a subset of given objects (or items) in such a way that the total profit of the selected objects is maximized, while a set of $m \geq 1$ knapsack constraints are satisfied. Due to its complexity, exact algorithms can only solve small to medium sized instances in a reasonable amount of time. For this reason, many heuristic procedures have been proposed in the literature, for example in the articles of Chu and Beasley [38], Glover and Kochenberger [80], Raidl and Gottlieb [179], Tavares et al. [200] and Vasquez and Vimont [207], only to mention a few.

Following the ideas presented in Chapter 3, transforming the m constraints into m minimizing objectives, we obtain a special case of the multiple objective unconstrained combinatorial optimization problem (cf. also Section 7.2.8). This problem is also $\mathcal{NP}$-hard to solve (cf. [54]) and to the best of our knowledge no algorithm has been proposed to solve it.

In this chapter, we consider the binary m-dimensional knapsack problem with binary weight coefficients and the $(m+1)$-objective unconstrained optimization problem with m binary criteria coefficients. We show that for $m = 2$, the problems above can be solved to optimality in polynomial time by following a simple greedy strategy.

Two additional aspects are worthwhile noting. First, the greedy algorithm for the triobjective unconstrained problem with binary criteria coefficients provides a constructive proof that the set of the efficient solutions for this problem is connected according to a combinatorial definition of adjacency of efficient solutions for the given problem. For the main concepts and results concerning the connectedness of the efficient set, we refer to Chapter 7. Second, the cardinality of the non-dominated set is bounded by a polynomial function of the number of items, which is often not the case in multiple objective combinatorial optimization (see, e.g., Ehrgott [53]). The greedy algorithm proposed in this chapter solves the triobjective problem in polynomial time and takes constant amount of time to find each efficient solution after a pre-processing step. We further show that the algorithm is optimal in terms of upper bound time complexity. Additional numerical experiments indicate that this approach is indeed

very efficient in practice.
The chapter is organized as follows. In Section 9.1, we introduce the notation and the problems. Furthermore, we present the pre-processing phase that is common to all algorithms described in this section. The greedy algorithm for the binary two-dimensional problem with equality constraints is presented in Section 9.2. The algorithms for the triobjective unconstrained problem and for the binary two-dimensional problem with inequality constraints are discussed in Sections 9.3 and 9.4, respectively. The results on the connectedness of the efficient set are proven in Section 9.5, and we conclude in Section 9.6.
Note that the results presented in this chapter are also published as a technical report in Gorski et al. [86].

9.1 Notation and Pre-Processing

Let $\mathcal{E}$ denote a set of n items. A subset $S \subseteq \mathcal{E}$ of items from $\mathcal{E}$ is called a *knapsack* in the following. The set of all feasible knapsacks is denoted by $\mathcal{X}$. For $j \in \{1, \ldots, m\}$, the profit of each item $s \in \mathcal{E}$ and its weight at dimension j is given by $p(s) > 0$ and $w^j(s) \geq 0$, respectively. The maximum capacity of a knapsack at dimension j is denoted by c^j. We define by $p(S) = \sum_{s \in S} p(s)$ and $w^j(S) = \sum_{s \in S} w^j(s)$ the total profit and the total weight at dimension j of the items in knapsack S, respectively. In our particular case, we assume that $w^j(s)$ can only take binary values for all items $s \in \mathcal{E}$.
We define the *binary m-dimensional knapsack problem with binary weights* (m-KP$_{\leq}$) as follows:

Definition 9.1 *Given a finite set $\mathcal{E}$, for each $s \in \mathcal{E}$ a profit $p(s) > 0$ and a weight $w^j(s) \in \{0, 1\}$, and a non-negative integer c^j, find a subset $S \in \mathcal{X}$ such that $p(S)$ is maximal and $w^j(S) \leq c^j$, for $j = 1, \ldots, m$.*

When $m = 1$, Problem (m-KP$_{\leq}$) simplifies to a sequential knapsack problem with divisible weights, which is solvable in $\mathcal{O}(n \log n)$ time (cf. Hartmann and Olmstead [99]). We introduce the following special case of (m-KP$_{\leq}$), the *binary m-dimensional knapsack problem with equality constraints and binary weights* (m-KP$_{=}$):

Definition 9.2 *Given a finite set $\mathcal{E}$, for each $s \in \mathcal{E}$ a profit $p(s) > 0$ and a weight $w^j(s) \in \{0, 1\}$, and a non-negative integer c^j, find a subset $S \in \mathcal{X}$ such that $p(S)$ is maximal and $w^j(S) = c^j$, for $j = 1, \ldots, m$.*

If we transform the m constraints of Problems (m-KP$_{=}$) and (m-KP$_{\leq}$) into m objectives to minimize (cf. also Chapter 3 for this idea), we obtain a variant of the multiple objective unconstrained combinatorial optimization problem, the *$(m+1)$-objective unconstrained combinatorial optimization problem with m binary weights* (m-MP), that is defined as follows.

Definition 9.3 *Given a finite set $\mathcal{E}$, for each $s \in \mathcal{E}$ a profit $p(s) > 0$ and a weight $w^j(s) \in \{0, 1\}$, find a subset $S \in \mathcal{X}$ such that $p(S)$ is maximal and $w^j(S)$ is minimal, for $j = 1, \ldots, m$.*

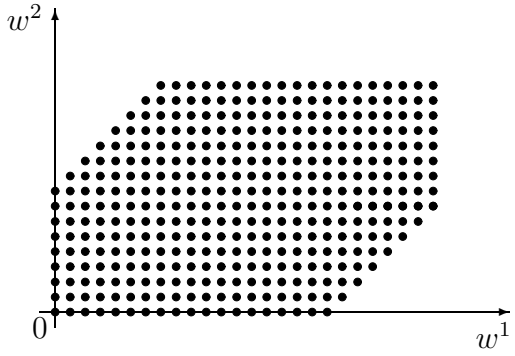


Figure 9.1: *The hexagonal grid $\mathcal{G}$ in the $\mathbb{R}^2$-plane.*

Note that Problem (m-MP) is nothing else then the associated multiple objective optimization problem of Problem (m-KP$_{\leq}$) (cf. Section 3.1). The set of efficient knapsacks of Problem (m-MP) is denoted by $\mathcal{X}_E$ in the following, while its corresponding set of non-dominated points in the objective space is given by $\mathcal{Y}_N$.
We recall from Chankong and Haimes [36] that a given efficient knapsack of Problem (m-MP) corresponds to an optimal knapsack for Problem (m-KP$_{\leq}$) for appropriately chosen, non-negative integers $c^1, \ldots, c^m$. Moreover, if there exists an optimal knapsack for Problem (m-KP$_{\leq}$) with non-negative integers $c^1, \ldots, c^m$, then this knapsack is at least weakly-efficient for Problem (m-MP). Note that if there exists an optimal knapsack for Problem (m-KP$_{=}$) with non-negative integers $c^1, \ldots, c^m$, this knapsack may not even be weakly-efficient for Problem (m-MP) in general.
In the following, we are particularly interested in the two-dimensional case ($m = 2$) of Problems (m-KP$_{\leq}$) and (m-KP$_{=}$). For the latter problem, we denote the set of all optimal knapsacks for the constraint (c^1, c^2) by $\mathcal{S}(c^1, c^2)$. If $c^1 = 0$ (or $c^2 = 0$) we call $\mathcal{S}(0, c^2)$ (or $\mathcal{S}(c^1, 0)$) a *basis* with respect to c^2 (or c^1). Moreover, $\rho(c^1, c^2)$ denotes the optimal profit value for this problem with constraint (c^1, c^2), i.e. $\rho(c^1, c^2) = p(S)$ where $S \in \mathcal{S}(c^1, c^2)$.
For the Problem (m-MP) with $m = 2$, we consider the problem of finding the set $\mathcal{Y}_N$ since to find all efficient knapsacks is an intractable task (cf. Ehrgott [54]).
To illustrate the ideas developed in the following, we use a geometric interpretation of the image of all feasible solutions for the two weight objectives w^1 and w^2 as a set

$$\mathcal{G} := \left\{ \left(w^1(S), w^2(S) \right), S \subseteq \mathcal{E} \right\}$$

that forms an hexagonal grid in the $\mathbb{R}^2$-plane (cf. Figure 9.1). Since for each combination of the weight objectives w^1 and w^2 there can exist at most a single non-dominated solution, $|\mathcal{G}| \in \mathcal{O}(n^2)$ obviously defines a strict upper bound on the cardinality of the non-dominated set.
In the following sections, we present greedy algorithms to solve the three problems defined above. After a pre-processing phase, the algorithm solves Problem (2-KP$_{=}$) by inserting items into the knapsack according to a given sequence of items. The non-dominated set of Problem (2-MP) is found by iteratively solving the previous problem for several constraints. Finally, Problem (2-KP$_{\leq}$) is solved based on the results for Problem (2-MP).
Note that the following pre-processing step is common to all algorithms presented in this section. It consists of partitioning the set of items and sorting its elements with

respect to their profit values. Without loss of generality, we assume that $w^1(s) + w^2(s) \geq 1$ for all items s of the problem. Note that items with null weights will only augment the profit value of a knapsack. Therefore, we can remove those items, store the sum of their profits and solve the problem for the remaining ones.
We partition the set of items according to their weights $(w^1(s), w^2(s))$ for all items s and obtain three different sets where all elements in a set have the weights $(1, 0)$, $(0, 1)$ and $(1, 1)$, respectively. We denote these sets by R, U and D, respectively, and their cardinalities by n_R, n_U and n_D. Without loss of generality, we assume that $n_U \leq n_R$. Next, we sort the elements of each set in non-increasing order of the profit values. We store the profit values of these items in the sequences $r = (r_1, r_2, \ldots, r_{n_R})$, $u = (u_1, u_2 \ldots, u_{n_U})$ and $d = (d_1, d_2, \ldots, d_{n_D})$.
In the following subsections, we will interleave between sequences of items and sequences of profits. The correspondence between an item in the sequence U, R or D and its profit in the sequence u, r or d should be clear from the context.

9.2 The Knapsack Problem with Two Equality Constraints

The greedy algorithm described in this section returns an optimal knapsack for Problem (2-KP$_=$) for an arbitrary constraint (c^1, c^2) in $\mathcal{G}$. The algorithm starts by finding an optimal knapsack for a given basis and proceeds by filling it with items taken from sets U, R and D based on the decomposition

$$\begin{pmatrix} w^1(S) \\ w^2(S) \end{pmatrix} = (c^1 - c^2) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + c^2 \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} c^1 \\ c^2 \end{pmatrix} \tag{9.1}$$

for $c^1 \geq c^2$ and a similar decomposition for $c^2 \geq c^1$, respectively.
For the sake of the explanation, we call as *super-item* a pair of items where one is taken from set R and the other is taken from set U. A super-item has a weight $(1, 1)$ and its profit is the sum of the profits of the two items.
For the remaining results of this section, we state the following remark.

Remark 9.4 *For a given constraint $(c^1, c^2) \in \mathcal{G} \setminus \{(0, 0)\}$, there is at least one optimal knapsack that contains the first items from set U, R or D.*

The first lemma states that it is easy to find the optimal profit value of knapsacks in a basis.

Lemma 9.5 *Let $c^1 \in \{1, \ldots, n_R\}$ and $c^2 \in \{1, \ldots, n_U\}$. Then, for all knapsacks $S \in \mathcal{S}(c^1, 0)$ and $S' \in \mathcal{S}(0, c^2)$ it holds*

$$p(S) = \sum_{i=1}^{c^1} r_i \quad \text{and} \quad p(S') = \sum_{i=1}^{c^2} u_i.$$

Proof: Let $S \in \mathcal{S}(c^1, 0)$. Obviously, knapsack S cannot contain any items of sets U and D. Hence, we assume that S corresponds to the knapsack containing the first c^1 items

of R. Since the sequence r is sorted in non-increasing order, $p(S) = \sum_{i=1}^{c^1} r_i$ is the optimal profit criterion value of (2-KP$_=$). A similar reasoning applies for $S' \in \mathcal{S}(0, c_2)$. □

Starting from an optimal knapsack in a basis, we establish the connection between these knapsacks and optimal knapsacks in $\mathcal{S}(c^1, c^2)$.

Lemma 9.6 *Let $(c^1, c^2) \in \mathcal{G}$ with $0 < c^2 < c^1$. Then, there exists an optimal knapsack $S \in \mathcal{S}(c^1, c^2)$ that contains all items of an optimal knapsack $S' \in \mathcal{S}(c^1 - c^2, 0)$.*

Proof: Assume that the statement is false. Following Remark 9.4 and Lemma 9.5, let $S' \in \mathcal{S}(c^1 - c^2, 0)$ contain the first $c^1 - c^2$ items of R and let $S \in \mathcal{S}(c_1, c_2)$ contain the first k items of R. Clearly, $k < c^1 - c^2$, since otherwise S would contain all items of S'. Consider a knapsack $\tilde{S}$ that contains only the items from S that belong to R and a knapsack $\bar{S}$ that contains the remaining items from S. Then, $\bar{S}$ only contains items from U and D, i.e. $w^1(\bar{S}) - w^2(\bar{S}) \leq 0$. However, since S is feasible, we have that

$$\begin{aligned} w^1(\bar{S}) - w^2(\bar{S}) &= w^1(S) - w^1(\tilde{S}) - \Big(w^2(S) - w^2(\tilde{S})\Big) = c^1 - k - \big(c^2 - 0\big) \\ &= c^1 - c^2 - k > c^1 - c^2 - \big(c^1 - c^2\big) = 0. \end{aligned}$$

This contradicts our previous result. □

Clearly, a similar result to Lemma 9.6 can be obtained for the case that $0 < c^1 < c^2$. Lemma 9.6 suggests a greedy algorithm to solve Problem (2-KP$_=$) for a given $(c^1, c^2) \in \mathcal{G}$. Assume that $0 < c^2 \leq c^1$. First, fill the knapsack with the first $c^1 - c^2$ items from R. Then repeat the following procedure c^2 times:

(i) Select the three items with the largest profit in R, U and D, respectively, that are not in the knapsack; let the two items from R and U correspond to a super-item.

(ii) From the item of D or the super-item of R and U, insert the one with the largest profit into the knapsack.

Let $\bar{D}$ denote the sequence of these last c^2 (super-)items. The following theorem states that the application of the above given procedure results in an optimal knapsack for a given instance of (2-KP$_=$) where $0 < c^2 \leq c^1$.

Theorem 9.7 *Let $(c^1, c^2) \in \mathcal{G}$ such that $0 < c^2 \leq c^1$, and let S denote the knapsack that includes the first $c^1 - c^2$ items from R and the c^2 (super-)items from $\bar{D}$. Then $S \in \mathcal{S}(c^1, c^2)$.*

Proof: According to the description above, S is a feasible knapsack and satisfies the decomposition (9.1). Now, assume that S is not optimal, i.e. there exists another feasible knapsack S' such that $p(S') > p(S)$. According to the construction of S as well as the result from Remark 9.4 and Lemma 9.6, we may assume that S' includes the first $c^1 - c^2$ elements of R. Therefore, the weights of the remaining items in S' must sum up to (c^2, c^2), since otherwise S' would not be a feasible knapsack. However, this is only possible if the cardinality of items from U and the cardinality of additional items from R coincide since they do not augment the value of the two weight coefficients simultaneously as all the items in D do. Hence, S' must contain a combination of

items (aside from the first $c^1 - c^2$ items of R) whose total profit is equal to the sum of the c^2 profit values of (super-)items in $\bar{D}$. However, this means that $p(S') = p(S)$, which contradicts the assumption that S is not in $\mathcal{S}(c^1, c^2)$. □

Using the same reasoning as above, we can construct an optimal knapsack for the case where $0 < c^1 < c^2$. This optimal knapsack contains the first $c^2 - c^1$ items of U and the c^2 items of the sequence $\bar{D}$, which corresponds to the sequence of items from D and appropriately combined super-items of R and the remaining items in U.
The algorithm for computing the optimal profit value for Problem (2-KP$_=$) with constraint (c^1, c^2) is given by Algorithm A.1 in Appendix A. We assume without loss of generality that d_j, r_{c+i} and u_i always exist in the outline of Algorithm A.1. We omit to give an additional outline for the case that $0 < c^1 < c^2$.
Note that since sequences R, U and D are sorted according to the profits, we can find the optimal knapsack in linear time after the pre-processing phase. The algorithm takes $\mathcal{O}(n \log n)$ time due to the sorting step at the pre-processing phase.

9.3 The Unconstrained Triobjective Optimization Problem with Two Binary Weights

Based on the results of the previous section, we can derive a straight-forward algorithm that finds the non-dominated set for Problem (2-MP) in polynomial time: First, call the greedy algorithm to solve Problem (2-KP$_=$) for every constraint $(c^1, c^2) \in \mathcal{G}$ (see Section 9.2) and store each optimal profit found and corresponding weight vector as a tuple; then, remove the dominated tuples. Clearly, the remaining set of tuples corresponds to the non-dominated set of Problem (2-MP). Since the removal of dominated tuples can be performed in $\mathcal{O}(n \log n)$ [121], this algorithm has $\mathcal{O}(n^3 \log n)$ time complexity. In this section we present an improvement on the algorithm above that reduces the time complexity to $\mathcal{O}(n^2)$. The resulting algorithm is optimal in terms of upper bound time complexity.
The remaining parts of this section are organized as follows: After proving important dominance relations for different pairs of $(c^1, c^2) \in \mathcal{G}$, we derive a strict lower bound on the cardinality of the non-dominated set. Using these results we further show that we can find the non-dominated set of Problem (2-MP) without even applying the filtering step to remove dominated knapsacks at the end.
In the following we will focus on the case that $0 < c^2 \leq c^1$ and only mention equivalent results for the case that $0 < c^1 < c^2$ briefly. Hence, let $0 < c^2 \leq c^1$, let $c = c^1 - c^2$ and let $\lambda = \min\{n_R - c, n_U\}$. For a given basis $\mathcal{S}(c, 0)$, $c \in \{0, \ldots, n_R\}$, let $\bar{D}^c$ denote the sequence of λ (super-)items that are chosen according to the greedy algorithm described in Section 9.2. Moreover, we store the profits of the elements in $\bar{D}^c$ in the sequence $\bar{d}^c$ and the profits of the super-items in the sequence $\tilde{d}^c$. In this case we say that these sequences *correspond* to the given basis c. For the remaining results of this section, we introduce the following corollary.

Corollary 9.8 *Let $(c^1, c^2) \in \mathcal{G}$ such that $c^1 = n_R + n_D$. Then there exists a sequence $\{S^i\}_i$ of knapsacks such that $S^i \in \mathcal{S}(c^1 - c^2 + i, i)$ for $i = 0, \ldots, c^2$ and S^i and S^{i+1} differ in exactly one (super-)item in the sequence $\bar{D}^{c^1 - c^2}$.*

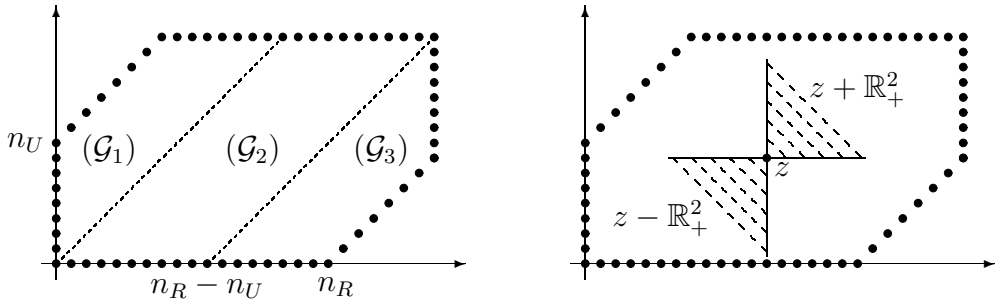


Figure 9.2: *Sectors and Dominance.*

Proof: The proof follows directly from Theorem 9.7. Let $S^0 \in \mathcal{S}(c^1 - c^2, 0)$ be the knapsack that contains the first $c^1 - c^2$ items of R and let S^i contain all items from S^0 and the first i (super-)items in sequence $\bar{D}^{c^1-c^2}$ for $i \in \{1, \ldots, c^2\}$. Then, $S^i \in \mathcal{S}(c^1 - c^2 + i, i)$ for $i = 0, \ldots, c^2$ and S^i and S^{i+1} differ in exactly one (super-)item in the sequence $\bar{D}^{c^1-c^2}$. □

Clearly, a similar result holds for the case that $c^2 = n_U + n_D$. Corollary 9.8 suggests that Problem (2-KP$_=$) can be solved for several constraint values by starting from a knapsack in a basis and subsequently adding c^2 items from the sequence $\bar{D}^{c^1-c^2}$ for $c^1 = n_R + n_D$. By repeating this procedure for each basis, we obtain an algorithm that finds the profit values of Problem (2-KP$_=$) for all constraints in the grid $\mathcal{G}$ in $\mathcal{O}(n^2)$ time. Since $\mathcal{O}(n^2)$ have to be filtered for dominance, the non-dominated set for Problem (2-MP) can be found in $\mathcal{O}(n^2 \log n)$.
In the remaining part of this section, we will present an improved approach that solves Problem (2-MP) in $\mathcal{O}(n^2)$ time. In more detail, we will show in the following that we may not need to consider the complete sequence $\bar{D}^c$ as defined above and that the removal of dominated knapsacks does not even need to be performed. For this purpose, we have to split the grid $\mathcal{G}$ into three sectors $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ defined as follows: $\mathcal{G}_1$ corresponds to the points of the grid that do not lie under the line segment connecting the points $(0, 0)$ and $(n_U + n_D, n_U + n_D)$; $\mathcal{G}_2$ consists of all points in the grid between the line segments connecting the points $(0, 0)$ and $(n_U + n_D, n_U + n_D)$ and the points $(n_R - n_U, 0)$ and $(n_R + n_D, n_U + n_D)$, respectively; the remaining points of the grid form $\mathcal{G}_3$ (cf. also the left subfigure of Figure 9.2). Note that all integer points on the border of two sectors belong to both sectors and that it is assumed that $n_U \leq n_R$.

9.3.1 Dominance Relations in $\mathcal{G}$

In the following we establish *dominance relations* among points in $\mathcal{G}$. For a constraint $(c^1, c^2) \in \mathcal{G}$ we know that a knapsack $S \in \mathcal{S}(c^1, c^2)$ can be dominated by any other knapsack in $\mathcal{S}((c^1, c^2) - \mathbb{R}^2) \cap \mathcal{G})$ and that S potentially dominates knapsacks in $\mathcal{S}((c^1, c^2) + \mathbb{R}^2) \cap \mathcal{G})$ by definition of Problem (2-MP) (cf. also Figure 9.2). In addition, while knapsack S can never dominate (or be dominated by) knapsacks in $\mathcal{S}(c^1 + 1, c^2 + 1)$ (or in $\mathcal{S}(c^1 - 1, c^2 - 1)$) by the construction of the sequences in Corollary 9.8, this can be the case for all knapsacks in $\mathcal{S}(c^1 + 1, c^2)$ or in $\mathcal{S}(c^1, c^2 + 1)$ (or in $\mathcal{S}(c^1 - 1, c^2)$ or in $\mathcal{S}(c^1, c^2 - 1)$). In the following we will focus on these cases.
To simplify the notation, we say that there exists *horizontal dominance*, *to the right* or

from the left when an optimal knapsack to Problem (2-KP$_=$) with constraint $(c^1, c^2) \in \mathcal{G}$ dominates (or is dominated by) his right (left) neighbor, i.e. an optimal knapsack to Problem (2-KP$_=$) with constraint $(c^1+1, c^2) \in \mathcal{G}$ (or $(c^1-1, c^2) \in \mathcal{G}$, respectively). Furthermore, we say that we have *vertical dominance*, *to the top* or *from the bottom* whenever an optimal knapsack to Problem (2-KP$_=$) with constraint $(c^1, c^2) \in \mathcal{G}$ dominates (or is dominated by) his top (bottom) neighbor, i.e. an optimal knapsack to Problem (2-KP$_=$) with constraint $(c^1, c^2+1) \in \mathcal{G}$ (or $(c^1, c^2-1) \in \mathcal{G}$, respectively). We will show in the following that vertical dominance can apply in $\mathcal{G}_1$ and $\mathcal{G}_2$ while horizontal dominance is not possible in these sectors. For Sector $\mathcal{G}_3$ we will prove that horizontal dominance is possible but never vertical dominance.
We start by introducing two lemmas that will be useful for deriving the main results about vertical and horizontal dominance in the three sectors. Starting from a knapsack $S^0 \in \mathcal{S}(0, c)$ where $c \in \{0, \ldots, n_U\}$ we have that

$$\rho(n_U + n_D - c, n_U + n_D) = \sum_{i=1}^{n_U - c} r_i + \sum_{i=1}^{n_U} u_i + \sum_{i=1}^{n_D} d_i, \tag{9.2}$$

in $\mathcal{G}_1$, according to Corollary 9.8. For $c \in \{0, \ldots, n_R - n_U\}$ and $S^0 \in \mathcal{S}(c, 0)$ Corollary 9.8 provides that

$$\rho(n_U + n_D + c, n_U + n_D) = \sum_{i=1}^{n_U + c} r_i + \sum_{i=1}^{n_U} u_i + \sum_{i=1}^{n_D} d_i, \tag{9.3}$$

in $\mathcal{G}_2$, whereas in $\mathcal{G}_3$ we have that

$$\rho(n_R + n_D, n_R + n_D - c) = \sum_{i=1}^{n_R} r_i + \sum_{i=1}^{n_R - c} u_i + \sum_{i=1}^{n_D} d_i, \tag{9.4}$$

where $c \in \{n_R - n_U, \ldots, n_R\}$ and $S^0 \in \mathcal{S}(c, 0)$. Since all profits are positive by assumption, it follows from (9.2), (9.3) and (9.4):

Lemma 9.9 *The finite sequences*

$$\{\rho(n_D + i, n_U + n_D)\}_{i=0}^{n_R} \text{ and } \{\rho(n_D + n_R, n_D + i)\}_{i=0}^{n_U}$$

are strictly increasing.

Focusing on the Sectors $\mathcal{G}_2$ and $\mathcal{G}_3$ we establish the following relations.

Lemma 9.10 *Let $c \in \{0, \ldots, n_R\}$ and let the sequence $\bar{d}^c$ correspond to the basis $\mathcal{S}(c, 0)$.*

1. *Let $c \in \{0, \ldots, n_R - 1\}$ and $i \in \{1, \ldots, n_R - c\}$. Then $\bar{d}^c_k \geq \bar{d}^{c+i}_k$ for all $k \in \{1, \ldots, n_D + \min(n_R - (c+i), n_U)\}$.*
2. *Let $c \in \{0, \ldots, n_R - n_U - 1\}$. Then $\bar{d}^c_{k+1} \leq \bar{d}^{c+1}_k$ for all $k \in \{1, \ldots, n_U + n_D - 1\}$.*
3. *Let $c \in \{n_R - n_U, \ldots, n_R - 1\}$. Then $\bar{d}^c_{k+1} \leq \bar{d}^{c+1}_k$ for all $k \in \{1, \ldots, n_R + n_D - (c+1)\}$.*

Proof: The proof of (1.) follows immediately from the construction of the sequences $\bar{d}^c$ and $\bar{d}^{c+i}$.
To prove (2.) we distinguish two cases. For the first case we assume that the first element of the sequence $\bar{d}^c$ corresponds to the profit of a super-item. Then, n_D of the remaining $n_U + n_D - 1$ elements of $\bar{d}^c$ coincide with n_D elements of $\bar{d}^{c+1}$ since both sequences contain the n_D profit values of the items in D. In addition, by the construction of the super-items it holds that

$$\tilde{d}^c_{k+1} \;=\; r_{c+k+1} + u_{k+1} \;\leq\; r_{c+k+1} + u_k \;=\; \tilde{d}^{c+1}_k$$

for all $k \in \{1, \ldots, n_U - 1\}$. Since the elements of the sequences are sorted in non-increasing order, this implies that $\bar{d}^{c+1}_k \geq \bar{d}^c_{k+1}$ for all $k \in \{1, \ldots, n_U + n_D - 1\}$. For the second case, let $\ell > 1$ correspond to the index of the profit of the first super-item contained in the sequence $\bar{d}^c$. For $k \in \{1, \ldots, \ell-2\}$ it holds that $\bar{d}^c_k = d_k \geq d_{k+1} = \bar{d}^c_{k+1}$ by construction and, in addition, we have that $\bar{d}^c_\ell = \tilde{d}^c_1 = r_{c+1} + u_1 \geq r_{c+2} + u_1 = \tilde{d}^{c+1}_1$. This implies that $\bar{d}^{c+1}_k = d_k$ for $k = 1, \ldots, \ell - 1$ and Part (2.) is true at least until $k = \ell$. For the remaining indices we can apply the same reasoning as in the first case taking into account that the remaining elements of the sequence $\bar{d}^c$ and $\bar{d}^{c+1}$ only coincide in the $n_D - (\ell - 1)$ profit values $d_\ell, \ldots, d_{n_D}$ of items contained in D. This completes the proof for the second case.
The proof of (3.) is similar to the proof of Part (2.). □

We will use Lemma 9.9 and Lemma 9.10 to derive results for vertical and horizontal dominance between neighbor points in the same sector.

Theorem 9.11 *For Sector $\mathcal{G}_3$ it holds:*

1. *Let $(c^1, c^2) \in \mathcal{G}$ such that $n_R - n_U \leq c^1 - c^2 \leq n_R - 1$ and $n_R - n_U \leq c^1 \leq n_R + n_D - 1$. If $S \in \mathcal{S}(c^1, c^2)$ dominates $S' \in \mathcal{S}(c^1+1, c^2)$ then $\bar{S} \in \mathcal{S}(c^1+j, c^2+j)$ also dominates $\bar{S}' \in \mathcal{S}(c^1+j+1, c^2+j)$, where $j \in \{0, \ldots, n_D + n_R - (c^1+1)\}$.*

2. *Let $c \in \{n_R - n_U, \ldots, n_R - 1\}$ and let $S \in \mathcal{S}(c+i+1, i)$ and $S' \in \mathcal{S}(c+i+1, i+1)$, where $i \in \{0, \ldots, n_R + n_D - (c+1)\}$. Then S does not dominate S'.*

Proof: To prove (1.), let S dominate S', i.e. $p(S) \geq p(S')$. Since according to Lemma 9.10 it holds that $\bar{d}^{c^1-c^2}_k \geq \bar{d}^{c^1-c^2+1}_k$ for all $k \in \{1, \ldots, n_D + n_R - (c^1 - c^2 + 1)\}$, we have that

$$\begin{aligned} p(\bar{S}) \;&=\; p(S) + \sum_{k=c^2+1}^{c^2+j} \bar{d}^{c^1-c^2}_k \;\geq\; p(S) + \sum_{k=c^2+1}^{c^2+j} \bar{d}^{c^1-c^2+1}_k \\ &\geq\; p(S') + \sum_{k=c^2+1}^{c^2+j} \bar{d}^{c^1-c^2+1}_k \;=\; p(\bar{S}'), \end{aligned}$$

for $j \in \{1, \ldots, n_D + n_R - (c^1 + 1)\}$. This implies that $\bar{S}$ dominates $\bar{S}'$.
To prove (2.), we assume that there exists an index $i \in \{0, \ldots, n_R + n_D - (c+1)\}$ such that $S \in \mathcal{S}(c+i+1, i)$ dominates $S' \in \mathcal{S}(c+i+1, i+1)$. Using the fact that $\bar{d}^{c+1}_k \geq \bar{d}^c_{k+1}$ for all $k \in \{1, \ldots, n_R + n_D - (c+1)\}$ from Lemma 9.10 this would imply that $\bar{S} \in \mathcal{S}(n_R + n_D, n_R + n_D - (c+1))$ dominates $\bar{S}' \in \mathcal{S}(n_R + n_D, n_R + n_D - c)$ and

hence $\rho(n_R+n_D, n_R+n_D-(c+1)) \geq \rho(n_R+n_D, n_R+n_D-c)$, which is a contradiction to Lemma 9.9. □

We established that there can exist horizontal dominance but never vertical dominance in Sector $\mathcal{G}_3$. Note that the same reasoning as in Part (1.) of Theorem 9.11 can be applied for Sector $\mathcal{G}_2$ assuming that there is dominance to the right somewhere in this sector. Nevertheless, the next theorem shows that the assumption of dominance to the right is never met in $\mathcal{G}_2$.

Theorem 9.12 *For Sector $\mathcal{G}_2$ it holds:*

1. *Let $c \in \{0, \ldots, n_R - n_U - 1\}$ and let $S \in \mathcal{S}(c+i, i)$ and $S' \in \mathcal{S}(c+i+1, i)$, where $i \in \{0, \ldots, n_U + n_D\}$. Then S does not dominate S'.*
2. *Let $(c^1, c^2) \in \mathcal{G}$ such that $1 \leq c^1 - c^2 \leq n_R - n_U$ and $0 \leq c^2 \leq n_U + n_D - 1$. If $S \in \mathcal{S}(c^1, c^2)$ dominates $S' \in \mathcal{S}(c^1, c^2 + 1)$, then $\bar{S} \in \mathcal{S}(c^1 + j, c^2 + j)$ also dominates $\bar{S}' \in \mathcal{S}(c^1 + j, c^2 + j + 1)$, where $j \in \{0, \ldots, n_U + n_D - (c^2 + 1)\}$.*

Proof: The proofs for (1.) and (2.) follow the same line of argument as the proofs of (2.) and (1.) in Theorem 9.11, respectively. To prove (1.), the first part of Lemma 9.10 has to be used, while the second result can be deduced from (2.) of Lemma 9.10. □

For Sector $\mathcal{G}_1$ we briefly state an analogous result to Lemma 9.10.

Lemma 9.13 *Let $c \in \{0, \ldots, n_U\}$ and let the sequence $\bar{d}^c$ correspond to the basis $\mathcal{S}(0, c)$.*

1. *Let $c \in \{0, \ldots, n_U - 1\}$ and $i \in \{1, \ldots, n_U - c\}$. Then $\bar{d}^c_k \geq \bar{d}^{c+i}_k$ for all $k \in \{1, \ldots, n_U + n_D - (c + i)\}$.*
2. *Let $c \in \{0, \ldots, n_U - 1\}$. Then $\bar{d}^c_{k+1} \leq \bar{d}^{c+1}_k$ for all $k \in \{1, \ldots, n_U + n_D - (c+1)\}$.*

Proof: The proofs follow the same idea as the proofs for Part (1.) and Part (3.) in Lemma 9.10. □

We use Lemma 9.13 to derive that there cannot exist horizontal dominance in $\mathcal{G}_1$ but that vertical dominance is possible.

Theorem 9.14 *It holds:*

1. *Let $(c^1, c^2) \in \mathcal{G}$ such that $0 \leq c^2 - c^1 \leq n_U - 1$ and $c^2 \leq n_U + n_D - 1$. If $S \in \mathcal{S}(c^1, c^2)$ dominates $S' \in \mathcal{S}(c^1, c^2+1)$ then $\bar{S} \in \mathcal{S}(c^1+j, c^2+j)$ also dominates $\bar{S}' \in \mathcal{S}(c^1 + j, c^2 + j + 1)$, where $j \in \{0, \ldots, n_U + n_D - (c^2 + 1)\}$.*
2. *Let $c \in \{0, \ldots, n_U - 1\}$ and let $S \in \mathcal{S}(i, c + i + 1)$ and $S' \in \mathcal{S}(i + 1, c + i + 1)$, where $i \in \{0, \ldots, n_U + n_D - (c + 1)\}$. Then S does not dominate S'.*

Proof: The proofs follow the same line of argument as the proofs for Theorem 9.12 using Lemma 9.13 instead of Lemma 9.10. □

To summarize, we have shown that there can be vertical dominance in Sectors $\mathcal{G}_1$ and $\mathcal{G}_2$ and there can be horizontal dominance in Sector $\mathcal{G}_3$. We will use these properties to show the following technical theorem which is very important for the next sections.

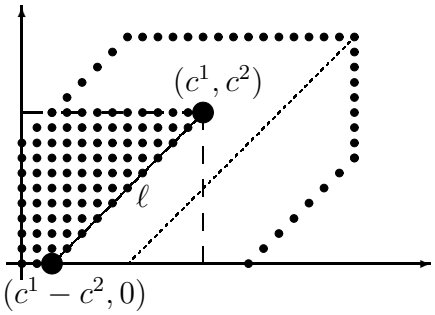


Figure 9.3: *Construction of the line segment ℓ in the proof of Theorem 9.15 for $(c^1, c^2) \in \mathcal{G}_2$.*

Theorem 9.15 *Let $S \in \mathcal{S}(c^1, c^2)$, where $c^1 \cdot c^2 \neq 0$.*

1. *Let $(c^1, c^2) \in \mathcal{G}_1 \cup \mathcal{G}_2$. Then, S cannot be dominated by any knapsack S' satisfying $c^2 \geq w^2(S') \geq w^1(S') + c^2 - c^1$. Furthermore, if S is dominated, then there exists a knapsack $\bar{S} \in \mathcal{S}(\bar{c}^1, \bar{c}^2)$, where $\bar{c}^1 = c^1$, $\bar{c}^2 < c^2$, $(\bar{c}^1, \bar{c}^2) \in \mathcal{G}_1 \cup \mathcal{G}_2$ and $p(\bar{S}) \geq p(S)$ that dominates S.*

2. *Let $(c^1, c^2) \in \mathcal{G}_3$. Then S cannot be dominated by any knapsack S' satisfying $c^1 \geq w^1(S') \geq w^2(S') + c^1 - c^2$. Furthermore, if S is dominated, then there exists a knapsack $\bar{S} \in \mathcal{S}(\bar{c}^1, \bar{c}^2)$, where $\bar{c}^1 < c^1$, $\bar{c}^2 = c^2$, $(\bar{c}^1, \bar{c}^2) \in \mathcal{G}_3$ and $p(\bar{S}) \geq p(S)$ that dominates S.*

Proof: We start with the proof of (1.). Let $S \in \mathcal{S}(c^1, c^2)$ be a dominated knapsack and let $(c^1, c^2) \in \mathcal{G}_1 \cup \mathcal{G}_2$ with $c^1 \cdot c^2 \neq 0$. Then, there exists a knapsack $\bar{S}$ such that $p(\bar{S}) \geq p(S)$, $w^1(\bar{S}) \leq c^1$ and $w^2(\bar{S}) \leq c^2$ with at least one strict inequality. Without loss of generality we may assume that $\bar{S} \in \mathcal{S}(\bar{c}^1, \bar{c}^2)$ where $\bar{c}^i = w^i(\bar{S})$ for $i = 1, 2$.
Since $S \in \mathcal{S}(c^1, c^2)$, we have that $\bar{S} \notin \mathcal{S}(c^1, c^2)$. Hence, $\bar{c}^1 < c^1$ or $\bar{c}^2 < c^2$. Assume that $\bar{c}^2 \geq \bar{c}^1 + c^2 - c^1$, i.e. the point $(\bar{c}^1, \bar{c}^2)$ does not lie underneath the line segment ℓ connecting the point (c^1, c^2) with $(0, c^2 - c^1)$, if $(c^1, c^2) \in \mathcal{G}_1$, and $(c^1 - c^2, 0)$ if $(c^1, c^2) \in \mathcal{G}_2$, respectively (cf. also Figure 9.3). Using Equation (9.2) and Equation (9.3), we conclude that the point $(\bar{c}^1, \bar{c}^2)$ must lie above the line segment.
Without loss of generality we may assume now that $\bar{c}^2 = c^2$ and $\bar{c}^1 < c^1$ since otherwise we can construct a new knapsack $\tilde{S} \in S(\tilde{c}^1, c^2)$ by applying Corollary 9.8, such that $p(\tilde{S}) > p(\bar{S}) \geq p(S)$ and $\tilde{c}^1 < c^1$.
First assume that $p(\bar{S}) > p(S)$. This implies that there exist two knapsacks $\bar{S}_1 \in \mathcal{S}(\bar{c}^1 + j, c^2)$ and $\bar{S}_2 \in \mathcal{S}(\bar{c}^1 + j + 1, c^2)$ for a fixed $j \in \{0, \ldots, c^1 - \bar{c}^1 - 1\}$ such that $p(\bar{S}_1) > p(\bar{S}^2)$ and hence, $\bar{S}_1$ would dominate $\bar{S}_2$ to the right. However, this is not possible in $\mathcal{G}_1 \cup \mathcal{G}_2$ due to Theorem 9.12 and Theorem 9.14, respectively.
Hence, we have that $p(\bar{S}) = p(S)$. Assume that $\bar{c}^1 = c^1 - 1$. This implies that $\bar{S}$ dominates S to the right which is not possible. Hence, $\bar{c}^1 \leq c^1 - 2$ and there exists another knapsack $\bar{S}_1 \in \mathcal{S}(\bar{c}^1 + 1, c^2)$ such that $p(\bar{S}_1) > p(\bar{S})$ since otherwise $\bar{S}_1$ would be dominated from the left by $\bar{S}$. But this implies once more that now $p(\bar{S}_1) > p(S)$, which is not possible in $\mathcal{G}_1 \cup \mathcal{G}_2$.
We conclude that $(\bar{c}^1, \bar{c}^2)$ must lie underneath the line segment ℓ. If $\bar{c}^1 < c^1$ we use once more Corollary 9.8 to construct a knapsack $\bar{S} \in \mathcal{S}(\bar{c}^1, \bar{c}^2)$, where $\bar{c}^1 = c^1$, $\bar{c}^2 < c^2$ is satisfied. Obviously, $\bar{S}$ dominates S. Since there is no dominance to the top in Sector $\mathcal{G}_3$ according to Theorem 9.11, we finally can assume that $(\bar{c}^1, \bar{c}^2) \in \mathcal{G}_1 \cup \mathcal{G}_2$. This proves (1.).

By a similar line of argument, (2.) can be proven. □

9.3.2 Lower Bound on the Cardinality of the Non-dominated Set

In this second part we establish a strict lower bound on the cardinality of the non-dominated set $\mathcal{Y}_N$ using the dominance relations in the different sectors.

Remark 9.16 *Without loss of generality we assume in the following that super-items are always included first in an optimal knapsack if there exist other items in D having the same profit value.*

Applying the rule stated in Remark 9.16 does not change the profit value of an optimal solution, but simplifies the reasoning in the following since tedious case differentiations can be omitted. We start with the knapsacks that belong to a basis.

Lemma 9.17 *Let $c' \in \{0, \ldots, n_R\}$ and $c^\star \in \{1, \ldots, n_U\}$. Then, $S' \in \mathcal{S}(c', 0)$ and $S^\star \in \mathcal{S}(0, c^\star)$ are efficient knapsacks of Problem (2-MP) with non-dominated objective vectors $(p(S'), c', 0)$ and $(p(S^\star), 0, c^\star)$.*

Proof: The efficiency of a knapsack that is contained in a basis follows immediately from Lemma 9.5. □

Next we state that all optimal knapsacks to Problem (2-KP$_=$) for constraints corresponding to integer points on the common boundary line between $\mathcal{G}_2$ and $\mathcal{G}_3$ are also efficient knapsacks of Problem (2-MP).

Lemma 9.18 *Let $(c^1, c^2) \in \mathcal{G}$ such that $c^1 - c^2 = n_R - n_U$ and $S \in \mathcal{S}(c^1, c^2)$. Then $S \in \mathcal{X}_E$.*

Proof: The lemma is an immediate consequence of Theorem 9.15 and Lemma 9.17. □

To complete the second part of this section, we finally show that optimal knapsacks of Problem (2-KP$_=$) for constraint vectors (c^1, c^2) contained in the rectangle

$$\mathcal{Q} = \left\{(x, y) \in \mathbb{R}^2 : x \in \{0, 1, \ldots, n_R\}, y \in \{0, 1, \ldots, n_U\}\right\},$$

also correspond to efficient knapsacks of Problem (2-MP).

Theorem 9.19 *Let $S \in \bigcup_{(c^1,c^2)\in\mathcal{Q}} \mathcal{S}(c^1, c^2)$. Then $S \in \mathcal{X}_E$.*

Proof: We have to distinguish the three cases $(c^1, c^2) \in \mathcal{G}_i$ for $i \in \{1, 2, 3\}$. We give a proof for Sector $\mathcal{G}_2$. The proofs for the two other sectors follow the same ideas as the proof given below.
If $(c^1, c^2) \in \mathcal{G}_2 \cap \mathcal{G}_3$, there is nothing to show according to Lemma 9.18. So let $(c^1, c^2) \in \mathcal{G}_2 \setminus \mathcal{G}_3$. We assume that there exists a knapsack $\bar{S}$ that dominates S. According to Theorem 9.15 there exists a well-defined index $\bar{c}^2 \in \{0, \ldots, c^2 - 1\}$ such that $\bar{S} \in \mathcal{S}(c^1, \bar{c}^2)$, and $(c^1, \bar{c}^2) \in \mathcal{G}_1 \cup \mathcal{G}_2$. Since $(c^1, c^2) \in \mathcal{G}_2$, also $(c^1, \bar{c}^2) \in \mathcal{G}_2$ must hold.
Next we show that we may assume that $\bar{S} \in \mathcal{S}(c^1, c^2 - 1)$. Otherwise there must exist

a fixed index $i \in \{0, \dots, c^2 - \bar{c}^2 - 1\}$ and $\bar{S}_i \in \mathcal{S}(c^1, \bar{c}^2 + i)$ and $\bar{S}_{i+1} \in \mathcal{S}(c^1, \bar{c}^2 + i + 1)$ such that $p(\bar{S}_i) > p(\bar{S}_{i+1})$. Setting $S = \bar{S}_{i+1}$ and $\bar{S} = \bar{S}_i$ implies that S is dominated by its neighbor $\bar{S}$ from below.
Now let $c = c^1 - c^2$ and let μ denote the number of super-items contained in $\bar{S}$. Note that since $(c^1, c^2) \in \mathcal{Q}$ it holds that $0 \leq \mu \leq c^2 - 1 \leq n_U - 1 < n_U$. According to Lemma 9.10 we know that $\bar{d}_k^c \geq \bar{d}_k^{c+1} \geq \bar{d}_{k+1}^c$ for all $k \in \{1, \dots, n_D + n_U - 1\}$. This implies that S must contain at least μ but at most $\mu + 1$ super-items, assuming Remark 9.16 is valid. We get:

$$
\begin{aligned}
p(\bar{S}) - p(S) &= r_{c+1} + \sum_{i=1}^{c^2-1} \bar{d}_i^{c+1} - \sum_{i=1}^{c^2} \bar{d}_i^c && (9.5)\\
&= r_{c+1} + \sum_{i=1}^{\mu} \tilde{d}_i^{c+1} + \sum_{i=1}^{c^2-1-\mu} d_i - \left(\sum_{i=1}^{\mu} \tilde{d}_i^c + \sum_{i=1}^{c^2-1-\mu} d_i + \bar{d}_{c^2}^c \right) \\
&= r_{c+1} + \sum_{i=1}^{\mu} r_{c+1+i} + \sum_{i=1}^{\mu} u_i - \left(\sum_{i=1}^{\mu} r_{c+i} + \sum_{i=1}^{\mu} u_i \right) - \bar{d}_{c^2}^c \\
&= r_{c+1+\mu} - \bar{d}_{c^2}^c \\
&= r_{c+1+\mu} - \max\left(d_{c^2-\mu}, r_{c+\mu+1} + u_{\mu+1}\right) && (9.6)\\
&< r_{c+1+\mu} - r_{c+1+\mu} = 0.
\end{aligned}
$$

Note that the element $d_{c^2-\mu}$ is not guaranteed to exist, but $\tilde{d}_{\mu+1}^c$ always exists since $\mu + 1 \leq n_U$. Hence, $p(\bar{S}) < p(S)$ and S cannot be dominated by $\bar{S}$. □

We summarize the last results in the following corollary.

Corollary 9.20 $|\mathcal{Y}_N| \geq (n_U + 1) \cdot (n_R + 1) + n_D$.

Proof: The proof follows immediately from Lemma 9.17, Lemma 9.18 and Theorem 9.19. □

At the end of the next section we will show that the stated lower bound on the cardinality of the set of all non-dominated solutions is tight, i.e. there exist instances such that equality holds in Corollary 9.20.

9.3.3 Avoiding the Filtering Step

The aim of this subsection is to show that we can omit the filtering step to remove dominated knapsacks at the end. We state necessary and sufficient conditions on the value of the profits of the items contained in R, U and D, respectively, which allow to decide whether an optimal knapsack for (2-KP$_=$) given by an element of the sequence $\{S^i\}_i$ stated in Corollary 9.8 is also an efficient knapsack of (2-MP).
We first concentrate on Sector $\mathcal{G}_2$ in the following and give a detailed outline of the proofs implying our necessary and sufficient condition for this sector. Note that for $(c^1, c^2) \in \mathcal{G}_2$ the maximal number of super-items contained in a feasible knapsack for Problem (2-KP$_=$) is restricted to n_U. We start with the following lemma.

Lemma 9.21 *Let $(c^1, c^2) \in \mathcal{G}_2$ where $c^2 > n_U$ and let $S' \in \mathcal{S}(c^1, c^2 - 1)$ dominate $S \in \mathcal{S}(c^1, c^2)$. Then, S and S' contain all n_U super-items and $r_{c^1-c^2+n_U+1} \geq d_{c^2-n_U}$.*

Proof: Let μ and μ' denote the number of super-items that are contained in S and S', respectively. Applying Lemma 9.10 and Remark 9.16, we conclude that $\mu \in \{\mu', \mu'+1\}$ whenever $\mu < n_U$. By applying the same reasoning as in the proof of Theorem 9.19, it implies that $p(S') < p(S)$, and S' cannot dominate S. Hence, S must contain all n_U super-items.
Now, assume that S' does not contain all n_U super-items but only $n_U - 1$ although it dominates S. Analyzing the elements of R, U, and D that are contained in S and S' leads to

$$0 \leq p(S') - p(S) = -u_{n_U} < 0.$$

Hence, also S' must contain all super-items. Using Equation (9.6) from the proof of Theorem 9.19 and knowing that $\mu = n_U$ is maximal, we conclude that

$$0 \leq p(S') - p(S) = r_{c^1-c^2+1+n_U} - d_{c^2-n_U}.$$

Hence, $r_{c^1-c^2+1+n_U} \geq d_{c^2-n_U}$. □

Note that Lemma 9.21 is valid for $0 \leq c^1 - c^2 < n_R - n_U$, whereas for $c^1 - c^2 = n_R - n_U$, $S \in \mathcal{S}(c^1, c^2)$ is always an efficient solution according to Lemma 9.18. We also take care of this fact in the following lemma.

Lemma 9.22 *Let $(c^1, c^2) \in \mathcal{G}_2$ with $0 \leq c^1 - c^2 < n_R - n_U$ and $n_U < c^2 < n_U + n_D$, and let $S' \in \mathcal{S}(c^1, c^2 - 1)$ dominate $S \in \mathcal{S}(c^1, c^2)$. Then S dominates $\bar{S} \in \mathcal{S}(c^1, c^2 + 1)$.*

Proof: Let S, S' and $\bar{S}$ be given as defined above. According to Lemma 9.21, S must contain all n_U super-items and it holds that $r_{c+1+n_U} \geq d_{c^2-n_U}$. We assume first that $(c^1, c^2) \in \mathcal{G}_2 \setminus \mathcal{G}_1$, i.e. $c^1 > c^2$. We use Lemma 9.10 to deduce that $\bar{S} \in \mathcal{S}(c^1 - c^2 - 1, 0)$ must also contain all n_U super-items, since it contains an additional item compared to S. We conclude that

$$\begin{aligned} p(S) - p(\bar{S}) &= r_{c^1-c^2+n_U} - d_{c^2-n_U+1} \geq r_{c^1-c^2+n_U+1} - d_{c^2-n_U+1} \\ &\geq r_{c^1-c^2+n_U+1} - d_{c^2-n_U} \geq 0. \end{aligned}$$

If $(c^1, c^2) \in \mathcal{G}_2 \cap \mathcal{G}_1$, i.e. $c^1 = c^2$, $\bar{S}$ contains all $n_U - 1$ super-items with respect to its basis $\mathcal{S}(0, 1)$ according to Lemma 9.21. Adapting the chain of inequalities stated above to this special case shows that S dominates $\bar{S}$. □

We are now able to derive the main result for $\mathcal{G}_2$.

Theorem 9.23 *Let $(c^1, c^2) \in \mathcal{G}_2$ with $0 \leq c^1 - c^2 \leq n_R - n_U - 1$ and $n_U + 1 \leq c^2 \leq n_U + n_D$, and let $S \in \mathcal{S}(c^1, c^2)$. Then the following statements are equivalent:*

(A) *$S \notin \mathcal{X}_E$.*

(B) *S is dominated by $S' \in \mathcal{S}(c^1, c^2 - 1)$.*

(C) *$r_{n_U+c^1-c^2+1} \geq d_{c^2-n_U}$.*

Proof: '(B)⇒(A)' is obviously true and the proof for '(B)⇒(C)' was already shown in Lemma 9.21. We prove the implications '(A)⇒(B)' and '(C)⇒(B)' in the following.
To prove '(A)⇒(B)', let $S \in \mathcal{S}(c^1, c^2)$ be dominated by $\bar{S} \in \mathcal{S}(\bar{c}^1, \bar{c}^2)$ with $\bar{c}^1 \leq c^1$ and $\bar{c}^2 \leq c^2$ with at least one strict inequality. According to Theorem 9.15 we may

assume that $(\bar{c}^1, \bar{c}^2) \in \mathcal{G}_2$, $\bar{c}^1 = c^1$ and $\bar{c}^2 < c^2$. Assume that S is not dominated by $S' \in \mathcal{S}(c^1, c^2 - 1)$, i.e. $\bar{c}^2 < c^2 - 1$ and $p(S') < p(S) \leq p(\bar{S})$. Then, there must exist a fixed index $i \in \{0, \ldots, c^2 - \bar{c}^2 - 2\}$ and $\bar{S}_i \in \mathcal{S}(c^1, \bar{c}^2 + i)$ and $\bar{S}_{i+1} \in \mathcal{S}(c^1, \bar{c}^2 + i + 1)$ such that $p(\bar{S}_i) > p(\bar{S}_{i+1})$ holds. But this implies that $\bar{S}_{i+1}$ is dominated by its neighbor $\bar{S}_i$ from below. By applying Lemma 9.22, also S must be dominated by its neighbor from below, i.e. S' dominates S, which contradicts our assumption.
We show that '(C)⇒(B)'. Since

$$r_{c^1 - c^2 + n_U} + u_{n_U} \; > \; r_{c^1 - c^2 + n_U} \; \geq \; r_{c^1 - c^2 + n_U + 1} \; \geq \; d_{c^2 - n_U},$$

S consists of all first $c^1 - c^2$ items of R, all n_U super-items and the first $c^2 - n_U$ items of D. Furthermore, it holds that

$$r_{c^1 - c^2 + n_U + 1} + u_{n_U} \; > \; r_{c^1 - c^2 + n_U + 1} \; \geq \; d_{c^2 - n_U},$$

and S' contains at most $c^2 - n_U - 1$ items of D, at most n_U super-items and the first $c^1 - c^2 + 1$ items of R. But since $(c^2 - n_U - 1) + n_U = c^2 - 1$ and this is the number of elements that are added to the knapsack in basis $\mathcal{S}(c^1 - c^2, 0)$, S' must contain exactly the above mentioned items. We conclude that

$$p(S') - P(S) \; = \; r_{c^1 - c^2 + n_U + 1} - d_{c^2 - n_U} \; \geq \; 0.$$

This completes the proof. □

In Theorem 9.23 we have proven a necessary and sufficient condition for the efficiency of an optimal knapsack $S \in \mathcal{S}(c^1, c^2)$ where $(c^1, c^2) \in \mathcal{G}_2$, which supersedes the filtering for dominated solutions. Given the sequence $\{S^i\}_i$ stated in Corollary 9.8 we stop calculating the elements of this sequence when $r_{c^1 - c^2 + n_U + 1} \geq d_{c^2 - n_U}$ is satisfied for the first time. Starting from this element, all remaining knapsacks of the sequence will be dominated by their neighbor from below. Hence, an additional filtering is no longer needed.
By a similar line of argument as used in Lemma 9.21 and Lemma 9.22, it is possible to prove similar results of Theorem 9.23 for $\mathcal{G}_1$ and $\mathcal{G}_3$.

Theorem 9.24 *Let $(c^1, c^2) \in \mathcal{G}_1$ with $1 \leq c^2 - c^1 \leq n_U$ and $n_U + 1 \leq c^2 \leq n_U + n_D$, and let $S \in \mathcal{S}(c^1, c^2)$. Then the following statements are equivalent:*

(A) *$S \notin \mathcal{X}_E$.*

(B) *S is dominated by $S' \in \mathcal{S}(c^1, c^2 - 1)$.*

(C) *$r_{n_U + c^1 - c^2 + 1} \geq d_{c^2 - n_U}$.*

For Sector $\mathcal{G}_3$ we get:

Theorem 9.25 *Let $(c^1, c^2) \in \mathcal{G}_3$ with $n_R - n_U + 1 \leq c^1 - c^2 \leq n_R$ and $n_R + 1 \leq c^1 \leq n_R + n_D$, and let $S \in \mathcal{S}(c^1, c^2)$. Then the following statements are equivalent:*

(A) *$S \notin \mathcal{X}_E$.*

(B) *S is dominated by $S' \in \mathcal{S}(c^1 - 1, c^2)$.*

			74	104	132	156	178	194	203	211
		71	101	129	153	175	191	200	208	203
	67	97	125	149	171	187	196	204	200	178
62	92	120	144	166	182	191	199	196	175	149
54	84	112	136	158	174	183	191	171	146	
29	59	87	111	133	149	158	166	142		
0	30	58	82	104	120	129	137			

Table 9.1: *Grid representation of all* 65 *optimal solutions for Problem (2-KP$_=$) considered in Section 9.3.4.*

(C) $u_{n_R-c^1+c^2+1} \geq d_{c^1-n_R}$.

The pseudo-code of the resulting algorithm for $\mathcal{G}_2$ is described by Algorithm A.2 in Appendix A. For the sake of simplicity of the explanation, we assume that the non-dominated vector $(0, 0, 0)$ is computed at sector $\mathcal{G}_2$ in Algorithm A.2. We omit to state the pseudo-code for $\mathcal{G}_1$ and $\mathcal{G}_3$ since it can be easily derived from Algorithm A.2 and the results stated in this section.
Note that the last results imply that the lower bound on the number of non-dominated solutions that was stated in Corollary 9.20 is tight. Assume that for a given instance of (2-MP) we have that $\min\{r_{n_R}, u_{n_U}\} > d_1$. Then the third criterion stated in the Theorems 9.23 to 9.25 for the different sectors immediately implies that the bound is tight. If, in contrast, $\max\{r_1, u_1\} < d_{n_D}$, the stated criteria imply that solving (2-KP$_\leq$) for any $(c^1, c^2) \in \mathcal{G}$ will lead to a non-dominated solution of (m-MP).
We finally summarize our main result in the following theorem.

Theorem 9.26 *Let an instance of Problem (2-MP) be given. Then, the set of all non-dominated solutions $\mathcal{Y}_N$ for this problem can be determined within $\mathcal{O}(n^2)$ time, where no additional filtering for dominated solutions has to be performed.*

Theorem 9.26 implies that the greedy algorithm presented for Problem (2-MP) is optimal in terms of upper bound time complexity, since the cardinality of the non-dominated set is bounded by $\mathcal{O}(n^2)$.

9.3.4 Example

In this subsection we present a short example for the results obtained so far. Furthermore, we show that the lower bound on the number of non-dominated solutions given in Corollary 9.20 is really tight in this case. So, let an instance of Problem (2-KP$_=$) be given by

$$R = [30, 28, 24, 22, 16, 9, 8],\ U = [29, 25, 8] \text{ and } D = [5, 4, 3].$$

The 65 different solutions for all possible right hand side vectors $c \in \mathcal{G}$ of Problem (2-KP$_=$) are shown in Table 9.1. Concerning the notation used in this table, it

			−−	−−	−−	−−	−−	−−	−−	211
		−−	−−	−−	−−	−−	−−	−−	208	−−
	−−	−−	−−	−−	−−	−−	−−	204	−−	−−
62	92	120	144	166	182	191	199	−−	−−	−−
54	84	112	136	158	174	183	191	−−	−−	
29	59	87	111	133	149	158	166	−−		
0	30	58	82	104	120	129	137			

Table 9.2: *Grid representation of all* 35 *non-dominated solutions for Problem (2-MP) considered in Section 9.3.4.*

is assumed that each replacement character in the scheme corresponds to an integer two-dimensional vector in the plane where the origin can be found in the bottom left corner. The complete scheme forms the hexagonal grid $\mathcal{G}$. The given numbers represent the optimal profit values of Problem (2-KP$_=$) where the two coordinates of the replacement character correspond to the right hand side vector (c^1, c^2) of the specific problem.
For example, the maximum profit value of the optimal knapsack $S \in \mathcal{S}(4, 2)$ is given by $p(S) = 158$, while for $S' \in \mathcal{S}(2, 5)$ it holds true that $p(S') = 71$, and so on.
Applying the domination rules stated proven in this section, it can easily be verified that the corresponding Problem (2-MP) hast exactly 35 non-dominated solutions, that are shown in Table 9.2. Note that in this table, replacement character represented by "−−" correspond to dominated solutions.
For the given example, the number of non-dominated solutions coincides with the lower bound on the cardinality of the non-dominated set stated in Corollary 9.20, since

$$|\mathcal{Y}_N| \;=\; 35 \;=\; 4 \cdot 8 + 3 \;=\; (n_U + 1) \cdot (n_R + 1) + n_D.$$

Since $\min\{r_{n_R}, u_{n_U}\} = \min\{8, 8\} = 8 > 5 = d_1$, this result is also implied by the criterion that is stated at the end of the last subsection.

9.3.5 Experimental Results

To verify the efficiency of our approach in practice, we implemented it in C and tested it on a set of randomly generated instances.
We generated 100 instances for each of the sizes $n \in \{10\times10^i, 25\times10^i, 50\times10^i, 75\times10^i\}$ with $i = 2, 3, 4$. The profit values are positive integers uniformly distributed in the range $[1, 11]$. Note that we chose a small range of profit values to avoid number overflow for larger instances. In order to generate values for n_R, n_U and n_D, we first generated three real numbers randomly and uniformly distributed in the range $[0, 1]$; then, we normalized these values with respect to their sum; finally, we multiplied each normalized value by n to obtain n_R, n_U and n_D, respectively.
We ran our implementation on an Intel Core 2 Duo 2.33Ghz, 4MB cache L2, 4GB RAM, with Windows Vista 32 bits, compiler MSVC 2008.

n	CPU-time (in secs.)		$\lvert\mathcal{Y}_N\rvert$	
	avg.	std.	avg.	std.
1 000	0.00	0.00	182 329	42 472
2 500	0.01	0.00	1 116 855	256 644
5 000	0.04	0.01	4 478 076	1 192 776
7 500	0.09	0.02	9 786 613	2 388 547
10 000	0.15	0.02	18 211 143	3 919 746
25 000	0.97	0.19	117 716 464	26 611 280
50 000	3.80	0.79	448 329 030	110 881 051
75 000	8.10	2.12	980 416 031	275 287 540
100 000	14.73	3.62	1 766 044 758	469 409 790
250 000	93.81	22.74	11 267 090 109	3 036 879 464
500 000	381.11	89.58	44 872 436 605	11 436 272 293
750 000	877.49	178.36	102 117 237 786	23 926 930 515
1000 000	1542.83	333.52	179 661 247 582	41 598 896 050

Table 9.3: *Average (avg.) and standard deviation (std.) of CPU-time in seconds taken by the greedy algorithm and the size of the non-dominated set for randomly generated instances of Problem (2-MP).*

Table 9.3 shows the average and standard deviation of CPU-time in seconds taken by our greedy algorithm, as well as the cardinality of the non-dominated set for the randomly generated instances. They clearly indicate that our approach can perform very fast.
For example, the algorithm solves instances with one million items and over 179 billion of non-dominated solutions within less than 30 minutes on average.

9.4 The Knapsack Problem with Two Inequality Constraints

Based on the results for solving Problem (2-MP), we can derive an efficient algorithm to solve Problem (2-KP$_{\leq}$). From Chapter 3 we recall that if Problem (2-KP$_{\leq}$) is feasible, i.e. c^1 and c^2 are chosen to be non-negative, an optimal knapsack $\bar{S}$ to this problem is contained in the efficient set of Problem (2-MP) (cf. Theorem 3.1). Obviously, for such a knapsack it holds that $p(\bar{S}) \geq p(S)$ for all $S \in \mathcal{X}_E$ where $w^j(S) \leq c^j$ and $w^j(\bar{S}) \leq c^j$, $j = 1, 2$, respectively. Hence, the results of Section 9.3 can be used to solve Problem (2-KP$_{\leq}$).
We will show that we only need to consider one of the sequences $\{S^i\}_i$ starting from an efficient knapsack contained in an appropriately chosen basis to find an optimal solution for Problem (2-KP$_{\leq}$). The superscripts of the optimal knapsacks in the following theorems indicate which element of the sequence $\{S^i\}_i$ has to be calculated.

To apply the results of Section 9.3 we need to consider an additional partition of each sector $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ as illustrated in Figure 9.4 and that is taken into account in the following results. We start with Sector $\mathcal{G}_2$ and recall that there is no dominance to the right in this sector and that an optimal knapsack of Problem (2-KP$_=$) with respect to $(c^1, c^2) \in \mathcal{G}_2$ correspond to a dominated knapsack of Problem (2-MP) if and only if it is dominated by its neighbor from below.

Theorem 9.27 *Let $(c^1, c^2) \in \mathcal{G}_2$.*

1. *If $(c^1, c^2) \in \mathcal{Q}$ or $c^1 - c^2 = n_R - n_U$, then $S^{c^2} \in \mathcal{S}(c^1, c^2)$ is an optimal knapsack of Problem (2-KP$_\leq$).*

2. *Let $(c^1, c^2) \notin \mathcal{Q}$, $c^1 - c^2 \neq n_R - n_U$ and $c^1 \leq n_R$. If there exists an index j such that $j = \min\{i \in \{0, \ldots, (c^2 - 1) - n_U\} : d_{c^2-i-n_U} > r_{n_U+c^1-c^2+i+1}\}$, then $S^{c^2-j} \in \mathcal{S}(c^1, c^2 - j)$ is an optimal knapsack of Problem (2-KP$_\leq$). Otherwise $S^{n_U} \in \mathcal{S}(c^1, n_U)$ is optimal.*

3. *Let $(c^1, c^2) \notin \mathcal{Q}$, $c^1 - c^2 \neq n_R - n_U$ and let $c^1 > n_R$. If there exists an index j such that $j = \min\{i \in \{0, \ldots, n_R - n_U - c^1 + c^2 - 1\} : d_{c^2-i-n_U} > r_{n_U+c^1-c^2+i+1}\}$, then $S^{c^2-j} \in \mathcal{S}(c^1, c^2 - j)$ is an optimal knapsack of Problem (2-KP$_\leq$). Otherwise $S^{c^1-n_R+n_U} \in \mathcal{S}(c^1, c^1 - n_R + n_U)$ is optimal.*

Proof: To prove (1.), let $(c^1, c^2) \in \mathcal{Q}$ and we assume that $S^{c^2} \in \mathcal{S}(c^1, c^2)$ is not optimal for Problem (2-KP$_\leq$). Then there must exist another feasible knapsack $S \notin \mathcal{S}(c^1, c^2)$ satisfying $p(S) > p(S^{c^2})$ and $w^j(S) \leq w^j(S^{c^2})$, $j = 1, 2$ with at least one strict inequality. But this implies that S dominates S^{c^2}, which is not possible due to Theorem 9.19. Hence, S^{c^2} must be optimal. A similar reasoning in combination with Lemma 9.18 can be applied for the case that $c^1 - c^2 = n_R - n_U$.
For (2.), let $(c^1, c^2) \notin \mathcal{Q}$ and $c^1 - c^2 \neq n_R - n_U$, but $c^1 \leq n_R$. Furthermore, let S be an optimal knapsack of Problem (2-KP$_\leq$). Without loss of generality we may assume that $S \in \mathcal{X}_E$ and that there exist $\bar{c}^1 \in \{0, \ldots, c^1\}$ and $\bar{c}^2 \in \{0, \ldots, c^2\}$ such that $S \in \mathcal{S}(\bar{c}^1, \bar{c}^2)$ where either $\bar{c}^1 = c^1$ or $\bar{c}^2 = c^2$ due to the Equations (9.2), (9.3) and (9.4). Since there cannot be dominance to the right in the Sectors $\mathcal{G}_1$ and $\mathcal{G}_2$ according to Theorem 9.12 and Theorem 9.14 we conclude that $\bar{c}^1 = c^1$. Now, define $j = \min\{i \in \{0, \ldots, (c^2 - 1) - n_U\} : d_{c^2-i-n_U} > r_{n_U+c^1-c^2+i+1}\}$ if the minimum exists. Otherwise, let $j = \infty$. Note that the index $n_U + c^1 - c^2 + i + 1$ is well-defined for all $i \in \{0, \ldots, (c^2 - 1) - n_U\}$, since obviously $n_U + c^1 - c^2 + i + 1 \geq 1$ and further

$$n_U + c^1 - c^2 + i + 1 \leq n_U + n_R - c^2 + (c^2 - n_U - 1) + 1 = n_R.$$

If $j = 0$, $S^{c^2} \in \mathcal{S}(c^1, c^2)$ must be optimal, since S^{c^2} is efficient due to Theorem 9.23 and hence, $p(S)$ must be maximal for all knapsacks S satisfying $w^j(S) \leq c^j$ for $j = 1, 2$. Now, assume that $1 \leq j < \infty$ and let $S^{c^2-j} \in \mathcal{S}(c^1, c^2 - j)$. Theorem 9.23 states that a knapsack $S \in \mathcal{S}(c^1, \tilde{c}^2)$ is dominated by its neighbor from below whenever $\tilde{c}^2 \in \{c^2 - j + 1, \ldots, c^2\}$. But this implies that $p(S) \leq p(S^{c^2-j})$ for all $S \in \mathcal{S}(c^1, \tilde{c}^2)$. Since S^{c^2-j} is efficient according to the same theorem, it follows that $p(S) < p(S^{c^2-j})$ for all $S \in \mathcal{S}(c^1, \tilde{c}^2)$ whenever $\tilde{c}^2 \in \{0, \ldots, c^2 - j - 1\}$. This shows, that $p(S^{c^2-j})$ is optimal for Problem (2-KP$_\leq$). If $j = \infty$ this implies that all knapsacks $S \in \mathcal{S}(c^1, \bar{c}^2)$ are dominated by their neighbors from below for $\bar{c}^2 \in \{n_U + 1, \ldots, c^2\}$. Let $S^{n_U} \in$

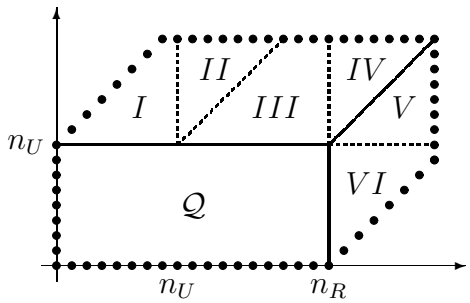


Figure 9.4: *Illustration of the partition of $\mathcal{G}$ for Theorems 9.27, 9.28 and 9.29.*

$\mathcal{S}(c^1, n_U)$. Since S^{n_U} is contained in $\mathcal{Q}$, S^{n_U} is efficient by Theorem 9.19, and hence it must be optimal for Problem (2-KP$_\leq$).
For (3.) we note that the proof for the case that $c^1 > n_R$ follows the same line of argument as the proof for the case $c^1 \leq n_R$. In this case, Lemma 9.18 has to be used instead of Theorem 9.19. Moreover, we remark that the index $c^2 - i - n_U$ is well-defined for all $i \in \{0, \ldots, n_R - n_U - c^1 + c^2 - 1\}$ since $c^2 - i - n_U \leq c^2 - n_U \leq n_D$ and further

$$c^2 - i - n_U \geq c^2 - (n_R - n_U - c^1 + c^2 - 1) - n_U = c^1 - n_R + 1 > 1.$$

This completes the proof. □

For $(c^1, c^2) \in \mathcal{G}_1$ it is easy to verify that we can find an optimal knapsack S to Problem (2-KP$_\leq$) such that $S \in \mathcal{S}(c^1, \bar{c}^2)$ and $(c^1, \bar{c}^2)$ is also contained in $\mathcal{G}_1$, whenever $c^1 \leq n_U$ holds. For the case that $c^1 > n_U$, it may happen that $(c^1, \bar{c}^2)$ is no longer contained in $\mathcal{G}_1$ but in $\mathcal{G}_2$.

Theorem 9.28 *Let $(c^1, c^2) \in \mathcal{G}_1$.*

1. *If $(c^1, c^2) \in \mathcal{Q}$, then $S^{c^1} \in \mathcal{S}(c^1, c^2)$ is an optimal knapsack of Problem (2-KP$_\leq$).*
2. *Let $(c^1, c^2) \notin \mathcal{Q}$ and let $c^1 \leq n_U$. If there exists an index j such that $j = \min\{i \in \{0, \ldots, (c^2 - 1) - n_U\} : d_{c^2 - i - n_U} > r_{n_U + c^1 - c^2 + i + 1}\}$, then $S^{c^1} \in \mathcal{S}(c^1, c^2 - j)$ optimally solves Problem (2-KP$_\leq$). Otherwise $S^{c^1} \in \mathcal{S}(c^1, n_U)$ is optimal.*
3. *Let $(c^1, c^2) \notin \mathcal{Q}$ and let $c^1 > n_U$ If there exists an index j such that $j = \min\{i \in \{0, \ldots, c^2 - c^1 - 1\} : d_{c^2 - i - n_U} > r_{n_U + c^1 - c^2 + i + 1}\}$, then $S^{c^1} \in \mathcal{S}(c^1, c^2 - j)$ is an optimal knapsack of Problem (2-KP$_\leq$). Otherwise there exists $\bar{c}^2 \in \{n_U, \ldots, c^1\}$, such that $(c^1, \bar{c}^2) \in \mathcal{G}_2$ and $S^{\bar{c}^2} \in \mathcal{S}(c^1, \bar{c}^2)$ is optimal for Problem (2-KP$_\leq$).*

Proof: The proofs of the three cases are similar to the proofs of the corresponding statements in Theorem 9.27. In the last two cases, the minimum may not exist since either $c^1 = c^2$ or all knapsacks $S \in \mathcal{S}(c^1, \tilde{c}^2)$ with $\tilde{c}^2 \in \{c^1 + 1, \ldots, c^2\}$ are dominated by their neighbors from below. But both cases imply that there must exist an efficient knapsack $S \in \mathcal{S}(c^1, \bar{c}^2)$, where $(c^1, \bar{c}^2) \in \mathcal{G}_2$ and $\bar{c}^2 \in \{n_U, \ldots, c^1\}$ holds. □

For the Sector $\mathcal{G}_3$ we find similar results compared to the other two sectors.

Theorem 9.29 *Let $(c^1, c^2) \in \mathcal{G}_3$.*

1. *If $(c^1, c^2) \in \mathcal{Q}$ or $c^1 - c^2 = n_R - n_U$, then $S^{c^2} \in \mathcal{S}(c^1, c^2)$ is an optimal knapsack of Problem (2-KP$_\leq$).*

2. *Let* $(c^1, c^2) \notin \mathcal{Q}$, $c^1 - c^2 \geq n_R - n_U + 1$ *and let* $c^2 \leq n_U$. *If there exists an index* j *such that* $j = \min\{i \in \{0, \ldots, (c^1 - 1) - n_R\} : d_{c^1-i-n_R} > u_{n_R-c^1+c^2+i+1}\}$, *then* $S^{c^2} \in \mathcal{S}(c^1 - j, c^2)$ *is an optimal knapsack of Problem (2-KP*$_{\leq}$*). Otherwise* $S^{c^2} \in \mathcal{S}(n_R, c^2)$ *is optimal.*

3. *Let* $(c^1, c^2) \notin \mathcal{Q}$, $c^1 - c^2 \geq n_R - n_U + 1$ *and let* $c^2 > n_U$. *If there exists an index* j *such that* $j = \min\{i \in \{0, \ldots, n_U - n_R + c^1 - c^2 - 1\} : d_{c^1-i-n_R} > r_{n_R-c^1+c^2+i+1}\}$, *then* $S^{c^2} \in \mathcal{S}(c^1 - j, c^2)$ *is an optimal knapsack of Problem (2-KP*$_{\leq}$*). Otherwise* $S^{c^2} \in \mathcal{S}(n_R - n_U + c^2, c^2)$ *is optimal.*

The Theorems 9.27 to 9.29 show that we can determine an optimal knapsack to Problem (2-KP$_{\leq}$) by calculating a fixed number of elements of a sequence $\{S^i\}_i$ used in Corollary 9.8 for a knapsack S^0 contained in an appropriately chosen basis. Therefore, the algorithm for solving this problem has the same time complexity as the greedy algorithm for (2-KP$_{=}$) which is given by $\mathcal{O}(n \log(n))$ according to the results of Subsection 9.2. The pseudo-code of the resulting algorithm for $\mathcal{G}_2$ is described by Algorithm A.3 in Appendix A. The pseudo-code for $\mathcal{G}_1$ and $\mathcal{G}_3$ is omitted since it can be easily derived from Algorithm A.3 and the results stated in this section.

9.5 Connectedness of the Efficient Set

Corollary 9.8 introduces an important result in terms of connectedness of efficient knapsacks. For a deeper discussion on the connectedness of the efficient set of multiple objective combinatorial optimization problems, we refer to Chapter 7. We define a graph where the nodes represent the efficient knapsacks and edges are introduced between all pairs of nodes that are adjacent with respect to the following definition of *k-change* neighborhood:

Definition 9.30 (k-change neighborhood) *Two knapsacks are neighbors with respect to the k-change neighborhood if and only if one knapsack can be obtained from the other by either adding or removing at most k items.*

We say that the efficient set of Problem (2-MP) is connected if and only if the corresponding graph is connected. Note that Definition 9.30 generalizes the definition of adjacent efficient solutions used in Section 7.2.8 for unconstrained multiple objective combinatorial optimization problems (cf. Definition 7.29). For $k = 1$ both definitions and the MILP-based definition of adjacency on the unit cube $[0, 1]^n$ (where $n = n_R + n_U + n_D$) coincide for this problem, while for $k > 2$ an enlarged neighborhood is considered. In this case, Definition 9.30 implies that a combinatorial definition of adjacency considered. For the special problem discussed in Section 9.3, we state the following result for the 2-change neighborhood:

Corollary 9.31 *There exists a set of efficient knapsacks of Problem (2-MP) that is connected with respect to the 2-change neighborhood and its image in the objective space coincides with the non-dominated set.*

Corollary 9.31 only states that a subset of the set of efficient knapsacks is connected but not the complete set itself. However, this property applies to the complete set of

efficient knapsacks if we consider also the following extended definition of neighborhood:

Definition 9.32 (k-exchange neighborhood) *Two knapsacks are neighbors with respect to the k-exchange neighborhood if and only if one knapsack can be obtained from the other by exchanging and adding or removing at most k items.*

Note that in a k-change neighborhood it is only allowed to either add or remove k items to or from a knapsack, respectively, while in a k-exchange neighborhood, we can exchange a number of items and either add or remove another fixed number. For example, exchanging a super-item by another one is considered as a 2-exchange, as well as an exchange of a super-item by an item contained in D. In this case we have to exchange one item and either add or remove another one. By the definition of a k-exchange, a k-change is always a k-exchange, but not necessarily the other way round.

Theorem 9.33 *The set of efficient knapsacks of Problem (2-MP) is connected with respect to the 2-exchange neighborhood.*

Proof: According to Corollary 9.31 it suffices to show that for an efficient $S \in \mathcal{S}(c^1, c^2)$ and an alternative efficient knapsack $\bar{S} \in \mathcal{S}(c^1, c^2)$, there exist a finite sequence of efficient knapsacks starting from S and ending by $\bar{S}$, such that all knapsacks of this sequence are contained in $\mathcal{S}(c^1, c^2)$ and subsequent knapsacks are neighbors with respect to the 2-exchange.
For the case that S is contained in a basis $\mathcal{S}(c, 0)$ (or $\mathcal{S}(0, c)$), alternative optima exist if and only if the profit value r_c (or u_c) is not unique, i.e. if there exists $\tilde{c} \in \{c{+}1, \ldots, n_R\}$ (or $\tilde{c} \in \{c{+}1, \ldots, n_U\}$) such that $r_c = \ldots = r_{\tilde{c}}$ (or $u_c = \ldots = u_{\tilde{c}}$). But obviously all the resulting efficient knapsacks are connected with respect to a 1-exchange neighborhood, since we can exchange items having the same profit value one by one to construct an appropriate sequence.
Now, let $\min\{c^1, c^2\} > 0$. If for an efficient $S \in \mathcal{S}(c^1, c^2)$, there exists an alternative efficient knapsack $\bar{S} \in \mathcal{S}(c^1, c^2)$, a similar reasoning as in the case of a basis applies. Either a number of profit values in the sequences R and U are not unique or the same property applies to the profit value $\bar{d}^{c^1-c^2}_{c^2}$, if $c^2 \leq c^1$, or $\bar{d}^{c^2-c^1}_{c^1}$, if $c^1 < c^2$ of the sequence $\bar{D}^{c^1-c^2}$ or $\bar{D}^{c^2-c^1}$. But all the resulting alternative knapsacks are connected with respect to a 2-exchange neighborhood, since in the worst case we have to exchange a super-item by another super-item to construct an alternative efficient knapsack. Hence, an appropriate sequence, starting from S and ending by $\bar{S}$ such that subsequent knapsacks are neighbors with respect to the 2-exchange neighborhood can always be found within the set $\mathcal{S}(c^1, c^2)$. □

It is worth mentioning that the (proof of the) connectedness of the set of efficient knapsacks is not based on the connectedness of supported efficient knapsacks that are always connected (cf. Chapter 7), but that the proof is constructive.

9.6 Conclusions and Further Ideas

In this chapter we presented efficient algorithms to solve interesting special cases of three $\mathcal{NP}$-hard optimization problems within a polynomial amount of time. In par-

Step	Greedy Solution		Optimal Solution	
	profit value	171	174	profit value
1	L	22	22	L
2	$L+R$	$16+12=28$	$16+12=28$	$L+R$
3	$L+R$	$9+5=14$	8	RL
4	D	40	40	D
5	D	39	39	D
6	$R+LU$	28	$9+28=37$	$L+RU$

Table 9.4: *Steps of the greedy algorithm for an instance of Problem (3-KP$_=$).*

ticular, for the case of Problem (2-MP), our implementation of the algorithm is able to find the complete non-dominated set in half an hour for instances with one million items, which corresponds to more than 400 millions of distinct solutions on average. This result is based on the fact that the non-dominated set of Problem (2-MP) can be found within $\mathcal{O}(n^2)$ time, where no additional filtering for dominated solutions has to be applied during the course of the suggested algorithm. This further implies that the presented algorithm is optimal in terms of upper bound time complexity. In addition to this result, we proved that the set of efficient knapsacks is connected with respect to a combinatorial definition of adjacency.

It is further important to mention that in this chapter we intensively made use of the ideas developed in Chapter 3 to construct solution concepts for single objective problems based on multiple objective approaches and vice versa. Starting from an algorithm for the single objective problem (2-KP$_=$), we developed an algorithm that solves the triobjective unconstrained combinatorial optimization problem (2-MP). Then, we went the way back to the single objective optimization problem (2-KP$_\leq$) and derived an efficient algorithm for this problem, based on the results for the associated multiple objective problem (2-MP).

Concerning further ideas for research on the problems presented in this chapter, one could think about an extension of the presented algorithms to higher dimensional problems involving more than two binary weight constraints. Unfortunately, we have to remark that although the proposed greedy algorithm for Problem (2-KP$_=$) may suggest that this approach can be easily extended to higher dimensions (i.e. to more than two binary constraints), depending only on the way the items are partitioned in the pre-processing step and using a similar decomposition as in (9.1), this is not the case, not even for the Problem (3-KP$_=$), as shown in the following example.

Example 9.34 Consider an instance of Problem (3-KP$_=$). We partition the set of items according to their weights $(w^1(s), w^2(s), w^3(s))$ for all items s and obtain seven different sets where all elements in a set have the weights $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, $(1,1,0)$, $(1,0,1)$, $(0,1,1)$ and $(1,1,1)$, respectively. We denote these sets by R, L, U, RL, RU, LU and D, respectively. Consider the following partitioning of items: $R = (12,5,4)$, $L = (22,16,9)$, $U = (7,6,5)$, $RL = (8,5,4)$, $RU = (28,8,7)$, $LU = (24,9,7)$,

$D = (40, 39, 20)$, and a constraint given by $c = (5, 6, 3)$, it holds that

$$\begin{pmatrix} 5 \\ 6 \\ 4 \end{pmatrix} = 1 \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 2 \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + 3 \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

The left hand side of Table 9.4 shows which solutions would be included by the greedy approach using the above decomposition while the right hand side shows the corresponding decomposition of the optimal solution. Choosing the last element from L in step 3 for the greedy solution is "the wrong choice" since it blocks all elements from UR for further inclusion in better solutions.

From Example 9.34 we conclude that an extension to higher dimensional problems cannot be longer based on a greedy strategy, but decision rules have to be found, under which conditions specific items have to be included in an optimal knapsack for a generalized Problem (m-KP$_=$).
Furthermore, it could be of interest to prove positive results for the connectedness of the set of efficient knapsacks in the case that $m \geq 3$. Obviously, instead of using a 2-exchange neighborhood, an m-exchange neighborhood is potentially need to prove connectedness results for the general case, whenever they may exist. Based on a positive result, simple local search strategies (like the one proposed in Paquete et al. [159]) could be used to solve the higher dimensional problem that are not directly based on the greedy strategies suggested in this chapter for the case $m = 2$.
We finally remark that solution concepts for Problem (2-KP$_\leq$) with an additional cardinality constraint could be investigated because of the following reason: Consider a triobjective k-max optimization problem (cf. Chapter 5), involving a profit function p to maximize and two k-max objectives to minimize, where $|S| = m$ (where $m < n$) holds for all S contained in the feasible set X. The solution approach for solving multiple objective k-max problems, suggested in Section 5.2, implies that the given triobjective problem has to be transformed into a sequence of ε-constraint problems (A_J) (cf. Section 5.2), that have to be solved. Unfortunately, the structure of these problems does not directly coincide with the structure of Problem (2-KP$_\leq$) for an appropriate right hand side vector $c \in \mathcal{G}$, since an additional cardinality constraint on the feasible set is involved. Hence, developing an algorithm that solves Problem (2-KP$_\leq$) with an additional cardinality constraint would imply that also the above stated triobjective problem can be solved efficiently.
Note that one might also think about relaxing the cardinality constraint of the given triobjective k-max problem to derive a new problem that can be solved by Algorithm A.2 for Problem (2-KP$_\leq$) suggested in Appendix A. But this relaxation is obviously not reasonable, since the involved k-max objectives are no longer well-defined for the resulting problem, since a minimum cardinality of feasible solutions can no longer be guaranteed.

Chapter 10

Biobjective Optimization Problems on Matroids with Binary Costs

Combinatorial optimization problems on matroids are frequently studied in the literature of classical combinatorial optimization. Indeed, the fundamental ideas, matroid theory is based on, can already be found in the article of Whitney [217] from the middle 1930's. Reviewing the complete literature of this field is beyond the scope of this chapter. For a deeper insight into the theory of matroids and its history we refer the interested reader, for example, to the books of Kung [122] and Oxley [156].
In this chapter we discuss a specially structured biobjective optimization problem on matroids. While the first objective can take arbitrary non-negative integer values, the second objective is restricted to take binary values only. We prove that the nondominated set of such a problem can be determined based on swaps between elements contained in different (efficient) bases of the problem. We take advantage of an algorithm stated in Gabow and Tarjan [67] to derive a modified version of this algorithm that is guaranteed to generate a complete set of efficient solutions of the given problem. Furthermore, we prove that the set of efficient solutions is always connected for this special type of problem. To the best of our knowledge, this is the first non-trivial problem on matroids where connectedness of $\mathcal{X}_E$ can be established.
The remainder of this chapter is organized as follows. In Section 10.1 we recall the main definitions and results from matroid theory that are relevant for the subsequent sections. In Section 10.2 we introduce the biobjective matroid problem involving a binary cost objective, and we give a short review of the existing literature related to this special type of matroid problem. In Section 10.3 we show how the problem under consideration can be solved efficiently, based on an algorithm developed by Gabow and Tarjan [67]. We prove the connectedness of the efficient set for this special type of problem in Section 10.4 and finally conclude in Section 10.5 with a short summary of our results and further ideas of research.

10.1 Matroid Preliminaries

In this section we review some basic facts and concepts from matroid theory. We concentrate on basic definitions and results that we need in the following sections. For more details on matroid theory we refer to the books of Kung [122] and Oxley [156].

Let $\mathcal{E} = \{e_1, \ldots, e_n\}$ denote a finite set of $n \in \mathbb{N}$ elements and $|S| \in \{0, \ldots, n\}$ the cardinality of a subset $S \subseteq \mathcal{E}$. A pair $(\mathcal{E}, \mathcal{I})$, where $\mathcal{I}$ is a subset of the power set $\mathcal{P}(\mathcal{E})$ of $\mathcal{E}$, is called *independence system*, if $\mathcal{I} \neq \emptyset$, and if for $S \in \mathcal{I}$ and $T \subseteq S$, it is implied that also $T \in \mathcal{I}$ holds. In this case, $S \in \mathcal{I}$ is called an *independent set*. A subset of $\mathcal{E}$ that is not contained in $\mathcal{I}$ is called *dependent*.
For a given independence system $(\mathcal{E}, \mathcal{I})$, an independent set $S \in \mathcal{I}$ is called *maximal*, if $S \cup \{e\} \notin \mathcal{I}$ for all $e \in \mathcal{E} \setminus S$. For $T \subseteq \mathcal{E}$, $\operatorname{rank}(T) := \max\{|S| : S \subseteq T, S \in \mathcal{I}\}$ is called *rank* of T.
An independence system $(\mathcal{E}, \mathcal{I})$ is called *matroid* $\mathcal{M}$, if for $S, T \in \mathcal{I}$ and $|S| < |T|$ it holds that there exists $t \in T \setminus S$ such that $S \cup \{t\} \in \mathcal{I}$. A maximal independent subset of $\mathcal{E}$ is called *basis* of the matroid. The set of all bases is denoted by $\mathcal{X}$ in the following. If $S, T \in \mathcal{X}$ and $e_S \in S \setminus T$, then there exists another element $e_T \in T \setminus S$ such that $(S \cup \{e_T\}) \setminus \{e_S\}$ also forms a basis of $\mathcal{M}$ *(basis exchange property)*. Furthermore, all sets contained in $\mathcal{X}$ have the same cardinality, i.e. for $S, T \in \mathcal{X}$ it holds that $|S| = \operatorname{rank}(S) = \operatorname{rank}(T) = |T| = m$ for a fixed integer $m \in \{0, \ldots, |\mathcal{E}|\}$.
A minimal dependent set of a matroid $\mathcal{M}$ is called a *cycle* of $\mathcal{M}$. Given a basis $B \in \mathcal{X}$ and an element $e \in (\mathcal{E} \setminus B)$, then $B \cup \{e\}$ contains a uniquely determined cycle $C(e, B)$ (or only C for short) containing e. This cycle is also called the *fundamental cycle* of e with respect to B.
If $S \subseteq \mathcal{E}$, *deleting* S from $\mathcal{M}$ defines the matroid $\mathcal{M} - S$. The ground set of this matroid consists of all elements contained in $\mathcal{E} \setminus S$. The independent sets of $\mathcal{M} - S$ consist of the independent sets of $\mathcal{M}$ that do not intersect S.
If S is an independent set of $\mathcal{M}$, *contracting* S gives the matroid $\mathcal{M}/S$. The ground set of this matroid once again consists of all elements contained in $\mathcal{E} \setminus S$. Its independent sets (and bases) are given by the sets $T \subseteq (\mathcal{E} \setminus S)$ such that $T \cup S$ is an independent set (basis) of $\mathcal{M}$.
For each element $e \in \mathcal{E}$ we introduce p non-negative weights $w_i(e) \in \mathbb{R}_0^+$ $(i = 1, \ldots, p)$. For $S \subseteq \mathcal{E}$ and $i = 1, \ldots, p$ we define $w_i(S) = \sum_{e \in S} w_i(e)$ and set $w(S) = (w_1(S), \ldots, w_p(S))$. The *multiple objective matroid problem* (MOMP) is given by

$$\begin{aligned} \min w(S) &= (w_1(S) \ldots, w_p(S)) \\ \text{s.t. } & S \in \mathcal{X}. \end{aligned} \tag{MOMP}$$

The set of efficient bases of this problem is denoted by $\mathcal{X}_E$, and its corresponding set of non-dominated solutions by $\mathcal{Y}_N = w(\mathcal{X}_E)$. We recall that an efficient basis S is called *supported efficient*, if it is a minimizer of the non-trivial weighted sum problem $\min\{\sum_{i=1}^p \lambda_i w_i(S),\ S \in \mathcal{X}\}$ for $\lambda_i \in (0, 1)$, $i = 1, \ldots, p$ and $\sum_{i=1}^p \lambda_i = 1$. Equivalently, one can show that S is supported if and only if $w(S)$ is an element of the non-dominated frontier that is defined as non-dominated set of $\operatorname{conv}(\mathcal{Y})$, where $\mathcal{Y} = w(\mathcal{X})$ (cf. also Figure 4.1).
For the main concepts on the connectedness of the efficient set for Problem (MOMP) as well as the definition of the efficiency graph for combinatorial problems, we refer to Chapter 7. We recall from Section 7.2 that the efficiency graph $\mathcal{G}$ for optimization problems on matroids is not connected in general, even for the case that also weakly efficient bases are considered. Nevertheless, $\mathcal{G}$ contains a connected subgraph, since the set of all supported efficient bases is always connected as shown in Ehrgott [52].

Concerning the complexity of matroid problems we note that the single objective version of Problem (MOMP) can be solved efficiently by a simple greedy strategy (cf. Oxley [156]). By contrast, the decision problem of MOMP is proven to be $\mathcal{NP}$-complete in general Ehrgott [52].
As a special case of a matroid $(\mathcal{E}, \mathcal{I})$, we introduce the *graphic matroid* $\mathcal{M}(G)$ of a connected graph $G = (V, A)$, where V denotes the set of vertices and A is the set of edges of G. For a graphic matroid, the ground set $\mathcal{E}$ is given by A, $\mathcal{I}$ corresponds to all subgraphs of G that do not contain a cycle, and the set of bases $\mathcal{X}$ is given by the set of all spanning trees of G. Note that an independent set of $\mathcal{M}$ is also called *forrest* in graph theory. For a graphic matroid, Problem (MOMP) is called *multiple objective minimum spanning tree problem* (MSTP). In the next sections we will use graphic matroids to illustrate the discussion on general matroids. Concerning the complexity of this special matroid, Chazelle [37] showed that the single objective minimum spanning tree problem can be solved in $\mathcal{O}(\alpha(|V|, |A|) \cdot |A|)$, where α is the classical functional inverse of the Ackermann's function. Although MSTP is only a special case of a multiple objective problem on matroids, it is also proven to be $\mathcal{NP}$-complete in general (cf. Camerini et al. [32]). For a detailed survey of existing results and algorithms for MSTPs, we refer to the survey of Ruzika and Hamacher [185].

10.2 Problem Formulation and Notation

In this section we present the problem formulation of the biobjective matroid problem involving a binary cost function, and we give a short survey of the existing literature for this special type of problem. Furthermore, we introduce the notation we need for the proofs in the subsequent sections.
Let a matroid $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ be given. We denote the set of all bases of $\mathcal{M}$ by $\mathcal{X}$, and we assume that $\operatorname{rank}(B) = m > 0$ for all $B \in \mathcal{X}$. Furthermore, let two different cost functions $c : \mathcal{E} \to \mathbb{N}$ and $b : \mathcal{E} \to \{0, 1\}$ on the elements of $\mathcal{E}$ be given. While the first function c is assumed to have arbitrary non-negative integer coefficients, we assume that the cost function b only takes binary values on elements of the ground set. Then, the two different costs of a basis $B \in \mathcal{X}$ are given by $c(B) = \sum_{e \in B} c(e)$ and $b(B) = \sum_{e \in B} b(e)$. The corresponding *biobjective matroid problem with binary costs* (BBMP) is given by

$$\min_{B \in \mathcal{X}} (c(B), b(B)) . \tag{BBMP}$$

The set of efficient bases of Problem $(BBMP)$ is denoted by $\mathcal{X}_E$ and its image under the two cost objectives (c, b) by $\mathcal{Y}_N$ in the following. Note that, since $b(\mathcal{X}) := \{b(B) : B \in \mathcal{X}\} \subseteq \{0, \dots, m\}$ is of size $\mathcal{O}(m)$, this also holds true for $\mathcal{Y}_N$, i.e. $\mathcal{Y}_N$ is of polynomial size.
One of the two ε-constraint versions of Problem $(BBMP)$ is given by

$$\begin{aligned} &\min c(B) \\ \text{s.t. } &b(B) \leq k, \\ &B \in \mathcal{X}, \end{aligned} \tag{BMP$_\leq$}$$

where $k \in \{0, \ldots, m\}$ is a fixed bound on the binary cost function b. We recall from Chankong and Haimes [36] that each optimal solution of Problem $(BMP_{\leq})$ is at least weakly efficient for Problem $(BBMP)$. If the inequality constraint is substituted by an equality constraint, we obtain

$$\begin{aligned} & \min c(B) \\ \text{s.t. } & b(B) = k, \\ & B \in \mathcal{X}. \end{aligned} \tag{$BMP_{=}$}$$

While an optimal basis of Problem $(BMP_{\leq})$ is at least weakly efficient for Problem $(BBMP)$, an optimal solution of Problem $(BMP_{=})$ may correspond to a dominated solution of Problem (MOMP) in general.

We give another interpretation of Problems $(BMP_{\leq})$ and $(BMP_{=})$ that is used in the articles of Gabow and Tarjan [67] and Gusfield [91]: Given a matroid $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ and a cost function $c : \mathcal{E} \to \mathbb{N}$ on the elements of $\mathcal{E}$, we additionally assign two different colors, red and green, to the elements of $\mathcal{E}$. This defines a partition of the set $\mathcal{E}$ in red and green elements. To establish a connection to the binary objective function b of Problem $(BMP_{=})$, we identify red elements $r \in \mathcal{E}$ with the binary cost $b(r) = 0$, while green elements $g \in \mathcal{E}$ are of binary cost $b(g) = 1$. Then, Problem $(BMP_{\leq})$ (and $(BMP_{=})$) consists of determining a minimum cost basis $B \in \mathcal{X}$ containing at least (or exactly) $m - k$ red elements from $\mathcal{E}$. Hence, especially Problem $(BMP_{=})$ can be seen as a generalized version of a single objective matroid problem with an additional constraint, where the original problem is obtained when $\mathcal{E}$ only consists of red elements and $k = 0$.

Gabow and Tarjan [67] as well as Gusfield [91] presented an algorithm that solves Problem $(BMP_{=})$ for fixed $k \in \mathbb{N}$. Note that while general constrained matroid problems are proven to be $\mathcal{NP}$-complete (cf. Camerini et al. [32] for graphic matroids), we will show in the following section that based on the (complexity) results of Gabow and Tarjan [67], the non-dominated set of Problem $(BBMP)$ can be found in polynomial time.

From a different point of view, Problem $(BMP_{=})$ can additionally be interpreted as a special case of a matroid intersection problem. We define a matroid $\mathcal{M}^{\star} = (\mathcal{E}, \mathcal{I}^{\star})$ where every basis of $\mathcal{M}^{\star}$ contains exactly k green and $n - k$ red elements from $\mathcal{E}$. Obviously, $(\mathcal{M}^{\star}, \mathcal{E})$ is a partition matroid (cf., e.g., Gabow and Tarjan [67] for a general definition). Hence, Problem $(BMP_{=})$ consists of finding a minimum cost basis of both matroids $\mathcal{M}$ and $\mathcal{M}^{\star}$. For more details on general matroid intersection problems as well as solution approaches we refer, amongst others, to the articles of Frank [66] and Brezovec et al. [26].

Concerning the literature on problems on matroids that are closely related to those studied in this chapter, the results for matroid problems with two colors were extended to problems with multiple colors by Rendl and Leclerc [182] and Brezovec et al. [27], respectively. In these articles, the ground set $\mathcal{E}$ is partitioned into $k \geq 2$ disjoint subsets such that elements from different subsets are assigned different colors. The multicolor matroid problem aims to find a minimum cost basis such that a fixed upper bound on the cardinality of elements from different subsets is not exceeded.

In Rendl and Leclerc [182], the authors start from an (in general infeasible) cost minimal basis and perform cost minimal exchanges with elements that are not yet contained

in the basis, until all cardinality constraints on the different colors are satisfied. Based on similar ideas, Brezovec et al. [27] derived an algorithm that also takes a lower bound on the number of unicolored elements in a basis into account. Srinivas [194] extended the results for the case that linear inequalities on the cardinality of different colors are given that are allowed to be in a feasible basis.
Finally, Hamacher and Rendl [95] considered general combinatorial optimization problems where a subdivision of the ground set into $k \geq 2$ not necessarily disjoint subsets is given such that each subset is assigned a different color. This implies that an element of $\mathcal{E}$ may have different colors at the same time. The authors aimed to find a cost minimal basis such that a fixed cardinality constraint on each color is satisfied. Amongst others, the authors showed that this problem can be reduced to a parametric problem with pairwise disjoint colors, and they derived polynomial time bounds for special classes of combinatorial optimization problems including colored bipartite matching problems.
In the case of a graphic matroid, Darmann and Pferschy [45] developed a so called *cycle improvement algorithm* to find an optimal solution for the constrained version of Problem $(BBMP)$, where a fixed bound is set on the cost function c, while b is treated as the only objective.
A short example of their approach is given in Example 10.5, contained in the next section. We note that the theoretical results of this unpublished article are closely related to those of Gabow and Tarjan [67].In more detail, the algorithm that is proposed by Darmann and Pferschy [45] is outperformed by the algorithm stated in [67]. For more details on this topic, we also refer to Section 10.3.
Finally, Climaco et al. [39] presented an algorithm to solve the biobjective minimal cost/ minimal label spanning tree problem. In this problem, each edge of a given graph is assigned a cost value and a label (color). While the first criterion aims to minimize the cost of a spanning tree, the second criterion intends to find a solution with a minimum number of different labels. Since it is already $\mathcal{NP}$-hard to determine the minimum label spanning tree on a given graph due to a result of Chang and Leu [35], the considered problem is also $\mathcal{NP}$-hard to solve.
To simplify the notation for the next sections, we introduce the following simple abbreviations for operations on sets: Let $S \subseteq \mathcal{E}$ be a set and let $e, f \in \mathcal{E}$, then $S+e$ denotes the set $S \cup \{e\}$, while $S - f$ denotes the set $S \setminus \{f\}$, provided that f is contained in S. To omit the use of multiple parenthesis, we constitute that operations on sets are always performed from the left to the right. For example, $S - f + e = (S \setminus \{f\}) \cup \{e\}$ means that first f is excluded from S, followed by the inclusion of e. Furthermore, given $S \subseteq \mathcal{E}$, $S^c := \mathcal{E} \setminus S$ denotes the complement of S in $\mathcal{E}$.
In the following, we assume that the elements of the ground set $\mathcal{E}$ are partitioned with respect to their binary value into the two subsets E_0 and E_1, where

$$E_0 = \{e \in \mathcal{E} : b(e) = 0\} \text{ and } E_1 = \{e \in \mathcal{E} : b(e) = 1\}.$$

For $i \in \{0, \ldots, m\}$, we define $\mathcal{X}_i := \{B \in \mathcal{X} : |B \cap E_0| = i\}$ as the set of bases containing exactly i elements of binary value zero. Furthermore, let

$$\mathcal{S}_i = \{B \in \mathcal{X}_i : c(B) \leq c(B') \, \forall B' \in \mathcal{X}_i\}$$

denote the set of minimum cost bases contained in $\mathcal{X}_i$. Note that any $B \in \mathcal{S}_i$ is an optimal basis of Problem $(BMP_=)$ with right hand side value $k = m - i$.

10.3 Solving Biobjective Matroid Problems with Binary Costs

In this section we present an algorithm that solves Problem ($BBMP$) efficiently. In more detail, we deduce from the results stated in Gabow and Tarjan [67] that each instance of Problem ($BBMP$) is solvable in a polynomial amount of time. Based on the algorithm stated in [67], we present an adapted version of this algorithm that generates the complete non-dominated set of Problem ($BBMP$) based on a generated sequence of optimal solutions for Problems ($BMP_{=}$).
In the following Subsection 10.3.1, we present the main results we need to prove the correctness of the proposed algorithm, as well as to show that the set of efficient bases for Problem ($BBMP$) is always connected (see Section 10.4), as it consists of supported efficient bases only. We present our algorithm in Subsection 10.3.2.

10.3.1 Bases and Minimal Swaps

Before we define what is meant by the notion of a *minimal swap* for a given basis, we recall the following stronger version of the simple basis exchange property for matroids, that is needed in several proofs in this section. Note that this stronger version was firstly proven in Brualdi [28].

Lemma 10.1 ([28]) *Let $B, B' \in \mathcal{X}$. For $e \in B \setminus B'$ there exists $f \in B' \setminus B$ such that both $B - e + f$ and $B' - f + e$ are bases of the given matroid.*

Given the partition $\mathcal{E} = E_0 \cup E_1$ stated in Section 10.2, we introduce the notion of a *swap* between elements from E_0 and E_1.

Definition 10.2 *Let $B \in \mathcal{X}$. Then the* swap (e, f) *(*involving e*,* for f *or* for basis B*) is an ordered pair of elements, where it is assumed that $e \in B \cap E_1$, $f \in E_0 \setminus B$ and $B - e + f$ forms a basis of the given matroid. The* cost *of a swap (e, f) is defined by $c(e, f) := c(f) - c(e)$. A swap (e, f) is called* minimal*, if $c(e, f) \leq c(e, f')$ for all $f' \in E_0 \setminus B$.*

Note that a swap (e, f) for B always guarantees that the binary objective value of the new basis $B - e + f$ is improved, since an element from E_1 is replaced by an element from E_0. Extending the notion of adjacency of efficient bases to the complete feasible set $\mathcal{X}$ (i.e. to all bases) implies that B and $B - e + f$ are always adjacent. Based on this observation, we will prove the connectedness of the efficient set in Section 10.4.
Another important result needed for this proof is the following theorem already proven in Gabow and Tarjan [67]: Given an optimal solution $B \in \mathcal{S}_{i-1}$, we can construct an optimal solution contained in $\mathcal{S}_i \neq \emptyset$, based on a simple minimal swap for B.

Theorem 10.3 ([67]) *Let $B \in \mathcal{S}_{i-1}$ for an $i \in \{1, \ldots, m\}$ and assume that $\mathcal{S}_i \neq \emptyset$. If the swap (e, f) is minimal, then $B - e + f$ is contained in $\mathcal{S}_i$.*

Proof: We follow the ideas of the proof stated in Gabow and Tarjan [67]. Obviously, it suffices to show that there exists a swap (e, f) for B such that $B - e + f \in \mathcal{S}_i$ holds. Let $B \in \mathcal{S}_{i-1}$ and assume that $\mathcal{S}_i \neq \emptyset$. We choose $B' \in \mathcal{S}_i$ such that $|B \cap B'|$ is maximum. Since $b(B') = m - i < m - i + 1 = b(B)$, there exists $f \in (B' \setminus B)$ such that

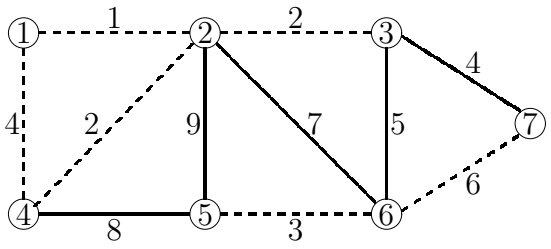


Figure 10.1: *Graph $G = (V, A)$. Solid lines correspond to edges e with $b(e) = 0$ while the dashed lines correspond to edges with $b(e) = 1$.*

$f \in E_0$. Applying the stronger version of the basis exchange property (Lemma 10.1), there must exist an element $e \in B \setminus B'$ such that both $T := B - e + f$ as well as $T' := B' - f + e$ are bases of the given matroid. We show that $e \in E_1$, i.e. $b(e) = 1$. Assume that this is not the case, i.e. $b(e) = 0$. Then, $b(T) = b(B)$ and $b(T') = b(B')$, and since $B \in \mathcal{S}_{i-1}$ and $B' \in \mathcal{S}_i$ it holds that

$$\begin{aligned} c(B) \leq c(T) &= c(B) - c(e) + c(f)\,, \text{ i.e. } c(e) \geq c(f), \\ c(B') \leq c(T') &= c(B') - c(f) + c(e), \text{ i.e. } c(f) \geq c(e). \end{aligned}$$

This implies that $c(e) = c(f)$ and hence, $c(T') = c(B')$, i.e. $T \in \mathcal{S}_i$. Since by construction, $e \in (T' \setminus B')$, this implies that T' and B have more elements in common than B' and B do. But this contradicts the choice of B'.
Since $e \in E_1$, this implies that $b(T) = b(B')$ and $b(T') = b(B)$. From the optimality of B and B' we finally conclude that

$$c(B') \leq c(T) = c(B) - (c(e) - c(f)) = c(B) - c(T') + c(B') \leq c(B').$$

Hence, $c(T) = c(B')$, i.e. $T \in \mathcal{S}_i$, as desired. □

The proof of Theorem 10.3 implies that the constructed swap (e, f) for B is minimal and further that any minimal swap for $B \in \mathcal{S}_{i-1}$ leads to an optimal basis $B' \in \mathcal{S}_i$. Following the ideas in Gabow and Tarjan [67], we deduce:

Corollary 10.4 *Let $s, t \in \mathbb{N}$ with $0 \leq s < t \leq m$ such that $\mathcal{S}_s \neq \emptyset \neq \mathcal{S}_t$. Then $\mathcal{S}_i \neq \emptyset$ for all $i \in \{s, \ldots, t\}$.*

Proof: The proof of the corollary follows immediately from the proof of Theorem 10.3: Let $B \in \mathcal{S}_s \neq \emptyset$ arbitrary but fixed. We choose $B' \in \mathcal{S}_t \neq \emptyset$ such that $|B \cap B'|$ is maximum. The proof of Theorem 10.3 implies that $\mathcal{S}_{s+1} \neq \emptyset$. By a simple induction it follows that $\mathcal{S}_i \neq \emptyset$ for all $i \in \{s, \ldots, t\}$. □

Given bases with a minimum and a maximum number of elements contained in E_0, Corollary 10.4 implies that there exist fixed lower and upper bounds l and u (satisfying $0 \leq l \leq u \leq m$), such that $\mathcal{S}_i \neq \emptyset$ for all $i \in \{l, \ldots, u\}$. If B corresponds to a basis such that $|B \cap E_0| \leq |B^\star \cap E_0|$ for all $B^\star \in \mathcal{X}$, we have that $l = |B \cap E_0|$, while $u = |B' \cap E_0|$, where B' is a basis such that $|B' \cap E_0|$ is maximum for all bases contained in $\mathcal{X}$.
Given an instance of Problem $(BBMP)$, Theorem 10.3 and Corollary 10.4 immediately imply a simple algorithm to determine the non-dominated set of the problem: Since

each efficient basis B of Problem $(BBMP)$ is also optimal for Problem $(BMP_{=})$ for the right hand side value $k = b(B)$, i.e. $B \in \mathcal{S}_{m-k}$, a superset of representatives of the non-dominated set can be determined by swapping between the optimal bases contained in $\mathcal{S}_i$ for $i = \{l, \dots, u\}$.
Let $B_l \in \mathcal{S}_l$, and set $A_0 := E_0 \setminus B_l$. We iteratively apply the following procedure first to B_l and then to the basis B that was generated in the last iteration, until no further replacement is possible:

(1.) For all elements $f_i \in A_0$ find a minimal swap $c(e_i, f_i)$ for f_i and B, if it exists.

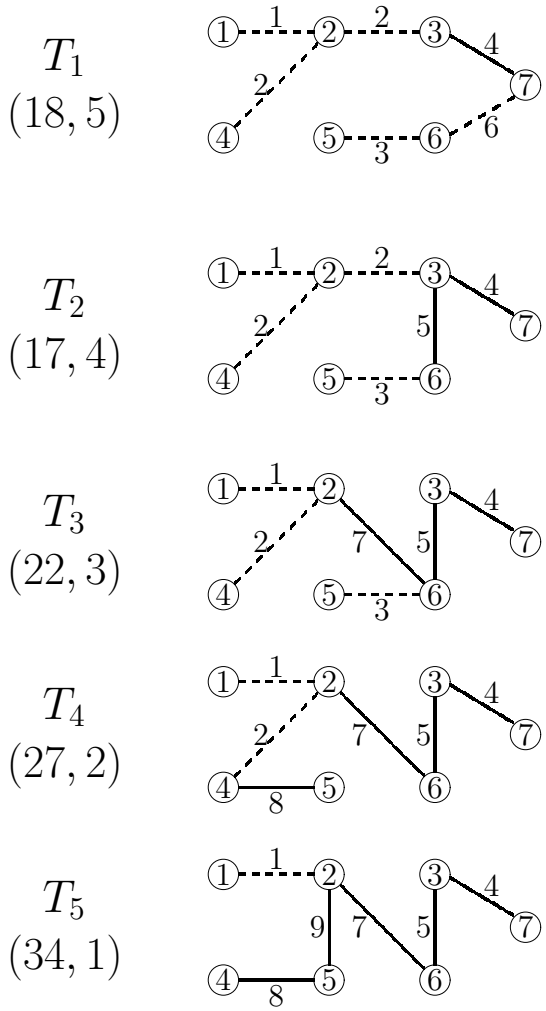
(2.) Determine an element $f_j \in A_0$ such that $c(e_j, f_j) \leq c(e_i, f_i)$ for all $f_i \in A_0$.

(3.) Generate a new basis B' by performing the swap $c(e_j, f_j)$ and delete f_j from A_0.

Note that the minimal swap in the first step of the above given procedure may not be unique. In this case, we choose a swap with the minimal costs with respect to $f \in A_0$. We give a short example of this procedure on a graphic matroid $\mathcal{M}(G)$.

Example 10.5 Consider the graph $G = (V, A)$ given in Figure 10.1, where solid lines correspond to edges of binary costs zero, while the dashed lines represent edges of binary costs one. Furthermore, the costs in the second objective c are given as stated next to the particular edges in the figure. We aim to find the complete non-dominated set for the graphic matroid $\mathcal{M}(G)$ using the above stated procedure.
The sequence of (optimal) spanning trees $\{T_1, \dots, T_5\}$ generated by this procedure can be found in Figure 10.2. Obviously, $T_i \in \mathcal{S}_i$ holds true for all $i \in \{1, \dots, 5\}$. In the left subfigure of Figure 10.2, the index of the optimal spanning tree T_i as well as its corresponding objective vector are given. The corresponding spanning trees are depicted in the middle of the figure, while on the right the minimal swap $c(e, f)$ for f and T_i for the current iteration is determined. Note that the edges stated in the "in"-column correspond to the edges $f \in A_0$, while the edges stated in the "out"-column correspond to edges $e \in T_i \cap E_1$. A swap with the minimum cost $c(e, f)$ is applied to T_i to generate an optimal spanning tree $T_{i+1} \in \mathcal{S}_{i+1}$. Furthermore, $l = 1$ and $u = 5$ in this example. The set of non-dominated solutions for the considered problem is given by $\mathcal{Y}_N = \{(17, 4); (22, 3); (27, 2); (34, 1)\}$.
Note that the optimal spanning tree $T_1 \in \mathcal{S}_1$ that contains exactly one edge from E_0 is dominated by the tree T_2, since the swap with the minimum cost is negative, i.e. the binary cost objective b as well as the cost objective c is improved by performing the swap. Based on the index $i = 2$, the remaining spanning trees correspond to efficient spanning trees of the problem and form a complete set of efficient solutions.

Darmann and Pferschy [45] showed in their unpublished article that applying the above described procedure to a graphic matroid $\mathcal{M}(G)$ on a connected graph $G = (V, A)$, it is possible to find the minimum spanning tree with respect to the binary objective b and a constraint on the cost objective c in $\mathcal{O}(|V|^2|A|)$. We omit further details on their algorithm here, as their approach is outperformed by the algorithm stated in [67].
The main drawback of the above described procedure is the fact that in each iteration a number of $\mathcal{O}(m)$ minimal swaps has to be calculated, resulting in only a single swap that is really performed at the end of the iteration. In this context, the calculation of a minimal swap for a fixed element $f \in A_0$ and a given basis $B_i \in \mathcal{S}_i$ includes the



in	out	$c(e,f)$
$[2,5]$	$[6,7]$	3
$[2,6]$	$[6,7]$	1
$[\mathbf{3},\mathbf{6}]$	$[\mathbf{6},\mathbf{7}]$	$\mathbf{-1}$
$[4,5]$	$[6,7]$	2

in	out	$c(e,f)$
$[2,5]$	$[5,6]$	6
$[\mathbf{2},\mathbf{6}]$	$[\mathbf{2},\mathbf{3}]$	$\mathbf{5}$
$[4,5]$	$[5,6]$	5

in	out	$c(e,f)$
$[2,5]$	$[5,6]$	6
$[\mathbf{4},\mathbf{5}]$	$[\mathbf{5},\mathbf{6}]$	$\mathbf{5}$

in	out	$c(e,f)$
$[\mathbf{2},\mathbf{5}]$	$[\mathbf{2},\mathbf{4}]$	$\mathbf{7}$

in	out	$c(e,f)$

Figure 10.2: *The sequence of optimal spanning trees* $\{T_1, \ldots, T_5\}$ *for the considered problem in Example 10.5. On the left: the index of the tree and the corresponding objective vector. In the middle: Associated optimal spanning tree. On the right: Cost of the minimal swap* $c(e, f)$ *for* f *and* T_i*.*

determination of the uniquely defined fundamental cycle, when f is added to B_i, as well as the calculation of the element $e \in B_i \cap E_1$ with the maximum cost in c that (potentially) will be excluded from B_i, when the swap (e, f) is chosen to be performed at the end of the iteration.

We will show in Section 10.3.2 that the recursive procedure stated in Gabow and Tarjan [67], where the size of the given matroid is bisected in each iteration, prevents the calculation of numerous (minimal) swaps that will not be performed at the end of an iteration. This clearly improves the running time of the algorithm.

In the remainder of this section we concentrate on the main results that will be needed to prove that our modified version of the algorithm stated in [67] correctly determines the complete set of non-dominated solutions of a given instance of Problem $(BBMP)$. Based on the idea of swapping between optimal bases contained in $\mathcal{S}_i$ for $i = l, \ldots, u$, we will show in the following that there exists a fixed index $j \in \{l, \ldots, u\}$ such that

$B \in \mathcal{S}_i$ is efficient, whenever $i \geq j$ holds.
Considering Example 10.5 once again, we notice that the sequence of costs that result from the minimal swaps performed at the end of the individual iterations is strictly increasing in the number of iterations. We prove in the following that this sequence is non-decreasing in general. First, we state a lemma that is taken from Gabow and Tarjan [67].

Lemma 10.6 ([67]) *Let B denote a basis containing the element $e \in (B \cap E_1)$. Let (e, f) denote a minimal swap for B involving e and define $B' = B - e + f$. Given $g \in E_1 \cap (B - e)$ arbitrary but fixed, let (g, h) and (g, h') denote the minimal swap for B and B' involving g, respectively. Then, it holds that $c(g, h) \leq c(g, h')$.*

Proof: We follow the ideas of the proof stated in Gabow and Tarjan [67]. To simplify the notation, we set $T = B - g + h$ and $T' = B' - g + h'$.
If (g, h') is a feasible swap for B involving g, there is nothing to show, since the swap (g, h) is minimal by definition.
Hence, we may assume that the swap (g, h') is not a valid swap for B. Since $g \in B \setminus T'$, Lemma 10.1 implies that there exists an element $z \in T' \setminus B = \{f, h'\}$ such that both, $B - g + z$ and $T' - z + g$ are bases of the given matroid. Since the swap (g, h') is assumed to be infeasible for B, it follows that $z = f$. Since (g, h) is minimal for B involving g, we conclude that $c(g, h) \leq c(g, f)$, i.e. $c(h) \leq c(f)$ holds true. Since $T' - f + g = B - e + h'$ and (e, f) is minimal for B involving e, we further conclude that $c(e, f) \leq c(e, h')$, i.e. $c(f) \leq c(h')$. Combining these two results implies that $c(h) \leq c(f) \leq c(h')$, and hence,

$$c(g, h) \;=\; c(h) - c(g) \;\leq\; c(h') - c(g) \;=\; c(g, h'),$$

This completes the proof. □

Based on Lemma 10.6 we finally prove that the sequence of costs induced by the minimal swaps is non-decreasing for increasing $i \in \{l, \ldots, u\}$, whenever $u \geq l + 2$ holds.

Theorem 10.7 *Let $u \geq l + 2$. For $i \in \{l, \ldots, u - 1\}$ let (e_i, f_i) denote a minimal swap for $B_i \in \mathcal{S}_i$ to generate $B_{i+1} \in \mathcal{S}_{i+1}$. Then every sequence of minimal swaps $\{c(e_i, f_i)\}_{i=l}^{u-1}$ is non-decreasing, i.e. $c(e_i, f_i) \leq c(e_{i+1}, f_{i+1})$ for all $i \in \{l, \ldots, u - 2\}$.*

Proof: Let a sequence of minimal swaps be given. We choose $i \in \{l, \ldots, u - 2\}$ arbitrary but fixed. Since $e_i \in B_i \setminus B_{i+1}$, it holds that $e_i \neq e_{i+1}$. Furthermore, $e_{i+1} \in B_i$ since otherwise e_{i+1} would be contained in $B_{i+1} \setminus B_i$, i.e. $e_{i+1} = f_i$, where $1 = b(e_{i+1}) = b(f_i) = 0$, which is not possible. Hence, e_{i+1} is contained in $B_i - e_i$.
Let (e_{i+1}, f) denote a minimal swap for B involving e_{i+1}. Note that such a swap always exists, since $e_{i+1} \in B_i \setminus B_{i+2}$, so there must exist an element $z \in B_{i+2} \setminus B_i$ such that $B_i - e_{i+1} + z \in \mathcal{X}$, according to the basis exchange property for matroids. But since $B_{i+2} \setminus B_i = \{f_i, f_{i+1}\}$, (e_{i+1}, z) always defines a feasible swap for B involving e_{i+1}.
Since the minimal swap (e_i, f_i) is performed to generate B_{i+1} from B_i, we conclude that $c(e_i, f_i) \leq c(e_{i+1}, f)$. If $f = f_{i+1}$, there is nothing more to show. If $f \neq f_{i+1}$, we use Lemma 10.6 to deduce that $c(e_{i+1}, f) \leq c(e_{i+1}, f_{i+1})$, since the swap (e_{i+1}, f_{i+1}) is minimal for B_{i+1}. Combining these two results also leads to $c(e_i, f_i) \leq c(e_{i+1}, f_{i+1})$. □

Since

$$c(B_{i+1}) - c(B_i) \;=\; c(f_i) - c(e_i) \;=\; c(e_i, f_i), \tag{10.1}$$

we deduce from Theorem 10.7 that the minimal costs of bases $B \in \mathcal{S}_i$ define a convex function in $i = |B \cap E_0| \in \{l, \ldots, u\}$. This fact helps to solve the ε-constraint version of Problem $(BBMP)$ (cf. Problem $(BMP_{\leq})$) stated in Section 10.2.

Corollary 10.8 *Let a feasible instance of Problem $(BMP_{\leq})$ with right hand side value $k \in \{m-u, \ldots, m-l\}$ be given. If the basis with minimum costs is feasible, it is also optimal. Otherwise, a basis $B \in \mathcal{S}_{m-k}$ is the optimal solution of the problem.*

Proof: If the minimum cost basis B is feasible, it must be optimal. Otherwise, $|B \cap E_1| > k$, and some elements from this set have to be exchanged by elements from $E_0 \setminus B$ to find an optimal solution. But since every sequence of minimal swaps is non-decreasing according to Theorem 10.7, the optimal solution is obtained, when an optimal solution $B^\star$ contains exactly k elements from E_1. But this implies that $B^\star \in \mathcal{S}_{m-k}$. □

From Theorem 10.7 we further deduce that if c and b are conflicting, there must exist an index $j \in \{l, \ldots, u\}$ such that, based on this index, all subsequent bases contained in the sequence $\{B_i\}_{i=j}^{u}$ correspond to at least weakly efficient solutions of Problem $(BBMP)$, since the value of the binary objective function b is decreased by one unit when a swap from B_i to B_{i+1} is performed, while the corresponding value of the cost function c is not decreased. Obviously, this specific index j corresponds to the minimum index $i \in \{l, \ldots, u-1\}$, such that $c(e_i, f_i) \geq 0$ holds for the first time. Consequently, we state:

Corollary 10.9 *Let $\{B_i\}_{i=l}^{u}$ denote the sequence of optimal bases such that $B_i \in \mathcal{S}_i$ for $i \in \{l, \ldots, u\}$. Assume that $u \geq l+2$. If there exists an index $j \in \{l+1, \ldots, u\}$ such that $c(B_{j-1}) < c(B_j)$, then $c(B_i) < c(B_{i+1})$ holds true for all $i \in \{j-1, \ldots, u-1\}$.*

Proof: Let $j \in \{l+1, \ldots, u\}$ denote the index, where $c(B_{j-1}) < c(B_j)$ holds true for the first time. If $j = u$ there is nothing to show. Otherwise, it suffices to prove that $c(B_j) < c(B_{j+1})$ holds, too. From Equation (10.1) we deduce that $c(e_{j-1}, f_{j-1}) > 0$. From Theorem 10.7 it follows that

$$c(B_{j+1}) - c(B_j) \;=\; c(e_j, f_j) \;\geq\; c(e_{j-1}, f_{j-1}) \;>\; 0,$$

which implies that $c(B_j) < c(B_{j+1})$ holds true. □

Corollary 10.9 implies a method, how a minimal complete set X of efficient bases can be generated. Starting from an optimal basis contained in $\mathcal{S}_l$, we calculate a sequence of minimal swaps $\{(e_i, f_i)\}_{i=l}^{u-1}$, called the *swap sequence* in the following, that generates a sequence of bases B_i contained in $\mathcal{S}_i$ for $\in \{l+1, \ldots, u\}$. Based on the index j where $c(e_j, f_j) > 0$ holds for the first time, we add B_j as well as all subsequently calculated bases to X. Note that in this case, B_j is lexicographically optimal with respect to c and b. Corollary 10.9 implies that the remaining bases $B_{j+1}, \ldots, B_u$ also correspond to efficient solutions of the problem, and hence they form a system of representatives of the remaining non-dominated solutions contained in $\mathcal{Y}_N$. Since no non-dominated solution is missed by construction, the set X is a minimal complete set of efficient bases of Problem $(BBMP)$. Hence, we have proven:

Corollary 10.10 *Let $\{B_i\}_{i=l}^{u}$ denote the sequence of bases generated by a swap sequence and assume that $u \neq l$. If there exists a minimum index $j \in \{l, \ldots, u-1\}$ such that $c(B_j) < c(B_{j+1})$, then B_j is lexicographically optimal with respect to c and b. Furthermore, $X = \{B_j, \ldots, B_u\}$ forms a minimal complete set of efficient solutions and $\mathcal{Y}_N = \{(c(B_i), b(B_i)),\ i = j, \ldots, u\}$.*

10.3.2 The Modified Algorithm of Gabow and Tarjan

In this subsection we present a modified version of the algorithm of Gabow and Tarjan that takes into account the biobjective nature of Problem $(BBMP)$. We combine their algorithm with the results of Corollary 10.10 to construct a new algorithm that efficiently determines the complete set of non-dominated solutions for a given BBMP. Since the algorithm of Gabow and Tarjan is already proven to generate a complete swap sequence $\{(e_i, f_i)\}_{i=l}^{u-1}$ starting from $B_l \in \mathcal{S}_l$ and leading to $B_u \in \mathcal{S}_u$, we omit detailed proofs for the correctness of this part of the algorithm, since it can be found in all details in [67]. We rather explain the idea of how a complete swap sequence is generated without calculating a multiplicity of unnecessary swaps that do not lead to new efficient bases of the biobjective problem. After having stated our modified algorithm, we give a small example of how it works in practice. In more detail, we apply the algorithm to the graphic matroid already considered in Example 10.5.
To omit the calculation of unnecessary swaps in each iteration, Gabow and Tarjan use the idea that the proofs of Theorem 10.3 and Corollary 10.4 are based on: Besides $B_l \in \mathcal{S}_l$, another optimal basis $B_u \in \mathcal{S}_u$ is determined such that B_l and B_u coincide on a maximum number of elements. According to the proofs of Theorem 10.3 and Corollary 10.4 only those elements have to be swapped that are not contained in both bases simultaneously. Note that given B_l, B_u coincides with B_l in a maximum number of elements if the following two criteria are satisfied:

- B_u contains all elements from $B_l \cap E_0$.
- If $u \neq m$, i.e. $B \cap E_1 \neq \emptyset$ for $B \in \mathcal{S}_u$, then $(B_u \cap E_1) \subseteq (B_l \cap E_1)$ must hold.

Obviously, these two criteria ensure that both bases have as many elements in common as possible. Furthermore, the remaining elements from $\mathcal{E}$ that are not contained in $B_u \cup B_l$ are redundant and can be removed from the problem. Starting from this reduced problem, the swap sequence is calculated by a recursive procedure based on the following simple idea illustrated for the basis B_l:
Adding an element f from $B_u \setminus B_l \subseteq E_0$ with minimum costs to B_l generates a fundamental cycle $C(f, B_l)$. If all the remaining elements of this fundamental cycle (that are obviously all contained in B_l) are also elements from E_1, the minimal swap (e, f) for f induced by $C(f, B_l)$ must be contained in the swap sequence, since no other element of this cycle can give a better swap than the swap (e, f) does, when f is added to B_l. If there exists an element $f \in B_u \setminus B_l$ such that $C(f, B_l)$ also contains elements $e \in (B_l \cap E_0)$ (as it is always the case for the tree T_1 in Example 10.5), a minimal swap for f that will be contained in a final swap sequence cannot be deduced immediately. In this case, the complete problem is split up into two new subproblems on contracted matroids that do not intersect on the original ground set $\mathcal{E}$, with the hope that now adding f to the reduced problem implies that all remaining elements in

Algorithm 10.1 Algorithm for Biobjective Matroid Problems with Binary Costs

Input: An instance $((\mathcal{M}, \mathcal{X}, (c, b))$ of Problem $(BBMP)$.
Output: $\mathcal{Y}_N$ and a complete set X of efficient solutions.

1: $X = \emptyset$, $\mathcal{Y}_N = \emptyset$.
2: Determine a minimal basis B_l with respect to c such that B_l contains a minimum number of elements from E_0.
3: Determine a minimal basis B_u with respect to c such that B_u contains a maximum number of elements from E_0, all elements from $B_l \cap E_0$ and only those elements from E_1 that are also contained in B_l.
4: Call $P((\mathcal{M} - (B_l \cup B_u)^c)/(B_l \cap B_u), B_l \setminus B_u, B_u \setminus B_l)$ to generate a swap sequence.
5: Let $\{(e_i, f_i)\}_{i=l}^{u-1}$ denote the swap sequence found by Procedure P, where the swaps are sorted in non-decreasing order with respect to their costs.
6: Set $i = l$, $B = B_l$, $\gamma = c(B_l)$ and $\beta = b(B_l)$.
7: **while** $c(e_i, f_i) \leq 0$ **do**
8: Set $B = B - e_i + f_i$, $\gamma = \gamma + c(e_i, f_i)$, $\beta = \beta - 1$ and $i = i + 1$.
9: **end while**
10: **for** $j = i$ **to** $u - 1$ **do**
11: Set $B = B - e_j + f_j$, $\gamma = \gamma + c(e_j, f_j)$ and $\beta = \beta - 1$.
12: Set $X = X \cup \{B\}$ and $\mathcal{Y}_N = \mathcal{Y}_N \cup \{(\gamma, \beta)\}$.
13: **end for**
14: **return** X and $\mathcal{Y}_N$.

the newly generated cycle are contained in E_1. If this is still not the case, the problem is split up once more.

Since the splitting of the problem can be done by preserving swaps that are already guaranteed to be contained in a final swap sequence by the criterion given above (see [67] for further details), the problem can be split until the ground set of the contracted matroids consists of two single elements $e \in B_l$ and $f \in B_u$ only. Then, the swap (e, f) is proven to be minimal and hence, must be contained in the final swap sequence.

To summarize more formally, the algorithm of Gabow and Tarjan is based on splits of the original matroid into smaller parts, induced by a subsequent bisection of the sets $L = B_l \setminus B_u$ and $U = B_u \setminus B_l$. To keep the matroid problem feasible, the split of a matroid $\mathcal{M}$ is performed by first partitioning the sets U and L into two subsets U_1, U_2 and $G, L \setminus G$, respectively, such that on the one hand, $U_1 \subseteq E_0$ consists of the $\lfloor |U|/2 \rfloor$ smallest elements with respect to c and on the other hand, the set $B = (L \setminus G) \cup U_1$ forms a minimal basis for $\mathcal{M}$ with respect to c satisfying $B \cap E_0 = U_1$. The problem is split into two different subproblems on the contracted matroids $(\mathcal{M} - U_2)/(L \setminus G)$ (where $L = G$ and $U = U_1$ in the next iteration) and $(\mathcal{M} - G)/U_1$ (with $L = L \setminus G$ and $U = U_2$).

It is proven in [67] that this bisection of the problem preserve swaps that are contained in the swap sequence we are looking for. Hence, it it guaranteed that when the ground set of a contracted matroid finally consists of exactly one element $e \in L \subseteq (B_l \setminus B_u) \cap E_1$ and one element $f \in U \subseteq (B_u \setminus B_l) \cap E_0$, the swap (e, f) has to be an element of the desired swap sequence.

Our algorithm that solves Problem $(BBMP)$ is formulated in Algorithm 10.1, while a short outline of the bisection procedure that is recursively called during the course

Algorithm 10.2 Procedure $P(\mathcal{M}, L, U)$ to generate the swap sequence ([67])

Input: A matroid $\mathcal{M}$ and two sets of elements $L \subseteq B_l \setminus B_u$ and $U \subseteq B_u \setminus B_l$, $|L| = |U|$.
Output: A minimal swap (e, f) or two recursive calls of the procedure P.

1: **if** $|U| = 1$ **then**
2: **return** the swap (e, f), where $L = \{e\}$ and $U = \{f\}$.
3: **else**
4: Let U_1 be the set of $\lfloor |U|/2 \rfloor$ smallest elements with respect to c (contained in E_0) and set $U_2 = U \setminus U_1$.
5: Determine $G \subseteq L$ such that $B = (L \setminus G) \cup U_1$ forms a minimal basis for $\mathcal{M}$ with respect to c satisfying $B \cap E_0 = U_1$.
6: Call $P((\mathcal{M} - U_2)/(L \setminus G), G, U_1)$ to find the swaps for the elements in U_1.
7: Call $P((\mathcal{M} - G)/U_1, L \setminus G, U_2)$ to find the swaps for the elements in U_2.
8: **end if**

of Algorithm 10.1 is given in Algorithm 10.2.
Applying Algorithm 10.1, we start from a (contracted) matroid that is given by $(\mathcal{M} - (B_l \cup B_u)^c)/(B_l \cap B_u))$, whose ground set consists of all elements $(B_l \cup B_u) \setminus (B_l \cap B_u)$. In the next step, Algorithm 10.2 is called and recursively applied until a complete swap sequence is determined and returned. Since the calculated swaps may not be sorted in non-decreasing order of their costs, we have to sort them. Then, we finally apply the result of Corollary 10.10 to determine the complete non-dominated set $\mathcal{Y}_N$ and a minimal set X of representatives.
However, in Line 5 of Algorithm 10.1 as well as in Line 4 of Algorithm 10.2 it may happen that ties in the costs need to be broken. In [67], the following decision rule is proposed to solve this problem:
We assume that the elements of E_0 are sorted and indexed according to their costs c in non-decreasing order. Then, in Line 4 of Algorithm 10.2 we always choose the first $\lfloor |U|/2 \rfloor$ elements from U, while, if there exist ties in the costs of the swap sequence generated by Procedure P in Algorithm 10.1, we order those swaps in increasing order of the indices with respect to the elements that are contained in E_0. For further details on the implementation (e.g., how the bases B_l and B_u can be found and how a matroid can be contracted efficiently) and an additional iterative approach to generate a swap sequence we once more refer to [67].
Furthermore, we remark that in Algorithm 10.1 we could replace the starting basis $B_l \in \mathcal{S}_l$ by a lexicographically optimal basis $B^\star \in \mathcal{S}_i$ with respect to c and b, where $l \leq i \leq u$. But since by the recursive calls of Algorithm 10.2, the desired swap sequence can be generated efficiently, implying that a lexicographically optimal basis is automatically found, the overall complexity of the algorithm will not be affected by this simplification, also taking into account that already the basis B_l may be efficient. We summarize the above given results for Algorithm 10.1 in the following theorem.

Theorem 10.11 *Algorithm 10.1 is correct and returns a complete set of efficient solutions as well as the non-dominated set.*

Proof: The correctness of the algorithm follows immediately from the correctness of the original algorithm stated in [67] in combination with Corollary 10.10. □

Concerning the complexity of Algorithm 10.1, it is not possible to state a general

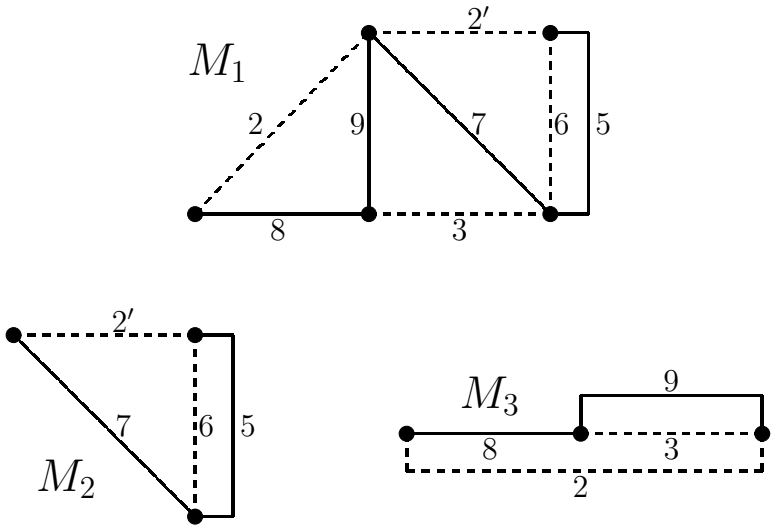


Figure 10.3: *Contracted graphic matroids M_1, M_2 and M_3 from Example 10.12. Solid lines correspond to edges e with $b(e) = 0$ while the dashed lines correspond to edges with $b(e) = 1$. The edges are identified by their cost value c.*

time bound that applies to all matroid problems simultaneously, since an efficient implementation of the algorithm always depends on the special type of the considered matroid problem. For graphic matroids $G = (V, A)$, Gabow and Tarjan [67] showed that their algorithm solves the spanning tree problem within $\mathcal{O}(m + n \cdot \log(n))$ time, where $|V| = n$ and $|A| = m$. Since the construction of X and $\mathcal{Y}_N$ additionally takes at most $\mathcal{O}(m)$ time, the stated time bound is also valid for Algorithm 10.1 and outperforms the solution procedure given in Section 10.3.1.
We close this section by applying Algorithm 10.1 to the graphic matroid of Example 10.5. To simplify the notation, we identify the edges of the graph in Figure 10.1 by their associated costs. Since the edge $[1, 4]$ is not contained in any optimal spanning tree, we can neglect it in the following. To distinguish the edge $[2, 3]$ from the edge $[2, 4]$ that are both of cost 2, we denote the cost of $[2, 3]$ by $2'$.

Example 10.12 Consider the graphic matroid shown in Figure 10.1. We apply Algorithm 10.1. The optimal bases B_l and B_u that have to be determined at the beginning of the algorithm, correspond to the spanning trees T_1 and T_5, respectively, depicted in Figure 10.2. Hence, $B_l = \{1, 2, 2', 3, 4, 6\}$ and $B_u = \{1, 4, 5, 7, 8, 9\}$ and the procedure P is called with $P(M^1, L^1, U^1)$, where $L^1 = \{2, 2', 3, 6\}$, $U^1 = \{5, 7, 8, 9\}$ and M^1 corresponds to the graphic matroid depicted in Figure 10.3. Since $|U^1| = 4 < 1$, the minimal basis $B^1 = \{2, 3, 5, 7\} \supset \{5, 7\} = U_1^1$ is calculated, and hence, $G^1 = \{2', 6\}$.
Then, the procedure is recursively called with $P(M^2, L^2, U^2)$ and $P(M^3, L^3, U^3)$, respectively, where $L^2 = G^1$, $L^3 = \{2, 3\}$, $U^2 = U_1^1$, $U^3 = \{5, 7\}$ and M^2 and M^3 correspond to the contracted graphic matroids that can be found in the bottom part of Figure 10.3. According to the above stated criteria, the first call of P returns the swaps $(6, 5)$ and $(2', 7)$ after two more recursive calls of P.
Finally, the swaps $(3, 8)$ and $(2, 9)$ are returned by the call of P with M_3. Also in this case, P has to be called recursively twice. Hence, the sorted swap sequence is given by the ordered set $\{(6, 5); (2', 7); (3, 8); (2, 9)\}$. Obviously, this sequence corresponds

to the swaps performed in Example 10.5. Applying the filter step of Algorithm 10.1 yields that $\mathcal{Y}_N = \{(17,4);(22,3);(27,2);(34,1)\}$ and $X = \{T_2,\ldots,T_5\}$ is the returned complete set of efficient solutions.

Example 10.12 completes this section. To summarize our results, we have proven that based on Corollary 10.4 it is possible to generate a complete set of efficient solutions by consecutively exchanging elements from efficient bases with elements that are not yet contained in any previously generated basis.

10.4 Connectedness of the Efficient Set

In this section we show that the set of efficient bases for BBMPs is always connected. We recall from Chapter 7 that two efficient bases are said to be adjacent, if they have $m-1$ elements in common, where m corresponds to the rank of the matroid. The efficient set is said to be connected, if its efficiency graph $\mathcal{G}$ is connected (cf. Definition 7.1). The proof given in the following is based on the fact that each non-dominated solution of the problem is a supported one, i.e. each point from $\mathcal{Y}_N$ is an element of the non-dominated frontier of the given problem. We combine this result with the result of Ehrgott [52], where it is shown that the set of supported efficient solutions is always connected with respect to the above given definition of adjacency for efficient bases.
We start with a short proof of a sufficient condition that indicates, whether the non-dominated set consists of supported non-dominated solutions only or not. For a general biobjective combinatorial minimization problem with non-dominated set $\mathcal{Y}_N = \{z_1,\ldots,z_n\} \subset \mathbb{R}^2$, where $n \geq 3$ and $z_i = (x_i, y_i) \in \mathbb{R}^2$, with $x_1 < \ldots < x_n$ and $y_1 > \ldots > y_n$, we define the sequence of slopes $\{m_i\}_{i=1}^{n-1}$ of subsequent points of $\mathcal{Y}_N$ by setting

$$m_i \;=\; \frac{y_{i+1}-y_i}{x_{i+1}-x_i}.$$

Note that by this definition, it holds that $m_i \in (-\infty, 0)$ for all $i \in \{1,\ldots,n-1\}$.

Theorem 10.13 *If the sequence of slopes $\{m_i\}_{i=1}^{n-1}$ is non-decreasing, the set of non-dominated solutions $\mathcal{Y}_N$ consists of supported non-dominated points only.*

Proof: We prove the theorem by contradiction. So assume, that there exists a non-dominated solution $z_t \in \mathcal{Y}_N$ that does not correspond to a supported one, where $t \in \{2,\ldots,n-1\}$. Since a non-dominated solution is supported if and only if it is an element of the non-dominated frontier of the given problem, there must exist supported non-dominated solutions $z_i, z_j \in \mathcal{Y}_N$ and $\lambda \in (0,1)$ such that the point $z_\lambda = (x_\lambda, y_\lambda) := \lambda z_i + (1-\lambda) z_j \in \mathbb{R}^2$ strongly dominates z_t, where $1 \leq i < t < j \leq n$ holds. Note that z_λ does not correspond to a feasible outcome of the given problem, since otherwise z_t would be dominated by z_λ (cf. also Figure 4.1). Without loss of generality, we may assume that $i = 1$ and $t = 2$.
Since $x_1 < x_\lambda < x_2$ and $y_\lambda < y_2 < y_1$ holds, we conclude that

$$(y_\lambda - y_1)\cdot(x_2 - x_1) \;<\; (y_2 - y_1)\cdot(x_2 - x_1) \;<\; (y_2 - y_1)\cdot(x_\lambda - x_1) \;<\; 0.$$

Since z_λ is an element of the straight line connecting z_1 and z_j, it follows that

$$m^\star := \frac{y_j - y_1}{x_j - x_1} = \frac{y_\lambda - y_1}{x_\lambda - x_1} < \frac{y_2 - y_1}{x_2 - x_1} = m_1.$$

But this is not possible, since by assumption $m_1 \leq m_i$ for all $i \in \{1, \ldots, n\}$, and hence

$$\begin{aligned} y_j &= y_1 + \sum_{i=1}^{j-1}(y_{i+1} - y_i) = y_1 + \sum_{i=1}^{j-1} m_i \cdot (x_{i+1} - x_i) \\ &\geq y_1 + m_1 \cdot \sum_{i=1}^{j-1}(x_{i+1} - x_i) = y_1 + m_1 \cdot (x_j - x_1). \end{aligned}$$

But this implies that $m^\star \geq m_1$ which contradicts our previous result. □

Note that it can easily be proven that the converse of Theorem 10.13 also holds true. This means that the non-dominated set $\mathcal{Y}_N$ consists of supported non-dominated points only, if and only if the sequence of slopes $\{m_i\}_{i=1}^{n-1}$ is non-decreasing. In this case, $z^i \in \mathcal{Y}_N$ corresponds to a breakpoint of the non-dominated frontier if and only if $m_{i-1} < m_i$ holds true for $i \in \{2, \ldots, n-1\}$.
Corollary 10.9 and Corollary 10.10 in combination with Theorem 10.13 imply one of the main results of this chapter.

Theorem 10.14 *Let a feasible instance of Problem $(BBMP)$ be given. Then the non-dominated set $\mathcal{Y}_N$ of this problem consists of supported non-dominated points only.*

Proof: We use the notation from Section 10.3. Let $\{(e_i, f_i)\}_{i=l}^{u-1}$ denote the swap sequence for optimal $B_i \in \mathcal{S}_i$, $i = l, \ldots, u$. Furthermore, let $j \in \{l, \ldots, u\}$ denote the minimum index such that $c(B_i) < c(B_{i+1})$ holds for the first time. According to Corollary 10.10, we have that $\mathcal{Y}_N = \{(c(B_i), b(B_i)),\ i = j, \ldots, u\}$. If $|\mathcal{Y}_N| \leq 2$, there is nothing more to show. Otherwise, it suffices to show that the sequence of slopes $\{m_i\}_{i=j}^{u-1}$, where

$$m_i = \frac{b(B_{i+1}) - b(B_i)}{c(B_{i+1}) - c(B_i)} = \frac{-1}{c(B_{i+1}) - c(B_i)}$$

is non decreasing, according to Theorem 10.13.
Let $\mathcal{Y}_N \geq 3$, i.e. $j \leq u - 2$. Furthermore, let $i \in \{j, \ldots, u-2\}$ arbitrary, but fixed. According to Theorem 10.7 and Corollary 10.9 we have that

$$c(B_{i+2}) - c(B_{i+1}) = c(e_{i+1}, f_{i+1}) \geq c(e_i, f_i) = c(B_{i+1}) - c(B_i) > 0,$$

But this implies that

$$m_{i+1} = \frac{-1}{c(B_{i+2}) - c(B_{i+1})} \geq \frac{-1}{c(B_{i+1}) - c(B_i)} = m_i.$$

Hence, the sequence of slopes $\{m_i\}_{i=j}^{u-1}$ of the non-dominated frontier of $\mathcal{Y}_N$ is non-decreasing and consequently, $\mathcal{Y}_N$ consists of supported non-dominated points only. □

Note that supported non-dominated solutions may exist that do not correspond to breakpoints of the non-dominated frontier of $\mathcal{Y}_N$. For example, if the value of two

consecutive minimal swaps (e_i, f_i) and (e_{i+1}, f_{i+1}) is the same, the non-dominated solution $(c(B_{i+1}), b(B_{i+1}))$ is supported but does not correspond to a breakpoint of the non-dominated frontier, since the slope of this part of the frontier does not change.
Since all points of $\mathcal{Y}_N$ are supported non-dominated solutions, i.e. all efficient bases correspond to supported efficient solutions, we conclude, based on the fact that the set of supported efficient solutions is always connected (cf. Ehrgott [52]), that the set of efficient bases $\mathcal{X}_E$ must be connected, where the adjacency of two efficient bases is based on the definition used in Subsection 7.2.2 of Chapter 7.

Corollary 10.15 *Let a feasible instance of Problem (BBMP) be given. Then the set of efficient solutions $\mathcal{X}_E$ is connected.*

Proof: Follows immediately from the fact that the restriction of the efficiency graph (cf. Definition 7.1) to the (sub)graph of supported efficient solutions is always connected due to Ehrgott [52], combined with the result of Theorem 10.14. □

10.5 Conclusions and Further Ideas

In this chapter we dealt with biobjective matroid problems involving a binary cost objective. While the general biobjective matroid problem is proven to be $\mathcal{NP}$-complete (cf. Ehrgott [52]), we proved that the special structure of the binary objective allows to solve the problem efficiently. For example, given a graphic matroid on a graph $G = (V, A)$ an upper time bound of $\mathcal{O}(m + n \cdot \log(n))$ can be established, where $|V| = n$ and $|A| = m$.
The presented solution approach for solving biobjective matroid problems involving a binary cost objective is based on an algorithm developed in Gabow and Tarjan [67] to find a minimum cost basis satisfying an equality constraint on the number of elements with binary cost zero. We showed that the main ideas of this algorithm can be used to solve the given biobjective problem, too. Based on swaps between optimal solutions for the equality constrained problem, we can derive a simple decision rule on the costs of the involved swaps that allows to filter dominated solution from the problem.
While the set of efficient solutions for biobjective matroid problems is not connected in general (cf. Chapter 7), we proved that in the special case of BBMPs, the efficient set is always connected, since its non-dominated set consists of supported non-dominated solutions only. To the best of our knowledge this is the first non-trivial problem on matroids where connectedness of $\mathcal{X}_E$ can be established.
This result should be seen as a starting point for a deeper discussion of the connectedness of the efficient set for this special type of problem for the case that not only one but p binary objective functions ($p \geq 2$) are considered. Since the non-dominated set of this $(p+1)$-objective problem is still of polynomial size, the simple idea of exchanging elements contained in different (efficient) bases may also apply to this problem to generate the complete non-dominated set using simple swaps only. Since matroid problems with multiple labels (or colors) can be seen as a special case of this more general multiple objective problem, the articles of Rendl and Leclerc [182] and Brezovec et al. [27], as well as the article of Hamacher and Rendl [95], where elements are related to more than one label, may build a good starting point for further research on this topic.

Part II

Biconvex Optimization

Chapter 11

Biconvex Sets and Optimization with Biconvex Functions

Biconvex optimization problems frequently occur in industrial applications, for example, in the field of multifacility location or medical image registration. We review theoretical results for biconvex sets and biconvex functions and survey existing methods and results for general biconvex optimization problems.
We recall that a set $S \subseteq \mathbb{R}^k$ is said to be convex if for any two points $s_1, s_2 \in S$ the line segment joining s_1 and s_2 is completely contained in S. A function $f : S \to \mathbb{R}$ on a convex set S is called convex, if

$$f(\lambda s_1 + (1-\lambda)s_2) \leq \lambda f(s_1) + (1-\lambda) f(s_2)$$

is valid for all $\lambda \in [0,1]$ and $s_1, s_2 \in S$.
For the definition of biconvex sets and biconvex functions, let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty, convex sets, and let $B \subseteq X \times Y$. We define x- and y-sections of B as follows:

$$\begin{aligned} B_x &:= \{y \in Y : (x,y) \in B\}, \\ B_y &:= \{x \in X : (x,y) \in B\}. \end{aligned}$$

Definition 11.1 *The set $B \subseteq X \times Y$ is called a* biconvex set *on $X \times Y$, or* biconvex *for short, if B_x is convex for every $x \in X$ and B_y is convex for every $y \in Y$.*

The most important results on biconvex sets are summarized in Section 11.1.

Definition 11.2 *A function $f : B \to \mathbb{R}$ on a biconvex set $B \subseteq X \times Y$ is called a* biconvex function *on B or* biconvex *for short, if*

$$f_x(\cdot) := f(x, \cdot) : B_x \to \mathbb{R}$$

is a convex function on B_x for every fixed $x \in X$ and

$$f_y(\cdot) := f(\cdot, y) : B_y \to \mathbb{R}$$

is a convex function on B_y for every fixed $y \in Y$.

From this definition, the definitions of *biconcave*, *bilinear* and *biaffine functions* are obtained by replacing the property for f_x and f_y of being convex by the property of being concave, linear, or affine, respectively. Since for a biconvex function $f : B \to \mathbb{R}$, the function $g := -f$ is biconcave on B, i.e., $g(x, y)$ is concave on B_x in y for fixed $x \in X$ and $g(x, y)$ is concave on B_y in x for fixed $y \in Y$, most of the results and methods mentioned in this paper can directly be transferred to the biconcave case, too.
In the first part of Section 11.2, we survey general properties of biconvex functions, like arithmetical properties or results on the continuity of such functions, which mostly result from the convex substructures of a biconvex function. In the second part, we discuss results on biconvex maximization problems and show that a biconvex function which attains its maximum in the relative interior of a given biconvex set B must be constant throughout B, assuming rather weak topological properties on B. Furthermore, we survey separation theorems for biconvex functions which are mostly applied in probability theory.

Definition 11.3 *An optimization problem of the form*

$$\min \{f(x, y) : (x, y) \in B\} \tag{11.1}$$

is said to be a biconvex optimization problem *or* biconvex *for short, if the feasible set B is biconvex on $X \times Y$, and the objective function f is biconvex on B.*

Different from convex optimization problems, biconvex problems are in general global optimization problems which may have a large number of local minima. However, the question arises whether the convex substructures of a biconvex optimization problem can be utilized more efficiently for the solution of such problems than in the case of general non-convex optimization problems. For this purpose, we discuss existing methods and algorithms, specially designed for biconvex minimization problems which primarily exploit the convex substructures of the problem and give examples for practical applications in Section 11.3.
In this context, we only briefly mention bilinear problems, as there exist plenty of literature and methods, see, e.g., Horst and Tuy [105] for a survey. We rather concentrate on minimization methods and algorithms which can be applied to general constrained as well as unconstrained biconvex minimization problems. In particular, we review the *alternate convex search* method, stated, e.g., in Wendell and Hurter Jr. [215], the *global optimization algorithm*, developed by Floudas and Visweswaran [62] and an algorithm for a special class of jointly constrained biconvex programming problems, given in Al-Khayyal and Falk [5]. Note that the above mentioned methods and algorithms can also be and are applied to bilinear problems in practice, (cf., e.g., Visweswaran and Floudas [210]). We finally conclude in Section 11.4.
We additionally remark that the main results of this chapter are additionally published in Gorski et al. [87].

11.1 Biconvex Sets

The goal of this section is to recall the main definitions and results obtained for biconvex sets. Only a few papers exist in the literature where biconvex sets are investigated.

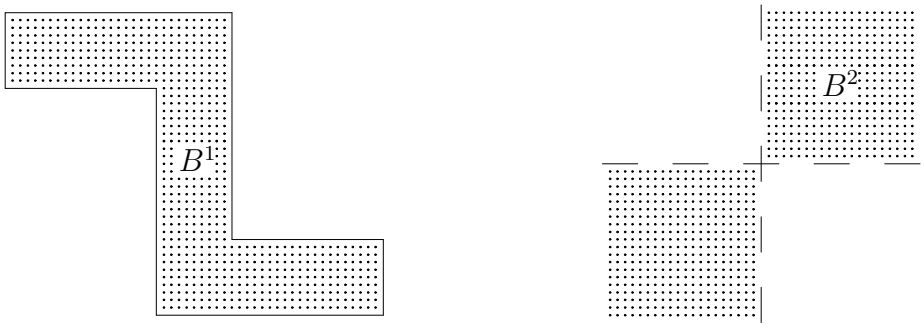


Figure 11.1: *Examples of biconvex sets which are non-convex (B^1) and non-convex and non-connected (B^2), respectively.*

The results presented here can be found in the papers of Aumann and Hart [9] and Goh et al. [82]. In addition we give a short comparison between convex and biconvex sets.

11.1.1 Elementary Properties

In this first subsection we recall elementary properties of biconvex sets. We start with a characterization.

Theorem 11.4 (Aumann and Hart [9]) *A set $B \subseteq X \times Y$ is biconvex if and only if for all quadruples (x_1, y_1), (x_1, y_2), (x_2, y_1), $(x_2, y_2) \in B$ it holds that for every $(\lambda, \mu) \in [0,1] \times [0,1]$*

$$(x_\lambda, y_\mu) := ((1-\lambda)x_1 + \lambda x_2, (1-\mu)y_1 + \mu y_2) \in B.$$

Obviously, a biconvex set is not convex in general. As an example we consider the letters "L" or "T" as a subset of $\mathbb{R} \times \mathbb{R}$, which are biconvex but not convex. Even worse, a biconvex set does not have to be connected in general, as the example

$$B^2 = \{(x,y) \in \mathbb{R}^2 : x, y > 0\} \cup \{(x,y) \in \mathbb{R}^2 : (-x), (-y) > 0\} \tag{11.2}$$

shows (see Figure 11.1). If in contrast B is convex, we derive the following result.

Theorem 11.5 *Let $k > 1$, let $B \subset \mathbb{R}^k$ be a convex set, and let (V_1, V_2) be an arbitrary partition of the variable set $V := \{x_1, \ldots, x_k\}$ into two non-empty subsets. Then B is biconvex on $\operatorname{span}(V_1) \times \operatorname{span}(V_2)$, where $\operatorname{span}(V_i)$ denotes the linear space generated by V_i $(i = 1, 2)$.*

The converse of the last theorem is obviously false. For a counter-example in $\mathbb{R}^2$ consider again the letters "L" or "T". For a more general counter-example in $\mathbb{R}^n$, we generalize the set B given in (11.2).

Example 11.6 Let $k \geq 2$, and let the set $B \subset \mathbb{R}^k$ be given by

$$B = \{z \in \mathbb{R}^k : z_i > 0,\ i = 1, \ldots, k\} \cup \{z \in \mathbb{R}^k : z_i < 0,\ i = 1, \ldots, k\}.$$

Since B is not connected, it cannot be convex. Now let (V_1, V_2) be an arbitrary, but fixed partition of the variable set $V := \{x_1, \ldots, x_k\}$ into two non-empty subsets. The given set B is symmetric in all variables, thus we can rearrange the variables such that we can suppose without loss of generality that the partition of V is given by $V_1 = \{x_1, \ldots, x_\nu\}$ and $V_2 = \{x_{\nu+1}, \ldots, x_k\}$ with $1 \leq \nu \leq k-1$, i.e., $X := \operatorname{span}(V_1) = \mathbb{R}^\nu$ and $Y := \operatorname{span}(V_2) = \mathbb{R}^{k-\nu}$. Now choose $\hat{x} \in X$ arbitrary, but fixed. Then

$$B_{\hat{x}} = \begin{cases} \{y \in Y : y_j > 0,\, j = 1 \ldots, k-\nu\} & : \quad \hat{x}_i > 0,\, i = 1, \ldots, \nu. \\ \emptyset & : \quad \exists\, i, j \in \{1 \ldots, \nu\}, i \neq j : x_i \cdot x_j \leq 0 \\ \{y \in Y : y_j < 0,\, j = 1 \ldots, k-\nu\} & : \quad \hat{x}_i < 0,\, i = 1, \ldots, \nu. \end{cases}$$

Obviously, in all the three cases, $B_{\hat{x}}$ is convex. Similarly, it can be shown that $B_{\hat{y}}$ is convex for every fixed $\hat{y} \in Y$. Hence, B is biconvex for the chosen partitioning of V.

11.1.2 Biconvex Combinations and the Biconvex Hull

In convex analysis, the concept of convex combinations of k given points in $\mathbb{R}^n$ and their convex hull is well known and straight forward (see, e.g., Rockafellar [183]). In Aumann and Hart [9] the concept of biconvex combinations as a special case of a convex combination of k given points is introduced and investigated. We recall the main ideas and results here.

Definition 11.7 *Let $(x_i, y_i) \in X \times Y$ for $i = 1, \ldots, k$. A convex combination*

$$(x, y) = \sum_{i=1}^{k} \lambda_i (x_i, y_i),$$

(with $\sum_{i=1}^{k} \lambda_i = 1$, $\lambda_i \geq 0$ for $i = 1, \ldots, k$) is called biconvex combination *or* biconvex *for short, if $x_1 = \cdots = x_m = x$ or $y_1 = \cdots = y_k = y$ holds.*

With the help of biconvex combinations another characterization for biconvex sets can be formulated:

Theorem 11.8 (Aumann and Hart [9]) *A set $B \subseteq X \times Y$ is biconvex if and only if B contains all biconvex combinations of its elements.*

As in the convex case, it is possible to define the *biconvex hull* of a given set $A \subseteq X \times Y$. To do this, we proceed as in the convex case and denote by H the intersection of all biconvex sets that contain A.

Definition 11.9 *Let $A \subseteq X \times Y$ be a given set. The set*

$$H := \bigcap \{A_I : A \subseteq A_I,\, A_I \text{ is biconvex}\}$$

is called biconvex hull *of A and is denoted by* $\operatorname{biconv}(A)$.

Theorem 11.10 (Aumann and Hart [9]) *The set H as defined in Definition 11.9 is biconvex. Furthermore, H is the smallest biconvex set (in the sense of set inclusion), which contains A.*

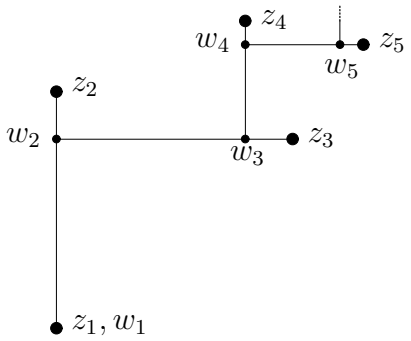


Figure 11.2: *Illustration of Example 11.13.*

As biconvex combinations are, by definition, a special case of convex combinations and the convex hull $\operatorname{conv}(A)$ of a given set A consists of all convex combinations of the elements of A (see, e.g., Rockafellar [183]), we have:

Lemma 11.11 *Let $A \subseteq X \times Y$ be a given set. Then*

$$\operatorname{biconv}(A) \subseteq \operatorname{conv}(A)$$

Aumann and Hart proposed in their paper another way to construct the biconvex hull of a given set A. They defined an inductively given sequence $\{A_n\}_{n\in\mathbb{N}}$ as follows:

$$\begin{aligned} A_1 &:= A \\ A_{n+1} &:= \{(x,y) \in A_n : (x,y) \text{ is a biconvex combination of elements of } A_n\}. \end{aligned}$$

Let $H' := \bigcup_{n=1}^{\infty} A_n$ denote the limit of this sequence.

Theorem 11.12 (Aumann and Hart [9]) *The above constructed set H' is biconvex and equals H, the biconvex hull of A.*

It is important to mention that when applying the above procedure to the convex case (i.e., for the construction of the convex hull of A), one iteration is sufficient as the convex hull consists exactly of all convex combinations of its elements. In general, there does not necessarily exist a finite number of sets A_n such that the union of these sets build the biconvex hull of the given set A. To see this, consider the following example.

Example 11.13 (Aumann and Hart [9]) Let $X = Y = [0,1]$. For $m \in \mathbb{N}$ we define

$$\begin{aligned} z_1 &= (0,0), & w_1 &= (0,0) \\ z_{2m} &= \left(1 - \frac{1}{2^{m-1}}, 1 - \frac{3}{2^{m+2}}\right), & w_{2m} &= \left(1 - \frac{1}{2^{m-1}}, 1 - \frac{1}{2^m}\right), \\ z_{2m+1} &= \left(1 - \frac{3}{2^{m+2}}, 1 - \frac{1}{2^m}\right), & w_{2m+1} &= \left(1 - \frac{1}{2^m}, 1 - \frac{1}{2^m}\right). \end{aligned}$$

For $n \geq 2$, w_n is a biconvex combination of the points z_n and w_{n-1}, namely

$$w_n = \frac{4}{5}\, z_n + \frac{1}{5}\, w_{n-1}.$$

Now, let the set A be given by $\{z_n\}_{n\in\mathbb{N}}$. Then it is easy to see that $w_n \in A_n$, but $w_n \notin A_{n-1}$ for every $n \geq 2$ (see also Figure 11.2).
By adding the point $(1, 1)$ to the set A, we obtain a closed and bounded set A with $A_n \subsetneq \operatorname{biconv}(A)$ for all $n \in \mathbb{N}$.

11.2 Biconvex Functions

In this section we present important properties of biconvex functions. As these types of functions regularly appear in practice, biconvex functions and optimization problems are widely discussed in the literature. Since we are interested in optimization with biconvex functions f on subsets of $\mathbb{R}^{n+m}$ here, we focus on properties which are related to these optimization problems.
Note that biconvex functions are of importance in other mathematical contexts, too. For example, biconvex functions can be used to derive results on robust stability of control systems in practical control engineering. For further details see Geng and Huang [75] and Geng and Huang [76]. Furthermore, biconvex functions play an important role in martingale theory and can be used to characterize whether a Banach space B is UMD (i.e., the space B has the unconditionality property for martingale differences), or whether B is a Hilbert space or not. Here, we refer to Burkholder [30], Aumann and Hart [9], Burkholder [31] and Lee [123]. Finally, Thibault [202], Jouak and Thibault [112] and Borwein [24] published results concerning the continuity and differentiability of (measurable) biconvex operators in topological vector spaces.
This section is organized as follows: The first subsection briefly reviews elementary properties of biconvex functions. We extend these properties by a comparison to convex functions. The next subsection summarizes results concerning continuity of biconvex functions given in Aumann and Hart [9]. The last subsection deals with the maximization of biconvex functions. Several known and some new results are presented.

11.2.1 Elementary Properties

We start our summary with the most important elementary properties of biconvex functions. Note that as mentioned at the beginning of this chapter, it is possible to transform a biconvex function to a biconcave one, and vice versa, by multiplying the given function by (-1). Similar to convex functions, biconvex functions can be characterized by an interpolation property:

Theorem 11.14 (Goh et al. [82]) *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty, convex sets, and let f be a real-valued function on $X \times Y$. f is biconvex if and only if for all quadruples (x_1, y_1), (x_1, y_2), (x_2, y_1), $(x_2, y_2) \in X \times Y$ it holds, that for every $(\lambda, \mu) \in [0, 1] \times [0, 1]$*

$$\begin{aligned} f(x_\lambda, y_\mu) \;\leq\; & (1-\lambda)\,(1-\mu)\, f(x_1, y_1) + (1-\lambda)\,\mu\, f(x_1, y_2) + \\ & + \lambda\,(1-\mu)\, f(x_2, y_1) + \lambda\,\mu\, f(x_2, y_2), \end{aligned}$$

where $(x_\lambda, y_\mu) := ((1-\lambda)x_1 + \lambda x_2, (1-\mu)y_1 + \mu y_2)$.

So, as one-dimensional interpolation always overestimates a convex function, two-dimensional interpolation always overestimates a biconvex function. Moreover, as convex functions have convex level sets, we state for the biconvex case:

Theorem 11.15 (Goh et al. [82]) *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty, convex sets, and let f be a real-valued function on $X \times Y$. If f is biconvex on $X \times Y$, then its level sets*

$$\mathcal{L}_c := \{(x, y) \in X \times Y : f(x, y) \leq c\}$$

are biconvex for every $c \in \mathbb{R}$.

Like in the convex case, the converse of the last theorem is not true in general:

Example 11.16 Let the function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $f(x, y) = x^3 + y^3$ be given, and let $c \in \mathbb{R}$. Then,

$$\begin{aligned}(\mathcal{L}_c)_{\bar{x}} &= \{y \in Y : y^3 \leq c - \bar{x}^3\} =]-\infty, \operatorname{sign}(c - \bar{x}^3) \sqrt[3]{|c - \bar{x}^3|}] \\ (\mathcal{L}_c)_{\bar{y}} &= \{x \in X : x^3 \leq c - \bar{y}^3\} =]-\infty, \operatorname{sign}(c - \bar{y}^3) \sqrt[3]{|c - \bar{y}^3|}]\end{aligned}$$

are convex sets and hence, the level set $\mathcal{L}_c$ of f is biconvex. But obviously f is not biconvex on $\mathbb{R} \times \mathbb{R}$, since $f_0(x) = f(x, 0) = x^3$ is not a convex function on $\mathbb{R}$.

Also many arithmetic properties that are valid for convex functions can be transferred to the biconvex case.

Lemma 11.17 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty, convex sets, let $\mu \in \mathbb{R}^+$ be a non-negative scalar, and let $f, g : X \times Y \to \mathbb{R}$ be two biconvex functions. Then the functions $h, t : X \times Y \to \mathbb{R}$ with $h(x, y) := f(x, y) + g(x, y)$ and $t(x, y) := \mu f(x, y)$ are biconvex, too.*

Proof: The biconvexity of h and t follows immediately from Definition 11.2 for biconvex functions and the fact that the above stated lemma is valid for convex functions, hence for f_x and g_x (f_y and g_y, respectively), too, as they are convex for every fixed $x \in X$ ($y \in Y$) by definition. □

For the composition of convex and biconvex functions we have:

Lemma 11.18 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty, convex sets, let $f : X \times Y \to \mathbb{R}$ be a biconvex function, and let $\varphi : \mathbb{R} \to \mathbb{R}$ be a convex, non-decreasing function. Then $h(x, y) := \varphi(f(x, y))$ is biconvex on $X \times Y$.*

Proof: For fixed $x \in X$ and fixed $y \in Y$ we consider the functions $h_x(y) := \varphi(f_x(y))$ and $h_y(x) := \varphi(f_y(x))$, respectively. Since Lemma 11.18 is valid for f convex (cf. Rockafellar [183]), f_x and f_y are both convex functions by definition and φ is convex and non-decreasing, h_x and h_y are convex, too. Hence, h is a biconvex function on $X \times Y$. □

Finally, we state a lemma concerning the pointwise supremum of biconvex functions.

Lemma 11.19 *The pointwise supremum of an arbitrary collection of biconvex functions is biconvex.*

Proof: Let I be an arbitrary index set, let $f^i : X \times Y \to \mathbb{R}$ be biconvex for all $i \in I$, and let $f(x,y) := \sup\{f^i(x,y),\, i \in I\}$ be the pointwise supremum of these functions. For fixed $\bar{y} \in Y$ and arbitrary $x \in X$ we have:

$$f_{\bar{y}}(x) \;=\; f(x,\bar{y}) \;=\; \sup_{i\in I}\{f^i(x,\bar{y})\} \;=\; \sup_{i\in I}\{f^i_{\bar{y}}(x)\}.$$

Since the functions $f^i_{\bar{y}}$ are convex for all $i \in I$ by assumption, $f_{\bar{y}}$, as pointwise supremum of convex functions, is convex by Rockafellar [183], too. Similarly it can be shown that $f_{\bar{x}}$ is convex on Y for every fixed $\bar{x} \in X$. Hence, f is biconvex. □

We close this subsection by a comparison between convex and biconvex functions. Obviously:

Theorem 11.20 *Let $f : \mathbb{R}^k \to \mathbb{R}$ ($k > 1$) be a convex function, and let (V_1, V_2) be an arbitrary partition of the variable set $V := \{x_1, \ldots, x_k\}$ into two non-empty subsets. Then f is biconvex on* $\mathrm{span}(V_1) \times \mathrm{span}(V_2)$.

As in the case of biconvex sets, if a given function f is biconvex for every arbitrary partition of the variable set, it has not to be convex in general. To see this, consider the following example:

Example 11.21 Let $n \geq 2$, let $b := \sqrt{\frac{2n-1}{2n(n-1)}}$, and let $f : \mathbb{R}^{n+1} \to \mathbb{R}$ be defined by

$$f(x_1, \ldots, x_{n+1}) \;=\; \frac{1}{2}(x_1^2 + \cdots + x_n^2 + x_{n+1}^2) + b \cdot x_{n+1} \cdot (x_1 + \cdots + x_n).$$

The partial derivatives of f are given by

$$\frac{\partial f}{\partial x_i}(x_1, \ldots, x_n) \;=\; \begin{cases} x_i + b \cdot x_{n+1} & , \text{if} \quad i \neq n+1 \\ x_{n+1} + b \cdot (x_1 + \cdots + x_n) & , \text{if} \quad i = n+1 \end{cases}$$

and the Hessian matrix of f is

$$H(x) \;:=\; \mathrm{Hess}(f)(x) \;=\; \begin{pmatrix} 1 & 0 & \ldots & 0 & b \\ 0 & 1 & \ldots & 0 & b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & b \\ b & b & \ldots & b & 1 \end{pmatrix} \in M((n+1)\times(n+1), \mathbb{R}).$$

First, we show that f is biconvex for any partition of the variable set $\{x_1, \ldots, x_{n+1}\}$ into two non-empty disjoint subsets. So let (V_1, V_2) be such a partition and let $Y^i = \mathrm{span}(V_i)$, $i = 1, 2$. We assume that $x_{n+1} \in V_2$. Let I^i denote the index set of the

variables of V_i, and let $c_i := |I^i|$ be the cardinality of I^i $(i = 1, 2)$. Then, the Hessian matrix of f_{Y^i} is given by

$$\text{Hess}(f_{Y^2})(y^1) = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \text{ and } \text{Hess}(f_{Y^1})(y^2) = \begin{pmatrix} 1 & 0 & \dots & 0 & b \\ 0 & 1 & \dots & 0 & b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & b \\ b & b & \dots & b & 1 \end{pmatrix},$$

where

$$y^1 = (x_{i_1^1}, \dots, x_{i_{c_1}^1}) \in Y^1, \qquad i_j^1 \in I^1 \; \forall\, j = 1, \dots, c_1,$$
$$y^2 = (x_{i_1^2}, \dots, x_{i_{c_2-1}^2}, x_{n+1}) \in Y^2, \; i_k^2 \in I^2 \; \forall\, k = 1, \dots, c_2 - 1,$$

and $\text{Hess}(f_{Y^1}) \in M(c_2 \times c_2, \mathbb{R})$ and $\text{Hess}(f_{Y^2}) \in M(c_1 \times c_1, \mathbb{R})$. Obviously, $\text{Hess}(f_{Y^2})$ is positive definite for all $y^1 \in Y^1 = \mathbb{R}^{c_1}$ and hence, f_{Y^2} is convex (cf. Floudas [64]). To show the convexity of f_{Y^1}, we calculate the eigenvalues of $\text{Hess}(f_{Y^1})$. They are given by $\lambda_1 = 1$ and $\lambda_{2,3} = 1 \pm b\sqrt{c_2 - 1}$. Since $c_2 \leq n$, it holds that $\lambda_2 > 0$ and

$$\lambda_3 \;=\; 1 - b\,\sqrt{c_2 - 1} \;\geq\; 1 - b\,\sqrt{n-1} \;=\; 1 - \sqrt{1 - \frac{1}{2n}} \;>\; 0.$$

So, all eigenvalues of $\text{Hess}(f_{Y^1})$ are positive, i.e., $\text{Hess}(f_{Y^1})$ is positive definite, and hence, f_{Y^1} is convex, too.
Finally, we calculate the eigenvalues of $H(x)$. They are given by $\lambda_1 = 1$ and $\lambda_{2,3} = 1 \pm b\sqrt{n}$. Since

$$\lambda_3 \;=\; 1 - b\,\sqrt{n} \;=\; 1 - \sqrt{1 + \frac{1}{2\,(n-1)}} \;<\; 0,$$

$H(x)$ has a negative eigenvalue. Hence, $H(x)$ is indefinite for all $x \in \mathbb{R}^{n+1}$ and f is not convex on every open, convex set $X \subseteq \mathbb{R}^{n+1}$.
Note that for a counter-example for a function from $\mathbb{R}^2$ to $\mathbb{R}$, one can use the above given function with $b := 2$.

11.2.2 Continuity of Biconvex Functions

One of the central results in convex analysis is the fact that a finite, real-valued, convex function f is continuous throughout the interior of its domain $C \subseteq \mathbb{R}^n$ (cf. Rockafellar [183]). Aumann and Hart [9] transferred this result to the biconvex case.

Definition 11.22 *Let $B \subseteq X \times Y$ and let $z = (x, y) \in B$. The point z is called a* bi-relatively interior point of B, *if z is in the interior of B relative to* $\text{aff}(\text{proj}_X(B)) \times \text{aff}(\text{proj}_Y(B))$, *where* $\text{proj}_X(B)$ *and* $\text{proj}_Y(B)$ *denote the projection of B into the X- and Y-space, respectively, and* $\text{aff}(C)$ *is the affine space, generated by the set C.*

From Rockafellar [183] we recall that an *m-dimensional simplex* is the convex hull of m affinely independent vectors $b_1 \dots, b_m \in \mathbb{R}^n$. A set $S \subseteq \mathbb{R}^n$ is called *locally simplicial*,

if for each $x \in S$ there exists a finite collection of simplices $S_1, \ldots, S_m$ such that, for some neighborhood U of x,

$$U \cap (S_1 \cup \cdots \cup S_m) = U \cap S.$$

Examples of locally simplicial sets are line segments, polyhedral convex sets, or relatively open, convex sets. Note that a locally simplicial set does not need to be convex or closed in general.

Definition 11.23 *Let $B \subseteq X \times Y$ and let $z = (x, y) \in B$. We say that B is* locally bi-simplicial at z*, if there exists a neighborhood U of x in X and a neighborhood V of y in Y, a collection of simplices $S_1, \ldots, S_k$ in X and a collection of simplices $T_1, \ldots, T_l$ such that for $S := \bigcup_{i=1}^{k} S_i$ and $T := \bigcup_{i=1}^{l} T_i$, $S \times T \subseteq B$ and $(U \times V) \cap B = (U \times V) \cap (S \times T)$.*

It holds:

Theorem 11.24 (Aumann and Hart [9]) *Let f be a biconvex function on a biconvex set B and let $z \in B$.*

1. *If z is a bi-relatively interior point of B, then f is lower-semi-continuous at z.*

2. *If B is locally bi-simplicial at z, then f is upper-semi-continuous at z.*

Since for all bi-relatively interior points z of B, B is locally bi-simplicial at z as well, it holds:

Corollary 11.25 *Let f be a biconvex function on a biconvex set B. Then f is continuous at all bi-relatively interior points $z \in B$.*

Note that only "directional continuity" (i.e., $f(x, \cdot) : Y \to \mathbb{R}$ and $f(\cdot, y) : x \to \mathbb{R}$ are continuous for all $x \in X$ and $y \in Y$) is not sufficient for a function f to be continuous on an open set $B \subseteq X \times Y$. A counter-example to this is given, for example, in the book of Gelbaum and Olmsted [74].

11.2.3 The Maximum of a Biconvex Function

This subsection deals with the problem of finding the maximum of a biconvex function over a given set contained in $X \times Y$. We recall known results for this problem and present a new result for the case that the maximum of a biconvex function is attained in the interior of a biconvex set B when B has some additional topological properties. In the convex case, it is well-known that the set of all points where the supremum of a convex function relative to a given convex set C is attained, is given by a union of faces of C (cf. Rockafellar [183]), i.e., that the supremum of a convex function over a convex set C is attained at a boundary point of C if it exists. Al-Khayyal and Falk [5] showed that this result is also valid for a continuous, biconvex function f over a compact and convex set $K \subseteq X \times Y$. (Actually, the result was proven for the minimum of a biconcave function, which is equivalent.)

Theorem 11.26 (Al-Khayyal and Falk [5]) *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty sets, let $K \subseteq X \times Y$ be compact and convex, and let $f : K \to \mathbb{R}$ be a continuous, biconvex function. Then the problem*

$$\max \{f(x, y) : (x, y) \in K\} \tag{11.3}$$

has always a solution on ∂K, the boundary of K.

If the given set K is a product of two polytopes in X and Y, respectively, Geng and Huang [76] stated:

Theorem 11.27 (Geng and Huang [76]) *Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be biconvex and let $S \subset \mathbb{R}^n$, $T \subset \mathbb{R}^m$ be polytopes with vertex sets S^* and T^*, respectively. Then*

$$\max_{(x,y)\in S\times T} f(x, y) = \max_{(x,y)\in S^*\times T^*} f(x, y). \tag{11.4}$$

Note that in Geng and Huang [75] and Geng and Huang [76] the authors referred to a proof of the above theorem given in Barmish [11]. Another proof and an outer approximation algorithm for Problem (11.4), based on the above theorem, can be found in Gao and Xu [70].
Horst and Thoai [104] presented a decomposition approach for the minimization of a biconcave function over a polytope $P \subset \mathbb{R}^{n+m}$ where P is not separable in the sense that it cannot be written as a product of two polytopes in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. The authors used a combination of global optimization techniques such as branch and bound, polyhedral outer approximation and projection of polyhedral sets onto a subspace to design an algorithm for problems of the form

$$\min \{f(x, y) : x \in X, y \in Y, (x, y) \in D\}, \tag{11.5}$$

where $X \subset \mathbb{R}^n$ and $Y \subset \mathbb{R}^m$ are polytopes, $D \subset \mathbb{R}^{n+m}$ is a polyhedral set and f is biconcave. As special cases of Problem (11.5), jointly constrained bilinear programming problems and separated jointly constrained biconcave programming problems of the form $f(x, y) = f_1(x) + f_2(y)$ are considered, amongst others.
Next we consider problems where the maximum of a biconvex function over a biconvex set B lies in the relative interior $\mathrm{ri}(B)$ of the set B. For the convex case we recall:

Theorem 11.28 (Rockafellar [183]) *Let f be a convex function and let C be a convex set. If f attains its supremum relative to C at some point of the relative interior of C, then f is constant throughout C.*

Our aim is to prove that this result is also valid for the biconvex case if we make some more topological assumptions on the given biconvex set B. In order to derive a proof for this result we need some preliminary lemmas and definitions.

Definition 11.29 *Let $I = [a, b] \subseteq \mathbb{R}$ be an interval and let $\gamma : I \to M$ be a continuous function. Then γ is called a* path *with* initial point $\gamma(a)$ *and* terminal point $\gamma(b)$.

Definition 11.30 *Let $M \subseteq \mathbb{R}^n$ be a non-empty set. M is called* path-connected *if for any two points m_1, $m_2 \in M$ there exists a path $\gamma : [a, b] \to M$ with $\gamma(a) = m_1$, $\gamma(b) = m_2$ and $\gamma(t) \in M$ for all $t \in [a, b]$.*

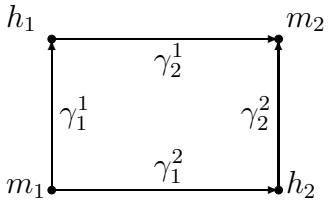


Figure 11.3: *Example of two* L*-shaped paths γ^1 and γ^2 joining m_1 and m_2 with inflection points h_1 and h_2, respectively.*

Definition 11.31 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty sets, let $M \subseteq X \times Y$, let $m_1 := (x_1, y_1) \in M$ and $m_2 := (x_2, y_2) \in M$, and let γ be a path in M joining m_1 and m_2. We call γ* L-shaped *if we can partition γ into two subpaths γ_1 and γ_2 such that γ restricted to γ_1 consists of the line segment joining m_1 and the point $h_1 := (x_1, y_2)$ (or $h_2 := (x_2, y_1)$) and γ restricted to γ_2 consists of the line segment joining h_1 (or h_2)and m_2. The intermediate point h_1 (or h_2) is called* inflection point *of γ. An* L*-shaped path is said to be* degenerate *if $x_1 = x_2$ or $y_1 = y_2$.*

If $X, Y \subseteq \mathbb{R}$, an L-shaped path is a path of the form "L" or "¬" (cf. Figure 11.3). Furthermore, we define:

Definition 11.32 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty sets, let $M \subseteq X \times Y$, and let $m_1, m_2 \in M$. If there exists a (finite) sequence of* L*-shaped paths joining m_1 and m_2 which is completely contained in M, we say that m_1 and m_2 are* (finitely) L-connectable *or* (finitely) L-connected *in M. The set M is* (finitely) L-connected *if any two points in M are (finitely)* L*-connectable.*

Due to the last definition it is obvious that every finitely L-connected set is path-connected, whereas the converse is not true in general. If we consider, for example, the line segment

$$M := \{(x, y) \in [0; 1] \times [0; 1] : (x, y) = \lambda\,(1, 0) + (1 - \lambda)\,(0, 1),\ \lambda \in [0; 1]\}$$

in $[0; 1] \times [0; 1]$, then M is path-connected, but any two points of M are not finitely L-connectable in M. Now, for $\varepsilon > 0$ and $x \in \mathbb{R}^n$, let

$$K_\varepsilon(x) := \{y \in \mathbb{R}^n : \|x - y\| < \varepsilon\}$$

denote the open ball around x with radius ε. Then the following lemma can easily be proven by induction:

Lemma 11.33 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty sets and let $I := \{1 \ldots, k\}$ be an index set. Furthermore, let k points $m_i \in X \times Y$ be given and k positive numbers ε_i such that $K_{\varepsilon_i}(m_i) \subseteq X \times Y$ for all $i \in I$ and the intersection $K_{\varepsilon_i}(m_i) \cap K_{\varepsilon_{i+1}}(m_{i+1})$ is not empty for all $i = 1, \ldots, k-1$. Then m_1 and m_k are finitely* L*-connectable in $\bigcup_{i \in I} K_{\varepsilon_i}(m_i)$ such that the resulting path contains the points m_i, $i \in I$.*

The proof of this lemma is obvious and can be performed as indicated in Figure 11.4.

Theorem 11.34 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty sets and let $M \subseteq X \times Y$ be a non-empty, open, and path-connected set. Then M is finitely* L*-connected.*

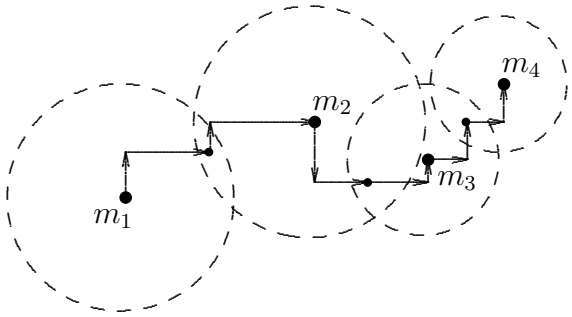


Figure 11.4: *Example of a sequence of* L*-shaped paths in Lemma 11.33.*

Proof: Let $m_1, m_2 \in M \subseteq X \times Y$ be two arbitrary chosen points in M. Since by assumption M is path-connected, there exist $a, b \in \mathbb{R}$ and a path $\gamma : [a, b] \to M$ with $\gamma(a) = m_1$, $\gamma(b) = m_2$, and $\gamma(t) \in M$ for every $t \in I := [a, b]$.
Since M is an open set, for every point $\gamma(t) \in M$ on the curve there exists $\varepsilon_t > 0$ such that $K_{\varepsilon_t}(\gamma(t))$ is completely contained in M. Hence, $\{\bigcup_{t\in I} K_{\varepsilon_t}(\gamma(t))\}$ builds an open covering of the image set $\gamma(I)$ of γ in M. Since $\gamma(I)$ is known to be compact, there exists $t_1, \ldots, t_n \in I$, such that $\gamma(I)$ is already covered by $\{\bigcup_{t=1}^{n} K_{\varepsilon_{t_i}}(\gamma(t_i))\}$.
Without loss of generality we suppose that $t_1 = a$ and $t_n = b$, otherwise we add the two balls $K_{\varepsilon_a}(\gamma(a))$ and $K_{\varepsilon_b}(\gamma(b))$ to the finite open covering of $\gamma(I)$. By eventually deleting and rearranging the order of the open balls $K_{\varepsilon_{t_i}}(\gamma(t_i))$ we can reorder the given finite covering in the way that the intersection of two consecutive open balls $K_{\varepsilon_{t_i}}(\gamma(t_i))$ and $K_{\varepsilon_{t_{i+1}}}(\gamma(t_{i+1}))$ is non-empty.
Let the resulting covering be denoted again by $\{\bigcup_{t=1}^{n} K_{\varepsilon_{t_i}}(\gamma(t_i))\}$. Then this covering satisfies all the assumptions made in Lemma 11.33. Hence, $\gamma(t_1) = m_1$ and $\gamma(t_n) = m_2$ are finitely L-connectable, which completes the proof. □

Now we can prove our main result:

Theorem 11.35 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty sets and let $B \subseteq X \times Y$ be a biconvex set such that the interior of B is non-empty and path-connected with $\partial(\text{int}(B)) = \partial B$. Furthermore, let $f : B \to \mathbb{R}$ be a continuous, biconvex function. If the problem*

$$\max\{f(x, y) : (x, y) \in B\} \tag{11.6}$$

has an optimal solution $z^* := (x^*, y^*) \in \text{int}(B)$, then f is constant throughout B.

Proof: We prove the theorem in two steps. First, we concentrate on points $z \in B$ lying in the interior of B and we show that $f(z^*) = f(z)$ holds for all points $z \in \text{int}(B)$. In the second step we extend our results to points situated in $B \setminus \text{int}(B)$.
So, let $z^* = (x^*, y^*) \in \text{int}(B)$ denote the optimal solution of problem (11.6). First, consider the two functions

$$f_{x^*} : B_{x^*} \to \mathbb{R} \text{ and } f_{y^*} : B_{y^*} \to \mathbb{R},$$

where $B_{x^*} := \{y \in Y : (x^*, y) \in B\}$ and $B_{y^*} := \{x \in X : (x, y^*) \in B\}$, respectively. Since B is biconvex by assumption, the sets B_{x^*} and B_{y^*} are convex. Obviously,

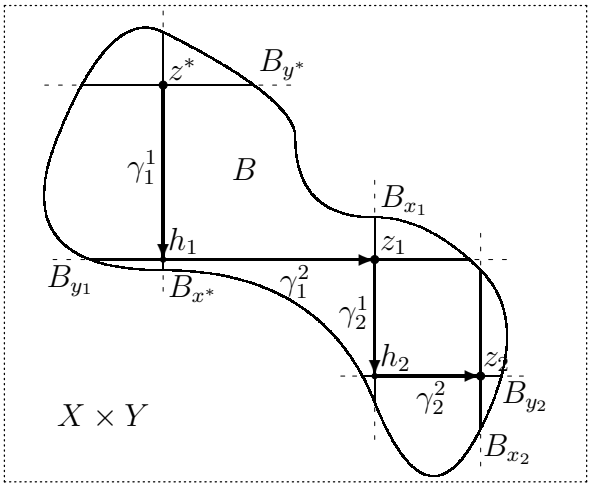


Figure 11.5: *Illustration of the proof of Theorem 11.35 with two intermediate points z_1 and z_2.*

$y^* \in B_{x^*}$ and $x^* \in B_{y^*}$ hold. But since (x^*, y^*) is a point in $\operatorname{int}(B)$, y^* and x^* are elements of $\operatorname{ri}(B_{x^*})$ and $\operatorname{ri}(B_{y^*})$, respectively. Hence, by Theorem 11.28, f_{x^*} and f_{y^*} are constant on B_{x^*} and B_{y^*}, respectively. So we have that

$$f(z) = f(z^*) \quad \forall\, z \in B_{z^*} := \{(x^*, y) : y \in B_{x^*}\} \cup \{(x, y^*) : x \in B_{y^*}\}.$$

Next, consider a point $z_1 = (x_1, y_1) \in \operatorname{int}(B)$ which is L-connectable to z^* throughout $\operatorname{int}(B)$ by exactly one L-shaped path γ, and let $h_1 := (x^*, y_1) \in \operatorname{int}(B)$ denote the inflection point of γ. Since $h_1 \in B_{z^*}$, $f(h_1) = f(z^*)$. Since B is biconvex, the set $B_{y_1} := \{x \in X : (x, y_1) \in B\}$ is convex. Since $h_1 \in \operatorname{int}(B)$, $x^* \in \operatorname{ri}(B_{y_1})$. Hence, $f_{y_1}(x^*) \geq f_{y_1}(x)$ holds for all $x \in B_{y_1}$, and f_{y_1} is constant on B_{y_1} by Theorem 11.28. Since $x_1 \in B_{y_1}$, $f(z_1) = f(z^*)$. So we have proven that $f(z) = f(z^*)$ for all z which are L-connectable to z^* by exactly one L-shaped path.
Finally, let $z = (x, y) \in \operatorname{int}(B)$ be arbitrarily chosen in $\operatorname{int}(B)$. Since $\operatorname{int}(B)$ is open by definition and non-empty and path-connected by assumption, z^* and z are finitely L-connectable in $\operatorname{int}(B)$ by k L-shaped paths γ_k by Theorem 11.34 (see Figure 11.5). Now, let $m_0 := z^*$, $m_k = z$, and let $m_i := (x_i, y_i) \in \operatorname{int}(B)$ and $h_i := (x_{i-1}, y_i) \in \operatorname{int}(B)$ $(i = 1, \ldots, k)$ denote the finite sequence of initial points and inflection points, respectively, obtained by the sequence of L-shaped paths from z^* to z.
Since $B_{x_i} := \{y \in Y : (x_i, y) \in B\}$ and $B_{y_i} := \{x \in X : (x, y_i) \in B\}$ are convex sets by assumption and $y_{i-1}, y_i \in \operatorname{ri}(B_{x_{i-1}})$ and $x_{i-1}, x_i \in \operatorname{ri}(B_{y_i})$ for $i = 1, \ldots, k$, respectively, we have, following the same argumentation as above, that f is subsequently constant on the L-shaped path γ_i joining m_{i-1} and m_i with inflection point h_i for $i = 1, \ldots, k$, i.e.,

$$f(z^*) = f(m_0) = f(m_1) = \ldots = f(m_{k-1}) = f(m_k) = f(z).$$

Hence, f is constant throughout $\operatorname{int}(B)$. This completes the first step of the proof.
Now suppose that the point $z \in B$ is an element of $B \setminus \operatorname{int}(B)$, i.e., $z \in \partial B \cap B$. Since, by assumption, $\partial(\operatorname{int}(B)) = \partial B$, $z \in \partial \operatorname{int}(B)$, i.e., there exists a sequence $\{z_n\}_{n \in \mathbb{N}}$ converging to z such that $z_n \in \operatorname{int}(B)$ for all $n \in \mathbb{N}$. Since f is continuous on B and

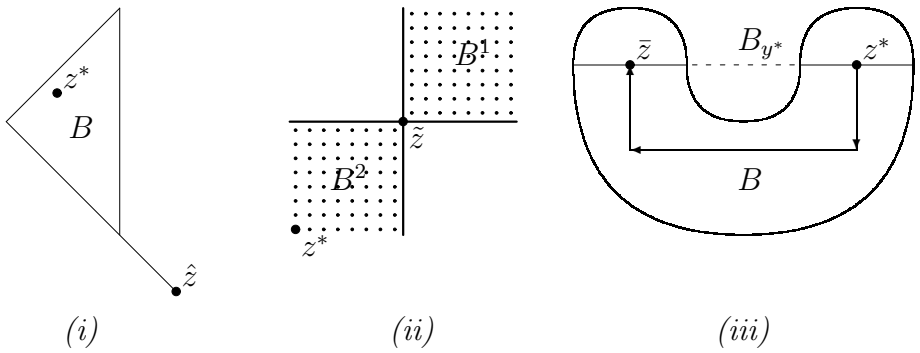


Figure 11.6: *Discussion of the assumptions made in Theorem 11.35.*

equal to the constant $f(z^*)$ on $\text{int}(B)$, we get that

$$f(z) = f\left(\lim_{n\to\infty} z_n\right) = \lim_{n\to\infty} f(z_n) = \lim_{n\to\infty} f(z^*) = f(z^*).$$

Hence, f is constant throughout B. □

Before we conclude this subsection by reflecting on the assumptions made in Theorem 11.35, we remark that for a set $A \subseteq \mathbb{R}^n$ it holds that

$$\partial(\text{int}(A)) = \partial A \quad \Longleftrightarrow \quad \text{cl}(\text{int}(A)) = \text{cl}(A).$$

Hence, the assumption $\partial(\text{int}(A)) = \partial A$ stated in the last theorem could be alternatively replaced by $\text{cl}(\text{int}(A)) = \text{cl}(A)$. Note that in set-theoretical topology a set which equals the closure of its interior is called a regular (closed) set. As immediate corollary of the last theorem, we additionally state:

Corollary 11.36 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty sets and let $B \subseteq X \times Y$ be a biconvex, regular closed set such that the interior of B is non-empty and path-connected. Furthermore, let $f : B \to \mathbb{R}$ be a continuous, biconvex function. Then, f attains its maximum in a boundary point of B.*

Now consider Figure 11.6. In Figure 11.6(i) a set B is shown where the assumption $\partial(\text{int}(B)) = \partial B$ is not valid, since the point $\hat{z}$ is an element of $\partial B \setminus \partial(\text{int}(B))$, and $f(\hat{z})$ can be chosen arbitrarily without effecting the biconvexity of f. In this case, a biconvex function having a global maximum in $z^* \in \text{int}(B)$ does not need to be constant throughout B since the point $\hat{z}$ is not L-connectable to z^* within B.
Figure 11.6(ii) shows the biconvex set $B := \mathbb{R}^2_+ \cup (-\mathbb{R}^2_+)$ where the interior of B is not path-connected any more. If a biconvex function f takes its maximum in $z^* \in \text{int}(B^2)$ ($B^2 := -\mathbb{R}^2_+$), then f has to be constant on $B^2 \cup \partial B^1$, where $B^1 := \mathbb{R}^2_+$, but not necessarily on $\text{int}(B^1)$. For a counter-example consider the function f given on B as follows:

$$f(z) = \begin{cases} 1 & , \text{ if } z \in B^2 \cup \partial B^1, \\ 1 - \min\{x, y\}, & \text{if } x \in [0,1] \text{ or } y \in [0,1], \\ 0 & , \text{ if } x > 1 \text{ and } y > 1. \end{cases}$$

Obviously, f is continuous and biconvex on B but not constant throughout B, although B (but not $\text{int}(B)$) is path-connected, using $\tilde{z}$ passing from B^1 to B^2 and the other way round.
Figure 11.6(*iii*) shows a set B which is not biconvex since the y^*-cut B_{y^*} is not convex. Hence, Theorem 11.35 is not applicable directly. Nevertheless, a biconvex function f taking its global maximum in $z^* \in \text{int}(B)$ is constant throughout the given set since every point $\bar{z} \in \text{int}(B)$ is still L-connectable to z^*. Hence, the line of argumentation of Theorem 11.35 is still valid, provided that the given set B can be partitioned into appropriate biconvex subsets such that Theorem 11.35 is applicable in these subsets. So, the biconvexity-assumption for the set B might be weakened.

11.2.4 Biconvexity and Separation

Aumann and Hart [9] stated several separation theorems for biconvex functions. In this context, separation does not mean that we separate two biconvex sets by a biconvex or bilinear function, but we determine the set of all points $z \in B$, B biconvex that cannot be separated from a subset $A \subset B$ of B by a biconvex function f. We give the main results and ideas here. For further details, we refer to the original article. The results for the convex case, which we state next, can also be found there.

Definition 11.37 *Let $C \subseteq \mathbb{R}^n$ be a convex set and $A \subseteq C$. Then a point $z \in C$ is* convex separated from A with respect to C *if there exists a bounded convex function f on C such that $f(z) > \sup f(A) := \sup\{f(a) : a \in A\}$. Furthermore, let* $\text{ncs}(C)$ $(= \text{ncs}_A(C))$ *denote the set of all points $z \in C$ that cannot be convex separated from A.*

For the set $\text{ncs}_A(C)$ we have:

Theorem 11.38 (Aumann and Hart [9]) *Let $C \subseteq \mathbb{R}^n$ be a convex set and let $A \subseteq C$, then* $\text{ncs}_A(C)$ *is a convex set and*

$$\text{conv}(A) \subseteq \text{ncs}_A(C) \subseteq \overline{\text{conv}(A)},$$

where $\overline{\text{conv}(A)}$ *denotes the closure of the convex hull of A.*

For biconvex sets this is as follows:

Definition 11.39 *Let $B \subseteq \mathbb{R}^n \times \mathbb{R}^m$ be a biconvex set and $A \subseteq B$. Then a point $z \in B$ is* biconvex separated from A with respect to B *if there exists a bounded biconvex function f on B such that $f(z) > \sup f(A) := \sup\{f(a) : a \in A\}$. Furthermore, let* $\text{nbs}(B)$ $(= \text{nbs}_A(B))$ *denote the set of all points $z \in B$ that cannot be biconvex separated from A.*

Obviously we have:

Lemma 11.40 (Aumann and Hart [9]) *Let $B \subseteq \mathbb{R}^n \times \mathbb{R}^m$ be a biconvex set and let $A \subseteq B$. Then $z \in \text{nbs}_A(B)$ if and only if $z \in B$ and, for all biconvex functions f defined on B, we have $f(z) \leq \sup f(A)$.*

As level sets of biconvex functions are biconvex by Theorem 11.15, for the set $\text{nbs}_A(B)$ we have:

Theorem 11.41 (Aumann and Hart [9]) *Let B be a biconvex set and let $A \subseteq B$. Then the set* $\mathrm{nbs}_A(B)$ *is biconvex and*

$$\mathrm{biconv}(A) \subseteq \mathrm{nbs}_A(B).$$

Different to the convex case, for biconvex separation we have $\mathrm{nbs}_A(B) \not\subset \overline{\mathrm{biconv}(A)}$ in general. For an example we refer to Aumann and Hart [9].
Furthermore, the set $\mathrm{nbs}_A(B)$ depends on the given domain B, i.e., if $A \subset B^* \subset B$ and B and B^* are biconvex sets, then $\mathrm{nbs}_A(B^*) \subsetneq \mathrm{nbs}_A(B)$ in general (cf. Aumann and Hart [9]).
For more theorems dealing with the concept of biconvex separability of a point $z \in B$ from a given set $A \subseteq B$, we refer again to Aumann and Hart [9]. For example, one can find results for the case that the separating biconvex function f additionally has to be continuous on A.

11.3 Biconvex Minimization Problems

In the following we discuss biconvex minimization problems of the form given in Definition 11.3. As mentioned in the beginning of this chapter, biconvex optimization problems may have a large number of local minima as they are global optimization problems in general. Nevertheless, there exist a couple of methods and algorithms which exploit the convex substructures of a biconvex optimization problem in order to solve such problems more efficiently than general global optimization methods do. Of course, such methods can also be used to solve biconvex problems. For example, in Goh et al. [82] subgradient descent methods or interior point methods were proposed to solve a special class of non-smooth, biconvex minimization problems. Since we are especially interested in biconvex optimization methods, we survey only those algorithms and methods which utilize the biconvex structure of the given problem.
This section is organized as follows: In the first subsection, we discuss the notion of partial optimality and recall a necessary optimality condition for biconvex problems with separable constraints. In the following subsections we present methods and algorithms for solving biconvex minimization problems of the form (11.1) that exploit the biconvex structure of the problem. We give short algorithmic descriptions for every solution approach and discuss convergence results and limitations of the considered methods. In detail, we present the *alternate convex search* method as a special case of *block-relaxation methods* (cf. Section 3.3), the *global optimization algorithm*, developed in Floudas and Visweswaran [62] and an algorithm for solving jointly constrained biconvex programming problems using the so called convex envelope of a function f.

11.3.1 Partial Optimality

In the following let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty sets, let $B \subseteq X \times Y$, and let B_x and B_y denote the x-sections and y-sections of B, respectively.

Definition 11.42 *Let $f : B \to \mathbb{R}$ be a given function and let $(x^\star, y^\star) \in B$. Then, $(x^\star, y^\star)$ is called a* partial optimum *of f on B, if*

$$f(x^\star, y^\star) \leq f(x, y^\star) \,\forall\, x \in B_{y^\star} \text{ and } f(x^\star, y^\star) \leq f(x^\star, y) \,\forall\, y \in B_{x^\star}.$$

We recall:

Definition 11.43 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a given function, let $\zeta \in \mathbb{R}^n$, and let the partial derivatives of f in ζ exist. If $\nabla f(\zeta) = 0$, then ζ is called a* stationary point *of f.*

Obviously we have:

Theorem 11.44 *Let $f : B \to \mathbb{R}$ be partial differentiable at $z^\star \in \operatorname{int}(B)$ and let $z^\star$ be a partial optimum. Then, $z^\star$ is a stationary point of f in B.*

Note that the converse of Theorem 11.44 is not true in general.

Example 11.45 Let $z^\star := (0,0) \in \mathbb{R}^2$ and let the function $f : \mathbb{R}^2 \to \mathbb{R}$ be given by

$$f(x,y) = x^3 \cdot (x-2) + y^2.$$

Then $\nabla f(z^\star) = 0$ holds true, but for fixed $y^\star = 0$ we have:

$$f(1,y^\star) \;=\; f(1,0) \;=\; -1 \;<\; 0 \;=\; f(0,0) \;=\; f(z^\star)$$

Hence, $z^\star$ is not a partial optimum.

However, if f is biconvex, we state:

Theorem 11.46 *Let B be a biconvex set and let $f : B \to \mathbb{R}$ be a differentiable, biconvex function. Then, each stationary point of f is a partial optimum.*

Proof: Let $z^\star := (x^\star, y^\star)$ be a stationary point of f in B. For fixed $y^\star$, the function $f_{y^\star} : B_{y^\star} \to \mathbb{R}$ is convex, so

$$f_{y^\star}(x) \;\geq\; f_{y^\star}(x^\star) + \left(\frac{\partial}{\partial x_1} f_{y^\star}(x^\star), \ldots, \frac{\partial}{\partial x_n} f_{y^\star}(x^\star)\right)^t (x - x^\star)$$

is valid for all $x \in B_{y^\star}$ (cf. Rockafellar [183]). Since $x^\star$ is also a stationary point of $f_{y^\star}$, the second summand equals zero and hence

$$f_{y^\star}(x) \;\geq\; f_{y^\star}(x^\star) \quad \forall x \in B_{y^\star}.$$

By symmetry of the problem we also have that

$$f_{x^\star}(y) \;\geq\; f_{x^\star}(y^\star) \quad \forall y \in B_{x^\star}$$

So, $z^\star$ is a partial optimum. □

Corollary 11.47 *Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be a differentiable, biconvex function. Then a point $z \in \mathbb{R}^{n+m}$ is stationary if and only if z is a partial optimum.*

Finally, we shortly review a necessary local optimality condition for the biconvex minimization problem with separable constraints

$$\min \{f(x,y) : \; x \in X \subseteq \mathbb{R}^n, y \in Y \subseteq \mathbb{R}^m\}. \tag{11.7}$$

In the case of separable constraints, the notion of partial optimality of a point $(x^\star, y^\star) \in X \times Y$ simplifies to

$$f(x^\star, y^\star) \;\leq\; f(x, y^\star) \,\forall\, x \in X \text{ and } f(x^\star, y^\star) \;\leq\; f(x^\star, y) \,\forall\, y \in Y.$$

Theorem 11.48 (Wendell and Hurter Jr. [215]) *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be convex sets and let $f : X \times Y \to \mathbb{R}$ be a biconvex function with a partial optimum in $(x^\star, y^\star) \in X \times Y$. Furthermore, let $U(y^\star)$ denote the set of all optimal solutions to Problem (11.7) with $y = y^\star$, and let $U(x^\star)$ be the set of optimal solutions to Problem (11.7) with $x = x^\star$. If $(x^\star, y^\star)$ is a local optimal solution to Problem (11.7), then it necessarily holds that*

$$f(x^\star, y^\star) \leq f(x, y) \quad \forall x \in U(x^\star)\ \forall y \in U(y^\star). \tag{11.8}$$

Note that the given local optimality condition is in general not sufficient.

Example 11.49 (Luenberger [125], mod.) We consider the biconvex minimization problem

$$\min\{x^3 - x^2 y + 2y^2 : x \geq 4, y \in [0; 10]\}.$$

This problem has a partial optimum at $(6, 9)$ that satisfies the condition (11.8) of the last theorem, but that is not a local optimum.

11.3.2 Alternate Convex Search

Alternate convex search (ACS) is a minimization method which is a special case of the *block-relaxation methods* already discussed in Section 3.3, where the variable set is divided into disjoint blocks (cf. de Leeuw [46]). In every step, only the variables of an active block are optimized while those of the other blocks are fixed. For ACS we only consider the two blocks of variables defined by the convex subproblems that are activated in cycles. Since the resulting subproblems are convex, efficient convex minimization methods can be used to solve these subproblems. In the case that $n = m = 1$, i.e., $f : B \subseteq \mathbb{R}^2 \to \mathbb{R}$, ACS can be seen as a special case of the *cyclic coordinate method (CCM)* which is stated, e.g., in Bazaraa et al. [13]. A survey on the ACS approach for convex as well as for biconvex objective functions can be found, e.g., in Wendell and Hurter Jr. [215].
In the following we will show that under weak assumptions the set of all accumulation points generated by ACS form a connected, compact set C and that each of these points is a stationary point of f but that no better convergence results (like local or global optimality properties) can be obtained in general.

Algorithm 11.1 (Alternate Convex Search)
Let a biconvex optimization problem in the sense of Definition 11.3 be given.

Step 1: Choose an arbitrary starting point $z_0 = (x_0, y_0) \in B$ and set $i = 0$.

Step 2: Solve for fixed y_i the convex optimization problem

$$\min\{f(x, y_i),\ x \in B_{y_i}\}. \tag{11.9}$$

If there exists an optimal solution $x^\star \in B_{y_i}$ to this problem, set $x_{i+1} = x^\star$, otherwise STOP.

Step 3: Solve for fixed x_{i+1} the convex optimization problem

$$\min\{f(x_{i+1}, y),\ y \in B_{x_{i+1}}\}. \tag{11.10}$$

If there exists an optimal solution $y^\star \in B_{x_{i+1}}$ to this problem, set $y_{i+1} = y^\star$, otherwise STOP.

Step 4: Set $z_{i+1} = (x_{i+1}, y_{i+1})$. If a stopping criterion is satisfied, then STOP, otherwise augment i by 1 and go back to Step 2.

Remarks:

1. The order of the optimization problems in Step 2 and Step 3 can be reversed, i.e., it is possible first to optimize in the y-variables, followed by an optimization in the x-variables.

2. There are several ways to define the stopping criterion in Step 4 of the algorithm. For example, one can consider the absolute value of the difference of z_{i-1} and z_i (or the difference in their objective values) or the relative increase in the z-variable compared to the last iteration. The stopping criterion may also depend on the special structure of the given biconvex objective function.

The following convergence properties of ACS are motivated by the results of Zangwill [221], Meyer [140] and de Leeuw [46]. The results stated in these papers cannot be applied directly to ACS, since in these papers it is assumed that the algorithmic map A (see Definition 11.52 below) is uniformly compact on B (i.e., there exists $B_0 \subseteq B$, compact, such that $A(z) \subseteq B_0$ for all $z \in B$) which is not true for ACS in general. Note that for most of the following results only continuity of f is needed.

Theorem 11.50 *Let $B \subseteq \mathbb{R}^n \times \mathbb{R}^m$, let $f : B \to \mathbb{R}$ be bounded from below, and let the optimization problems* (11.9) *and* (11.10) *be solvable. Then the sequence $\{f(z_i)\}_{i\in\mathbb{N}}$ generated by ACS converges monotonically.*

Proof: Since the sequence of objective values $\{f(z_i)\}_{i\in\mathbb{N}}$, generated by Algorithm 11.1, is monotonically decreasing and f is bounded from below, the sequence $\{f(z_i)\}_{i\in\mathbb{N}}$ converges to a limit value $a \in \mathbb{R}$. □

The statement of Theorem 11.50 is relatively weak. The boundedness of the objective function f only ensures the convergence of the sequence $\{f(z_i)\}_{i\in\mathbb{N}}$ but not automatically the convergence of the sequence $\{z_i\}_{i\in\mathbb{N}}$. Indeed, there exist biconvex functions where the sequence $\{f(z_i)\}_{i\in\mathbb{N}}$ generated by ACS converges while the sequence $\{z_i\}_{i\in\mathbb{N}}$ diverges. To see this, we consider the following example:

Example 11.51 Let the biconvex function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be given by:

$$f(x,y) := \begin{cases} (x-y)^2 + \frac{1}{x+y+1} & \text{, if} \quad x \geq -y \\ (x-y)^2 + 1 - x - y, \text{ if} & x < -y. \end{cases}$$

It is easy to check that for any starting point $(x_0, y_0) \in \mathbb{R}^2$ the generated sequence $\{f(z_i)\}_{i\in\mathbb{N}}$ converges to 0 while the sequence $\{z_i\}_{i\in\mathbb{N}}$ diverges to infinity.

To give convergence results for the generated sequence $\{z_i\}_{i\in\mathbb{N}}$ we introduce the *algorithmic map* of the ACS algorithm. For a general definition of algorithmic maps we refer to Bazaraa et al. [13].

Definition 11.52 *Let $B \subseteq \mathbb{R}^n \times \mathbb{R}^m$, let $z_k = (x_k, y_k) \in B$ for $k = 1, 2$, and let $f : B \to \mathbb{R}$ be given. The map $A : B \to \mathcal{P}(B)$ from B onto the power set $\mathcal{P}(B)$ of B defined by $z_2 \in A(z_1)$ if and only if*

$$f(x_2, y_1) \leq f(x, y_1) \quad \forall x \in B_{y_1} \text{ and } f(x_2, y_2) \leq f(x_2, y) \quad \forall y \in B_{x_2}$$

is called the algorithmic map *of the ACS algorithm.*

Using the algorithmic map, the ACS algorithm can be described as the iterative selection of a $z_{i+1} \in A(z_i)$. This means that z_{i+1} is a possible outcome of the algorithm with starting point z_i after one complete iteration.

Lemma 11.53 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be closed sets and let $f : X \times Y \to \mathbb{R}$ be continuous. Then the algorithmic map A is closed, i.e., it holds:*

$$\left.\begin{array}{ll} z_i := (x_i, y_i) \in X \times Y, & \lim\limits_{i\to\infty} (x_i, y_i) = (x^\star, y^\star) =: z^\star \\ z_i' := (x_i', y_i') \in A(z_i), & \lim\limits_{i\to\infty} (x_i', y_i') = (x', y') =: z' \end{array}\right\} \Longrightarrow z' \in A(z^\star).$$

Proof: Since $z_i' \in A(z_i)$ for all $i \in \mathbb{N}$ we have that

$$f(x_i', y_i) \leq f(x, y_i) \, \forall x \in X \quad \text{and} \quad f(x_i', y_i') \leq f(x_i', y) \, \forall y \in Y.$$

Since f is continuous by assumption we get that

$$f(x', y^\star) = \lim_{i\to\infty} f(x_i', y_i) \leq \lim_{i\to\infty} f(x, y_i) = f(x, y^\star) \quad \forall x \in X$$

and

$$f(x', y') = \lim_{i\to\infty} f(x_i', y_i') \leq \lim_{i\to\infty} f(x_i', y) = f(x', y) \quad \forall y \in Y.$$

Hence, $z' \in A(z^\star)$. □

The following theorem states a condition for the limit of the sequence of points generated by ACS.

Theorem 11.54 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be closed sets and let $f : X \times Y \to \mathbb{R}$ be continuous. Let the sequence $\{z_i\}_{i\in\mathbb{N}}$ generated by ACS converge to $z^\star \in X \times Y$. Then $z^\star$ is a partial optimum.*

Proof: The sequence $\{z_{i+1}\}_{i\in\mathbb{N}}$ is convergent with limit point $z^\star$. Since the algorithmic map A is closed by Lemma 11.53 and $z_{i+1} \in A(z_i)$ for all $i \in \mathbb{N}$, also $z^\star$ is contained in $A(z^\star)$. Hence,

$$f(x^\star, y^\star) \leq f(x, y^\star) \quad \forall x \in X \quad \text{and} \quad f(x^\star, y^\star) \leq f(x^\star, y) \quad \forall y \in Y$$

and $z^\star$ is a partial optimum. □

Note that a similar result is mentioned in Wendell and Hurter Jr. [215] for X and Y being compact sets. The next lemma ensures that, as long the algorithm generates new points that are no partial optima, a descent in the objective values can be achieved during one iteration.

Lemma 11.55 *Let $B \subseteq \mathbb{R}^n \times \mathbb{R}^m$ and $f : B \to \mathbb{R}$ be given. Let the optimization problems* (11.9) *and* (11.10) *be solvable and let $z_1 := (x_1, y_1) \in B$ and $z_2 := (x_2, y_2) \in A(z_1)$.*

1. *If the optimal solution of Problem* (11.9) *with $y = y_1$ is unique, then*

$$z_1 \text{ is not a partial optimum} \implies f(z_2) < f(z_1).$$

2. *If the optimal solution of Problem* (11.10) *with $x = x_2$ is unique, then*

$$z_2 \text{ is not a partial optimum} \implies f(z_2) < f(z_1).$$

3. *If the optimal solutions of both Problem* (11.9) *with $y = y_1$ and Problem* (11.10) *with $x = x_2$ are unique, then*

$$z_1 \neq z_2 \implies f(z_2) < f(z_1).$$

Proof: Obviously, it holds true that

$$f(z_2) = f(x_2, y_2) \leq f(x_2, y_1) \leq f(x_1, y_1) = f(z_1).$$

We assume that $f(x_2, y_2) = f(x_2, y_1) = f(x_1, y_1)$ and show the reversed statements. Since $z_2 \in A(z_1)$,

$$f(x_2, y_1) \leq f(x, y_1) \, \forall x \in B_{y_1} \quad \text{and} \quad f(x_2, y_2) \leq f(x_2, y) \, \forall y \in B_{x_2}.$$

If the optimal solution of Problem (11.9) with $y = y_1$ is unique, then $x_1 = x_2$ and z_1 is a partial optimum. If the optimal solution of Problem (11.10) with $x = x_2$ is unique, then $y_1 = y_2$ and z_2 is a partial optimum. If the optimal solutions of both Problem (11.9) with $y = y_1$ and Problem (11.10) with $x = x_2$ are unique, $x_1 = x_2$ and $y_1 = y_2$, hence $z_1 = z_2$. □

Now the following theorem about the convergence of the sequence $\{z_i\}_{i \in \mathbb{N}}$ can be stated and proven.

Theorem 11.56 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be closed sets and let $f : X \times Y \to \mathbb{R}$ be continuous. Let the optimization problems* (11.9) *and* (11.10) *be solvable.*

1. *If the sequence $\{z_i\}_{i \in \mathbb{N}}$ generated by the ACS algorithm is contained in a compact set, then the sequence has at least one accumulation point.*

2. *In addition, suppose that for each accumulation point $z^\star = (x^\star, y^\star)$ of the sequence $\{z_i\}_{i \in \mathbb{N}}$ the optimal solution of Problem* (11.9) *with $y = y^\star$ or the optimal solution of Problem* (11.10) *with $x = x^\star$ is unique, then all accumulation points are partial optima and have the same objective value.*

3. *Furthermore, if for each accumulation point $z^\star = (x^\star, y^\star)$ of the sequence $\{z_i\}_{i \in \mathbb{N}}$ the optimal solutions of both Problem* (11.9) *with $y = y^\star$ and Problem* (11.10) *with $x = x^\star$ are unique, then*

$$\lim_{i \to \infty} \|z_{i+1} - z_i\| = 0$$

and the accumulation points form a compact continuum C (i.e., C is a connected, compact set).

Proof: The first part of the theorem follows immediately from the Bolzano-Weierstrass' theorem (cf., e.g., Forster [65]).
By condition 1, the sequence $\{z_i\}_{i\in\mathbb{N}}$ has at least one accumulation point $z^\star := (x^\star, y^\star)$. Thus we have a convergent subsequence $\{z_k\}_{k\in\mathcal{K}}$ with $\mathcal{K} \subseteq \mathbb{N}$ that converges to $z^\star$. Similarly, $\{z_{k+1}\}_{k\in\mathcal{K}}$ has an accumulation point $z^+ := (x^+, y^+)$ to which a subsequence $(z_{l+1})_{l\in\mathcal{K}}$ with $\mathcal{L} \subseteq \mathcal{K}$ converges. By Lemma 11.53 and Theorem 11.50 it follows that $z^+ \in A(z^\star)$ and $f(z^+) = f(z^\star)$. In the same manner we see that the sequence $\{z_{k-1}\}_{k\in\mathcal{K}}$ has an accumulation point $z^- := (x^-, y^-)$ with $z^\star \in A(z^-)$ and $f(z^\star) = f(z^-)$.
Now suppose that $z^\star$ is not a partial optimum even though condition 2 is satisfied. Thus, one of the optimization problems (11.9) with $y = y^\star$ or (11.10) with $x = x^\star$ has a unique solution, and by Lemma 11.55, $f(z^+) < f(z^\star)$ or $f(z^\star) < f(z^-)$, which gives a contradiction. Therefore, $z^\star$ must be a partial optimum. If there exist further accumulation points, their objective values must equal $f(z^\star)$ due to Theorem 11.50.
Suppose that additionally condition 3 is satisfied, but $\|z_{i+1} - z_i\| > \delta$ for infinitely many $i \in \mathbb{N}$ and $\delta > 0$. Then the sequences $\{z_i\}_{i\in\mathbb{N}}$ and $\{z_{i+1}\}_{i\in\mathbb{N}}$ again have accumulation points $z^\star$ and z^+ with $\|z^+ - z^\star\| \geq \delta$. In particular, $z^+ \neq z^\star$. As above we see that $z^+ \in A(z^\star)$ and $f(z^+) = f(z^\star)$. But by Lemma 11.55 it follows that $f(z^+) < f(z^\star)$ which gives a contradiction. Thus the sequence $\{\|z_{i+1} - z_i\|\}_{i\in\mathbb{N}}$ converges to 0, and since $\{z_i\}_{i\in\mathbb{N}}$ is bounded the accumulation points form a compact continuum (cf. Ostrowski [154]). □

Note that in Theorem 11.56 the Problems (11.9) and (11.10) must be uniquely solvable only for the set of accumulation points but not for an arbitrary element of the sequence $\{z_i\}_{i\in\mathbb{N}}$. For a biconvex function f uniqueness of the solutions is automatically guaranteed in practice if, for example, f is strictly convex as a function of y for fixed x and vice versa.
Unfortunately, Theorem 11.56 still does not guarantee the convergence of the sequence $\{z_i\}_{i\in\mathbb{N}}$ but is close enough for all practical purposes. Note that statements similar to Theorem 11.56 can be found in the literature, e.g., for CCM in Bazaraa et al. [13]. But for ACS the assumptions of Theorem 11.56 are weaker since the biconvex structure is used.

Corollary 11.57 *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be closed sets and let $f : X \times Y \to \mathbb{R}$ be a differentiable function. Furthermore, let the sequence $\{z_i\}_{i\in\mathbb{N}}$ generated by the ACS algorithm be contained in a compact set, and for each accumulation point $z^\star = (x^\star, y^\star)$ of the sequence $\{z_i\}_{i\in\mathbb{N}}$ let the optimal solutions of both Problem* (11.9) *with $y = y^\star$ and Problem* (11.10) *with $x = x^\star$ be unique. Then all accumulation points $z^\star$ that lie in the interior of $X \times Y$ are stationary points of f.*

Proof: This is an immediate consequence of Theorem 11.46 and Theorem 11.56. □

It is obviously clear that for every stationary point $z^\star := (x^\star, y^\star) \in B$ of a differentiable, biconvex function f there exists a non-empty set S of starting points such that $z^\star$ is an outcome of the ACS algorithm when the optimal solutions of both Problem (11.9) with $y = y^\star$ and Problem (11.10) with $x = x^\star$ are unique, since all points of the form $(x, y^\star) \in B$ will lead to $z^\star$ within one iteration. So theoretically, all stationary points of f can be generated by ACS.
Furthermore, it can be shown that if the assumptions of Theorem 11.56 are satisfied and X and Y are subsets of $\mathbb{R}$ (i.e., ACS simplifies to CCM), the generated compact

continuum C simplifies to a singleton, i.e., the sequence $\{z_i\}_{i\in\mathbb{N}}$ is actually convergent. Although an accumulation or limit point $z^\star$, generated by Algorithm 11.1, might be a partial optimum, it neither has to be a global nor a local optimum to the given biconvex optimization problem even if $z^\star$ is stationary, as stationary points can be saddle points of the given function. To see this for the case when f is not everywhere differentiable over its whole domain we consider:

Example 11.58 (Goh et al. [82]) Let the function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be given by

$$f(x, y) := \max\{y - 2x,\, x - 2y,\, \frac{1}{4}(x^2 + y^2 - 16)\}.$$

As the pointwise maximum of three convex functions f is convex and thus also biconvex (cf. Lemma 11.19 and Theorem 11.20). Let $M^{(1)}$ and $M^{(2)}$ denote the points $(-4, 2)$ and $(2, -4)$, respectively, and define two sets $C^{(1)}$ and $C^{(2)}$ by

$$\begin{aligned} C^{(1)} &= \{z \in \mathbb{R}^2 : \|z - M^{(1)}\| \leq 6\} \\ C^{(2)} &= \{z \in \mathbb{R}^2 : \|z - M^{(2)}\| \leq 6\}. \end{aligned}$$

A calculation shows that

$$f(x, y) = \begin{cases} y - 2x & \text{for } (x, y) \in C^{(1)} \cap \{(x, y) \in \mathbb{R}^2 : x \leq y\} \\ x - 2y & \text{for } (x, y) \in C^{(2)} \cap \{(x, y) \in \mathbb{R}^2 : x \geq y\} \\ \frac{1}{4}(x^2 + y^2 - 16) & \text{for } (x, y) \in \mathbb{R}^2 \setminus (C^{(1)} \cup C^{(2)}). \end{cases}$$

Furthermore, f is continuous but not everywhere differentiable, since it has non-smooth transitions between the three regions defined above. Nevertheless, f has a global minimum at $z^\star = (2, 2)$ and the two convex optimization problems (11.9) and (11.10) given by the ACS algorithm are well-defined and always have unique solutions. If this procedure is applied to f with a starting-point $z_0 = (x_0, y_0) \in \mathbb{R} \times [2, -4]$, the algorithm will converge to the point $\zeta := (y_0, y_0)$ within one iteration. But ζ is clearly not a minimum of f for $y_0 \in]2, -4]$.
Due to the symmetry of the problem, the result remains true if the first optimization in Step 2 of the algorithm is performed over the y-variables and x_0 is given in the interval $[2, -4]$.

What might happen in the cases when the domain of f is not of the form $X \times Y$ or the set of accumulation points is not a part of the interior of $X \times Y$? As the next example shows it is possible that, depending on the starting point, the resulting limit point of the ACS algorithm need not to be the global or a local minimum in most cases.

Example 11.59 (Floudas and Visweswaran [62], mod.) Consider the biaffine, constrained minimization problem

$$\begin{aligned} \min_{x,y} \quad & -x + xy - y \\ \text{s.t.} \quad & -6x + 8y \leq 3 \\ & 3x - y \leq 3 \\ & x, y \in [0; 1.5] \end{aligned}$$

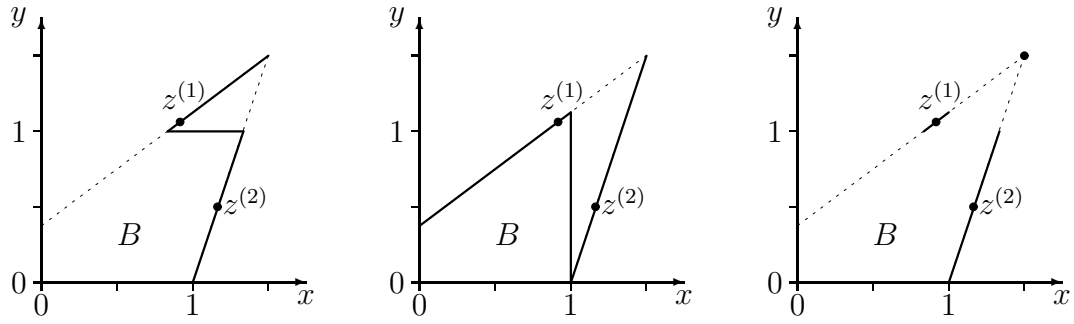


Figure 11.7: *The dashed lines mark the feasible set B of the problem in Example 11.59. The bold line in the first picture illustrates the set of optimal solutions for varying $y \in [0; 1.5]$ with f optimized in x-direction. The bold line in the second picture shows the optimal solutions for varying $x \in [0; 1.5]$ with f optimized in y-direction, while in the third picture the bold lines mark the set of possible outcomes M of the ACS algorithm depending on the chosen starting point.*

which has a local minimum at the boundary point $z^{(1)} = (0.916, 1.062)$ and a global one at the boundary point $z^{(2)} = (1.167, 0.5)$. Denote the feasible set by B and the objective function by f. Since the objective function is affine for fixed x or y in $[0; 1.5]$, the minimal value of f for fixed x or y is attained in a boundary point of B. If we apply a constrained version of the ACS method to solve the problem given above, a calculation shows that every point of the set

$$M := \{(x, y) : 3x - y = 3, y \in [0; 1]\} \cup \{(x, y) : -6x + 8y = 3, y \in]1; 1.125]\} \cup \{(1.5, 1.5)\}$$

is a possible outcome of the algorithm, depending on the chosen starting point (see Figure 11.7). Furthermore, the set of starting points which lead to the global as well as to the local optimum is a discrete point set, and for a starting point z_0 with $y_0 \in [0; 1[$, only the choice $y_0 = 0.5$ results in the global optimum. For $y_0 \in [0; 1[$, the local minimum $z^{(1)}$ is never obtained. Hence, the ACS algorithm applied to the problem given above can provide a point which is far away from being a global or local minimum of the problem.

To find the global optimum of a biconvex minimization problem by ACS, a multistart version of ACS can be used (like, e.g., suggested in Goh et al. [82]). But as we have seen in the last example, there is no guarantee to find the global optimum within a reasonable amount of time or to be sure that the actual best minimum is the global one.

We conclude this subsection with a survey of classes of optimization problems where variants of the ACS method are frequently used to solve biconvex minimization problems in practice. One of these classes are the *location-allocation problems (LAP)* in location theory. Examples for these types of problems are the *multisource Weber problem (MWP)*, first formally stated by Cooper [40], or the *K-connection location problem (KCLP)* in the plane (see Huang et al. [107]) which can be seen as an extension of the classical MWP and is further discussed in Chapter 13.

In these problems from location theory, m new facilities ($m > 1$) have to be located in the plane and allocated to a set of n given existing facilities (or demand points) such that given demands of the existing facilities are satisfied and the total transportation

cost between the existing and the new facilities is minimized. Note that the definition of the total transportation cost depends on the specific problem. This class of optimization problems has a biconvex objective function which is neither convex nor concave (cf. Cooper [40] for the MWP). A well-known heuristic approach to the classical MWP which can also be applied to general LAP's is the alternate location and allocation algorithm developed by Cooper [41] that alternates between a location and an allocation phase until no further improvement can be achieved. This corresponds to the ACS approach applied to the given LAP. A general survey on the application of location-allocation methods in location theory can be found, for example, in Hodgson et al. [103] and Plastria [167].

If we apply the above developed convergence results of the ACS algorithm to the special cases of MWPs and KCLAPs, respectively, we can state that Theorem 11.50 holds true in both cases since the objective function is always non-negative. So, the generated sequence of objective values always converges. Furthermore, since the m new facilities lie within the convex hull of all existing facilities if the distance function is chosen appropriately (this is true, e.g., for Euclidian distances) and the decision variables are restricted to $\{0,1\}$, also item (1.) in Theorem 11.56 applies in both cases. But since neither the position of the new locations nor the partition of the decision variables need to be unique in general, no further results in the decision space can be given in general.

In medical image analysis an ACS approach can be used to register two medical images. In general, a registration problem is a problem where two given data sets have to be rendered in a joint coordinate system such that corresponding features of the two data sets are aligned. In medical image registration the two data sets normally correspond to 2- or 3-dimensional images, the *template image* T which has to be mapped onto the *reference image* R by an appropriate transformation f which can be a rigid function, i.e., a combination of rotations and translations, or a non-rigid function. In practice, rigid transformations are used for registration when it is known that both images T and R show the same part of the body but from a different perspective. Furthermore, they are used to detect morphological changes of an organ (e.g. the growth of a tumor) while non-rigid transformations are normally applied to compensate those changes.

One way to formulate the described registration problem is to select a set of characteristical points $X = \{x_1, \ldots, x_I\}$ in the template image T and a set of corresponding characteristical points $Y = \{y_1, \ldots, y_J\}$ in the reference image R. Then, the transformation $f : T \to R$ is chosen from a set $\mathcal{F}$ of feasible transformations such that the sum of the distances between each image point $f(x_i) \in R$ ($x_i \in X$) and its closest point $y_j \in Y$ in the reference set is minimized. This approach leads to the following generalized biconvex assignment problem

$$\begin{aligned} \min_{f,z} \quad & \sum_{i=1}^{I} \sum_{j=1}^{J} z_{ij} \|f(x_i) - y_j\|^2 \\ \text{s.t.} \quad & \sum_{j=1}^{J} z_{ij} = 1, \qquad i = 1, \ldots, I, \\ & z_{ij} \in \{0,1\}, \quad i = 1, \ldots, I,\ j = 1, \ldots, J, \\ & f \in \mathcal{F}, \end{aligned} \tag{11.11}$$

where z_{ij} equals 1 if the point $x_i \in X$ is assigned to the point $y_j \in Y$, i.e., y_j is the closest point to $f(x_i)$ in Y. Otherwise, z_{ij} is set to 0. Note that the binary constraints

on the assignment variables z_{ij} can be relaxed to $z_{ij} \in [0,1]$ since the assignment matrix is totally unimodular.
A common solution approach to Problem (11.11) is the *iterative closest point (ICP)* algorithm which was developed by Besl and McKay [19] and corresponds to the ACS approach. The algorithm alternates between an assignment step in which the points $f(x_i) \in R$ are assigned to their closest neighbor in Y and a step in which a new transformation function f is chosen until no further improvement is achieved. For further details we refer to Zitová and Flusser [223] where a survey on image registration can be found, and to Besl and McKay [19] for information on the ICP algorithm. From the theoretical point of view, we get the same results as in the case of the LAPs. The given objective function is always non-negative, so the sequence of objective values produced by the ICP algorithm is convergent due to Theorem 11.50. Usually, the set of all feasible transformations $\mathcal{F}$ can be restricted such that the transformations are determined by only a finite number of parameters which are contained in a compact subset of $\mathbb{R}^n$. In this case, the feasible set of the problem is compact and the sequence generated by the ICP algorithm has an accumulation point in the decision space by Theorem 11.56. Since neither the chosen transformation nor the assignment variables need to be unique, no further results can be obtained in general.
Another field where ACS is frequently used as a standard approach is the field of (robust) control theory. For example, the *biaffine matrix inequality (BMI) feasibility problem*, stated, e.g., in Goh et al. [82], can be solved by the ACS method. But since the BMI problem has a non-smooth objective function, in general no global or local minimum can be determined by using the ACS approach (cf. Example 11.58). So, other non-convex optimization methods have to be considered to obtain an optimal solution for the BMI problem. For further details see, e.g., Goh et al. [82] and Goh et al. [81].

11.3.3 The Global Optimization Algorithm

In this subsection we review an algorithm for constrained biconvex minimization problems which exploits the convex substructure of the problem by a primal-relaxed dual approach. The algorithm is called *global optimization algorithm (GOP)* and was developed by Floudas and Visweswaran [62]. The method follows decomposition ideas introduced by Benders [16] and Geoffrion [77]. Like in the second step of the ACS method, the constrained problem is firstly solved for a fixed value of the y-variables which leads to an upper bound on the solution of the biconvex problem. This problem is called *primal problem*. To get a lower bound to the solution, duality theory and linear relaxation are applied. The resulting *relaxed dual problem* is solved for every possible combination of bounds in a subset of the x-variables, the set of *connected* x-variables. By iterating between the primal and the relaxed dual problem a finite ε-convergence to the global optimum can be shown.
In the following, we focus on the assumptions that have to be satisfied by the given biconvex minimization problem so that the GOP algorithm can be applied to this problem. A deeper description of the mathematical background and a detailed outline of the algorithm are given in Floudas and Visweswaran [62], Floudas and Visweswaran [63] and the books Floudas [61] and Floudas [64]. We shortly review convergence results and give a short survey on the optimization fields in which the algorithm is

used.
We consider an optimization problem of the form

$$\begin{aligned} & \min_{x,y} f(x,y) \\ \text{s.t.} \quad & g(x,y) \leq 0 \\ & h(x,y) = 0 \\ & x \in X,\ y \in Y, \end{aligned} \tag{11.12}$$

where X and Y are compact convex sets, and $g(x,y)$ and $h(x,y)$ are vectors of inequality and equality constraints. The functions must be differentiable and they must be given in explicit form. Furthermore, the following conditions, denoted by *Conditions (A)*, need to be satisfied (cf. Floudas [64])

1. f is biconvex on $X \times Y$.
2. g is biconvex on $X \times Y$.
3. h is biaffine on $X \times Y$.
4. An appropriate first order constraint qualification is satisfied for fixed y.

Note that, for example, in Floudas [64] partitioning and transformation methods for the variable set of quadratic programming problems are suggested so that it is possible to transform this class of problems into a problem of type (11.12) where Condition (A) is satisfied automatically.
Now, let

$$V := \{y : h(x,y) = 0,\ g(x,y) \leq 0 \text{ for some } x \in X\},$$

then the following ε-convergence result for the GOP algorithm holds:

Theorem 11.60 (Floudas [64]) *If X and Y are non-empty compact convex sets satisfying that $Y \subset V$, f, g, and h are continuous on $X \times Y$, the set $U(y)$ of optimal multipliers for the primal problem is non-empty for all $y \in Y$ and uniformly bounded in some neighborhood of each such point and Condition (A) is satisfied, then the GOP algorithm terminates in a finite number of steps for any given $\varepsilon > 0$.*

For the resulting solution it holds:

Corollary 11.61 (Floudas [64]) *If the conditions stated in Theorem 11.60 hold, the GOP algorithm will terminate at the global optimum of the biconvex minimization problem.*

What are the advantages and drawbacks of the GOP algorithm? As mentioned above, one of the advantages of the algorithm is the fact that the primal problem which has to be solved in the first step of each iteration is a convex problem. Hence, every local optimum is the global minimum of the subproblem. Furthermore, the set of constraints for the convex subproblem often simplifies to linear or quadratic constraints in the x-variables so that the primal problem can be solved by any conventional non-linear local optimization solver. As another advantage of this approach can be seen that the relaxed dual problem has only to be solved in the connected x-variables. This might

reduce the number of variables for which the relaxed dual problem has to be solved. For further details see, e.g., Floudas [64].
The main drawback of the GOP algorithm is the fact that in each iteration of the algorithm, $2^{|I|}$ in general non-linear subproblems have to be solved to obtain a new lower bound to the problem, where I denotes the set of the connected x-variables. In fact, in each iteration a total enumeration of all possible assignments of the connected x-variables to their lower and upper bounds is done and the relaxed dual problem is solved for every combination of these bounds. In Floudas [64], several improvements of the algorithm, depending on the structure of the given biconvex problem, are given to reduce the number of relaxed dual problems.
The GOP algorithm is a useful tool for different classes of practical optimization problems. Visweswaran and Floudas [209], Visweswaran and Floudas [210], and Floudas [64] discuss, amongst others, quadratic problems with linear constraints, quadratically constrained problems, and univariate polynomial problems. Furthermore, an application to bilevel linear and quadratic problems, a practical approach to phase and chemical equilibrium problems as well as an implementation and computational studies of the GOP algorithm can be found there.
In Barmish et al. [12] a solution algorithm for some open robustness problems including matrix polytope stability is stated which was influenced by the ideas of the GOP approach. There, the optimization on the x-variables is carried out in the usual way (i.e., for fixed y) to get a valid upper bound. The optimization on the y-variables is done by a "relaxation" process where the relaxation is refined at each subsequent iteration step. By an accumulation of the resulting constraints, better lower bounds on the problem are obtained in each step of the iteration and an ε-convergence to the optimum can be proven. Note that for the problems stated Barmish et al. [12], only a finite number of linear programs have to be solved to get an ε-global optimum.
A convex minimization problem with an additional biconvex constraint is considered in the paper of Tuyen and Muu [205]. There, a convex criterion function of a multiple objective affine fractional problem has to be minimized over the set of all weakly efficient solutions of the fractional problem. As in the GOP algorithm, Lagrangian duality and a simplicial subdivision is used to develop a branch and bound algorithm which is proven to converge to a global ε-optimal solution of the problem.

11.3.4 Jointly Constrained Biconvex Programming

In this subsection we concentrate on a special case of a jointly constrained biconvex programming problem introduced by Al-Khayyal and Falk [5]. The specific problem is given by

$$\begin{aligned} \min_{(x,y)} \quad & \Phi(x,y) \;=\; f(x) + x^t y + g(y) \\ & \text{s.t. } (x,y) \;\in\; S \cap \Omega, \end{aligned} \tag{11.13}$$

where

1. $\Omega \;=\; \{(x,y) : l \leq x \leq L,\, m \leq y \leq M\}$,
2. S is a closed, convex set, and
3. f and g are convex over $S \cap \Omega$.

Since the functions f and g are convex, the objective function Φ is biconvex on $S \cap \Omega$. Problem (11.13) can be seen as a generalization of an ordinary bilinear programming problem which is of the form

$$\begin{aligned} \min_{(x,y)} \quad & c^t x + x^t A y + d^t y \\ \text{s.t.} \quad & x \in X,\, y \in Y, \end{aligned} \tag{11.14}$$

where $c \in \mathbb{R}^n$ and $d \in \mathbb{R}^n$ are given vectors, A is an $(n \times n)$-matrix and X and Y are polytopes in $\mathbb{R}^n$. The above given biconvex problem is more general since it allows joint constraints in (x, y) and non-linear, convex subfunctions f and g in Φ. The bilinear Problem (11.14) can be transformed into the biconvex Problem (11.13) by replacing the term $x^t\,(Ay)$ by $x^t\,z$ and including the linear constraint $z = Ay$ among the constraints defining the feasible set.
While bilinear problems of the form of Problem (11.14) always have extreme-point solutions in $X^\star \times Y^\star$ (cf. Horst and Tuy [105]), where $X^\star$ and $Y^\star$ denote the set of extreme-points of the polytopes X and Y, respectively, this is no longer the case for biconvex problems of the form (11.13) (cf. Al-Khayyal and Falk [5]). Nevertheless, if the objective function Φ is also a biconcave function which is optimized over a compact, convex set $C \subset \mathbb{R}^n \times \mathbb{R}^n$, then it can be shown that if the minimum of Φ over C exists, it is achieved in at least one boundary point of C (cf. Al-Khayyal and Falk [5]).
Al-Khayyal and Falk [5] used a pattern of Falk and Soland [59] to develop a branch and bound algorithm for solving the jointly constrained biconvex Problem (11.13). The necessary bounds are obtained by employing convex envelopes.

Definition 11.62 (Floudas [64]) *Let f be a lower semicontinuous function defined on a non-empty convex set $C \subseteq \mathbb{R}^n$. Then the* convex envelope *of f on C is a function $\Psi_C(f) : C \to \mathbb{R}$ that satisfies:*

1. *$\Psi_C(f)$ is convex on C.*
2. *$\Psi_C(f(x)) \leq f(x)$ for all $x \in C$.*
3. *If h is any convex function defined on C such that $h(x) \leq f(x)$ for all $x \in C$, then $h(x) \leq \Psi_C(f(x))$ for all $x \in C$.*

Note that the convex envelope of f is obtained by taking the pointwise supremum of all convex (or linear) functions which underestimate f over C (cf. Al-Khayyal and Falk [5]).
Since

$$\begin{aligned} \Psi_\Omega(x^t y) &= \sum_{i=1}^{n} \Psi_{\Omega_i}(x_i y_i) \quad \forall\, (x,y) \in \Omega, \\ \Psi_\Omega(x^t y) &= x^t y \qquad\qquad\qquad \forall\, (x,y) \in \partial\Omega, \end{aligned}$$

and $\Psi_{\Omega_i}(x_i y_i)$ can easily be calculated (cf. Al-Khayyal and Falk [5]),

$$F(x,y) := f(x) + \Psi_\Omega(x^t y) + g(y)$$

is a convex underestimator of Φ on $S \cap \Omega$ that coincides with Φ on $\partial\Omega$ and is used to calculate lower bounds of the objective functions.

Now the algorithm works as follows: In the first step the minimization Problem (11.13) is solved with F instead of Φ as objective function which leads to an optimal point $z^1 = (x^1, y^1) \in S \cap \Omega$ and valid lower and upper bounds $F(z^1)$ and $\Phi(z^1)$. If $F(z^1) = \Phi(z^1)$, then z^1 is optimal. Otherwise, there exists at least one $i \in \{1, \ldots, n\}$ such that $\Psi_{\Omega_i}(x_i y_i) < x_i y_i$. So, the index i that leads to the largest difference between $x_i y_i$ and $\Psi_{\Omega_i}(x_i y_i)$ is chosen, and the i^{th} rectangle Ω_i is split up into four subrectangles. Then, new bounds are calculated in each of the resulting four new hyper-rectangles. This leads to a point z^2 and new lower and upper bounds for f which can be shown to be tighter than the bounds of the last iteration. If $F(z^2) = \Phi(z^2)$ the algorithm stops, otherwise a new refinement is performed. By iteratively applying this procedure, it can be shown that the algorithm converges to a global optimum of Problem (11.13).
Horst and Tuy [105] presented in their book a modified version of the algorithm which differs from the original one in the choice of the new iterate z^k and the subdivision rule which is based on bisection there. In Al-Khayyal [4] the author strengthened the algorithm by also evaluating the concave envelope of the problem. In Audet et al. [8] a short overview of papers that concentrate on the application of the algorithm to bilinear programming problems and quadratically constrained quadratic programming problems is given.
Another algorithm for a special type of functions f and g of Problem (11.13) is developed in Sherali et al. [191] and Sherali et al. [192] for risk management problems. Instead of working with the convex envelope, the authors used a specialized implementation of Geoffrion's generalized Benders decomposition (see Geoffrion [77]). With the help of a projection method and dual theory, an alternative graphical solution scheme is proposed that enables the decision maker to interact subjectively with the optimization process.

11.4 Conclusions and Further Ideas

In this chapter we gave a survey on optimization problems with biconvex sets and biconvex functions and reviewed properties of these sets and functions given in the literature. We stated a new result for the case that the maximum of a biconvex function f is attained in the relative interior of a biconvex set B by assuming further, rather weak topological properties on B. We showed that under these assumptions f must be constant throughout B.
Existing methods and algorithms, specially designed for biconvex minimization problems which primarily exploit the convex substructures of the problem, were discussed for the constrained as well as for the unconstrained case. In particular, we showed that an alternating convex search approach, a primal-relaxed dual approach, as well as an approach that uses the convex envelope of parts of the biconvex objective function are suitable for solving biconvex optimization problems using the special properties of these problems. For each of these methods different practical applications as well as applications to the bilinear and biaffine case were discussed. We recalled that under appropriate assumptions the primal-relaxed dual approach as well as the approach that uses the convex envelope lead to a global optimum, while the alternating approach in general only finds partial optima and stationary points of the objective function, respectively. However, the advantage of this approach can be seen in the fact that it

can be applied to any biconvex minimization problem while for the other approaches additional properties for the given objective function as well as for the feasible set must be satisfied.
Further fields of research related to biconvex sets and functions are separation theorems of disjoint biconvex sets with biconvex functions (cf. Aumann and Hart [9]). Moreover, improvements of the given minimization algorithms are of interest, especially of the ACS method. Concerning this topic, we also refer to the Chapters 3, 12 and 13 of this work.

Chapter 12

Augmented Alternate Convex Search for Biconvex Optimization Problems

In this chapter we present an enhanced version of the alternate convex search method for biconvex optimization problems. The original version of this approach was already discussed in further details in Section 11.3 of this work. We make use of the theoretical ideas developed in Section 3.3 for the enhanced version of the alternate block search strategy, to try to heuristically improve the performance of the original alternate convex search method.

As already stated in Section 11.3, this method can be seen as a special case of the more general alternate block search strategy, where only two disjoint blocks of variables are given. In addition, the two resulting subproblems are assumed to be convex, i.e. each local optimal solution of the induced subproblems is automatically a global optimum (cf. Rockafellar [183]). While the incorporation of the descent potential for the alternate block search strategy was only discussed from a theoretical point of view in Section 3.3, we go into further details for the biconvex case in this chapter. Amongst others, we show that gradient information can be used to define the descent potential for the case that f is assumed to be continuously differentiable on its domain.

However, although we restrict ourselves to the case of only two disjoint blocks of variables as well as to convex subproblems in the following, we note that all the ideas that are presented in the remainder of this chapter can easily be generalized to the case of more than two given blocks of variables and to the case of non-convex subproblems, respectively.

We further remark that in Chapter 13 we make use of the ideas presented in this chapter to heuristically improve the results that are obtained by applying the alternate convex search method to solve a biconvex optimization problem from location theory. In particular, we show how the ideas presented in this chapter can be used to solve the connection location-allocation problem in the plane (cf. Huang [106] and Bischoff [20]).

The remainder of this chapter is organized as follows: In the next section, we give a short introduction to the problem formulation and the notation used throughout this chapter. In Section 12.2 we discuss potential definitions of the descent potential for the block of fixed variables, while the enhanced alternate convex search strategy is presented in Section 12.3. We illustrate our ideas by means of an appropriate example

in Section 12.4 and we finally conclude in Section 12.5.

12.1 Notation and Problem Formulation

As we are dealing with biconvex functions in this chapter, we make use of the notation introduced in Chapter 11. However, to make this chapter self-contained, we recall the most important terms and definitions from Sections 3.3 and 11.3 of this work.
Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be two non-empty sets and let $B \subseteq X \times Y$. For $x \in X$ and $y \in Y$, we define the x-sections and y-sections of B by

$$B_x = \{y \in Y : (x, y) \in B\} \subseteq Y \text{ and } B_y = \{x \in X : (x, y) \in B\} \subseteq X,$$

respectively. Furthermore, let $f : B \to \mathbb{R}$ be a continuously differentiable function on $X \times Y$. To simplify the notation, we define $f_x : B_x \to \mathbb{R}$, $f_x(y) := f(x, y)$ for fixed $x \in X$ and $f_y : B_y \to \mathbb{R}$, $f_y(x) := f(x, y)$ for fixed $y \in Y$, respectively. The biconvex optimization problem considered in this chapter is given by

$$\min \{f(x, y) : (x, y) \in B\}. \tag{12.1}$$

From Definition 11.42 we recall that a point $(x^\star, y^\star) \in B$ is called a *partial optimum* of f in B if

$$f(x^\star, y^\star) \leq f(x, y^\star) \,\forall\, x \in B_{y^\star} \text{ and } f(x^\star, y^\star) \leq f(x^\star, y) \,\forall\, y \in B_{x^\star}.$$

For a detailed discussion of partial optimal solutions, we refer to Section 11.3. We recall from Theorem 11.44 that each partial optimum $(x^\star, y^\star) \in B$ of f is also a stationary point, i.e. $\nabla f(x^\star, y^\star) = 0$ holds true, while the converse is not true in general (cf. Example 11.45).
As an algorithmic description of the alternate convex search method is already given by Algorithm 11.1 in Chapter 11, we omit a detailed outline here. Note that an alternative description of this algorithm can be derived from Algorithm 3.1 in Chapter 3. We further recall from Section 11.3 that the point z that is returned by Algorithm 11.1 corresponds to a stationary point of f, whenever some mild assumptions on f as well as the feasible set are satisfied (cf. Corollary 11.57).
In the remainder of this section it is assumed that Problem (12.1) is hard to solve in general, while the subproblems $\min_{y \in B_x} f_x(y)$ and $\min_{x \in B_y} f_y(x)$ are much easier to handle for fixed $x \in X$ and $y \in Y$, respectively, compared to the overall problem. As by assumption, the functions f_x and f_y are convex for all $x \in B_x$ and $y \in B_y$, respectively, this especially holds true when the overall Problem (12.1) corresponds to a non-convex optimization problem. However, as already mentioned in the introduction of this chapter, the general ideas for the application of the enhanced alternate convex search strategy presented in the following are not limited to problems where the involved functions f_x and f_y are known to be convex, but they can also be applied to more general non-linear optimization problems with two or more blocks of variables.

12.2 The Descent Potential

Based on the ideas presented in Section 3.3 of this work, we show how the descent potential for the block of fixed variables can be defined in the case of a continuously

differentiable function. In the following we distinguish between the two cases whether $B = \mathbb{R}^n \times \mathbb{R}^m$, i.e. the given Problem (12.1) corresponds to an unconstrained problem, or whether B is given as a compact subset of $X \times Y$. Note that the biconvexity of the given objective function f is not a necessary criterion for the validity of the ideas presented in this section. However, it ensures that all local optima that are found during the iterations of the original alternate convex search method correspond to global optima of the solved subproblems.
To motivate the mathematical definition of the descent potential given below from a more general point of view, let $g : \mathbb{R}^n \to \mathbb{R}$ denote a continuously differentiable function, and let $x_0, v \in \mathbb{R}^n$. Using the Taylor expansion of g in a neighborhood of x_0, g can locally be rewritten as

$$g(x_0 + v) \;=\; g(x_0) + \langle \nabla g(x_0), v \rangle + o(\|v\|),$$

where ∇g denotes the gradient of g and $\langle \cdot, \cdot \rangle$ corresponds to the standard scalar product in $\mathbb{R}^n$ (cf. Forster [65]). This implies that in a neighborhood $U(x_0)$ of x_0, g can locally be approximated by

$$g(x_0 + v) \;\approx\; g(x_0) + \langle g(x_0), v \rangle,$$

where the real number $\langle g(x_0), v \rangle$ indicates the potential change of g in the point x_0 and direction v. Hence, to estimate the potential improvement of g in $U(x_0)$, we may compute $v \in \mathbb{R}^n$ that corresponds to an optimal solution of the continuous optimization problem

$$\min\{\langle \nabla g(x_0), v \rangle : \; \|v\| = 1, \; v \in \mathbb{R}^n\}. \tag{12.2}$$

From Bazaraa et al. [13] we recall that the optimal solution of Problem (12.2) corresponds to the direction of the steepest descent given by $v_0 := -\nabla g(x_0)/\|\nabla g(x_0)\|$. Since

$$\langle \nabla g(x_0), v_0 \rangle \;=\; -\frac{\langle \nabla g(x_0), \nabla g(x_0) \rangle}{\|\nabla g(x_0)\|} \;=\; -\|\nabla g(x_0)\|,$$

the norm of the gradient of g is an appropriate choice for measuring the expected potential improvement of g in a neighborhood of x_0. If g is further assumed to be convex, we recall from convex analysis that the first order Taylor approximation of g in x_0 is a global underestimator of the given function (cf. Boyd and Vandenberghe [25]). In this case, the negative norm of the gradient of g in x_0 provides a strict lower bound on the expected improvement that can be achieved.
The situation completely changes if g is only defined on a compact subset $S \subseteq \mathbb{R}^n$. While the above stated results remain valid, whenever the point $x_0 \in S$ corresponds to an interior point of S, i.e. there exists a neighborhood of x_0 that is still completely contained in S, this is no longer the case, if x_0 belongs to the boundary of B. If for $v = -\nabla g(x, y) \in \mathbb{R}^n$ and arbitrary $\lambda > 0$, the point $x + \lambda v$ does not belong the feasible set S of Problem (12.1), the direction v does no longer correspond to a feasible (descent) direction of g in x_0. Hence, the information provided by the negative norm of the gradient of g can no longer be used to measure the estimated improvement of the objective value in a neighborhood of the point x_0. Instead, the optimal solution of the optimization problem

$$\min\{\langle \nabla g(x_0), v \rangle : \; \|v\| = 1, \; \exists \bar{\lambda} > 0 : \; x_0 + \lambda v \in S \quad \forall \lambda \leq \bar{\lambda}\}, \tag{12.3}$$

has to be used to derive the desired information. However, since we want to evaluate the descent information for a continuous set of points in the following, the above stated optimization problem has to be solved for each point of the feasible set S in general, as long as no further information on S or the function g are given.

In the following, we relate the above stated results to the optimization Problem (12.1) under consideration. We start with the unconstrained case. Let $B = \mathbb{R}^n \times \mathbb{R}^m$, and let $f : B \to \mathbb{R}$ be a continuously differentiable (biconvex) function. Furthermore, let $\nabla f(x, y)$ denote the gradient of f in the point $(x, y) \in B$. To simplify the notation, we define

$$\begin{aligned} \nabla_x f(x,y) &= \left(\frac{\partial f}{\partial x_1}(x,y), \ldots, \frac{\partial f}{\partial x_n}(x,y)\right)^\top \text{ and} \\ \nabla_y f(x,y) &= \left(\frac{\partial f}{\partial y_1}(x,y), \ldots, \frac{\partial f}{\partial y_m}(x,y)\right)^\top, \end{aligned}$$

i.e. $\nabla f(x, y) = (\nabla_x f(x, y), \nabla_y f(x, y))^\top$. According to the above stated results, the norm of the gradient yields an appropriate measure of the expected potential improvement in the point $(x, y) \in B$ for fixed $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, respectively.

Definition 12.1 *Let $(x, y) \in B$ be a feasible solution of Problem* (12.1)*. Then, the functions*

$$\pi_x : \begin{cases} B & \to & \mathbb{R} \\ (x,y) & \mapsto & \|\nabla_x f(x,y)\| \end{cases} \quad \textit{and } \pi_y : \begin{cases} B & \to & \mathbb{R} \\ (x,y) & \mapsto & \|\nabla_y f(x,y)\| \end{cases}$$

are called the descent potential *of f with respect to x and y in the point (x, y), respectively.*

Note that the norm $\|\cdot\|$ is not specified in Definition 12.1. In practice, the (squared) Euclidian or the maximum norm can be used.
To further simplify the notation, we define $\tilde{\pi}_y(x) = \pi_y(x, y)$ for a fixed given $y \in \mathbb{R}^m$ and all $x \in \mathbb{R}^n$, as well as $\tilde{\pi}_x(y) = \pi_x(x, y)$ for a fixed given $x \in X$ and all $y \in B_x$. Since f is assumed to be continuously differentiable, the functions π_x and π_y are both continuous within their domain $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. Furthermore, if the point $(x, y) \in B$ corresponds to a stationary point of f, this additionally implies that $\pi_x(x, y) = \pi_y(x, y) = 0$, i.e. each stationary point is a point of minimum potential.
According to the results for the constrained case stated above, we remark that the values of the above defined descent potentials have to be replaced by

$$\begin{aligned} \pi_x(x,y) &= |\min\{0, \langle \nabla_x f(x,y), v_0 \rangle\}| \text{ and} \\ \pi_y(x,y) &= |\min\{0, \langle \nabla_y f(x,y), v_0 \rangle\}|, \end{aligned} \tag{12.4}$$

respectively, where v_0 corresponds to an optimal solution of Problem (12.3), when the point (x, y) belongs to a boundary point of B for fixed x and y, respectively. While for the unconstrained case, the negative gradient always points into the direction of the steepest descent and is of positive length as long as no partial optimum of f is reached, the optimal solution of Problem (12.3) may indicate that no further descent

is possible in the considered point, i.e. $\langle \nabla_x f(x,y), v_0 \rangle > 0$ and $\langle \nabla_y f(x,y), v_0 \rangle > 0$ holds true, respectively. In this case, the descent potential for this point has to be considered as zero.
Having a closer look at the alternate convex search algorithm (cf. Algorithm 11.1), the main drawback of this approach can be seen in the fact that the quality of the returned solution (i.e. its objective value compared to the global optimum of the given problem) strongly depends on the initial starting solution $(x_0, y_0) \in B$. If the sequence of points that is generated during the course of the algorithm is stuck in a neighborhood of a local minimum of Problem (12.1), it will never leave this neighborhood, but the generated sequence of points in the decision space converges to the local minimum. Hence, already the initial solution chosen at the beginning of Algorithm 11.1 determines the solution that will be found at the end, while potentially better local minima located in other neighborhoods around the initial starting solution are disregarded by this method.
However, especially in the first iterations of the algorithm, also the expected descent with respect to the block of fixed variables may be of interest, when the problem is solved for the active block. One might rather prefer stopping in a non-optimal point with respect to the active variables, when the potential descent with respect to the fixed variables is expected to be higher in this point compared to the potential descent in the optimal solution of the subproblem. Hence, restarting from this non-optimal point for the next subproblem may result in a better local or even the global optimum of the given problem at the end of the algorithm.
Unfortunately, it cannot be guaranteed in general that the global optimum of the problem can be found when additional gradient information is used, nor that the final solution is a better local minimum as compared to the point that is generated without using this additional information. Hence, the above described idea should be seen as a heuristic approach to improve the solution that is obtained when the alternate convex search method is applied. Numerical studies for the connection location-allocation problem, provided in Chapter 13 of this work, show that this works very well in practice, i.e., that an improvement with respect to the overall objective value can be expected in many cases. For more details on this problem, we refer to Section 13.3.

12.3 The Augmented Alternate Convex Search Algorithm

In this section, we further discuss the enhanced alternate convex search strategy based on a biobjective approach. Let $B \subseteq X \times Y$, where $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$. Furthermore, let $(x,y) \in B$. For fixed $y \in Y$, we consider the two objectives f_y and $\tilde{\pi}_y$, where the first has to be minimized, while we are interested in a maximum of the second. The resulting biobjective minimization problem is given by

$$\begin{aligned} \min F_y(x) &= (f_y(x), -\tilde{\pi}_y(x))^\top \\ \text{s.t. } x &\in B_y. \end{aligned} \tag{12.5}$$

Conversely, for fixed $x \in \mathbb{R}^n$ we have to consider the two objectives f_x and $\tilde{\pi}_x$ which results in the biobjective minimization problem

Algorithm 12.1 Augmented Alternate Convex Search Algorithm

Input: An instance of Problem (12.1).
Output: A point $(x, y) \in B$.

1: Choose an arbitrary starting point $(x_0, y_0) \in B$ and set $i = 0$.
2: **while** no stopping criterion is satisfied **do**
3: Determine a representative set $X' \subseteq B_{y_i}$ of efficient solutions of Problem (12.5).
4: Choose $x \in X'$ according to a prescribed rule, and set $x_{i+1} = x$.
5: Determine a representative set $Y' \subseteq B_{x_{i+1}}$ of efficient solutions of Problem (12.6).
6: Choose $y \in Y'$ according to a prescribed rule, and set $y_{i+1} = y$.
7: Augment i by one unit.
8: **end while**
9: **return** (x_i, y_i).

$$\begin{aligned} \min\, & F_x(y) = (f_x(y), -\tilde{\pi}_x(y))^\top \\ \text{s.t. } & y \in B_x. \end{aligned} \tag{12.6}$$

Unfortunately, Problems (12.5) and (12.6) normally correspond to biobjective non-linear optimization problems. Even for the case that $B = \mathbb{R}^n \times \mathbb{R}^m$ and both functions f_x and f_y are known to be convex for all $(x, y) \in B$ as it is the case here, the given problems correspond to non-convex problems, since the derivative of a convex function must not be convex in general. Hence, solution methods from non-linear optimization have to be used to derive efficient solutions of the given problems. However, if the main interest is to improve the performance of Algorithm 11.1, local efficient solutions of Problems (12.5) and (12.6) may suffice to find better local minima at the end of the algorithm.

A detailed outline of the *augmented alternate convex search algorithm* can be found in Algorithm 12.1. By symmetry of the given Problems (12.5) and (12.6), we only consider Problem (12.5), i.e. Lines 3 and 4 of Algorithm 12.1, in more detail.

As it is assumed that the subproblems for a fixed block of variables can be solved efficiently, spending too much time in the determination of an appropriate representative set of efficient solutions for Problem (12.5) is not favorable in general. Although further information on the descent potential $\tilde{\pi}_y$ are of interest, time-consuming calculations of the non-dominated set should be avoided. This especially holds true if the number of expected local minima of Problem (12.1) is small. In this case, also a multi-start version of Algorithm 11.1 may be favorable. So, only a small number of representatives of non-dominated solutions should be calculated for Problem (12.5) in Line 3 of Algorithm 12.1.

In the subsequent step of Algorithm 12.1 (cf. Line 4) an appropriate solution candidate from the set X' is chosen that has to satisfy a prescribed decision rule specified by the decision maker. While during the first iterations, intermediate solutions $x_i \in X'$ with a large descent potential $\tilde{\pi}_y$ are of interest, one is rather interested in optimal solutions of the original single objective problem in later iterations of the algorithm to guarantee its convergence. In practice, the specific implementation of this rule normally depends on the considered optimization problem as well as on the specific interests of the decision maker, as the applied rule has a large impact on the CPU time spent to solve the problem. For a detailed discussion of appropriate, problem-dependent definitions

of this rule, we also refer to Chapter 13, where an adapted version of Algorithm 12.1 is applied to derive local optima for the connection location-allocation problem in the plane.

In the following, we shortly discuss the pros and cons of the two most frequently used scalarization techniques (the ε-constraint and the weighted sum approach) to determine (weakly) efficient solutions of Problem (12.5). In general, an ε-constraint approach seems to be suitable to solve Problem (12.5). Formally, this problem is given by

$$\begin{aligned} &\min\ f_y(x) \\ &\text{s.t. } \tilde{\pi}_y(x) \geq \varepsilon \\ &\qquad x \in B_y. \end{aligned} \tag{12.7}$$

On the one hand, the main advantage of this solution technique can be seen in the fact that the desired level of $\tilde{\pi}_y$ can easily be handled by an adaptive choice of the parameter ε. If a high value of $\tilde{\pi}_y$ is of interest, the parameter ε can be chosen accordingly. If a faster convergence is desired at the end of the algorithm, the value on ε can be decreased. Actually, fixing $\varepsilon > 0$ throughout the application of the alternate convex search method implies that the algorithm will not converge to a stationary point of f, since the additional side constraint on $\tilde{\pi}_y$ automatically implies that $\nabla_y f(x, y) > 0$ for all feasible solutions of Problem (12.7). However, if B_y is compact and f_y attains its minimum in a boundary point, this point still may satisfy the constraints for $\tilde{\pi}_y$.
On the other hand, if the original subproblem with objective f_y can be solved efficiently, the additional constraint on $\tilde{\pi}_y$ may destroy the simple structure of the given optimization problem. This is especially the case, when Problem (12.5) corresponds to an unconstrained problem, i.e. $B_y = \mathbb{R}^n$. Furthermore, the calculated solution of Problem (12.7) may only correspond to a weakly non-dominated outcome of the biobjective problem.
For this reason, also a weighted sum approach can be used to calculate supported efficient solutions of Problem (12.7). For $\lambda \in [0, 1]$ this approach is formally given by

$$\min \{\lambda\, f_y(x) - (1 - \lambda)\tilde{\pi}_y(x) :\ x \in B_y\}. \tag{12.8}$$

However, while this approach preserves the underlying structure of the feasible set B_y, an appropriate choice of the parameter λ seems to be critical in general, when no further information on the objectives are given. Hence, Problem (12.8) must be solved for several values of λ in $[0, 1]$ to guarantee a prescribed bound on the descent potential $\tilde{\pi}_y(x)$. Note that, analogous to the approach presented in Section 3.1 for constrained single objective optimization problem, Problem (12.8) can be seen as the Lagrangian relaxation of Problem (12.7).
As already mentioned above, if a non-zero descent potential is requested in the block of fixed variables for each subproblem in Algorithm 11.1, the generated sequence of points may never converge to a stationary point of Problem (12.1), as $\|\nabla f(x, y)\| > 0$ holds true at the end of each iteration. Furthermore, if, for fixed y, the points (x, y) correspond to points on the boundary of B for all $x \in B_y$, solving Problems (12.7) and (12.8) may implicitly imply that an additional optimization problem for each point $x \in B_y$ has to be solved, according to the definition of the descent potential (12.4) for points on the boundary, as long as no further information on g and B_y are given.

Hence, a mixed strategy involving steps that incorporate a non-zero descent potential and steps that disregard the information contained in the block of fixed variables can be used instead. While in the first iterations of Algorithm 12.1, higher values for $\tilde{\pi}_y$ and $\tilde{\pi}_x$ may be favorable to avoid a fast convergence to an undesired local minimum of the problem as well as generating points on the boundary of the feasible set, the descent potential should be totally disregarded after a certain number of iterations to guarantee the convergence of the algorithm. This can be done, for example, by applying appropriate decision rules in Lines 4 and 6 of Algorithm 12.1. In this case, the same convergence results as stated for Algorithm 11.1 in Section 11.3 can be achieved. Of course, further variants like applying the biobjective approach only to one of two subproblems in Algorithm 12.1 or using an additional annealing process for ε, when an ε-constraint approach is used to solve the biobjective problem, can be of interest.
To further decrease the additional CPU time that has to be spent to derive efficient solutions of the involved biobjective problems, the calculation of a complete set of representatives can be omitted, as in the end of an iteration we are interested in a single (efficient) solution only. Hence, the number of calculated non-dominated points of the biobjective subproblems could be restricted to only a small number of points.
However, since the presented enhanced alternate convex search strategy is a purely heuristic concept to improve the performance of Algorithm 11.1, no profound changes of the convergence results (like the guaranteed convergence to the global optimum of the problem) compared to the original algorithm can be expected in general.
Before we close this chapter on the theoretical background of the enhanced alternate convex search strategy with an illustrative example for the two-dimensional case, we once more remark that the biconvexity of f is not a necessary property for the validity of the ideas presented in this section. For a practical application of the proposed solution concept to a biconvex optimization problem, we refer to Chapter 13, where intensive numerical studies are presented for the connection location-allocation problem in the plane.

12.4 Illustrative Example

In this final section, we present an illustrative example of the enhanced alternate convex search strategy suggested in Section 12.3. We consider the two-dimensional, unconstrained optimization problem

$$\min_{x,y\in\mathbb{R}} f(x,y) = x^2y^2 + 2x^2y + 3x^2 + 3xy^2 - 2xy - x + 4y^2.$$

For the given problem we have that

$$\begin{aligned} f_y(x) &= (y^2 + 2y + 3)x^2 + (3y^2 - 2y - 1)x + 4y^2, \\ f_x(y) &= (x^2 + 3x + 4)y^2 + (2x^2 - 2x)y + 3x^2 - x. \end{aligned}$$

Since $y^2 + 2y + 3 > 0$ for all $y \in \mathbb{R}$ and $x^2 + 3x + 4 > 0$ for all $x \in \mathbb{R}$, f is a biconvex function. However, since the determinant of the Hessian H of f is given by

$$\det(H(x,y)) = -12x^2y^2 - 36y^2x - 20y^2 - 24yx^2 - 8yx + 56y - 4x^2 + 52x + 44,$$

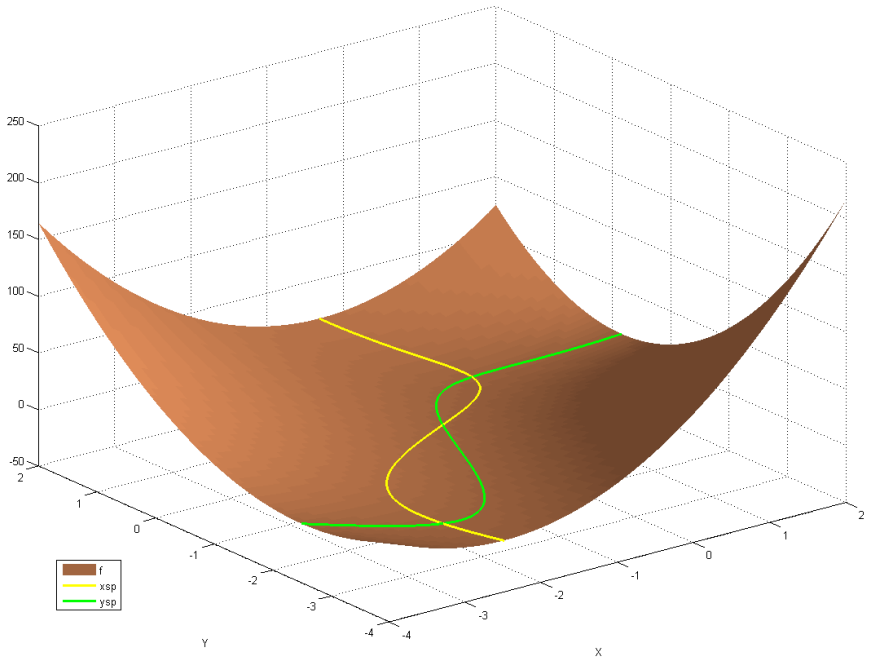
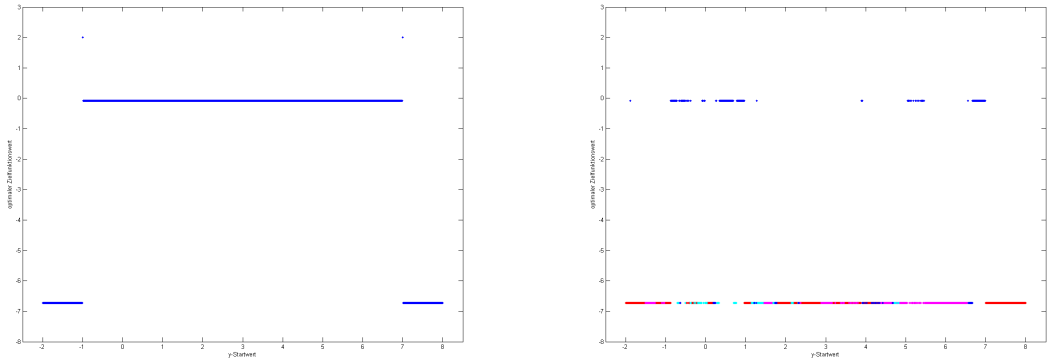


Figure 12.1: *Trajectories of the local minima of f_y* (yellow) *and f_x* (green)*, respectively. The intersection points of the trajectories correspond to the stationary points of the given problem.*

we have that $\det(H(0,4)) = -52$. Hence, f is non-convex in general. Solving the involved subproblems $\min\{f_y(x) : x \in \mathbb{R}\}$ and $\min\{f_x(y) : y \in \mathbb{R}\}$, respectively, show that the trajectories of the local minima are given by

$$x_{\text{sp}}(y) = \left(\frac{-3y^2 + 2y + 1}{2y^2 + 4y + 6}, y\right)^\top \quad \text{and} \quad y_{\text{sp}}(x) = \left(x, \frac{-x^2 + x}{x^2 + 3x + 4}\right)^\top,$$

for fixed $y \in \mathbb{R}$ and $x \in \mathbb{R}$, respectively. The two trajectories are depicted in Figure 12.1, where the yellow curve corresponds to the trajectory x_{sp}, while the green one belongs to y_{sp}. The trajectories intersect in the points $z^0 = (x^0, y^0) = (-1, -1)$, $z^1 = (x^1, y^1) = (0.17, 0.03)$ and $z^2 = (x^2, y^2) = (-3.15, -2.65)$ that correspond to the stationary points of f. Further calculations show that z^0 with $f(z^0) = 2$ corresponds to a saddle point of f, while z^1 with $f(z^1) = -0.88$ and z^2 with $f(z^2) = -6.73$ represent a local and the global minimum of the given problem, respectively.
When Algorithm 11.1 is applied to the two different starting points shown in Figure 12.3a, we see that for the point where $y < -1$ (blue lines), the global maximum of the problem is reached, while the sequence of generated points only converge to the local maximum for the starting point satisfying $y > -1$ (white lines). A detailed analysis of the resulting minima, depending on the initial value of y can be found in Figure 12.2a. Note that since the first subproblem is solved for fixed $y \in \mathbb{R}$, the final point that is calculated by Algorithm 11.1 does not depend on the choice of $x \in \mathbb{R}$.
It can be seen that for $-1 < y < 7$, the local minimum of f is obtained, while only for the choice $y \in \{-1, 7\}$ the saddle point of f is reached. For the remaining values



(a) *Evaluation of the obtained objective values by applying Algorithm 11.1 to different initial values of $y \in \mathbb{R}$.*

(b) *Evaluation of the minimum objective values obtained for different setups of Algorithm 12.1.*

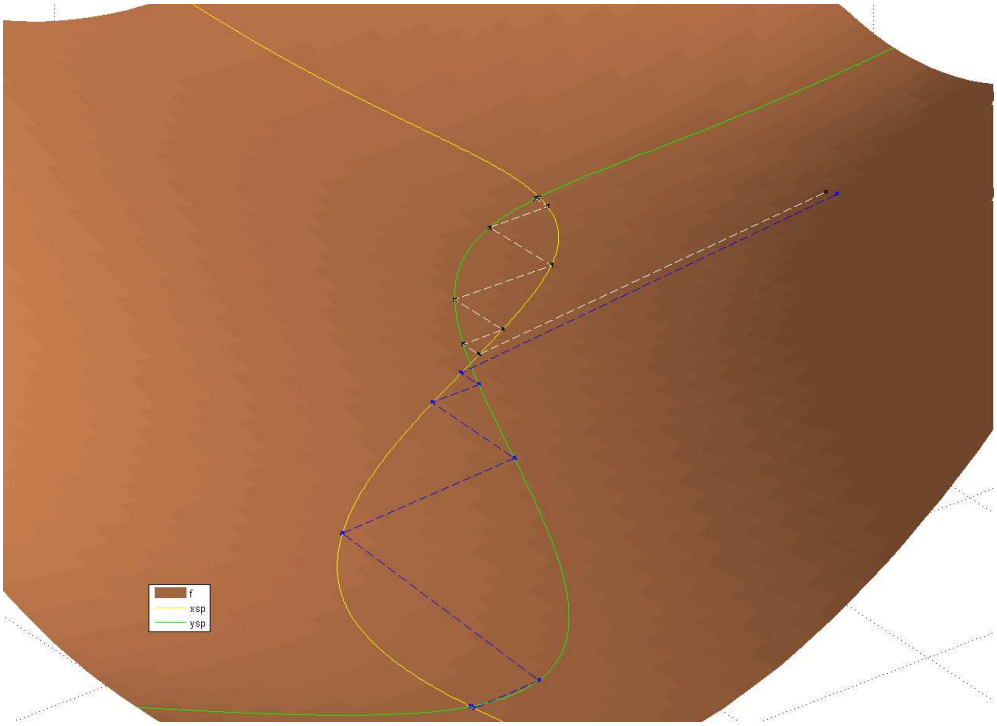
Figure 12.2: *Numerical comparison of Algorithm 11.1 and Algorithm 12.1, depending on the chosen starting value of $y \in \mathbb{R}$.*

of y, Algorithm 11.1 already converges to the global minimum of the given problem. An improved result can be obtained, when the augmented alternate convex search algorithm (cf. Algorithm 12.1) is applied. In Figure 12.3b, the ε-constraint approach is used to enforce a minimal descent potential of $\tilde{\pi}_y = 9$ (blue line) and $\tilde{\pi}_y = 10$ (white line) for the first iteration of Algorithm 12.1. Note that the red line results from applying the original Algorithm 11.1. While for the choice $\tilde{\pi}_y = 9$, still the local minimum z^1 is reached, Algorithm 12.1 converges to the global minimum z^2 if $\tilde{\pi}_y = 10$ is chosen.
A final evaluation of different setups for Algorithm 12.1 can be found in Figure 12.2b. Depending on the initial value of $y \in \mathbb{R}$, the overall minimum objective value is depicted that is obtained by solving four different setups of Algorithm 12.1. These setups enclose the cases that a minimal descent potential of $\varepsilon = 10$ or $\varepsilon = 15$ has to be satisfied in the first iteration or in the first two iterations of Algorithm 12.1. The obtained results clearly show an improved convergence rate to the global minimum of the given problem, when Algorithm 12.1 is applied. However, there still exist initial values of y that do not imply a convergence to the global optimum of the given problem.

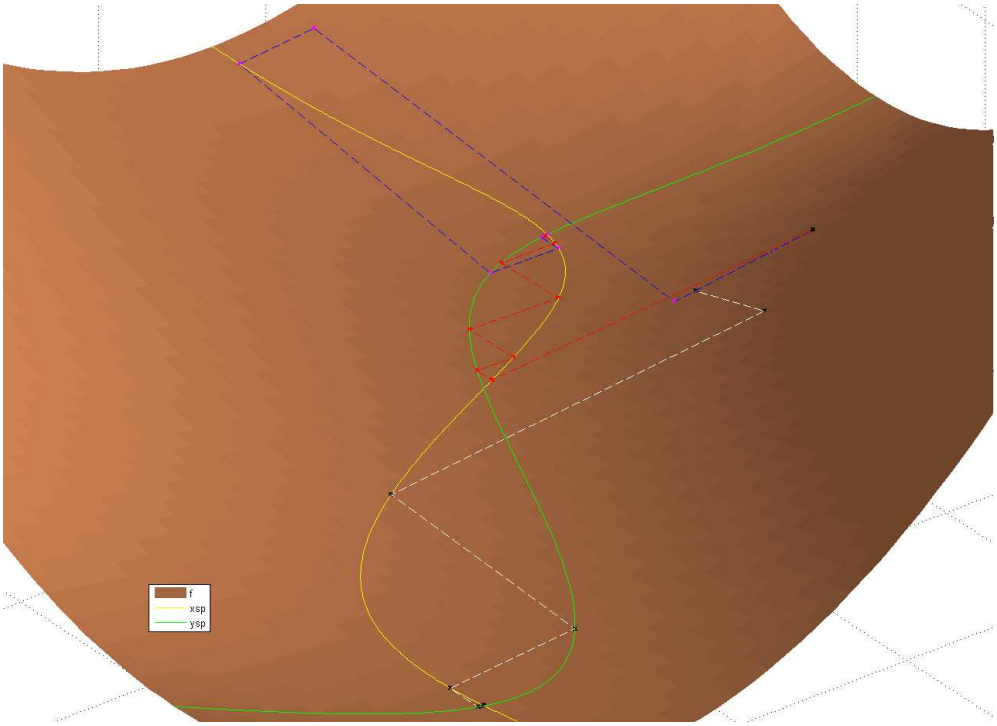
12.5 Conclusions and Further Ideas

In this chapter we discussed an enhanced version of the alternate convex search technique for biconvex optimization problems that makes use of a multiple objective approach to the problem. Based on the ideas presented in Section 3.3 for the alternate block search method, we derived an augmented alternate convex search algorithm that additionally incorporates the idea of measuring the descent potential with respect to the block of fixed variables during the iterations of the original method. While gradient information is used for the unconstrained case, additional optimization problems may have to be considered, when the feasible set of the convex subproblem is compact. As instead of a single objective problem a multiple objective problem has to be solved in each iteration of the presented enhanced approach, we further discussed

how combinations of both methods could be used to decrease the additional CPU time that has to be spent to solve the involved biobjective subproblems. We further gave an illustrative example for the enhanced approach in the $\mathbb{R}^2$-plane.
As for the constrained case, the presented approach implies that an additional optimization problem has to be solved for non-interior points of the feasible set, appropriate approximations of the descent potential for such points could be in the focus of further research. Due to the linearity of the involved objective that has to be minimized (cf. Problem (12.3)), it may be possible to derive lower bounds on the potential objective improvement for such points (cf. also Chapter 13).
In addition, we restricted ourselves to optimization problems with only two blocks of variables in this chapter. However, the combination of the ideas on an enhanced approach for problems with more than two blocks from Section 3.3 with the definitions for the descent potential presented in this chapter can be used to derive an enhanced alternate block search strategy for an arbitrary number of blocks of variables. As a consequence this new method can be applied to derive local optima for any non-linear (unconstrained) single objective optimization problem. A practical application of this approach to a real world problem could be of further interest.



(a) *Algorithm 11.1 applied to two different starting points. While for the starting point with $y > -1$ (white lines), the algorithm converges only to the local minimum, the global one is obtained for the point satisfying $y < -1$ (blue lines).*



(b) *Algorithm 12.1 applied to the starting point $y > -1$. While for the choice $\varepsilon = 9$ (blue line) for the minimal potential descent in the first iteration of the algorithm, still the local minimum is obtained, the global one is reached for the choice $\varepsilon = 10$ (white line).*

Figure 12.3: *An illustrative comparison of Algorithm 11.1 and Algorithm 12.1*

Chapter 13

The Connection Location-Allocation Problem in the Plane

In this chapter we deal with a special problem from location theory, the so-called *connection location-allocation problem in the plane*. Given a finite set of existing facilities, we consider the problem where directed flows between different source and target facilities have to be routed through connection facilities. In addition, these connection facilities have to be located in the $\mathbb{R}^2$-plane such that the resulting total transportation costs are minimized. In practical applications, these flows may correspond to goods that have to pass a factory during the manufacturing process, or to people that have to pass a control point, for example when they want to enter a stadium or cross the border between two countries. For further real-world applications we refer amongst others to the article of Montreuil and Ratliff [146] and the book of Mattfield [130].

The connection location-allocation problem is related to a variety of other well-known location problems: the facility location-allocation problem, the p-median problem, the hub location problem and the round-trip location problem. See, amongst others, Wesolowski [216], Mirchandani and Francis [145], Alumur and Kara [6] as well as Chan and Francis [34] for a detailed discussion of the mentioned location problems. For an overview on location theory in general, we refer to Drezner and Hamacher [48] and Nickel and Puerto [153].

However, the connection location-allocation problem itself is treated rarely in the literature. Huang [106] introduced the problem in his dissertation. The author mainly focused on discrete location problems with capacitated and uncapacitated connections, respectively. In addition, also the continuous case was discussed shortly. Furthermore, different distance functions, amongst others also polyhedral gauges, were used to model the problem. The results of Huang on the (un)capacitated N-connection location problem have also been published in Huang et al. [107].

In the recent literature, the continuous problem was tackled by Bischoff [20] in his dissertation. Based on a generalized description of the given problem, the author discussed theoretical properties like, for example, cluster properties and showed the $\mathcal{NP}$-hardness of the problem. Furthermore, a variety of solution methods both exact and heuristic were presented, and extensions to problems with barriers and queues were discussed. Parts of the author's dissertation have been published in Bischoff and Bayer [21], where connection location-allocation problems with polyhedral gauge

distances were considered, in Bischoff and Dächert [22], where various meta-heuristic approaches to the problem were compared and in Bischoff and Klamroth [23], where two branch & bound methods for the connection location-allocation problems were presented.
In the following, we show that the connection location-allocation problem can be modeled as a biconvex optimization problem that often has a large number of local minima in practice. Due to the biconvex structure of the problem, an alternate block search strategy (cf. Section 3.3), i.e. the alternate convex search method (cf. Section 11.3) in particular, is suitable to derive local minima of the given problem. In Location Theory, this solution method is also known as *alternate location-allocation algorithm* (cf. Cooper [40]). Given a partition of the set of variables into location variables and allocation variables, a local minimum of the problem can be derived by alternately solving the given problem for fixed location and allocation variables, respectively.
Based on the original version of the alternate location-allocation algorithm for the connection location-allocation problem, we present modified versions of this algorithm that incorporate the idea of the enhanced alternate convex search strategy, developed in Chapter 12 of this work.
We further show numerically that the quality of the resulting local minima can be improved, when the potential improvement of the objective value in the block of fixed variables is taken into account as an additional criterion that has to be maximized during parts of the optimization process. We further provide numerical results comparing the quality of the local minima obtained by the common and the modified approach, as well as the time taken to find these solutions.
This chapter is organized as follows: In the first section, we present a detailed problem formulation. In Section 13.2 we show that the given location problem can be formulated as a biconvex optimization problem. We present our augmented location-allocation algorithm in Section 13.3, based on the ideas presented in Section 3.3 and Chapter 12, including numerical comparisons between the original and the enhanced solution approach. Note that the detailed numerical results of our study can be found in Appendix B of this work. We finally summarize our results and give a short outlook in Section 13.4.

13.1 Notation and Problem Formulation

Let a set $\mathcal{A} = \{a_1, \dots, a_L\}$ of existing facilities in the $\mathbb{R}^2$-plane be given, i.e. $a_l \in \mathbb{R}^2$ for all $l \in \mathcal{L}$, where $\mathcal{L}$ is the set of indices of these facilities. Furthermore, let a set $\mathcal{M}$ of M flows be given, where each flow $m \in \mathcal{M}$ is associated with a source facility $a_{s_m} \in \mathcal{A}$ and a target facility $a_{t_m} \in \mathcal{A}$ and an intensity $w_m > 0$. Let $\mathcal{X} = \{x_1, \dots, x_N\}$ denote the set of the N connection facilities given by their coordinates that have to be located in the $\mathbb{R}^2$-plane, i.e. $x_n \in \mathbb{R}^2$ for all $n \in \mathcal{N}$. In this case, $\mathcal{N}$ denotes the set of indices of these facilities.
We further assume that an arbitrary amount of flow can be routed through a connection facility, i.e. a limitation of the flow capacity for the new facilities is disregarded. Furthermore, let $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ denote a distance function that measures the distance between the existing and the connection facilities. Then, the *connection location-allocation problem in the plane* (ConLoc) is given by

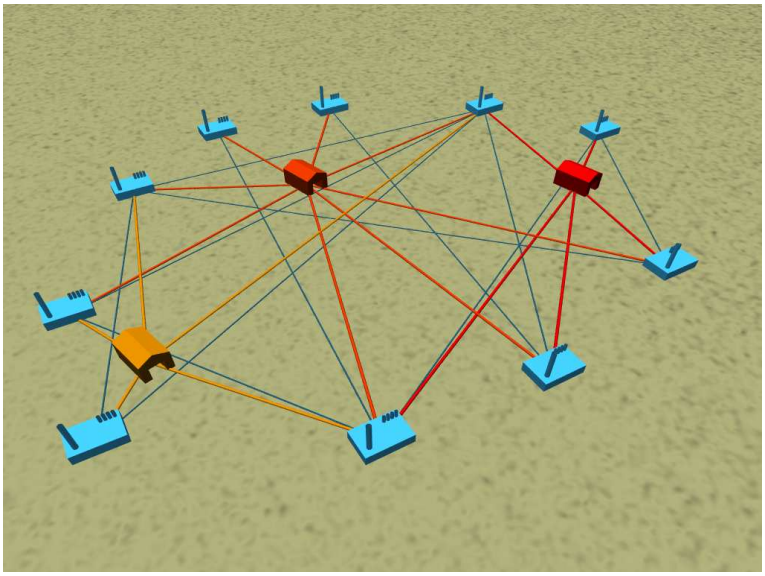


Figure 13.1: *An example for the connection location-allocation problem with* 10 *existing facilities (blue),* 10 *flows (blue) and* 3 *connection facilities (orange and red, respectively). Flows allocated to a specific connection facility have the same color as the corresponding facility.*

$$\begin{aligned} \min \sum_{m=1}^{M} \sum_{n=1}^{N} y_{mn} w_m \left(d(a_{s_m}, x_n) + d(x_n, a_{t_m}) \right) & \\ \text{s.t.} \sum_{n=1}^{N} y_{mn} = 1, \qquad & \forall\, m \in \mathcal{M} \\ y_{mn} \in \{0,1\}, & \forall\, m \in \mathcal{M}\ \forall\, n \in \mathcal{N} \\ x_n \in \mathbb{R}^2, \qquad & \forall\, n \in \mathcal{N}. \end{aligned} \tag{ConLoc}$$

In this case, the binary variable y_{mn} indicates, whether the flow m is allocated to the new facility x_n, i.e.

$$y_{mn} = \begin{cases} 1, \text{ if flow } m \text{ is assigned to connection facility } n, \\ 0, \text{ otherwise.} \end{cases}$$

The side constraint $\sum_{n=1}^{N} y_{mn} = 1$ in Problem (ConLoc) ensures, that each flow $m \in \mathcal{M}$ is assigned to a connection location exactly once. Hence, the set of all feasible assignments $Y \in \{0,1\}^{M \times N}$ is given by

$$\mathcal{Y} = \left\{ Y \in \{0,1\}^{M \times N} : \sum_{n=1}^{N} y_{mn} = 1\, \forall m \in \mathcal{M} \right\}.$$

An illustrative example of Problem (ConLoc) can be found in Figure 13.1. In Bischoff [20] it is shown that Problem (ConLoc) is $\mathcal{NP}$-hard to solve in general. However, due to the discrete structure of the decision variable $Y \in \mathcal{Y}$, the given optimization problem can theoretically be solved by a total enumeration of all feasible assignments contained in $\mathcal{Y}$. Unfortunately, the cardinality of $\mathcal{Y}$, i.e. the number of feasible assignments, is

bounded from below by the number of ways that a set of M elements can be partitioned into N non-empty subset (cf., e.g., Cooper [40] and Bischoff [20]). This number is also known as the *Stirling number of the second kind* and is given by

$$S(M,N) \;=\; \frac{1}{N!}\sum_{n=0}^{N}(-1)^n\binom{N}{n}(N-n)^M.$$

As $S(M,N)$ grows exponentially with increasing problem size, a total enumeration approach is limited to very small instances of the given problem. However, $S(M,N)$ provides an upper bound on the number of local optima of Problem (ConLoc) if it is assumed that for each given allocation vector the resulting location problem has a unique optimal solution. Otherwise, an infinite number of local optima to Problem (ConLoc) may exist.

In the remainder of this chapter, we consider *Euclidian distances* only. That is, for $a=(a_1,a_2)\in\mathcal{A}$ and $x=(x_1,x_2)\in\mathcal{X}$, let the distance function d in Problem (ConLoc) be given by

$$d_2(a,x) \;=\; \sqrt{(a_1-x_1)^2+(a_2-x_2)^2}.$$

Huang et al. [107] showed that in the case of Euclidian distances, any optimal solution of Problem (ConLoc) satisfies the nice property that all connection locations are located within the convex hull of the existing facilities.

If the flow $m\in\mathcal{M}$ is assigned to a connection facility located at $x\in\mathbb{R}^2$, it incurs the cost

$$c_m(x) = w_m\left(d_2(a_{s_m},x)+d_2(x,a_{t_m})\right). \tag{13.1}$$

If the considered connection facility corresponds to facility $n\in\mathbb{N}$, we refer to its special costs by $c_{mn}=c_m(x_n)$ in the following. Note that if m is assigned to n, c_{mn} forms one of the m non-zero summands in the objective function of Problem (ConLoc).

We further recall the definition of a cone in $\mathbb{R}^n$ (cf., e.g., Jahn [110] and Rockafellar [183]). Let $C\subseteq\mathbb{R}^n$ denote a non-empty subset of $\mathbb{R}^l$, where $n\in\mathbb{N}$. Then, the set C is called a *cone* if for all $x\in C$ and $\lambda\geq 0$, also $\lambda x\in C$ holds true. For $S\subseteq\mathbb{R}^n$,

$$\operatorname{cone}(S) \;=\; \{\lambda s:\ \lambda\geq 0 \text{ and } s\in S\}$$

is called the *cone generated* by S.

13.2 Biconvexity and the Location-Allocation Algorithm

In this section we show that the objective function

$$f(X,Y) \;=\; \sum_{m=1}^{M}\sum_{n=1}^{N} y_{mn}w_m\left(d_2(a_{s_m},x_n)+d_2(x_n,a_{t_m})\right) \;=\; \sum_{m=1}^{M}\sum_{n=1}^{N} y_{mn}c_m(x_n)$$

of Problem (ConLoc) is biconvex (cf. Chapter 11), i.e. that the two functions $f(\bar{X},Y)$ and $f(X,\bar{Y})$ are both convex for fixed $\bar{X}\in\mathbb{R}^2\times\mathbb{R}^N$ and fixed $\bar{Y}\in\{0,1\}^{M\times N}$, respectively. This implies that local search techniques like alternate convex search (cf. Section 11.3) as a special case of the more general alternate block search method (cf.

Section 3.3) can be applied to derive partial optima (cf. Definition 11.42) for the given problem. In this context, $(X^\star, Y^\star)$ corresponds to a partial optimum if $f(X^\star, Y^\star) \leq f(X, Y^\star)$ for all $X \in \mathbb{R}^2 \times \mathbb{R}^N$ and $f(X^\star, Y^\star) \leq f(X^\star, Y)$ for all $Y \in \{0,1\}^{M \times N}$. We mainly follow the ideas presented in Bischoff [20] in the following.

13.2.1 Fixed Location Variables

We start with the case that a fixed location $\bar{X} \in \mathbb{R}^2 \times \mathbb{R}^N$ for the N connection facilities is prescribed. Given the vector $\bar{X}$, the costs that would arise if flow $m \in \mathcal{M}$ was assigned to facility $n \in \mathcal{N}$ can easily be calculated by Equation (13.1) for all flows contained in $\mathcal{M}$. Let $\bar{c}_{mn}$ denote the resulting costs in the following. Using the costs $\bar{c}_{mn}$, Problem (ConLoc) simplifies to

$$\begin{aligned} \min \ & \sum_{m=1}^{M} \sum_{n=1}^{N} y_{mn} \bar{c}_{mn} \\ \text{s.t.} \ & \sum_{n=1}^{N} y_{mn} = 1, \qquad \forall\, m \in \mathcal{M} \\ & y_{mn} \in \{0,1\}, \, \forall\, m \in \mathcal{M} \, \forall\, n \in \mathcal{N}. \end{aligned} \tag{AP}$$

Since each flow m has to be assigned to exactly one connection facility (due to the constraint $\sum_{n=1}^{N} y_{mn} = 1$), we conclude that for fixed $m \in \mathcal{M}$ and $Y \in \mathcal{Y}$

$$\sum_{n=1}^{N} y_{mn} \bar{c}_{mn} \ \geq \ \min\{\bar{c}_{mn} : \, n \in \mathcal{N}\} \tag{13.2}$$

is satisfied. Moreover, equality in (13.2) is obtained, whenever $y_{mn^\star} = 1$, where $n^\star \in \mathcal{N}$ corresponds to the index such that $c_{mn^\star} = \min\{\bar{c}_{mn} : \, n \in \mathcal{N}\}$. As the objective function of Problem (AP) decomposes into M independent blocks of N non-negative summands of the same type as the left hand side of (13.2), the inequality can be applied to each block separately, i.e., we have that

$$\sum_{m=1}^{M} \sum_{n=1}^{N} y_{mn} \bar{c}_{mn} \ \geq \ \sum_{m=1}^{M} \min\{\bar{c}_{mn} : \, n \in \mathcal{N}\} \tag{13.3}$$

for all feasible assignments $Y \in \mathcal{Y}$. Hence, the right hand side value of (13.3) provides a lower bound on the optimal objective of Problem (AP). Considering $Y' \in \{0,1\}^{M \times N}$, where

$$y'_{mn^\star} = \begin{cases} 1, \text{ if } \bar{c}_{mn^\star} \leq \bar{c}_{mn} \, \forall n \in \mathcal{N} \quad \wedge \quad y'_{m1} = \ldots = y'_{m(n^\star - 1)} = 0, \\ 0, \text{ otherwise,} \end{cases} \tag{13.4}$$

for fixed $m \in \mathcal{M}$, further implies that

$$\sum_{m=1}^{M} \sum_{n=1}^{N} y'_{mn} \bar{c}_{mn} \ = \ \sum_{m=1}^{M} \min\{\bar{c}_{mn} : \, n \in \mathcal{N}\} \ \geq \ \min \left\{ \sum_{m=1}^{M} \sum_{n=1}^{N} y_{mn} \bar{c}_{mn}, \, Y \in \mathcal{Y} \right\}.$$

As in addition, Y' is feasible for Problem (AP), it follows that Y' provides an optimal solution of the considered assignment problem. Note that if the minimum costs for fixed $m \in \mathcal{M}$ are not unique in the definition of Y', the flow can be assigned to the facility with the minimum index n such that $\bar{c}_{mn}$ is optimal for m. We summarize the obtained results in the following theorem.

Theorem 13.1 ([20]) *Let a fixed location $\bar{X} \in \mathbb{R}^2 \times \mathbb{R}^N$ of the N connection facilities be given. Then, Problem* (ConLoc) *simplifies to the linear assignment problem* (AP) *with optimal solution Y' as defined in* (13.4)*. Its optimal objective value is given by the right hand side value of* (13.3)*.*

Moreover as shown in Bischoff [20], the optimal objective value of Problem (AP) additionally coincides with the optimal objective value of its linear relaxation. We further make use of this result in Section 13.3 of this chapter. Note that in this case, the binary constraints $y_{mn} \in \{0, 1\}$ are relaxed to $0 \leq y_{mn} \leq 1$ for all $m \in \mathcal{M}$ and $n \in \mathcal{N}$. Hence, Problem (AP) could be solved in fact by means of a standard LP-solver.

13.2.2 Fixed Allocation Variables

In the following we consider the case that the assignments of the M flows to the N connection facilities are given and fixed. For a source facility $i \in \mathcal{L}$, a target facility $j \in \mathcal{L}$ and a connection $n \in \mathcal{N}$ we define

$$\bar{w}_{ijn} := \begin{cases} w_m, \text{ if there exists } m \in \mathcal{M} : i = s_m,\ j = t_m,\ \bar{y}_{mn} = 1, \\ 0 \ , \text{ otherwise.} \end{cases}$$

Using this notation, Problem (ConLoc) can be rewritten as

$$\begin{aligned} \min \ & \sum_{n=1}^{N} \left(\sum_{i=1}^{L} \sum_{j=1}^{L} \bar{w}_{ijn} \left(d_2(a_i, x_n) + d_2(x_n, a_j) \right) \right) \\ \text{s.t. } & x_n \in \mathbb{R}^2, \, \forall \, n \in \mathcal{N}. \end{aligned} \tag{13.5}$$

Since each summand of the objective function only depends on a fixed facility $n \in \mathbb{N}$, Problem (13.5) decomposes into a sequence of N independent unrestricted single-facility location problems. In this case, each connection facility $n \in \mathcal{N}$ has to be located such that the sum of costs associated to this facility is minimized. Before we discuss how such problems can be solved efficiently in practice, we further conclude by the symmetry of the Euclidian distance that for fixed facility $n \in \mathbb{N}$ we have that

$$\begin{aligned} \sum_{i=1}^{L} \sum_{j=1}^{L} \bar{w}_{ijn} (d_2(a_i, x_n) + d_2(x_n, a_j)) &= \sum_{i=1}^{L} \sum_{j=1}^{L} \bar{w}_{ijn} d_2(a_i, x_n) + \sum_{i=1}^{L} \sum_{j=1}^{L} \bar{w}_{ijn} d_2(a_j, x_n) \\ &= \sum_{i=1}^{L} \sum_{j=1}^{L} \bar{w}_{ijn} d_2(a_i, x_n) + \sum_{j=1}^{L} \sum_{i=1}^{L} \bar{w}_{jin} d_2(a_i, x_n) \\ &= \sum_{i=1}^{L} \left(\sum_{j=1}^{L} (\bar{w}_{ijn} + \bar{w}_{jin}) \right) d_2(a_i, x_n) \\ &= \sum_{i=1}^{L} \bar{w}_{in} d_2(a_i, x_n), \end{aligned}$$

where $\bar{w}_{in} = \sum_{j=1}^{L} (\bar{w}_{ijn} + \bar{w}_{jin})$ equals the sum of flows associated with the source or target facility $i \in \mathcal{L}$. Hence, we have that

$$f(X, \bar{Y}) \ = \ \sum_{n=1}^{N} \sum_{i=1}^{L} \bar{w}_{in} d_2(a_i, x_n).$$

Algorithm 13.1 Weiszfeld Algorithm for the Weber Problem in $\mathbb{R}^2$

Input: Existing facilities $a_1, \ldots, a_l$ and corresponding positive weights $w_1, \ldots, w_l$.
Output: Approximate optimal location of the new facility x.

1: **for** $l = 1$ **to** L **do**
2: **if** Facility a_l satisfies Inequality (13.8) **then**
3: **return** a_l.
4: **end if**
5: **end for**
6: Choose a starting solution $x^0 \in \mathbb{R}^2$.
7: Set the iteration counter $i = 0$.
8: **while** no stopping criterion is satisfied **do**
9: Set $i = i + 1$.
10: Set $x_1^i = \left(\sum_{l=1}^{L} w_l \frac{a_{l1}}{d_2(a_l, x^{i-1})}\right)\left(\sum_{l=1}^{L} w_l \frac{1}{d_2(a_l, x^{i-1})}\right)^{-1}$.
11: Set $x_2^i = \left(\sum_{l=1}^{L} w_l \frac{a_{l2}}{d_2(a_l, x^{i-1})}\right)\left(\sum_{l=1}^{L} w_l \frac{1}{d_2(a_l, x^{i-1})}\right)^{-1}$.
12: **end while**
13: **return** x^i.

Since the Euclidian distance function d_2 is convex, $f(\cdot, \bar{Y}) : \mathbb{R}^2 \times \mathbb{R}^N \to \mathbb{R}$ is the sum of convex functions for each fixed assignment vector $\bar{Y} \in \mathcal{Y}$ and thus convex itself. Furthermore, we conclude that

$$\min f(X, \bar{Y}) \;=\; \sum_{n=1}^{N} \min_{x_n \in \mathbb{R}^2} \left\{ \sum_{i=1}^{L} \bar{w}_{in} d_2(a_i, x_n) \right\}. \tag{13.6}$$

We have proven:

Theorem 13.2 ([20]) *Let a fixed allocation vector $\bar{Y} \in \mathcal{Y}$ be given. Then, the given Problem* (ConLoc) *simplifies to a sequence of N independent and convex single-facility problems, where an optimal solution is given by the right hand side of* (13.6).

In the remainder of this subsection, we shortly discuss how single-facility location problems of the general form

$$\min_{x \in \mathbb{R}^2} g(x) \;:=\; \sum_{l=1}^{L} w_l d_2(a_l, x), \tag{13.7}$$

can be solved, assuming that $w_l > 0$ holds true for all $l \in \{1, \ldots, L\}$. Note that Problem (13.7) is a well-known problem in location theory. It is also denoted as *Fermat-Weber problem* and was introduced by Weber [212] in 1909. For a historical review on Weber problems, we refer to Wesolowski [216].
Assuming that $x = (x_1, x_2) \in \mathbb{R}^2$ and $a = (a_{l1}, a_{l2}) \in \mathbb{R}^2$ for all $l \in \{1, \ldots, L\}$, the partial derivatives of g are given by

$$\frac{\partial g(x)}{\partial x_i} \;=\; \sum_{l=1}^{L} w_l \cdot \frac{x_i - a_{li}}{d_2(a_l, x)}, \qquad i \in \{1, 2\},$$

Algorithm 13.2 Alternate Location-Allocation Algorithm for Problem (ConLoc)

Input: An instance of Problem (ConLoc).

Output: Location X of the connection facilities and corresponding assignment vector Y.

1: Choose initial assignment vector $\bar{Y} \in \mathcal{Y}$.
2: **while** no stopping criterion is satisfied **do**
3: **Location step for fixed allocation vector $\bar{Y}$:**
Solve the N single-facility location problems for the fixed allocation vector $\bar{Y}$ by applying the Weiszfeld algorithm 13.1 to generate the optimal locations X.
4: **Allocation step for fixed locations $\bar{X}$:**
Solve the Assignment Problem (AP) for the fixed locations $\bar{X}$ to generate the optimal assignment vector Y.
5: **end while**
6: **return** Locations X with corresponding assignment vector Y.

whenever $x \neq a_l$ for all $l \in \{1, \ldots, L\}$. Since the necessary condition for a local minimum $\frac{\partial g(x)}{\partial x_i} = 0$ for $i = 1, 2$ cannot be solved for x_i, a fix-point iteration method is typically applied to approximate the optimal solution of the convex problem with arbitrary exactness. This method is also known as *Weiszfeld algorithm* in the literature and was originally published by Weiszfeld [214] in 1937. Independently, also Cooper [40], Kuhn and Kuenne [120] as well as Miehle [142] published this solution approach 20 years later. A short outline of the algorithm can be found in Algorithm 13.1.
As the Weber function g is not differentiable in any point $a_l \in \mathbb{R}^2$, $l = 1, \ldots, L$, the existing facilities have to be checked for optimality in addition. Kuhn [119] proved that the location of the new facility x coincides with the location of an existing facility a_j, $j \in \{1, \ldots, L\}$, if and only if

$$\left(\left(\sum_{\substack{l=1 \\ l \neq j}}^{L} w_l \cdot \frac{a_{j1} - a_{l1}}{d_2(a_l, a_j)} \right)^2 + \left(\sum_{\substack{l=1 \\ l \neq j}}^{L} w_l \cdot \frac{a_{j2} - a_{l2}}{d_2(a_l, a_j)} \right)^2 \right)^{\frac{1}{2}} \leq w_j \tag{13.8}$$

holds true. If this is not the case, a starting solution x^0 is chosen and the while-loop of Algorithm 13.1 is performed, until a prescribed stopping criterion is satisfied. As the unique optimal solution of the Weber problem for the squared Euclidian distance is usually relatively close to the optimal solution of Problem (13.7), this location is frequently used as initial solution of Algorithm 13.1. In this case, x^0 is given as the center of gravity of the weighted locations of the existing facilities (cf., e.g., Drezner et al. [49]). Concerning an appropriate choice for the stopping criterion of Algorithm 13.1, a maximum number of iterations, a minimum change in the location of the new facility ($\|x^i - x^{i-1}\| \leq \delta$) or a minimum decrease in the objective value ($|f(x^i) - f(x^{i-1})| \leq \varepsilon$) are used in general. For a survey on acceleration techniques for the Weiszfeld algorithm, we refer to Bischoff [20].

13.2.3 The Location-Allocation Algorithm

Summarizing the results discussed in the last two subsections, we can conclude that the objective function f of Problem (ConLoc) is indeed biconvex.

Algorithm 13.3 Multi-Start Version of Algorithm 13.2

Input: An instance of Problem (ConLoc) and a prescribed number I of iterations.

Output: Best local minimum $(\hat{X}, \hat{Y}, f(\hat{X}, \hat{Y}))$ found for the prescribed number of iterations.

1: Set $i = 1$, $\hat{f} = \infty$ and initialize $\hat{X} \in \mathbb{R}^{2\times N}$ and $\hat{Y} \in \mathcal{Y}$.
2: **while** $i \leq I$ **do**
3: Apply Algorithm 13.2 to determine a local optimal solution (X, Y).
4: **if** $f(X, Y) < \hat{f}$ **then**
5: Set $(\hat{X}, \hat{Y}, \hat{f}) = (X, Y, f(X, Y))$.
6: **end if**
7: Set $i = i + 1$.
8: **end while**
9: **return** $(\hat{X}, \hat{Y}, \hat{f})$.

Corollary 13.3 *The connection location-allocation problem in the plane can be formulated as a biconvex optimization problem given by Problem* (ConLoc)*. For fixed locations, the problem simplifies to a simple assignment problem, while for fixed allocation variables, the problem reduces to a sum of N single-facility location problems in the plane.*

Due to the biconvex structure of the problem, the alternate convex search strategy introduced in Section 11.3 can be used to generate local optima of the given problem in an efficient amount of time. This strategy can be seen as a special case of the more general alternate block search technique, already described in Section 3.3. In the context of location theory, this method is also called *alternate location-allocation algorithm* in the literature and was initially introduced by Cooper [40] for the facility location-allocation problem with Euclidian distances. A short outline of the algorithm can be found in Algorithm 13.2.

Starting from a randomly generated allocation vector $\bar{Y} \in \mathcal{Y}$, the location and the allocation problems are iteratively solved for fixed $\bar{Y}$ and $\bar{X}$, respectively, until a stopping criterion is satisfied. This is for example the case, when the allocation vector of the last iteration coincides with the vector of the current iteration, since in this case no further improvement of the objective can be expected in further iterations. Nevertheless, especially for larger problem instances a maximum number of iterations can be prescribed, and the algorithm terminates, when the maximum number is exceeded. However, instead of initializing the algorithm with a fixed allocation vector $\bar{Y}$, also randomly generated locations $\bar{X}$ of the connection facilities can be used. In this case, the initial locations should preferably be chosen within the convex hull of the existing facilities, as an optimal solution of Problem (ConLoc) also satisfies this property (cf. Huang et al. [107]). Furthermore, the order of the location and the allocation step has to be reversed in Algorithm 13.2.

Since the objective value of Problem (ConLoc) is improved in each iteration of the algorithm, and the objective function f is bounded from below, the generated sequence of solutions is guaranteed to converge to a partial optimal solution, i.e. a local minimum of the problem (cf. also Chapter 11). However, the quality of the calculated local minimum crucially depends on the initial assignment vector $\bar{Y}$ or the given locations $\bar{X}$. Hence, the alternate location-allocation algorithm can only be seen as a local search

strategy to solve the given problem. To improve the overall performance, a multi-start version of Algorithm 13.2 can be applied to derive alternative local optima. The algorithm returns the best solution found after a prescribed number of runs of the original location-allocation algorithm. A short outline of this approach can be found in Algorithm 13.3.
We further remark that Algorithm 13.2 as well as Algorithm 13.3 were used by Bischoff and Dächert [22] as subroutines for heuristic approaches to solve Problem (ConLoc). For other solution approaches to solve Problem (ConLoc), like exact branch & bound techniques, we refer to Bischoff [20].

13.3 The Augmented Location-Allocation Algorithm

The location-allocation algorithm (cf. Algorithm 13.2) described in the last section yields an efficient and fast method to find a local optimum of Problem (ConLoc). However, due to the large number of potential local minima of the given problem, the quality of the solution that is found by the algorithm strongly depends on the initial starting solution that is normally chosen at random. Due to the biconvex structure of the problem, the enhanced alternate convex search strategy suggested in Chapter 12 can be applied to heuristically derive better local optima for the given location problem. Based on the ideas presented in Chapter 12, we adapt the theoretical background of the enhanced solution approach to the given problem from location theory in the following. Furthermore, we provide detailed numerical studies comparing the enhanced search concept to the original version of the location-allocation algorithm.
From Section 3.3 we recall that our enhanced solution concept is based on the idea to additionally exploit descent information that is contained in the block of fixed variables and that is normally disregarded when an alternate block search strategy like Algorithm 13.2 is applied to solve the given optimization problem. We have shown in Chapter 12 that for unconstrained subproblems, the norm of the gradient restricted to the block of fixed variables can be used to represent this descent information, also called the *descent potential* (cf. Definition 12.1).
While the original method tends to converge to a local minimum of the problem that is located in a direct neighborhood of the starting solution, the enhanced version of the method is capable to avoid a fast convergence to this local minimum, but heuristically exploit local descent information to determine a potentially better local optimum of the given problem. However, due to the large number of local minima and the heuristic nature of the enhanced approach, a convergence to the global optimum of a given problem instance cannot be guaranteed.
Contrary to the notation used in the previous sections, we write X instead of $\bar{X}$ and Y instead of $\bar{Y}$ in the remainder of this section, whenever the location and allocation variables are assumed to be fixed, respectively. However, it is always clear from the context, whether X or Y are fixed or not.
The remainder of this section is organized as follows: In the following two subsections, we present adapted versions of the definition of the descent potential given in Section 12.2 that incorporate the special structure of the connection location-allocation problem. Based on modified definitions of the descent potential for a fixed location

vector X and allocation vector Y, we derive three different enhanced variants of the original location-allocation algorithm (i.e. Algorithm 13.2) in Subsection 13.2. In the final subsection, we present detailed numerical comparisons between the suggested enhanced solution methods and (the multi-start version of) Algorithm 13.2.

13.3.1 The Descent Potential for Fixed Location Variables

We start with the definition of the descent potential π_X for a fixed given location vector $X \in \mathbb{R}^2 \times \mathbb{R}^N$. Unlike the set $\mathcal{Y}$ of feasible allocations that is contained in a compact subset of $\mathbb{R}^M \times \mathbb{R}^N$ by definition of Problem (ConLoc), the locations of the N connection facilities are unrestricted. However, as the Euclidian distance is used to measure the distance between the connection facilities contained in $\mathcal{X}$ and the existing facilities from $\mathcal{A}$, the objective function f of Problem (ConLoc) is not differentiable in the locations of the existing facilities from $\mathcal{A}$. We take care of this fact in the following. For $X \in \mathbb{R}^2 \times \mathbb{R}^N$ let $n \in \mathcal{N}$ such that $x_n = (x_{n1}, x_{n2}) \neq a_l$ for all $l \in \mathcal{L}$. Furthermore, let $m \in \mathcal{M}$. We use Equation (13.1) to state for $i \in \{1, 2\}$ that

$$\frac{\partial c_m(X)}{\partial x_{ni}} \;=\; w_m \left(\frac{x_{ni} - a_{s_m i}}{d_2(a_{s_m}, x_n)} + \frac{x_{ni} - a_{t_m i}}{d_2(x_n, a_{t_m})} \right).$$

This implies that

$$\frac{\partial f(X,Y)}{\partial x_{ni}} = \sum_{m=1}^{M} y_{mn} w_m \left(\frac{x_{ni} - a_{s_m i}}{d_2(a_{s_m}, x_n)} + \frac{x_{ni} - a_{t_m i}}{d_2(x_n, a_{t_m})} \right),$$

whenever $x_n \neq a_l$ for all $l \in \mathcal{L}$. Otherwise, f is not differentiable with respect to x_n. Hence, Definition 12.1 of the descent potential a for fixed location vector X cannot be directly applied here. However, since f is continuously differentiable in all points $X \in \mathbb{R}^2 \times \mathbb{R}^N$ where none of the components of X coincide with a location of an existing facility, we disregard these special components of the vector X in the definition of the descent potential π_X in the following. To simplify the notation, we set

$$\rho_{ni}(X,Y) \;=\; \begin{cases} \dfrac{\partial f(X,Y)}{\partial x_{ni}}, & \text{if } a_{s_m} \neq x_n \neq a_{t_m} \quad \forall m \in \mathcal{M}, \\ 0 \quad , & \text{otherwise,} \end{cases} \tag{13.9}$$

and define:

Definition 13.4 *Let a feasible solution (X, Y) of Problem* (ConLoc) *be given. Then, the descent potential π_X of f with respect to the location vector X is given by*

$$\pi_X(X,Y) \;=\; \|\nabla_X f(X,Y)\| \;=\; \left(\sum_{n=1}^{N} \sum_{i=1}^{2} \left(\rho_{ni}(X,Y)\right)^2 \right)^{\frac{1}{2}},$$

where $\rho_{ni}(X, Y)$ is defined as in (13.9).

Although the definition of the descent potential with respect to the location vector X is restricted to the components of X that do not coincide with the location of an existing facility, we will see in the numerical studies presented in Subsection 13.3.4 that Definition 13.4 is an appropriate definition of the descent potential π_X, especially for larger problem instances.

13.3.2 The Descent Potential for Fixed Allocation Variables

In this subsection, we discuss two different approaches how the descent potential π_Y of f with respect to a fixed allocation vector Y can be defined. Since the coefficients of Y are binary, a definition of the descent potential based on gradient information does not seem to be an appropriate choice for this problem at first sight. However, since there exists a solution of the linear programming relaxation of the problem that is also optimal for the discrete problem (cf. Bischoff [20]), a gradient-based definition of the descent potential π_Y can be derived, at least when relaxed binary constraints are considered.
We present two different types of definitions in the following: One that is based on the relaxation of the M individual allocation vectors to the complete interval $[0,1]$ in each component, and another that is based on the discrete structure of the given subproblem.
Let an instance of Problem (ConLoc) be given, and let $m \in \mathcal{M}$ and $n \in \mathcal{N}$. Then, the partial derivative of f with respect to the continuous allocation component y_{mn}, is given by

$$\frac{\partial f(X,Y)}{\partial y_{mn}} \;=\; w_m \left(d_2(a_{s_m}, x_n) + d_2(x_n, a_{t_m})\right) \;=\; c_m(x_n) \;>\; 0. \tag{13.10}$$

Hence, for a given allocation vector $Y \in \mathcal{Y}$ the objective function of the connection location-allocation Problem can be rewritten as

$$f(X,Y) \;=\; \langle \nabla_Y f(X,Y), Y \rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes the standard scalar product in $\mathbb{R}^{M \times N}$. In addition to the discrete definition of the feasible set $\mathcal{Y}$ of all allocation vectors, let

$$\tilde{\mathcal{Y}} \;=\; \left\{ Y \in [0,1]^{M \times N} : \sum_{n=1}^{N} y_{mn} = 1 \quad \forall m \in \mathcal{M} \right\}$$

denote the linear programming relaxation of $\mathcal{Y}$ with respect to the binary variables. For $m \in \mathcal{M}$, we further define $\tilde{\mathcal{Y}}_m = \left\{ y \in [0,1]^N : \sum_{n=1}^{N} y_{mn} = 1 \right\}$ and $\mathcal{Y}_m = \tilde{\mathcal{Y}}_m \cap \{0,1\}^N$. Note that the sets $\tilde{\mathcal{Y}}$ as well as $\tilde{\mathcal{Y}}_1, \ldots, \tilde{\mathcal{Y}}_M$ are convex, polyhedral sets by definition. Hence, we can use Minkowski's theorem (cf., e.g. Nemhauser and Wolsey [150]) to further conclude that

$$\tilde{\mathcal{Y}}_m = \left\{ y \in \mathbb{R}^N : y = \sum_{n=1}^{N} \lambda_n e_n,\ \lambda_n \geq 0 \ \ \forall n \in \mathcal{N},\ \sum_{n=1}^{N} \lambda_n = 1 \right\},$$

where $e_1, \ldots, e_N$ denote the N different unit vectors of $\mathbb{R}^N$. Due to the definition of $\tilde{\mathcal{Y}}$, we have that $\tilde{\mathcal{Y}} = \tilde{\mathcal{Y}}_1 \times \ldots \times \tilde{\mathcal{Y}}_M$. Unlike Subsection 13.3.1, where the locations of the connection facilities do not have to satisfy further restrictions, a feasible allocation vector to Problem (ConLoc) must be contained in the set $\tilde{\mathcal{Y}}$ that corresponds to a compact subset of $\mathbb{R}^M \times \mathbb{R}^N$. Hence, Equation (12.3) rather than Definition 12.1 has to be used to define the descent potential π_Y based on the relaxed set $\tilde{\mathcal{Y}}$.

Definition 13.5 *Let $Y \in \mathcal{Y}$ be arbitrary, but fixed. Then, the vector $V \in \mathbb{R}^{M\times N}$ is called a* feasible direction *in Y if there exists $\bar{\lambda}$ such that $Y + \lambda V \in \tilde{\mathcal{Y}}$ holds true for all $\lambda \in [0, \bar{\lambda}]$.*

By construction, $V \in \mathbb{R}^{M\times N}$ is a feasible direction in Y if and only if $y_m + \lambda v_m$ is an element of $\tilde{\mathcal{Y}}_m$ for all $\lambda \in [0, \bar{\lambda}]$ and $m \in \mathcal{M}$, where $y_m \in \mathcal{Y}_m \subseteq \{0,1\}^N$ and $v_m \in \mathbb{R}^N$. Given a feasible allocation vector $Y \in \mathcal{Y}$, a further improvement with respect to f can only be realized, when V corresponds to a feasible descent direction in Y, since otherwise the potential of an infeasible solution would be considered. Due to the results from Section 12.2, the optimal descent direction for fixed Y is given by $V = -\nabla_Y f(X, Y)$ if an unconstrained problem is considered. Unfortunately, we have that:

Lemma 13.6 *Let $Y \in \mathcal{Y}$. Then, $V = -\nabla_Y f(X, Y)$ does not correspond to a feasible direction in Y.*

Proof: For arbitrary $\lambda > 0$, let $\bar{Y} = Y - \lambda \nabla_Y f(X, Y)$. For arbitrary but fixed $m \in \mathcal{M}$, we have that

$$\sum_{n=1}^{N} \bar{y}_{mn} = \sum_{n=1}^{N} y_{mn} - \lambda \sum_{n=1}^{N} c_m(x_n) < \sum_{n=1}^{N} y_{mn} = 1,$$

since $c_m(x_n) > 0$ for all $n \in \mathcal{N}$. Hence, $\bar{Y} \notin \tilde{\mathcal{Y}}$ for all $\lambda > 0$. □

Due to Lemma 13.6, the negative gradient of f with respect to Y does not yield a feasible direction for any $Y \in \mathcal{Y}$. This shows that Definition 12.1 cannot be used to define the descent potential π_Y of f with respect to Y, and we have to use Equation (12.3) instead. Unfortunately, this implies that an additional optimization problem has to be solved for all $X \in \mathbb{R}^2 \times \mathbb{R}^N$ to derive an appropriate value of π_Y. However, we show in the following that due to the linear structure of Problem (ConLoc) with respect to the allocation variables, the optimal solution of the optimization problem that is induced by Equation (12.3) can be given explicitly.
Let $e_1, \ldots, e_N$ denote the N different unit vectors of $\mathbb{R}^N$, and let $m \in \mathcal{M}$ be arbitrary, but fixed. By definition of $Y \in \mathcal{Y}$, there exists $j \in \mathcal{N}$ such that $y_{mj} = 1$, while $y_{mi} = 0$ for all $i \in I_j := \mathcal{N} \setminus \{j\}$. Let this index be denoted by j_m in the following. Using this notation, we conclude that $y_m = e_{j_m}$ holds true. For $m \in \mathcal{M}$ we further define

$$S_{j_m} = (\text{conv}\{e_1 - e_{j_m}, \ldots, e_{j_m-1} - e_{j_m}, e_{j_m+1} - e_{j_m}, \ldots, e_N - e_{j_m}\})$$
$$\text{and } \mathcal{C}_m(e_{j_m}) = \text{cone}(S_{j_m}),$$

where $\text{conv}\{A\}$ denotes the convex hull of a set $A \subseteq \mathbb{R}^N$. By construction, $e_{j_m} + S_{j_m}$ corresponds to the facet of $\tilde{\mathcal{Y}}_m$ that faces the point $e_{j_m} \in \mathbb{R}^N$. We show:

Theorem 13.7 *Let $Y \in \mathcal{Y}$ and let $V \in \mathbb{R}^{M\times N}$. Then, V is a feasible direction with respect to Y if and only if $v_m \in \mathcal{C}_m(e_{j_m})$ holds true for all $m \in \mathcal{M}$.*

Proof: Let $V \in \mathbb{R}^{M\times N}$ be a feasible direction with respect to Y. Hence, there exists $\mu > 0$ such that the point $Y + \mu V \in \tilde{\mathcal{Y}}$, i.e. $y_m + \mu v_m \in \tilde{\mathcal{Y}}_m$ for all $m \in \mathcal{M}$. Using Minkowski's theorem, we conclude that there exists $\lambda \in \mathbb{R}^N$, $\lambda_n \geq 0$ for all $n \in \mathcal{N}$,

$\sum_{n=1}^{N} \lambda_n = 1$, such that $y_m + \mu v_m = \sum_{n=1}^{N} \lambda_n e_n$ for all $m \in \mathcal{M}$. Hence, we have that

$$\begin{aligned} v_m &= \frac{1}{\mu}\left(\sum_{n=1}^{N} \lambda_n e_n - e_{j_m}\right) = \frac{1}{\mu}\left(\sum_{n=1}^{N} \lambda_n (e_n - e_{j_m})\right) \\ &= \sum_{n \in I_{j_m}} \frac{\lambda_n}{\mu}(e_n - e_{j_m}) \in \mathcal{C}_m(e_{j_m}), \end{aligned}$$

since $\frac{\lambda_n}{\mu} \geq 0$ holds true for all $n \in I_{j_m}$.
Conversely, let $v_m \in \mathcal{C}_m(e_{j_m})$ for all $m \in \mathcal{M}$. For $n \in \mathcal{I}_{j_m}$ there exist $\lambda_n \geq 0$, such that

$$v_m = \sum_{n \in I_{j_m}} \lambda_n (e_n - e_{j_m}) = \sum_{n \in I_{j_m}} \lambda_n e_n - \sum_{n \in I_{j_m}} \lambda_n \cdot y_m = \sum_{n \in I_{j_m}} \lambda_n e_n - \tau_m y_m, \quad (13.11)$$

where $\tau_m = \sum_{n \in I_{j_m}} \lambda_n$. Note that if $\tau_m \leq 1$, Equation (13.11) automatically implies that $y_m + v_m \in \tilde{\mathcal{Y}}_m$. Hence, by convexity of $\tilde{\mathcal{Y}}_m$, this also holds true for all $y_m + \tilde{\mu} v_m$, where $\tilde{\mu} \in [0, 1]$.
We define

$$\mu = \min\left\{1, \frac{1}{\tau_1}, \ldots, \frac{1}{\tau_M}\right\}.$$

If $\mu = 1$, $y_m + \mu v_m \in \tilde{\mathcal{Y}}_m$ for all $m \in \mathcal{M}$, since $\tau_m \leq 1$ holds true for all $m \in \mathcal{M}$. Otherwise, there exists $m^\star \in \mathcal{M}$ such that $\mu = \tau_{m^\star}^{-1}$. For fixed $m \in \mathcal{M}$ we have that

$$\mu v_m = \sum_{n \in I_{j_m}} \frac{\lambda_n}{\tau_{m^\star}}(e_n - e_{j_m}) = \sum_{n \in I_{j_m}} \tilde{\lambda}_n e_n - \frac{\tau_m}{\tau_{m^\star}} \cdot y_m,$$

where

$$\tilde{\lambda}_n = \frac{\lambda_n}{\tau_{m^\star}} \leq \frac{\lambda_n}{\tau_m}$$

holds true for all $n \in I_{j_m}$. But this implies that

$$\sum_{n \in I_{j_m}} \tilde{\lambda}_n = \sum_{n \in I_{j_m}} \frac{\lambda_n}{\tau_{m^\star}} = \frac{\tau_m}{\tau_{m^\star}} \leq 1.$$

Hence, $y_m + \mu v_m \in \tilde{\mathcal{Y}}_m$ holds true for all $m \in \mathcal{M}$. Since $\tilde{\mathcal{Y}}_m$ is convex for all $m \in \mathcal{M}$, this implies that $Y + \tilde{\mu} V \in \tilde{\mathcal{Y}}$ for all $\tilde{\mu} \in [0, \mu]$. □

To derive a feasible *descent* direction V for fixed $Y \in \mathcal{Y}$ with a maximum potential, we search for all feasible directions in Y that minimize the linear objective $\langle \nabla_Y f(X, Y), V \rangle$ according to Equation (12.3). To guarantee feasibility of the resulting solution, the components v_m of V may no longer be arbitrary elements of the cone $\mathcal{C}_m(e_{j_m})$ for $m \in \mathcal{M}$, but they have to be contained within the set $\tilde{\mathcal{Y}}_m - \{y_m\} = \{y - y_m : y \in \tilde{\mathcal{Y}}_m\}$. Hence, due to the linearity of f with respect to Y we are interested in the optimal solution of the following linear optimization problem:

$$\begin{aligned} &\min \langle \nabla_Y f(X, Y), V \rangle \\ &\text{s.t. } V \in \tilde{\mathcal{Y}} - \{Y\}. \end{aligned} \quad (13.12)$$

Since the set $\tilde{\mathcal{Y}} - \{Y\}$ is a non-empty, compact polyhedron, Problem (13.12) is feasible. Furthermore, as the objective function is linear, this problem attains at least one global minimum at a boundary point of the feasible set, due to the *fundamental theorem of linear programming* (cf., e.g., Hamacher and Klamroth [92]). We give an alternative proof in the following. Since the M components of the scalar product are independent, we note that

$$\begin{aligned}\min\{\langle \nabla_Y f(X,Y), V\rangle : V \in \tilde{\mathcal{Y}} - \{Y\}\} \; = \\ = \sum_{m=1}^{M} \min\{\langle \nabla_{Y_m} f(X,Y), v_m\rangle : \; v_m \in \tilde{\mathcal{Y}}_m - \{y_m\}\},\end{aligned}$$

where $\nabla_{Y_m} f(X,Y)$ denotes the restriction of $\nabla_Y f(X,Y)$ to the components of y_m. Hence, we have decomposed Problem (13.12), into M linear problems of the form

$$\begin{aligned}&\min \langle \nabla_{Y_m} f(X,Y), v_m\rangle \\ &\text{s.t. } v_m \in \tilde{\mathcal{Y}}_m - \{y_m\}.\end{aligned} \tag{13.13}$$

To further simplify the notation, we set $r_{mn} := e_n - e_{j_m}$ for fixed $m \in \mathcal{M}$, all $n \in \mathcal{N}$ and given $Y \in \mathcal{Y}$.

Theorem 13.8 *For given $m \in \mathcal{M}$, the optimal solution of Problem* (13.13) *is given by*

$$r_m^\star = \min_{n\in\mathcal{N}}\{\langle \nabla_{Y_m} f(X,Y), r_{mn}\rangle\}. \tag{13.14}$$

Proof: Let $v_m \in \tilde{\mathcal{Y}}_m - \{y_m\}$, i.e. there exists $\lambda_n \geq 0$, $n \in \mathcal{I}_{j_m}$, where $\sum_{n\in\mathcal{I}_{j_m}} \lambda_n \leq 1$ such that $v_m = \sum_{n\in\mathcal{I}_{j_m}} \lambda_n r_{mn}$. Keeping in mind that r_{mj_m} equals the zero vector in $\mathbb{R}^N$, we have that

$$\begin{aligned}\langle \nabla_{Y_m} f(X,Y), v_m\rangle &= \langle \nabla_{Y_m} f(X,Y), \sum_{n\in\mathcal{I}_{j_m}} \lambda_n r_{mn}\rangle = \sum_{n\in\mathcal{I}_{j_m}} \lambda_n \langle \nabla_{Y_m} f(X,Y), r_{mn}\rangle \\ &\geq \min\left\{\sum_{n\in\mathcal{I}_{j_m}} \lambda_n \cdot \min_{n\in\mathcal{I}_{j_m}} \langle \nabla_{Y_m} f(X,Y), r_{mn}\rangle, 0\right\} \\ &\geq \min\left\{\min_{n\in\mathcal{I}_{j_m}} \langle \nabla_{Y_m} f(X,Y), r_{mn}\rangle, 0\right\} \\ &= \min\left\{\min_{n\in\mathcal{I}_{j_m}} \langle \nabla_{Y_m} f(X,Y), r_{mn}\rangle, \langle \nabla_{Y_m} f(X,Y), r_{mj_m}\rangle\right\} \\ &= r_m^\star.\end{aligned}$$

This completes the proof. □

Theorem 13.8 shows that each optimal solution of Problem (13.13) corresponds to an extreme-point of the feasible set $\tilde{\mathcal{Y}}_m - \{y_m\}$. Given $m \in \mathcal{M}$ and $n \in \mathcal{N}$, we further conclude from (13.10) that

$$\langle \nabla_{Y_m} f(X,Y), r_{mn}\rangle = \langle \nabla_{Y_m} f(X,Y), e_n\rangle - \langle \nabla_{Y_m} f(X,Y), e_{j_m}\rangle = c_m(x_n) - c_m(x_{j_m}).$$

Hence, $r_m^\star = \min\{c_m(x_n) - c_m(x_{j_m}) : n \in \mathcal{N}\}$. We summarize our results in the following corollary.

Corollary 13.9 *Let $Y \in \mathcal{Y}$. Then, the optimal solution of Problem* (13.12) *corresponds to an extreme-point of the feasible set $\tilde{\mathcal{Y}} - \{Y\}$. Its optimal objective value is given by*

$$\sum_{m \in \mathcal{M}} \min_{n \in \mathcal{N}} \{c_m(x_n) - c_m(x_{j_m})\}\,.$$

Based on Equation (12.3), we finally define the descent potential for a fixed allocation vector with respect to the relaxation of the feasible set $\mathcal{Y}$ to $\tilde{\mathcal{Y}}$. Let $Y \in \mathcal{Y}$ be given and let $V \in \mathbb{R}^{M \times N}$. Due to the linearity of f with respect to the allocation variables, we have that

$$f(X, Y + V) \;=\; f(X, Y) + \langle \nabla_Y f(X, Y), V \rangle.$$

Hence, the maximum descent potential of f for fixed Y in a feasible direction is given by the optimal solution of Problem (13.12). According to the result of Corollary 13.9, we define:

Definition 13.10 *Let a feasible solution (X, Y) of Problem* (ConLoc) *be given. Then, the descent potential π^c_Y of f with respect to the allocation vector Y is given by*

$$\pi^c_Y(X, Y) \;=\; \left| \sum_{m \in \mathcal{M}} \min_{n \in \mathcal{N}} \{c_m(x_n) - c_m(x_{j_m})\} \right|.$$

Note that if the allocation vector Y corresponds to an optimal solution, resulting from the allocation step in Algorithm 13.2 for fixed $X \in \mathbb{R}^2 \times \mathbb{R}^N$, we obviously have that $\pi^c_Y(X, Y) = 0$, according to the results from Section 13.2.
Besides the above stated definition for the descent potential, we further consider a definition that is based on counting the number of changes in the allocation vector in two different feasible solutions of the problem. If the allocation vectors of two subsequent iterations of Algorithm 13.2 differ strongly, a noticeable improvement of the objective value can be expected, due to the fact that many allocations have changed after the last location step. Hence, measuring the number of changes in the allocation vector yields a promising approach to estimate descent information.

Definition 13.11 *Let (X, Y) denote a feasible solution of Problem* (ConLoc)*, and let $\bar{Y} \in \mathcal{Y}$. Then, the descent potential π^d_Y of the point (X, Y) with respect to the allocation vector $\bar{Y}$ is given by*

$$\pi^d_Y(X, Y)[\bar{Y}] \;=\; \frac{1}{2} \sum_{m=1}^{M} \sum_{n=1}^{N} |y_{mn} - \bar{y}_{nm}|.$$

In contrast to Definition 13.10 the descent potential in the above definition is independent of the improvement of the objective values, but measures the differences among variables only. For the practical application of Definition 13.11 to the connection location-allocation problem, we refer to the next subsection.

13.3.3 The Enhanced Versions of Algorithm 13.2

Due to the results from Section 13.2, the location as well as the allocation step of Algorithm 13.2 can be implemented very efficiently. In contrast, applying the enhanced

alternate convex search strategy developed in Chapter 12 to the connection location-allocation problem would require to solve a large number of non-linear, mixed-integer optimization problems in each step of Algorithm 12.1 to derive a subset of (weakly-)efficient points for the biobjective optimization problems (12.5) and (12.6). Hence, solving these biobjective problems for the complete feasible set of the individual subproblems would be too time-consuming compared to the efficiency of Algorithm 13.2. In the following, we only concentrate on smaller subsets contained in the feasible sets of these individual subproblems, to incorporate the idea of the descent potential.
During the location step, we focus on the feasible points that are automatically generated during the solution process of the M individual Weiszfeld algorithms. The feasible set of the biobjective problem is restricted to the set of location vectors that result from the intermediate iterations of these M algorithms. For the allocation step, the feasible set of the biobjective problem involving π_X is restricted to the set of all assignment vectors that correspond to permutations of the changed assignments resulting from the assignment vectors of the last and the current iteration. In more detail, our considered modified versions of the location and allocation steps that incorporate the idea of the descent potential are given as follows:

Modified location step using π_Y^c: Let a fixed allocation vector $Y \in \mathcal{Y}$ be given. The Weiszfeld Algorithm 13.1 is used to solve the induced M location subproblems. After each iteration of the M individual Weiszfeld algorithms the descent potential π_Y^c of the intermediate locations x_n for $n \in \mathcal{N}$ is calculated. The location step is stopped when either the original stopping criteria for all M individual Weiszfeld algorithms are satisfied, or a decrease of π_Y^c with respect to the preceding iterations of the individual Weiszfeld algorithms is detected.

Modified location step using π_Y^d: Let a fixed allocation vector $Y \in \mathcal{Y}$ be given. The Weiszfeld Algorithm 13.1 is used to solve the induced M location subproblems. After each iteration of the M individual algorithms, the optimal allocations for the intermediate locations x_n ($n \in \mathcal{N}$) are calculated. Then, π_Y^d is used to determine the descent potential of these intermediate solutions. The location step is stopped, when either the original stopping criteria for all M individual Weiszfeld algorithms are satisfied, or a decrease in π_Y^d with respect to the preceding iteration is detected.

Modified allocation step using π_X: Let a fixed location vector $X \in \mathbb{R}^2 \times \mathbb{R}^N$ be given, and let $Y^1 \in \mathcal{Y}$ denote the assignment vector of the preceding iteration. The original allocation step is used to determine an optimal assignment Y^2 for fixed X. Then, the number of changed assignments is determined. The descent potential π_X is calculated for all assignment vectors that correspond to permutations of the changed assignments in the assignment vectors Y^1 and Y^2. The assignment vector with the maximum descent potential π_X is chosen as final vector returned by the modified allocation step.

In the following, we relate the solutions obtained by applying one of the above described modified location and allocation steps to the efficient solutions of the corresponding biobjective optimization problems (12.5) and (12.6). According to the descriptions given above, the modified location step is stopped, whenever a decrease in the particular descent potential is detected compared to the previous iteration. If it is assumed

that the objective value of f with respect to X is improved in each call of the Weiszfeld algorithms, the chosen solution X normally corresponds to an efficient solution of the biobjective problem, restricted to the points calculated by the M individual Weiszfeld algorithms for the different location problems. However, if the modified location step is stopped because a decrease of the descent potential is detected, there still might exist solutions that would be calculated in future iterations of the original method, that could lead to an improved descent potential, and hence could dominate the chosen solution. However, due to performance reasons for the enhanced solution approach, we do not consider this possibility in our modified location step.
For the modified allocation step, we remark that the feasible set of the biobjective subproblem is given by the set of all assignments from $\mathcal{Y}$ that correspond to an individual exchange of assignments given by the vectors Y^1 and Y^2. As the number of changes in the two assignment vectors is given by

$$\tau = \frac{1}{2} \sum_{m=1}^{M} \sum_{n=1}^{N} |y^1_{mn} - y^2_{mn}|,$$

the cardinality of the feasible set is given by 2^τ. All these 2^τ permutations are tested during the modified allocation step and the assignment that results in the highest descent potential is used as new assignment vector. Hence, the lexicographically optimal assignment with respect to π_X that is contained in the restricted feasible set of the biobjective subproblem is chosen for the next iteration.
In the remainder of this chapter, we use the notion of the *augmented location-allocation algorithm*, whenever one of the two steps of the original alternate location-allocation algorithm is replaced by the modified location and/or allocation step described above. An algorithmic description of this algorithm can easily be derived from the one of Algorithm 12.1 given in Chapter 12. Hence, we omit a detailed outline here. As in the original version of the alternate location-allocation algorithm, a maximum number of iterations should be prescribed to guarantee the termination of the algorithm. If the maximum number of iterations is reached, the original version of the algorithm is applied to the returned solution, to guarantee that a local optimum of Problem (ConLoc) is found.
However, our numerical studies show that for the considered instances the augmented location-allocation algorithm terminates before the maximum number of iterations is reached, when the termination criteria of the two different subproblems of Algorithm 13.2 are also applied to the modified steps of the enhanced version of the algorithm. One reason for this might be seen in the strongly restricted feasible sets of the individual biobjective subproblems. For example, if π^d_Y is used to calculate the descent potential in the modified location step, the allocation vectors that are calculated for the intermediate solutions of the M individual Weiszfeld algorithms may coincide in each iteration. Hence, the solution that is returned by the modified location step equals the solution that is obtained, when the descent potential π^d_Y is not considered.

13.3.4 Numerical Results

We present detailed numerical comparisons between the alternate location-allocation algorithm (cf. Algorithm 13.2) and the different variants of the augmented location-allocation algorithm in the following, where the aims of our numerical studies are

Symbol	Alternate Location-Allocation Algorithm 13.2 with ...
$(\mathcal{LA})$	original location and allocation step.
$(\mathcal{L}_Y^c\mathcal{A})$	modified location step applying descent potential π_Y^c.
$(\mathcal{L}_Y^d\mathcal{A})$	modified location step applying descent potential π_Y^d.
$(\mathcal{LA}_X)$	modified allocation step applying descent potential π_X.
$(\mathcal{L}_Y^d\mathcal{A}_X)$	modified loc. and alloc. step applying descent potential π_Y^d and π_X.

Table 13.1: *Notation used for the algorithms used in Subsection 13.3.4.*

twofold. In the first part of this subsection, Algorithm 13.2 is compared to four enhanced variants of the original algorithm that make use of the modified location and allocation steps defined in Subsection 13.3.3. Due to the additional CPU-time that has to be spent to incorporate the idea of the descent potential in the several variants of these algorithms, we further compare the solutions that are returned by the different approaches within a fixed prescribed amount of time, in the second part of this subsection.

To simplify the notation for the individual algorithms, we use the shortcuts that are defined in Table 13.1. The numerical setups as well as detailed numerical results of our study can be found in Appendix B. A summary of the main results concerning the comparisons between the different algorithms is additionally listed in Table 13.2. We mainly focus on the evaluation of the obtained results in the remainder of this subsection.

We start with a comparison of Algorithm $(\mathcal{LA})$ to its enhanced versions with modified location step (cf. Tables B.3 to B.6 in Appendix B). Having a closer look at Table B.3 we conclude that Algorithm $(\mathcal{L}_Y^c\mathcal{A})$ does not yield superior results to Algorithm $(\mathcal{LA})$. While Algorithm $(\mathcal{L}_Y^c\mathcal{A})$ found a better local optimum for 670 test instances, the original algorithm converged for 707 instances to a better local optimal solution, based on the same initial solution for both algorithms (cf. also Table 13.2). Furthermore, this result seems to be independent from the considered size of the test problems. Concerning the CPU time (cf. Table B.4), Algorithm $(\mathcal{LA})$ ran much faster than its enhanced counterpart especially for larger problem instances, due to the numerous additional calculations that have to be performed to evaluate the descent potential for larger problem sizes.

The situation changes, when the discrete version π_Y^d from Definition 13.11 is used to measure the descent potential in the modified location step. Table B.5 shows that the original algorithm was outperformed by Algorithm $(\mathcal{L}_Y^d\mathcal{A})$, especially when problem sizes larger than 19 were considered (cf. Table B.1 for the definition of the problem sizes). From Table B.6 we conclude that the CPU time spent by Algorithm $(\mathcal{L}_Y^d\mathcal{A})$ did not increase that much as compared to Algorithm $(\mathcal{L}_Y^c\mathcal{A})$. While for the latter, the CPU time was larger by a factor of 35 compared to the CPU time used for the original algorithm, this factor did not exceed a value of 6.3 for Algorithm $(\mathcal{L}_Y^d\mathcal{A})$. The reason for this can be seen in the fact that if the algorithm has nearly converged to a local minimum in the allocation step, the allocation vectors that are calculated for the intermediate solutions resulting from the individual iterations of the Weiszfeld algorithms may coincide throughout the iterations. This results in a faster convergence

	Algorithm ($\mathcal{LA}$)		
Augmented Algorithms	+	−	=
Algorithm ($\mathcal{L}^c_Y\mathcal{A}$)	670	707	273
Algorithm ($\mathcal{L}^d_Y\mathcal{A}$)	777	629	244
Algorithm ($\mathcal{LA}_X$)	845	761	44
Algorithm ($\mathcal{L}^d_Y\mathcal{A}_X$)	862	746	42
Best of ($\mathcal{L}^d_Y\mathcal{A}$), ($\mathcal{LA}_X$), ($\mathcal{L}^d_Y\mathcal{A}_X$)	1232	248	170

Table 13.2: *Summary of the numerical results from Appendix B. In each row the specific algorithm is compared to Algorithm ($\mathcal{LA}$) after 1650 runs of the involved algorithms, always initialized with the same starting solutions.*

of the enhanced algorithm for π^d_Y.
Tables B.7 and B.8 compare Algorithm ($\mathcal{LA}$) to its enhanced version with modified allocation step. It can be seen that in more than 80 cases a better objective value was obtained, when the enhanced version ($\mathcal{LA}_X$) was applied (cf. also Table 13.2). This is especially the case for larger problem sizes, while the original version of the algorithm was clearly superior, when only a few new connections have to be located in the plane. Concerning the CPU time, Algorithm ($\mathcal{LA}_X$) was much slower compared to Algorithm ($\mathcal{LA}$), especially for larger problem sizes. Since the feasible set of the biobjective problem is given as the set of all possible permutations of changes contained in two different allocation vectors, many additional calculations have to be performed to evaluate the involved gradient information during the course of the modified allocation step.
Finally, Tables B.9 and B.10 show the comparison between Algorithm ($\mathcal{LA}$) and the augmented location-allocation algorithm with modified location and allocation step, i.e. Algorithm ($\mathcal{L}^d_Y\mathcal{A}_X$). Due to the numerical results from Tables B.3 to B.6, only the descent potential π^d_Y from Definition 13.11 was used during the location steps of the enhanced version of the algorithm. The obtained results nearly coincide with the results for Algorithm ($\mathcal{LA}_X$). While the original algorithm was better for smaller instance sizes, it was clearly outperformed by the enhanced version for problem sizes larger than 20. The CPU time of the Algorithm ($\mathcal{L}^d_Y\mathcal{A}_X$) is mainly influenced by the modified allocation step. At maximum, the average time was longer by a factor of 100 as compared to the original version of the algorithm. However, the enhanced version found a better local optimum in 116 cases.
Comparing Algorithm ($\mathcal{L}^d_Y\mathcal{A}$) to Algorithm ($\mathcal{LA}_X$), we conclude from Table B.11 that none of the two versions outperformed the other. Both algorithms found a better optimal solution for more than 800 instances. As in the comparisons stated above, Algorithm ($\mathcal{LA}_X$) seems to perform better for larger problem instances.
Table B.12 compares the best objective value obtained from the three enhanced versions of Algorithm 13.2 to the objective value that is obtained for Algorithm ($\mathcal{LA}$), based on the same starting solution for all four algorithms (cf. also Table 13.2). The table shows that in almost 75% of all cases, at least one of the three enhanced versions found a better local optimal solution than the original method. Algorithm ($\mathcal{LA}$) was better for only 15% of the analyzed instances.

In Table B.13, the best objective values that were obtained, when all four algorithms had been started with the same initial solutions, are related to the algorithms that returned these specific objective values. It can be seen that Algorithm ($\mathcal{LA}$) is outperformed by all three enhanced versions. Furthermore, Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$) seems to be slightly better than the enhanced versions, where only one of the two modified steps was used, since it found the best local minimum in over 30% of all cases. Once more, this especially is the case for larger problem sizes.
The quality of the calculated local minima with respect to the used CPU time is investigated in Table B.14, where the multi-start version of Algorithm ($\mathcal{LA}$) (cf. Algorithm 13.3) is compared to Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$) for problem sizes larger than 14 (cf. Table B.1). The enhanced version was initialized with only 10 solutions and its CPU time to solve these 10 problems was measured. Then Algorithm 13.3 was used to repeatedly solve the given instance until the CPU time of the enhanced version had been exceeded. From Table B.14 we see that this time-based comparison of the algorithms resulted in at maximum nearly 1000 additional restarts of Algorithm ($\mathcal{LA}$) depending on the given problem size. Furthermore, it can be concluded that Algorithm ($\mathcal{LA}$) performs much better than Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$), when the CPU time is considered as additional criterion. However, the enhanced version still found a better local optimal solution for some problem instances although Algorithm ($\mathcal{LA}$) is restarted with up to 30 times more initial solutions.

Summarizing the numerical results stated in this subsection, we conclude that the proposed biobjective approach based on measuring the descent potential contained in the fixed set of variables can be seen as an alternative solution concept for Problem (ConLoc), when an alternate convex search strategy is used to solve the problem. While the augmented location-allocation Algorithm ($\mathcal{L}_Y^c\mathcal{A}$) was clearly outperformed by Algorithm ($\mathcal{LA}$), applying the descent potential π_Y^d from Definition 13.11 clearly improved the performance of the enhanced solution approach. The same observation holds true, when π_X from Definition 13.4 was used to measure the descent potential in the modified allocation step. The combination of the two modified steps in Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$) additionally improved the objective value of the resulting local optimal solution.
However, the numerical tests show that for all four variants of the augmented location-allocation algorithm it seems to be too restrictive to focus only on the points that are calculated during the individual iterations of the original method. This especially holds true, when π_Y^c is used to measure the descent potential for fixed allocation variables. Although the original method was outperformed for larger problem sizes, Algorithm ($\mathcal{LA}$) seems to be superior especially for smaller instances of the considered problem.
In contrast, Table B.13 show that in most cases at least one of the three enhanced search strategies outperformed the original version of the algorithm, when all four versions were initialized with the same starting solution, especially when larger problem sizes were considered. As the number of local minima of Problem (ConLoc) is expected to grow exponentially with the problem size (assuming that all solutions in the individual location steps are unique), considering only 10 different starting solutions for Algorithm ($\mathcal{LA}$) seems to be insufficient to derive good local optima for Problem (ConLoc) in general. In this case, the enhanced versions seem to be superior. However, our numerical studies also show that it is not evident in advance which

modified version of the algorithm should be used to improve the objective value of the resulting local optimum compared to Algorithm ($\mathcal{LA}$).
Furthermore, due to the additional amount of calculations during the course of the modified steps of Algorithm ($\mathcal{LA}$), the original algorithm clearly outperforms its enhanced versions with respect to the CPU time spent to solve the problem. Taking this as additional performance criterion to rate the four algorithms, we have seen that the multi-start version of the original location-allocation algorithm (cf. Algorithm 13.3) may converge to a better local optimum of Problem (ConLoc) in many more cases as compared to the enhanced versions of the original algorithm.
We conclude that if Problem (ConLoc) has to be solved for a prescribed set of initial solutions, one of the modified versions of the original alternate location-allocation algorithm should be used to solve Problem (ConLoc), especially when larger problem instances are considered, and the CPU time that is spent to solve the problem is not of further interest. In all these cases, the augmented location-allocation algorithm with modified location and allocation step, i.e. Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$) seems to be the favorable choice. However, if the calculation of a good local optimum of Problem (ConLoc) within a prescribed amount of time is of interest, the multi-start version of Algorithm ($\mathcal{LA}$) should be used instead.

13.4 Conclusions and Further Ideas

In this chapter we discussed the connection location-allocation problem in the plane. We showed that the problem can be formulated as a biconvex optimization problem involving the set of location and allocation variables, respectively. If the locations of the connections are fixed in the $\mathbb{R}^2$-plane, the optimization problem simplifies to a simple assignment problem, while for prescribed allocations a fixed number of single-facility location problems has to be solved. This can be done efficiently by means of the Weiszfeld algorithm applied to the individual single-facility location problems. Hence, the alternate convex search strategy described in Chapter 11 yields an efficient way to derive local minima for the given problem from location theory.
However, since the local optimal solution that is obtained by applying the alternate location-allocation algorithm strongly depends on the given initial solution, we applied the enhanced alternate convex search strategy suggested in Chapter 12 of this work to try to heuristically improve the quality of the calculated local minima. Due to the mixed-integer structure of the given problem, we adapted the general definition of the descent potential to fit the given problem.
We presented detailed numerical comparisons between the alternate location-allocation algorithm and the different versions of the augmented location-allocation algorithm developed in this chapter. We saw that in most cases, at least one of the modified versions of the algorithm is capable to improve the objective value of the resulting local minimum compared to the original version of the algorithm. However, due to the additional numerical effort to incorporate the idea of the descent potential in the modified location and allocation step, respectively, the multi-start version of the alternate location-allocation algorithm seems to be more than competitive compared to the multi-start version of the enhanced methods, especially when the main focus lies on the spent CPU time. However, especially for larger problem sizes, the augmented

location-allocation algorithm with modified location and allocation step outperforms the original method with respect to the objective value of the resulting local optimal solution.
For further fields of research, the application of the enhanced alternate convex search strategy to other location-allocation problems from location theory can be of interest. However, our numerical studies suggest that the application of the enhanced version of the general location-allocation algorithm should be limited to problems with a large number of local minima. Otherwise, a multi-start version of the original algorithm or other meta-heuristic approaches seems to be more favorable. In this context, the enhanced versions of the location-allocation algorithm for the connection location-allocation problem that were presented in this chapter could additionally be compared to the meta-heuristics proposed by Bischoff and Dächert [22].
Due to the biconvex structure of the connection location-allocation problem, also the application of the global optimization algorithm (cf. Section 11.3.3) can be of interest to derive the global optimum of the given problem. The main drawback of this approach can be seen in the large number of relaxed dual subproblems that have to be solved for every possible combination of bounds in the set of connected variables in each iteration of this algorithm. Mathematical investigations show that this set can be identified with the set of feasible allocation vectors of the problem that makes the direct application of this approach impractical in practice. However, a branch & bound procedure as suggested in Floudas [64] for the general global optimization algorithm could be investigated, at least for smaller problem instances.

Chapter 14

Conclusions

In this thesis we dealt with multiple objective optimization and its implications for single objective optimization problems. In more detail, we discussed how ideas and solution concepts from multiple objective optimization can be used to gain a new insight into special types of single objective optimization problems. In this context, we focused on combinatorial optimization problems in Part I and on biconvex optimization problems in Part II of this work. However, we already showed in the preliminary Chapter 3 that the theoretical background and ideas of our approaches are not exclusively limited to these two special fields of optimization. Independently from these ideas, we additionally investigated important structural properties of the efficient set of multiple objective combinatorial optimization problems.

While traditionally single objective approaches are used to solve multiple objective problems, we took the reverse approach in this thesis. Given the constrained version of a single objective problem, we showed that solution concepts from multiple objective optimization can be used to derive an optimal solution for the single objective problem. We further showed that this also holds true for single objective problems where the corresponding objective is given as a weighted sum of different types of objectives.

We summarized the most important exact solution methods for multiple objective combinatorial problems with sum objectives and generalized the existing results for combinatorial problems with bottleneck objectives. In addition, we discussed the notion of a generalized bottleneck objective yielding the k^{th} largest cost coefficient of a feasible solution. We showed that the resulting k-max optimization problem can be solved within a polynomial amount of time, whenever an associated auxiliary sum problem with binary costs satisfies this property.

We additionally used the presented solution approach for the single objective case to derive an efficient algorithm for the multiple objective k-max optimization problem with an additional sum objective. While combinatorial optimization problems with sum objectives are intractable in general, we showed that the considered k-max optimization problem can be solved efficiently, whenever an associated single objective optimization problem with binary constraints can be solved in a polynomial amount of time.

Using the formulation of multiple objective combinatorial optimization problems with bottleneck and k-max objectives, respectively, we further showed that most of the algorithms for solving balanced combinatorial optimization problems, minimum deviation

problems and k-sum optimization problems that can be found in the literature are implicitly based on a multiple objective reinterpretation of the specific single objective optimization problem. We made use of this idea and derived solution approaches for generalized versions of these optimization problems. We proved that the considered problems can be solved efficiently, whenever this holds true for their associated multiple objective problem formulation.

Although we restricted ourselves to the multiple objective reinterpretation of the ε-constraint and the weighted sum approach in this work, there exist other solution concepts from multiple objective optimization whose problem formulations can be seen as an associated formulation of a single objective problem. For example, we showed in Chapter 3 that the compromise solution method with squared Euclidian distance can be interpreted as an associated multiple objective problem formulation of a least squares problem from adjustment theory. Further research could focus on this and other reinterpretations of solution concepts from the field of multiple objective optimization.

Another main topic of this thesis was the analysis of the connectedness of the efficient set for multiple objective combinatorial optimization problems. If this set is known to be connected for a given class of combinatorial problems, this would imply that the complete set itself can be determined by means of simple local search techniques. Based on two different approaches for defining the adjacency of efficient solutions, we presented counter-examples that show that most of the classical combinatorial problems do not yield a connected efficient set in general. In addition, we proved that this further holds true, when multiple objective combinatorial problems with bottleneck objectives are considered.

On the other hand, based on a greedy-like algorithm for the triobjective unconstrained optimization problem with two binary objectives, we were able to prove the connectedness of the efficient set for this special type of problem. Moreover, we showed that our suggested algorithm is optimal in terms of upper bound time complexity. Applying this algorithm, we were able to solve instances with up to one million items and 180 billion non-dominated solutions within less than 30 minutes of CPU-time. In addition, we discussed the biobjective matroid problem with at least one binary sum objective. Also for this problem we proved the connectedness of the efficient set, based on a modified version of an algorithm already stated in the literature.

From our results on the connectedness of the efficient set we conclude that this property is a powerful tool that helps to solve multiple objective combinatorial optimization problems efficiently. However, our investigations suggest that the connectedness of this set mainly depends on the structure of the given cost coefficients involved in the problem. When these coefficients are chosen by random, our numerical tests for the biobjective binary knapsack problem with bounded cardinality from Chapter 7 showed that instances with unconnected efficient set are rare, but however they exist. This even holds true, when unconstrained biobjective optimization problems are considered. In contrast, our results for multiple objective problems with binary objectives show that whenever the values of the given cost coefficients are restricted to $\{0, 1\}$, i.e. the cardinality of the non-dominated set is polynomially bounded, connectedness of the efficient set for these specific types of problems can be proven.

Besides combinatorial optimization problems, we also dealt with biconvex optimization problems. Different from general non-linear optimization problems, a biconvex

problem has the nice property that the set of variables can be partitioned into two disjoint blocks of variables such that the given problem decomposes into two convex subproblems, whenever one of the blocks of variables is considered as fixed. From the literature on biconvex problems we recalled that the alternate convex search method can be used to derive stationary points of a given optimization problem. This solution approach can be seen as a special version of the more general alternate block search technique that was discussed in Chapter 3. In the latter chapter we additionally presented an enhanced version of this solution technique based on a multiple objective solution approach.

We revisited this approach in Chapter 12 for the biconvex case. Besides the two objectives of the convex subproblems, we used the idea of additionally exploiting descent information that is contained in the fixed block of variables when the problem is solved in the block of active variables. We discussed how this additional information can be used to heuristically improve the results obtained by applying the alternate convex search technique as a special variant of the more general alternate block search strategy.

We further made use of this approach to derive local minima for the connection location-allocation problem in the plane. Due to the biconvex structure of the problem from location theory, the location-allocation algorithm as a special version of the more general alternate convex search technique can be used to derive local minima of the considered problem. We numerically showed that the results of this algorithm can be improved by applying the proposed enhanced versions of the alternate convex search technique.

However, due to the complex structure of the considered location problem, we had to restrict the feasible set of our four enhanced versions of the original location-allocation algorithm to a set of points that is implicitly calculated during the iterations of the original method. Although this limitation seems quite restrictive, our numerical studies showed that at least one of the four proposed versions led to an improvement of the objective value in most cases, assuming that all algorithms are initialized with the same starting solution. If CPU-time is taken into account as additional criterion, we saw that the multi-start version of the location-allocation algorithm performs better compared to its enhanced versions, since numerous additional information has to be gathered to evaluate the descent potential that is contained in the fixed block of variables during the course of the original algorithm.

We mainly restricted ourselves to multiple objective approaches for solving biconvex optimization problems in Part II of this thesis. However, the idea of measuring the descent potential for a block of variables is not limited to the biconvex case, but can also be applied to more general non-linear optimization problems based on the general framework presented in Chapter 3. Due to the more complex definition of the descent potential when constrained optimization problems are considered, a further application of this approach should mainly focus on the unconstrained case.

To summarize our results, we finally conclude that multiple objective optimization yields a powerful tool to analyze and exploit the structure of special classes of single objective (combinatorial) optimization problems. Although a multiple objective-based approach for solving a single objective problem implicitly implies that additional and perhaps unnecessary information has to be gathered that may not be needed for further calculations, this additional information can be used to significantly improve

existing and derive new solution concepts for single objective problems, respectively. In addition, concepts from multiple objective optimization may be the only way to unify seemingly different approaches for single objective problems in a more general framework.

Appendix A

Outline of the Greedy Algorithms of Chapter 9

This part of the appendix contains the outline of the three greedy algorithms for Problem (2-KP$_=$), Problem (2-MP) and Problem (2-KP$_\leq$) discussed in Chapter 9. For the notation used in the following, we refer to the specific sections of Chapter 9.
While Algorithm A.1 can be applied to any instance of Problem (2-KP$_=$) satisfying $c^1 \geq c^2 > 0$ (cf. Section 9.2), Algorithm A.2 for Problem (2-MP) (cf. Section 9.3) and Algorithm A.3 for Problem (2-KP$_\leq$) (cf. Section 9.4) are restricted to instances of the individual problems where $(c^1, c^2) \in \mathcal{G}_2$ holds. Appropriate algorithms for the other sectors can be derived from the algorithms stated here in combination with the results presented in Chapter 9.
For Algorithm A.1 that computes the optimal profit value for Problem (2-KP$_=$) with constraint (c^1, c^2) , we assume without loss of generality that the profit values d_j, r_{c+i} and u_i always exist in the outline of the algorithm. For further details, we refer to Section 9.2.
To simplify the notation of Algorithm A.2, we omit the case that $c^1 - c^2 = 0$. Furthermore, it is assumed that the second stopping criterion given in Line 21 of Algorithm A.2 is evaluated if and only if the first criterion is satisfied. Otherwise, the considered indices of the sequences r and d may not be well-defined.

Algorithm A.1 Greedy Algorithm for Problem (2-KP$_=$)

Input: An instance of Problem (2-MP), assuming that $c^1 \geq c^2 > 0$.
Output: Maximum profit criterion value p.

Pre-processing step (see Section 9.1).
$i \leftarrow 1$
$j \leftarrow 1$
$c \leftarrow c^1 - c^2$
$p \leftarrow \sum_{k=1}^{c} r_k$
for $\ell = 1$ to c^2 **do**
  if $d_j < r_{c+i} + u_i$ **then**
    $p \leftarrow p + r_{c+i} + u_i$
    $i \leftarrow i + 1$
  else
    $p \leftarrow p + d_j$
    $j \leftarrow j + 1$
  end if
  $\ell \leftarrow \ell + 1$
end for
return p

Algorithm A.2 Greedy Algorithm for Problem (2-MP) in Sector $\mathcal{G}_2$

Input: An instance of Problem (2-MP).
Output: The non-dominated set $\mathcal{Y}_N(\mathcal{G}_2)$ in sector $\mathcal{G}_2$.

Pre-processing step (see Section 9.1).
$b \leftarrow 0$
$\mathcal{Y}_N(\mathcal{G}_2) \leftarrow \emptyset$
for $c = 1$ to $(n_R - n_U)$ **do**
  $b \leftarrow b + r_c$
  $\mathcal{Y}_N(\mathcal{G}_2) \leftarrow \mathcal{Y}_N(\mathcal{G}_2) \cup \{(c, 0, b)\}$
  $p \leftarrow b$
  $i \leftarrow 1$
  $j \leftarrow 1$
  if $c < n_R - n_U$ **then**
    repeat
      if $d_j < r_{c+i} + u_i$ **then**
        $p \leftarrow p + r_{c+i} + u_i$
        $\mathcal{Y}_N(\mathcal{G}_2) \leftarrow \mathcal{Y}_N(\mathcal{G}_2) \cup \{(c + i + j - 1, i + j - 1, p)\}$
        $i \leftarrow i + 1$
      else
        $p \leftarrow p + d_j$
        $\mathcal{Y}_N(\mathcal{G}_2) \leftarrow \mathcal{Y}_N(\mathcal{G}_2) \cup \{(c + i + j - 1, i + j - 1, p)\}$
        $j \leftarrow j + 1$
      end if
    until $n_U + 1 \leq i + j - 1 \leq n_U + n_D$ and then $r_{n_U+c+1} \geq d_{i+j-1-n_U}$
  else
    while $\leq i + j - 1 \leq n_U + n_D$ **do**
      if $d_j < r_{c+i} + u_i$ **then**
        $p \leftarrow p + r_{c+i} + u_i$
        $\mathcal{Y}_N(\mathcal{G}_2) \leftarrow \mathcal{Y}_N(\mathcal{G}_2) \cup \{(c + i + j - 1, i + j - 1, p)\}$
        $i \leftarrow i + 1$
      else
        $p \leftarrow p + d_j$
        $\mathcal{Y}_N(\mathcal{G}_2) \leftarrow \mathcal{Y}_N(\mathcal{G}_2) \cup \{(c + i + j - 1, i + j - 1, p)\}$
        $j \leftarrow j + 1$
      end if
    end while
  end if
end for
return $\mathcal{Y}_N(\mathcal{G}_2)$

Algorithm A.3 Greedy Algorithm for Problem (2-KP$_{\leq}$)

Input: An instance of Problem (2-KP$_{\leq}$), assuming $(c^1, c^2) \in \mathcal{G}_2$.
Output: Maximum profit criterion value p.

Pre-processing step (see Section 9.1).
$c \leftarrow c^1 - c^2$
if $(c^2 \leq n_U)$ or $(c = n_r - n_U)$ **then**
  $p \leftarrow$ value returned by Algorithm A.1 for (c^1, c^2)
else
  $i \leftarrow 0$
  if $c^1 \leq n_R$ **then**
    while $(i \leq c^2 - 1 - n_U)$ and $(d_{c^2-i-n_U} \leq r_{n_U+c+i+1})$ **do**
      $i \leftarrow i + 1$
    end while
    if $i > c^2 - 1 - n_U$ **then**
      $p \leftarrow$ value returned by Algorithm A.1 for (c^1, n_U)
    else
      $p \leftarrow$ value returned by Algorithm A.1 for $(c^1, c^2 - i)$
    end if
  else
    while $(i \leq n_R - n_U - c - 1)$ and $(d_{c^2-i-n_U} \leq r_{n_U+c+i+1})$ **do**
      $i \leftarrow i + 1$
    end while
    if $i > n_R - n_U - c - 1$ **then**
      $p \leftarrow$ value returned by Algorithm A.1 for $(c^1, c^1 - n_R + n_U)$
    else
      $p \leftarrow$ value returned by Algorithm A.1 for $(c^1, c^2 - i)$
    end if
  end if
end if
return p

Appendix B

Supplementary Numerical Results

We list the results of the numerical study for the connection location-allocation problem in the plane from Chapter 13 in this part of the appendix.
For our numerical studies, we investigated the test problems provided by Bischoff in [20]. The 33 randomly generated test problems range from 5 existing facilities with 10 flows to 100 existing facilities with 4950 flows, respectively (cf. Table B.1). Each test problem is once more subdivided into 5 samples of the same size.

Size	L	M	N	Size	L	M	N	Size	L	M	N
1	5	10	3	12	16	120	8	23	40	780	20
2	6	15	3	13	17	136	9	24	45	990	23
3	7	21	4	14	18	153	9	25	50	1225	25
4	8	28	4	15	19	171	10	26	55	1484	30
5	9	36	5	16	20	190	10	27	60	1770	35
6	10	45	5	17	22	231	11	28	65	2080	40
7	11	55	6	18	24	276	12	29	70	2415	45
8	12	66	6	19	26	325	13	30	75	2775	50
9	13	78	7	20	28	378	14	31	80	3160	55
10	14	91	8	21	30	435	15	32	90	4005	65
11	15	105	8	22	35	595	18	33	100	4950	75

Table B.1: *Problem sizes of the considered test instances. Here, L denotes the number of flows, M corresponds to the number of existing facilities, and N represents the number of connections to locate in the plane.*

For each test instance the coordinates of the existing facilities $a_l \in \mathbb{R}^2$, $l \in \mathcal{L}$, were randomly generated within the set $\{0, \ldots, 1000\}$. The intensity w_m of each flow $m \in \mathcal{M}$ between a pair of existing facilities a_i, a_j, $i, j \in \mathcal{L}$, $i < j$ was chosen within the set $\{5, \ldots, 25\}$. Furthermore, the number of connections N to be located in the $\mathbb{R}^2$-plane ranged within the set $\{3, \ldots, 75\}$, increasing with the size of the test problems. For more details, we refer to Table B.1.

Symbol	Objective value obtained for Algorithm 13.2 and the ...
F	original location and allocation step.
F_Y^c	modified location step applying descent potential π_Y^c.
F_Y^d	modified location step applying descent potential π_Y^d.
F_X	modified allocation step applying descent potential π_X.
F_{XY}^d	modified Loc. and Alloc. step applying descent potential π_Y^d and π_X.

Table B.2: *Notation used for the tables in this section.*

All algorithms were implemented in Matlab and a Sun Fire V20z machine with two AMD Opteron 2.4 GHz CPUs was used for the numerical studies.
The following tables show the comparison between the original alternate location-allocation algorithm (cf. Algorithm 13.2) and the enhanced versions of the algorithm. The notation that is used to evaluate the results can be found in Table B.2 and Table 13.1 in Section 13.3.4.
In Tables B.3 to B.6 the original alternate location-allocation Algorithm ($\mathcal{LA}$) is compared to the enhanced versions ($\mathcal{L}_Y^c\mathcal{A}$) and ($\mathcal{L}_Y^d\mathcal{A}$), where the descent potential π_Y^c and π_Y^d is used, respectively. 10 arbitrarily generated initial solutions were used for both algorithms and the 165 test problems and the resulting objective values were compared. Furthermore, the CPU time of the Algorithm ($\mathcal{LA}$) is related to its specific enhanced version.
Tables B.7 and B.8 and Tables B.9 and B.10 show the comparison between Algorithm ($\mathcal{LA}$) and the enhanced versions ($\mathcal{LA}_X$) and ($\mathcal{L}_Y^d\mathcal{A}_X$), with modified allocation step and modified location and allocation step, respectively. The same setting was used as for the numerical investigations described above.
Table B.11 provides a comparison between Algorithm ($\mathcal{L}_Y^d\mathcal{A}$) and Algorithm ($\mathcal{LA}_X$). Also for this study, the same setting was used as described above.
The best objective value obtained from the three enhanced versions of Algorithm ($\mathcal{LA}$) is compared to the objective value calculated by Algorithm ($\mathcal{LA}$) in Table B.12. All four algorithms were initialized with the same 10 starting solutions and the minimum objective value of the enhanced versions was compared to the objective value of Algorithm ($\mathcal{LA}$).
Table B.13 shows the number of the best objective values obtained for the four different algorithms, initialized with the same starting solution in each run.
Finally, the multi-start version of Algorithm ($\mathcal{LA}$) is compared to the augmented location-allocation Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$) in Table B.14. First, Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$) was started with 10 arbitrarily generated initial solutions and its CPU time was recorded. Then, Algorithm ($\mathcal{LA}$) was used to resolve the problem with at least the same 10 plus additional starting solutions until the CPU time of the enhanced version was reached. The table shows the number of restarts of the different methods and compares the best objective values found for both algorithms.
Note that due to the numerical results from Tables B.3 to B.6, only the descent potential π_Y^d from Definition 13.11 was considered for the augmented location-allocation Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$) with modified location and allocation step throughout the numerical studies.

	Example 1			Example 2			Example 3			Example 4			Example 5			Σ		
Size	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=
1	1	3	6	1	1	8	0	2	8	3	1	6	1	2	7	6	9	35
2	4	4	2	1	2	7	4	3	3	2	1	7	1	1	8	12	11	27
3	3	0	7	0	4	6	1	3	6	1	2	7	4	1	5	9	10	31
4	0	3	7	2	3	5	3	3	4	1	1	8	4	2	4	10	12	28
5	6	2	2	4	3	3	4	2	4	0	4	6	5	3	2	19	14	17
6	2	6	2	3	3	4	4	4	2	4	5	1	4	3	3	17	21	12
7	2	5	3	5	4	1	3	5	2	7	1	2	4	3	3	21	18	11
8	2	1	7	3	5	2	3	4	3	1	4	5	4	1	5	13	15	22
9	3	4	3	1	8	1	4	2	4	1	8	1	4	4	2	13	26	11
10	3	3	4	3	4	3	3	4	3	4	3	3	1	8	1	14	22	14
11	4	6	0	2	7	1	3	4	3	3	4	3	5	5	0	17	26	7
12	5	4	1	6	2	2	5	3	2	4	4	2	5	2	3	25	15	10
13	5	4	1	6	4	0	2	2	6	3	6	1	9	1	0	25	17	8
14	4	6	0	3	5	2	5	4	1	5	5	0	6	3	1	23	23	4
15	4	4	2	3	5	2	6	2	2	5	4	1	5	5	0	23	20	7
16	4	4	2	3	6	1	4	5	1	5	4	1	2	6	2	18	25	7
17	3	6	1	5	5	0	4	6	0	3	5	2	7	2	1	22	24	4
18	5	4	1	4	6	0	2	6	2	3	7	0	5	3	2	19	26	5
19	4	5	1	6	4	0	4	4	2	3	6	1	2	7	1	19	26	5
20	4	6	0	5	4	1	7	3	0	3	7	0	6	4	0	25	24	1
21	5	5	0	7	3	0	4	6	0	5	4	1	5	4	1	26	22	2
22	7	3	0	4	4	2	6	4	0	6	4	0	2	8	0	25	23	2
23	8	2	0	5	4	1	2	8	0	7	3	0	2	6	2	24	23	3
24	4	6	0	4	6	0	7	3	0	5	5	0	5	5	0	25	25	0
25	7	3	0	2	8	0	5	5	0	5	5	0	2	8	0	21	29	0
26	6	4	0	6	4	0	6	4	0	5	5	0	5	5	0	28	22	0
27	5	5	0	4	6	0	5	5	0	7	3	0	5	5	0	26	24	0
28	5	5	0	4	6	0	5	5	0	7	3	0	5	5	0	26	24	0
29	6	4	0	5	5	0	6	4	0	7	3	0	4	6	0	28	22	0
30	5	5	0	4	6	0	5	5	0	5	5	0	4	6	0	23	27	0
31	4	6	0	4	6	0	6	4	0	5	5	0	6	4	0	25	25	0
32	6	4	0	4	6	0	5	5	0	5	5	0	6	4	0	26	24	0
33	4	6	0	2	8	0	4	6	0	3	7	0	4	6	0	17	33	0

Table B.3: *Comparison of the objective values obtained from* 10 *runs of Algorithm* ($\mathcal{LA}$) *and Algorithm* ($\mathcal{L}_Y^c\mathcal{A}$). *Each run is initialized with the same starting solution. Total:* 1650 *runs, (+)* F_Y^c *better than* F*:* 670*, (−)* F_c^Y *worse than* F*:* 707*, (=)* F_y^c *equals to* F*:* 273

	Algorithm ($\mathcal{LA}$)			Algorithm ($\mathcal{L}^c_Y\mathcal{A}$)			Quotient
Size	$t^1_{\min}$	$t^1_{\max}$	t^1_{av}	$t^2_{\min}$	$t^2_{\max}$	t^2_{av}	t^2_{av}/t^1_{av}
1	0.00202	0.12928	0.00926	0.00194	0.10292	0.01135	1.22579
2	0.00444	0.03449	0.01288	0.00372	0.11248	0.04183	3.24751
3	0.00256	0.04628	0.01817	0.00253	0.13089	0.04078	2.24409
4	0.00627	0.03682	0.02086	0.00664	0.23592	0.07306	3.50232
5	0.01383	0.05476	0.02903	0.03828	0.28966	0.16029	5.52241
6	0.01282	0.09801	0.04099	0.01270	0.72664	0.25200	6.14792
7	0.02062	0.09790	0.05089	0.04083	0.63029	0.30743	6.04153
8	0.02260	0.10084	0.04614	0.06389	0.79103	0.32030	6.94232
9	0.01269	0.18089	0.05820	0.03335	0.88089	0.36994	6.35661
10	0.03137	0.12430	0.07270	0.25549	7.19239	0.69825	9.60439
11	0.03945	0.20553	0.08683	0.28116	1.83848	0.88417	10.18255
12	0.05312	0.18258	0.11258	0.43016	1.66408	1.11086	9.86739
13	0.06067	0.23709	0.10129	0.17352	2.56472	1.22766	12.11992
14	0.03626	0.25404	0.12778	0.19320	3.03597	1.33424	10.44214
15	0.08448	0.30821	0.15859	0.64888	4.26629	1.92818	12.15841
16	0.05621	0.27581	0.18813	0.93691	4.27380	2.18772	11.62898
17	0.14207	0.36207	0.23575	1.52903	5.41847	3.38712	14.36737
18	0.14741	0.55744	0.27688	1.66121	8.40566	4.00625	14.46904
19	0.23417	0.64870	0.32816	3.02130	10.23641	5.48388	16.71115
20	0.24990	0.86559	0.43395	3.46021	12.51452	7.36274	16.96667
21	0.23188	0.86929	0.45479	3.23481	16.56833	7.59787	16.70623
22	0.32229	1.20807	0.76569	6.60261	33.67643	15.95169	20.83301
23	0.58187	1.95168	0.97011	11.87721	40.02030	20.92713	21.57184
24	0.51736	2.01814	1.27666	16.26050	48.88119	29.68611	23.25302
25	1.03183	3.56492	1.77760	16.29898	89.68059	41.20576	23.18059
26	1.59495	4.71122	2.74198	40.45853	136.52315	66.25800	24.16431
27	1.94021	7.82469	3.32722	51.56203	194.32025	93.63637	28.14253
28	2.18603	6.30494	3.93397	52.51465	245.86611	114.98436	29.22859
29	3.66957	11.81777	5.59468	79.22246	393.26676	183.41932	32.78458
30	4.52615	16.78956	7.77156	130.02252	465.72653	244.55159	31.46752
31	5.54644	19.01240	9.28082	144.99561	479.48568	310.28586	33.43302
32	7.69273	30.39403	13.90885	203.82022	1096.73726	533.83034	38.38062
33	11.38564	41.83649	23.84498	365.19107	1849.56429	847.10644	35.52556

Table B.4: *Comparison of the CPU time of Algorithm ($\mathcal{LA}$) (t^1) and Algorithm ($\mathcal{L}^c_Y\mathcal{A}$) ($t^2$).*

	Example 1			Example 2			Example 3			Example 4			Example 5			Σ		
Size	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=
1	1	1	8	1	0	9	0	1	9	1	0	9	1	0	9	4	2	44
2	4	1	5	0	1	9	3	3	4	2	1	7	2	1	7	11	7	32
3	2	0	8	3	0	7	2	1	7	1	1	8	2	1	7	10	3	37
4	1	0	9	4	0	6	4	0	6	4	1	5	2	4	4	15	5	30
5	6	1	3	4	1	5	4	1	5	4	2	4	5	1	4	23	6	21
6	3	2	5	2	4	4	3	5	2	3	5	2	1	5	4	12	21	17
7	3	6	1	4	4	2	4	5	1	8	0	2	2	6	2	21	21	8
8	3	4	3	6	3	1	6	2	2	7	1	2	4	3	3	26	13	11
9	4	4	2	4	5	1	6	4	0	3	5	2	5	5	0	22	23	5
10	8	2	0	1	6	3	0	5	5	3	5	2	4	5	1	16	23	11
11	5	4	1	3	6	1	6	3	1	3	6	1	5	4	1	22	23	5
12	5	5	0	3	6	1	6	3	1	3	6	1	6	3	1	23	23	4
13	4	5	1	5	5	0	4	2	4	4	5	1	7	3	0	24	20	6
14	6	4	0	5	5	0	6	4	0	4	5	1	7	2	1	28	20	2
15	5	5	0	6	2	2	4	5	1	6	4	0	4	5	1	25	21	4
16	6	4	0	4	6	0	7	3	0	5	5	0	6	4	0	28	22	0
17	8	2	0	6	4	0	5	5	0	5	5	0	9	1	0	33	17	0
18	4	6	0	7	3	0	7	2	1	6	4	0	3	7	0	27	22	1
19	6	4	0	9	1	0	5	3	2	5	5	0	5	5	0	30	18	2
20	4	6	0	6	3	1	3	7	0	3	6	1	8	2	0	24	24	2
21	5	5	0	8	2	0	4	6	0	4	6	0	5	3	2	26	22	2
22	6	4	0	6	4	0	7	3	0	5	5	0	6	4	0	30	20	0
23	5	5	0	4	6	0	4	6	0	4	6	0	7	3	0	24	26	0
24	4	6	0	6	4	0	6	4	0	4	6	0	5	5	0	25	25	0
25	6	4	0	3	7	0	8	2	0	3	7	0	8	2	0	28	22	0
26	4	6	0	3	7	0	4	6	0	6	4	0	4	6	0	21	29	0
27	6	4	0	8	2	0	4	6	0	7	3	0	7	3	0	32	18	0
28	3	7	0	7	3	0	5	5	0	6	4	0	7	3	0	28	22	0
29	6	4	0	8	2	0	4	6	0	6	4	0	8	2	0	32	18	0
30	6	4	0	4	6	0	5	5	0	5	5	0	6	4	0	26	24	0
31	5	5	0	5	5	0	4	6	0	6	4	0	5	5	0	25	25	0
32	5	5	0	6	4	0	8	2	0	7	3	0	6	4	0	32	18	0
33	7	3	0	4	6	0	4	6	0	4	6	0	5	5	0	24	26	0

Table B.5: *Comparison of the objective values obtained from* 10 *runs of Algorithm* ($\mathcal{LA}$) *and Algorithm* ($\mathcal{L}_Y^d\mathcal{A}$). *Each run is initialized with the same starting solution. Total:* 1650 *runs, (+)* F_y^d *better than* F*:* 777*, (−)* F_y^d *worse than* F*:* 629*, (=)* F_y^d *is equal to* F*:* 244

	Algorithm ($\mathcal{LA}$)			Algorithm ($\mathcal{L}_Y^d\mathcal{A}$)			Quotient
Size	$t^1_{\min}$	$t^1_{\max}$	t^1_{av}	$t^2_{\min}$	$t^2_{\max}$	t^2_{av}	t^2_{av}/t^1_{av}
1	0.00203	0.15622	0.01247	0.00182	0.04827	0.02014	1.61507
2	0.00537	0.04749	0.01716	0.00849	0.09567	0.03774	2.19992
3	0.00253	0.05891	0.02263	0.00240	0.11642	0.04251	1.87902
4	0.00665	0.04965	0.02788	0.00949	0.12321	0.04872	1.74782
5	0.01543	0.06790	0.03863	0.02747	0.19888	0.09591	2.48280
6	0.01516	0.12451	0.05176	0.03203	0.33929	0.11662	2.25317
7	0.02464	0.11435	0.06339	0.05010	0.30117	0.16765	2.64451
8	0.02721	0.14392	0.05700	0.05666	0.37335	0.17315	3.03792
9	0.01566	0.22697	0.07561	0.05671	0.51424	0.25272	3.34232
10	0.03614	0.15069	0.09338	0.09111	0.51198	0.29979	3.21028
11	0.04639	0.26646	0.10946	0.14357	0.65335	0.34895	3.18785
12	0.06539	0.22102	0.13641	0.20950	0.94387	0.48278	3.53916
13	0.07684	0.27900	0.13738	0.12449	0.97735	0.45593	3.31868
14	0.04093	0.30161	0.16035	0.32665	1.03774	0.61257	3.82029
15	0.09730	0.37022	0.19130	0.36758	1.52182	0.71184	3.72096
16	0.07275	0.35323	0.23096	0.34573	1.52923	0.93367	4.04248
17	0.16164	0.43753	0.28513	0.55509	2.05087	1.09414	3.83736
18	0.18200	0.68000	0.33861	0.73672	2.38536	1.56006	4.60724
19	0.28029	0.75252	0.39737	0.93070	3.48230	1.87620	4.72148
20	0.28739	1.04054	0.51621	1.05160	4.12826	2.40885	4.66644
21	0.30740	1.06044	0.53631	1.45248	6.52625	2.87554	5.36174
22	0.40165	1.49979	0.92336	2.46241	9.61456	4.53446	4.91082
23	0.67683	2.30787	1.15826	4.08509	13.75457	6.66724	5.75625
24	0.59247	2.26815	1.49549	4.88890	22.03364	8.94810	5.98337
25	1.19778	4.01605	2.01867	6.93540	27.32444	12.04877	5.96867
26	1.89359	5.58069	3.19349	10.58886	32.97860	18.46635	5.78250
27	2.25508	9.18251	3.83377	14.82927	44.79809	26.70266	6.96513
28	2.57302	7.39033	4.59172	18.20970	52.29282	29.14138	6.34650
29	4.30439	13.35776	6.48891	27.54375	100.36404	38.34960	5.91002
30	5.30405	18.58331	8.85210	26.18843	119.41319	55.49840	6.26952
31	6.33473	21.23526	10.70743	42.02928	130.83510	64.88547	6.05985
32	8.60766	33.95924	15.66010	55.97478	215.94925	95.59561	6.10441
33	12.84767	46.04262	26.77315	71.05779	310.12853	164.44391	6.14212

Table B.6: *Comparison of the CPU time of Algorithm ($\mathcal{LA}$) (t^1) and Algorithm ($\mathcal{L}_Y^d\mathcal{A}$) ($t^2$).*

	Example 1			Example 2			Example 3			Example 4			Example 5			Σ		
Size	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=
1	1	9	0	1	5	4	0	8	2	1	9	0	0	6	4	3	37	10
2	0	10	0	2	4	4	4	6	0	3	4	3	4	1	5	13	25	12
3	1	5	4	5	4	1	4	6	0	4	4	2	2	6	2	16	25	9
4	4	5	1	4	6	0	2	6	2	3	4	3	2	7	1	15	28	7
5	4	5	1	3	6	1	4	4	2	1	8	1	4	6	0	16	29	5
6	3	7	0	6	4	0	3	7	0	7	3	0	3	7	0	22	28	0
7	2	8	0	5	5	0	6	4	0	5	5	0	4	6	0	22	28	0
8	5	5	0	6	4	0	5	5	0	8	2	0	4	6	0	28	22	0
9	2	8	0	6	4	0	4	6	0	6	4	0	3	7	0	21	29	0
10	3	7	0	3	7	0	3	7	0	4	6	0	7	3	0	20	30	0
11	5	5	0	2	8	0	9	1	0	3	7	0	7	3	0	26	24	0
12	3	7	0	4	6	0	2	8	0	1	9	0	4	6	0	14	36	0
13	3	6	1	7	3	0	7	3	0	4	6	0	7	3	0	28	21	1
14	5	5	0	2	8	0	4	6	0	4	6	0	5	5	0	20	30	0
15	2	8	0	5	5	0	4	6	0	7	3	0	4	6	0	22	28	0
16	7	3	0	2	8	0	4	6	0	5	5	0	7	3	0	25	25	0
17	5	5	0	5	5	0	4	6	0	7	3	0	5	5	0	26	24	0
18	3	7	0	2	8	0	5	5	0	6	4	0	3	7	0	19	31	0
19	7	3	0	5	5	0	6	4	0	8	2	0	6	4	0	32	18	0
20	6	4	0	6	4	0	4	6	0	6	4	0	5	5	0	27	23	0
21	5	5	0	8	2	0	6	4	0	5	5	0	7	3	0	31	19	0
22	5	5	0	5	5	0	6	4	0	7	3	0	6	4	0	29	21	0
23	7	3	0	6	4	0	8	2	0	7	3	0	9	1	0	37	13	0
24	9	1	0	9	1	0	8	2	0	7	3	0	8	2	0	41	9	0
25	7	3	0	5	5	0	7	3	0	9	1	0	8	2	0	36	14	0
26	2	8	0	9	1	0	6	4	0	10	0	0	7	3	0	34	16	0
27	7	3	0	4	6	0	6	4	0	6	4	0	8	2	0	31	19	0
28	10	0	0	8	2	0	7	3	0	7	3	0	6	4	0	38	12	0
29	8	2	0	7	3	0	8	2	0	4	6	0	8	2	0	35	15	0
30	7	3	0	6	4	0	5	5	0	4	6	0	3	7	0	25	25	0
31	3	7	0	3	7	0	8	2	0	6	4	0	6	4	0	26	24	0
32	4	6	0	9	1	0	9	1	0	6	4	0	7	3	0	35	15	0
33	9	1	0	6	4	0	7	3	0	5	5	0	5	5	0	32	18	0

Table B.7: *Comparison of the objective values obtained from* 10 *runs of Algorithm* ($\mathcal{LA}$) *and Algorithm* ($\mathcal{LA}_X$). *Each run is initialized with the same starting solution. Total:* 1650 *runs, (+)* F_X *better than* F*:* 845*, (−)* F_X *worse than* F*:* 761*, (=)* F_X *is equal to* F*:* 44

	Algorithm ($\mathcal{LA}$)			Algorithm ($\mathcal{LA}_X$)			Quotient
Size	$t^1_{\min}$	$t^1_{\max}$	t^1_{av}	$t^2_{\min}$	$t^2_{\max}$	t^2_{av}	t^2_{av}/t^1_{av}
1	0.00203	0.15622	0.01247	0.00100	0.03741	0.01279	1.02565
2	0.00537	0.04749	0.01716	0.00401	0.05118	0.01542	0.89861
3	0.00253	0.05891	0.02263	0.00285	0.06293	0.02683	1.18560
4	0.00665	0.04965	0.02788	0.00852	0.10013	0.03223	1.15630
5	0.01543	0.06790	0.03863	0.01801	0.09037	0.05090	1.31766
6	0.01516	0.12451	0.05176	0.02165	0.14004	0.06205	1.19883
7	0.02464	0.11435	0.06339	0.03857	0.16451	0.08608	1.35790
8	0.02721	0.14392	0.05700	0.05663	0.14982	0.09401	1.64937
9	0.01566	0.22697	0.07561	0.06668	0.23095	0.13729	1.81566
10	0.03614	0.15069	0.09338	0.10210	0.22619	0.17322	1.85491
11	0.04639	0.26646	0.10946	0.09764	0.39086	0.23472	2.14431
12	0.06539	0.22102	0.13641	0.15803	0.35926	0.25870	1.89649
13	0.07684	0.27900	0.13738	0.20334	0.54651	0.32394	2.35791
14	0.04093	0.30161	0.16035	0.16037	0.64148	0.38032	2.37185
15	0.09730	0.37022	0.19130	0.36381	0.72715	0.53752	2.80976
16	0.07275	0.35323	0.23096	0.39611	0.92507	0.61520	2.66362
17	0.16164	0.43753	0.28513	0.58540	1.29219	0.90071	3.15896
18	0.18200	0.68000	0.33861	0.91138	1.90129	1.30230	3.84601
19	0.28029	0.75252	0.39737	1.27363	2.95213	1.88111	4.73384
20	0.28739	1.04054	0.51621	1.71289	4.59404	2.58536	5.00838
21	0.30740	1.06044	0.53631	1.90339	6.83663	3.59644	6.70595
22	0.40165	1.49979	0.92336	4.52538	17.60455	8.49477	9.19984
23	0.67683	2.30787	1.15826	7.00516	23.99419	16.26965	14.04661
24	0.59247	2.26815	1.49549	19.42227	41.92320	30.09866	20.12623
25	1.19778	4.01605	2.01867	34.90645	85.74217	47.57345	23.56673
26	1.89359	5.58069	3.19349	34.36743	150.61091	89.86794	28.14102
27	2.25508	9.18251	3.83377	77.88469	234.72838	149.14225	38.90229
28	2.57302	7.39033	4.59172	117.46639	385.85798	238.05463	51.84428
29	4.30439	13.35776	6.48891	102.68581	604.01725	347.25682	53.51540
30	5.30405	18.58331	8.85210	329.40807	857.16449	555.45344	62.74819
31	6.33473	21.23526	10.70743	367.46909	1047.9815	694.93835	64.90243
32	8.60766	33.95924	15.66010	572.04506	1976.6717	1319.2815	84.24476
33	12.84767	46.04262	26.77315	842.84768	3865.3038	2657.1199	99.24568

Table B.8: *Comparison of the CPU time of Algorithm ($\mathcal{LA}$) (t^1) and Algorithm ($\mathcal{LA}_X$) (t^2).*

	Example 1			Example 2			Example 3			Example 4			Example 5			Σ		
Size	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=
1	2	8	0	0	6	4	0	8	2	1	8	1	1	4	5	4	34	12
2	0	10	0	2	3	5	2	8	0	4	3	3	4	2	4	12	26	12
3	1	5	4	4	5	1	7	3	0	5	4	1	3	5	2	20	22	8
4	3	6	1	4	6	0	1	6	3	2	6	2	1	9	0	11	33	6
5	7	3	0	3	7	0	4	5	1	1	8	1	4	6	0	19	29	2
6	4	6	0	5	5	0	3	7	0	6	4	0	2	8	0	20	30	0
7	3	7	0	5	5	0	6	4	0	6	4	0	1	9	0	21	29	0
8	3	7	0	8	2	0	3	7	0	4	6	0	4	5	1	22	27	1
9	2	8	0	6	4	0	2	8	0	5	5	0	6	4	0	21	29	0
10	4	6	0	1	9	0	3	7	0	5	5	0	7	3	0	20	30	0
11	4	6	0	4	6	0	6	4	0	4	6	0	6	4	0	24	26	0
12	6	4	0	3	7	0	5	5	0	2	8	0	4	6	0	20	30	0
13	5	4	1	7	3	0	6	4	0	4	6	0	8	2	0	30	19	1
14	6	4	0	1	9	0	2	8	0	5	5	0	6	4	0	20	30	0
15	5	5	0	7	3	0	7	3	0	5	5	0	2	8	0	26	24	0
16	6	4	0	3	7	0	4	6	0	7	3	0	4	6	0	24	26	0
17	5	5	0	5	5	0	4	6	0	8	2	0	8	2	0	30	20	0
18	3	7	0	5	5	0	5	5	0	6	4	0	6	4	0	25	25	0
19	6	4	0	5	5	0	4	6	0	7	3	0	5	5	0	27	23	0
20	6	4	0	7	3	0	6	4	0	4	6	0	7	3	0	30	20	0
21	5	5	0	5	5	0	7	3	0	5	5	0	4	6	0	26	24	0
22	6	4	0	5	5	0	9	1	0	5	5	0	7	3	0	32	18	0
23	7	3	0	7	3	0	7	3	0	8	2	0	10	0	0	39	11	0
24	6	4	0	8	2	0	6	4	0	6	4	0	9	1	0	35	15	0
25	7	3	0	5	5	0	7	3	0	8	2	0	9	1	0	36	14	0
26	3	7	0	8	2	0	8	2	0	9	1	0	7	3	0	35	15	0
27	7	3	0	5	5	0	6	4	0	9	1	0	7	3	0	34	16	0
28	8	2	0	7	3	0	6	4	0	5	5	0	6	4	0	32	18	0
29	4	6	0	8	2	0	8	2	0	3	7	0	8	2	0	31	19	0
30	7	3	0	6	4	0	7	3	0	8	2	0	4	6	0	32	18	0
31	4	6	0	5	5	0	9	1	0	6	4	0	7	3	0	31	19	0
32	8	2	0	9	1	0	9	1	0	4	6	0	6	4	0	36	14	0
33	10	0	0	7	3	0	8	2	0	9	1	0	3	7	0	37	13	0

Table B.9: *Comparison of the objective values obtained from* 10 *runs of Algorithm* ($\mathcal{LA}$) *and Algorithm* ($\mathcal{L}_Y^d\mathcal{A}_X$). *Each run is initialized with the same starting solution. Total:* 1650 *runs, (+)* F_{XY}^d *better than* F*:* 862*, (−)* F_{XY}^d *worse than* F*:* 746*, (=)* F_{XY}^d *is equal to* F*:* 42

	Algorithm ($\mathcal{LA}$)			Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$)			Quotient
Size	$t^1_{\min}$	$t^1_{\max}$	t^1_{av}	$t^2_{\min}$	$t^2_{\max}$	t^2_{av}	t^2_{av}/t^1_{av}
1	0.00203	0.15622	0.01247	0.00110	0.05326	0.02218	1.77839
2	0.00537	0.04749	0.01716	0.01375	0.09477	0.03587	2.09097
3	0.00253	0.05891	0.02263	0.00297	0.20722	0.05294	2.33992
4	0.00665	0.04965	0.02788	0.01482	0.14048	0.06591	2.36452
5	0.01543	0.06790	0.03863	0.04405	0.20562	0.10866	2.81283
6	0.01516	0.12451	0.05176	0.03691	0.36322	0.13330	2.57530
7	0.02464	0.11435	0.06339	0.06756	0.45571	0.18934	2.98675
8	0.02721	0.14392	0.05700	0.10121	0.41049	0.22291	3.91082
9	0.01566	0.22697	0.07561	0.12313	0.49436	0.28145	3.72232
10	0.03614	0.15069	0.09338	0.17876	0.59532	0.36899	3.95133
11	0.04639	0.26646	0.10946	0.27244	0.94478	0.48588	4.43884
12	0.06539	0.22102	0.13641	0.33556	0.94292	0.55356	4.05799
13	0.07684	0.27900	0.13738	0.44589	1.25720	0.75318	5.48230
14	0.04093	0.30161	0.16035	0.39069	1.12334	0.80814	5.03992
15	0.09730	0.37022	0.19130	0.70727	1.93294	1.10880	5.79599
16	0.07275	0.35323	0.23096	0.81653	2.07075	1.21832	5.27495
17	0.16164	0.43753	0.28513	0.94886	2.35689	1.82383	6.39654
18	0.18200	0.68000	0.33861	1.59050	3.49585	2.59837	7.67364
19	0.28029	0.75252	0.39737	2.23993	5.78489	3.21206	8.08321
20	0.28739	1.04054	0.51621	2.83969	7.27309	4.53987	8.79466
21	0.30740	1.06044	0.53631	3.76659	9.53091	6.36103	11.86082
22	0.40165	1.49979	0.92336	8.12776	19.89230	11.98379	12.97845
23	0.67683	2.30787	1.15826	9.95763	31.19271	21.37547	18.45479
24	0.59247	2.26815	1.49549	14.54023	62.52007	40.41172	27.02231
25	1.19778	4.01605	2.01867	40.86355	110.08388	67.06312	33.22144
26	1.89359	5.58069	3.19349	38.82490	174.79150	106.28305	33.28120
27	2.25508	9.18251	3.83377	100.57446	259.97573	186.93286	48.75960
28	2.57302	7.39033	4.59172	115.50853	480.18852	268.03126	58.37268
29	4.30439	13.35776	6.48891	172.16565	696.18293	403.89053	62.24316
30	5.30405	18.58331	8.85210	175.92486	832.79162	580.43482	65.57027
31	6.33473	21.23526	10.70743	434.11182	1238.3994	799.00967	74.62197
32	8.60766	33.95924	15.66010	528.01068	2232.8246	1439.4020	91.91524
33	12.84767	46.04262	26.77315	1183.7017	4138.6535	2707.6997	101.13488

Table B.10: *Comparison of the CPU time of Algorithm ($\mathcal{LA}$) (t^1) and Algorithm ($\mathcal{L}_Y^d\mathcal{A}_X$)($t^2$).*

	Example 1			Example 2			Example 3			Example 4			Example 5			Σ		
Size	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=
1	9	1	0	5	0	5	8	0	2	10	0	0	6	0	4	38	1	11
2	10	0	0	4	2	4	7	3	0	5	3	2	1	4	5	27	12	11
3	6	1	3	4	5	1	6	4	0	4	4	2	7	2	1	27	16	7
4	6	3	1	6	4	0	6	2	2	5	1	4	8	2	0	31	12	7
5	4	5	1	6	3	1	6	3	1	9	1	0	6	4	0	31	16	3
6	6	4	0	3	7	0	6	4	0	4	6	0	6	3	1	25	24	1
7	8	2	0	5	5	0	4	6	0	6	4	0	5	5	0	28	22	0
8	5	5	0	4	6	0	5	5	0	4	6	0	9	1	0	27	23	0
9	7	3	0	3	7	0	8	2	0	4	6	0	6	4	0	28	22	0
10	8	1	1	8	2	0	7	3	0	5	5	0	3	7	0	31	18	1
11	6	4	0	4	6	0	2	8	0	5	5	0	4	6	0	21	29	0
12	7	3	0	6	4	0	8	2	0	5	5	0	8	2	0	34	16	0
13	5	4	1	4	6	0	2	8	0	5	5	0	2	8	0	18	31	1
14	7	3	0	6	4	0	6	4	0	6	4	0	7	3	0	32	18	0
15	6	4	0	6	4	0	7	3	0	3	7	0	4	6	0	26	24	0
16	4	6	0	7	3	0	6	4	0	4	6	0	4	6	0	25	25	0
17	6	4	0	5	5	0	5	5	0	4	6	0	6	4	0	26	24	0
18	9	1	0	8	2	0	4	6	0	4	6	0	6	4	0	31	19	0
19	3	7	0	6	4	0	5	5	0	4	6	0	4	6	0	22	28	0
20	5	5	0	6	4	0	5	5	0	4	6	0	5	5	0	25	25	0
21	4	6	0	4	6	0	5	5	0	5	5	0	3	7	0	21	29	0
22	4	6	0	5	5	0	7	3	0	5	5	0	5	5	0	26	24	0
23	5	5	0	5	5	0	5	5	0	4	6	0	3	7	0	22	28	0
24	1	9	0	1	9	0	3	7	0	3	7	0	3	7	0	11	39	0
25	4	6	0	3	7	0	3	7	0	1	9	0	3	7	0	14	36	0
26	8	2	0	1	9	0	4	6	0	2	8	0	2	8	0	17	33	0
27	3	7	0	6	4	0	1	9	0	4	6	0	4	6	0	18	32	0
28	1	9	0	2	8	0	4	6	0	2	8	0	7	3	0	16	34	0
29	2	8	0	4	6	0	2	8	0	7	3	0	3	7	0	18	32	0
30	4	6	0	5	5	0	2	8	0	4	6	0	6	4	0	21	29	0
31	7	3	0	7	3	0	3	7	0	4	6	0	2	8	0	23	27	0
32	6	4	0	4	6	0	3	7	0	7	3	0	5	5	0	25	25	0
33	5	5	0	2	8	0	1	9	0	5	5	0	4	6	0	17	33	0

Table B.11: *Comparison of the objective values obtained from* 10 *runs of Algorithm* $(\mathcal{L}_Y^d\mathcal{A})$ *and Algorithm* $(\mathcal{L}\mathcal{A}_X)$*. Each run is initialized with the same starting solution. Total:* 1650 *runs, (+)* F_Y^d *better than* F_X*:* 802*, (−)* F_Y^d *worse than* F_X*:* 806*, (=)* F_Y^d *is equal to* F_X*:* 42

	Example 1			Example 2			Example 3			Example 4			Example 5			Σ		
Size	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=	+	−	=
1	2	1	7	1	0	9	0	1	9	1	0	9	2	0	8	6	2	42
2	4	1	5	2	1	7	5	2	3	6	0	4	5	1	4	22	5	23
3	2	0	8	8	0	2	7	0	3	6	0	4	4	0	6	27	0	23
4	4	0	6	7	0	3	6	0	4	6	0	4	3	2	5	26	2	22
5	9	0	1	4	1	5	5	0	5	5	0	5	8	0	2	31	1	18
6	6	1	3	7	2	1	6	4	0	7	1	2	4	3	3	30	11	9
7	5	5	0	7	2	1	6	3	1	8	0	2	6	3	1	32	13	5
8	7	1	2	9	1	0	9	0	1	9	1	0	5	1	4	39	4	7
9	6	2	2	7	2	1	6	4	0	7	2	1	8	2	0	34	12	4
10	8	2	0	3	5	2	3	4	3	6	4	0	8	2	0	28	17	5
11	8	2	0	5	4	1	10	0	0	5	4	1	8	2	0	36	12	2
12	9	1	0	7	3	0	8	1	1	4	5	1	6	3	1	34	13	3
13	7	2	1	8	2	0	9	1	0	7	3	0	9	1	0	40	9	1
14	8	2	0	5	5	0	7	3	0	7	3	0	8	1	1	35	14	1
15	7	3	0	8	0	2	7	2	1	9	1	0	6	4	0	37	10	3
16	8	2	0	6	4	0	8	2	0	9	1	0	8	2	0	39	11	0
17	9	1	0	8	2	0	7	3	0	10	0	0	10	0	0	44	6	0
18	8	2	0	8	2	0	8	2	0	9	1	0	7	3	0	40	10	0
19	9	1	0	9	1	0	6	2	2	10	0	0	8	2	0	42	6	2
20	7	3	0	10	0	0	7	3	0	7	3	0	9	1	0	40	10	0
21	9	1	0	10	0	0	9	1	0	7	3	0	8	2	0	43	7	0
22	7	3	0	9	1	0	9	1	0	7	3	0	10	0	0	42	8	0
23	9	1	0	8	2	0	9	1	0	8	2	0	10	0	0	44	6	0
24	10	0	0	9	1	0	9	1	0	9	1	0	10	0	0	47	3	0
25	8	2	0	6	4	0	9	1	0	9	1	0	10	0	0	42	8	0
26	6	4	0	10	0	0	8	2	0	10	0	0	8	2	0	42	8	0
27	9	1	0	9	1	0	9	1	0	10	0	0	10	0	0	47	3	0
28	10	0	0	9	1	0	8	2	0	8	2	0	8	2	0	43	7	0
29	10	0	0	10	0	0	9	1	0	9	1	0	9	1	0	47	3	0
30	9	1	0	9	1	0	8	2	0	9	1	0	6	4	0	41	9	0
31	7	3	0	8	2	0	9	1	0	9	1	0	9	1	0	42	8	0
32	9	1	0	10	0	0	10	0	0	9	1	0	10	0	0	48	2	0
33	10	0	0	9	1	0	8	2	0	10	0	0	5	5	0	42	8	0

Table B.12: *Comparison of the objective values obtained from* 10 *runs of Algorithm* $(\mathcal{LA})$ *and the best objective value obtained from Algorithms* $(\mathcal{L}_Y^d\mathcal{A})$, $(\mathcal{LA}_X)$ *and* $(\mathcal{L}_Y^d\mathcal{A}_X)$. *Total:* 1650 *runs, (+)* $\min\{F_Y^d, F_X, F_{XY}^d\}$ *better than* F*:* 1232*, (−)* $\min\{F_Y^d, F_X, F_{XY}^d\}$ *worse than* F*:* 248*, (=)* $\min\{F_Y^d, F_X, F_{XY}^d\}$ *is equal to* F*:* 170

Size	A	B	C	D
1	44	44	12	14
2	28	34	20	22
3	23	32	20	23
4	24	35	18	12
5	19	31	15	12
6	20	15	16	15
7	18	13	13	14
8	11	19	15	14
9	16	16	11	15
10	22	14	12	9
11	14	12	14	12

Size	A	B	C	D
12	16	13	6	18
13	10	11	19	14
14	15	16	11	9
15	13	15	11	15
16	11	11	11	17
17	6	14	11	20
18	10	12	10	18
19	8	16	13	15
20	10	13	13	14
21	7	7	19	17
22	8	9	10	23

Size	A	B	C	D
23	6	7	11	26
24	3	6	21	20
25	8	5	18	19
26	8	6	18	18
27	3	9	20	18
28	7	7	20	16
29	3	9	22	16
30	9	9	15	17
31	8	12	12	18
32	2	13	12	23
33	8	8	19	15

Table B.13: *Number of the best objective values found by the Algorithms* $(\mathcal{LA})$, $(\mathcal{L}_Y^d\mathcal{A})$, $(\mathcal{LA}_X)$ *and* $(\mathcal{L}_Y^d\mathcal{A}_X)$*.Total:* 1650 *runs, (A)* F *best:* 418*, (B)* F_Y^d *best:* 493*, (C)* F_X *best:* 488*, (D)* F_{XY}^d *best:* 548*.*

	Alg. $(\mathcal{LA})$		Alg. $(\mathcal{L}_Y^d\mathcal{A}_X)$	
Size	better	restarts	better	restarts
15	5	57	0	10
16	4	60	1	10
17	2	63	3	10
18	5	67	0	10
19	2	83	3	10
20	4	88	1	10
21	3	122	2	10
22	4	141	1	10
23	3	171	2	10
24	2	251	3	10

	Alg. $(\mathcal{LA})$		Alg. $(\mathcal{L}_Y^d\mathcal{A}_X)$	
Size	better	restarts	better	restarts
25	5	316	0	10
26	2	301	3	10
27	3	419	2	10
28	4	598	1	10
29	4	525	1	10
30	4	617	1	10
31	5	719	0	10
32	4	841	1	10
33	5	982	0	10

Table B.14: *Comparison of the objective values obtained from the multi-start version of Algorithm* $(\mathcal{LA})$ *and Algorithm* $(\mathcal{L}_Y^d\mathcal{A}_X)$ *for problem size 15 to 33. The modified version was started with* 10 *arbitrary starting solutions and its CPU time was recorded. Then, Algorithm* $(\mathcal{LA})$ *was started with the* 10 *starting solutions also used for the enhanced version of the algorithm plus additional starting solutions until the CPU time of the enhanced version was reached. The total number of restarts can also be found in the table.* F *better than* F_{XY}^d*:* 70*,* F_{XY}^d *better than* F*:* 25

Bibliography

[1] Aggarwal, V. (1985). A lagrangean-relaxation method for the constrained assignment problem. *Computers & Operations Research*, 12(1):97–106.

[2] Aggarwal, V., Aneja, Y. P., and Nair, K. P. K. (1982). Minimal spanning tree subject to a side constraint. *Computers & Operations Research*, 9(4):287–296.

[3] Ahuja, R. K. (1997). The balanced linear programming problem. *European Journal of Operational Research*, 101:29–38.

[4] Al-Khayyal, F. (1990). Jointly constrained bilinear programs and related problems: An overview. *Computers in Mathematical Applications*, 19(11):53–62.

[5] Al-Khayyal, F. and Falk, J. (1983). Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286.

[6] Alumur, S. and Kara, B. (2008). Network hub location problems: the state of the art. *European Journal of Operations Research*, 190(1):1–21.

[7] Aneja, P. Y. and Nair, K. P. K. (1978). The constrained shortest path problem. *Naval Research Logistics Quarterly*, 25(3):549–555.

[8] Audet, C., Hansen, P., Jaumard, B., and Savard, G. (2000). A branch and cut algorithm for non-convex quadratically constrained quadratic programming. *Mathematical Programming, Series A*, 87(1):131–152.

[9] Aumann, R. and Hart, S. (1986). Bi-convexity and bi-martingales. *Israel Journal of Mathematics*, 54(2):159–180.

[10] Balinski, M. L. and Russakoff, A. (1974). On the assignment polytope. *SIAM Review*, 16:516–525.

[11] Barmish, B. (1994). *New Tools for Robustness of Linear Systems*. Maxwell, Macmillan International, New York.

[12] Barmish, B., Floudas, C., Hollot, C., and Tempo, R. (1995). A global programming solution to some open robustness problems including matrix polytope stability. In *Proceedings of the American Control Conference Seattle, Washington*, pages 3871–3877.

[13] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. (2006). *Nonlinear Programming - Theory and Algorithms*. John Wiley & Sons, Inc., New York, third edition.

[14] Bazgan, C., Hugot, H., and Vanderpooten, D. (2009). Solving efficiently the 0-1 multi-objective knapsack problem. *Computers and Operations Research*, 36(1):260–279.

[15] Beasley, J. and Christofides, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394.

[16] Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252.

[17] Berman, O., Einav, D., and Handler, G. (1990). The constrained bottleneck problem in network. *Operations Research*, 38:178–181.

[18] Bertsekas, D. P. and Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation*. Prentice-Hall International Editions, Englewood Cliffs, NJ.

[19] Besl, P. and McKay, N. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:239–256.

[20] Bischoff, M. (2008). *Location of Connection Facilities*. Shaker Verlag GmbH, Germany. Ph.D.Thesis.

[21] Bischoff, M. and Bayer, Y. (2007). Construction line algorithms for the connection location-allocation problem with restrictions. *OR Proceedings*, Volume 2007(XV):345–350.

[22] Bischoff, M. and Dächert, K. (2009). Allocation search methods for a generalized class of location-allocation problems. *European Journal of Operational Research*, 192(3):793–807.

[23] Bischoff, M. and Klamroth, K. (2007). Two branch & bound methods for generalized classes of location-allocation problems. In Paias, A. and Saldanha da Gama, F., editors, *Proceedings on the EURO Winter Institute on Location and Logistics*, pages 45–61.

[24] Borwein, J. (1986). Partially monotone operators and the generic differentiability of convex-concave and biconvex mappings. *Israel Journal of Mathematics*, 54(1):42–50.

[25] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[26] Brezovec, C., Cornuéjols, G., and Glover, F. (1986). Two algorithms for weighted matroid intersection. *Mathematical Programming*, 36(1):39–53.

[27] Brezovec, C., Cornuéjols, G., and Glover, F. (1988). A matroid algorithm and its application to the efficient solution of two optimization problems on graphs. *Mathematical Programming*, 42:471–487.

[28] Brualdi, R. A. (1969). Comments on bases in dependence structures. *Bulletin of the Australian Mathematical Society*, 1(2):161–167.

[29] Burkard, R., Dell'Amico, M., and Martello, S. (2009). *Assignment Problem*. SIAM.

[30] Burkholder, D. (1981). A geometrical characterization of Banach spaces in which martingale difference sequences are unconditional. *The Annals of Probability*, 9(6):997–1011.

[31] Burkholder, D. L. (1986). *Lecture Notes in Mathematics*, volume 1206 of *Probability and Analysis (Varenna, 1985)*, chapter Martingales and Fourier analysis in Banach spaces, pages 61–108. Springer-Verlag, Berlin.

[32] Camerini, P. M., Galbiati, G., and Maffioli, F. (1984). The complexity of multi-constrained spanning tree problems. In Lovasz, L., editor, *Theory of Algorithms*, pages 53–101. North-Holland, Amsterdam.

[33] Carosi, L., Jahn, J., and Martein, L. (2003). On the connections between semidefinite optimization and vector optimization. *Journal of Interdisciplinary Mathematics*, 6:219–229.

[34] Chan, A. and Francis, R. (1976). A round-trip location problem on a tree graph. *Transportation Science*, 10:35–51.

[35] Chang, R. and Leu, S.-J. (1997). The minimum labeling spanning trees. *Information Processing Letters*, 63(6):277–282.

[36] Chankong, V. and Haimes, Y. Y. (1983). *Multiobjective Decision Making: Theory and Methodology*. Elsevier Science Publishing, New York.

[37] Chazelle, B. (2000). A minimum spanning tree algorithm with inverse-ackermann type complexity. *Journal of the ACM*, 47(6):1028–1047.

[38] Chu, P. C. and Beasley, J. (1998). A genetic algorithm for the multiconstrained knapsack problem. *Journal of Heuristics*, 4:63–86.

[39] Climaco, J. C. N., Captivo, M. E., and Pascoal, M. M. B. (2010). On the bicriterion - minimal cost/minimal label - spanning tree problem. *European Journal of Operational Research*, 204:199–205.

[40] Cooper, L. (1963). Location-Allocation Problems. *Operations Research*, 11:331 – 343.

[41] Cooper, L. (1964). Heuristic methods for location-allocation problems. *SIAM Review*, 6:37–53.

[42] Cormen, T. H., Stein, C., Leiserson, C. E., and Rivest, R. L. (2001). *Introduction to Algorithms*. B&T.

[43] da Silva, C. G., Clímaco, J., and Figueira, J. (2004). Geometrical configuration of the Pareto frontier of bi-criteria {0,1}-knapsack problems. Working paper 16-2004, Institute for Systems and Computers Engineering, Coimbra, Portugal.

[44] Dächert, K., Gorski, J., and Klamroth, K. (2010). An adaptive augmented weighted tchebycheff method to solve discrete, integer-valued bicriteria optimization problems. Technical Report BUW-AMNA-OPAP 10/06, Bergische Universität Wuppertal, Fachbereich Mathematik und Naturwissenschaften.

[45] Darmann, A. and Pferschy, U. (2008). The constrained minimum spanning tree problem with binary costs. private communication.

[46] de Leeuw, J. (1994). Block relaxation algorithms in statistics. In Bock, H., Lenski, W., and Richter, M., editors, *Information Systems and Data Analysis*, pages 308–325. Springer, Berlin.

[47] Della Croce, F., Paschos, V. T., and Tsoukias, A. (1999). An improved general procedure for lexicographic bottleneck problems. *Operations Research Letters*, 24(4):187–194.

[48] Drezner, Z. and Hamacher, H. W. (2002). *Facility Location*. Springer, New York.

[49] Drezner, Z., Klamroth, K., Schöbel, A., and Wesolowsky, G. O. (2002). The Weber Problem. In Drezner, Z. and Hamacher, editors, *Facility Location*, pages 1–36. Springer, New York.

[50] Duin, C. W. and Volgenant, A. (1991). Minimum deviation and balanced optimization: A unified approach. *Operations Research Letters*, 10(1):43–48.

[51] Edmonds, J. and Fulkerson, D. R. (1970). Bottleneck extrema. *Journal of Combinatorial Theory*, 8:299–306.

[52] Ehrgott, M. (1996). On matroids with multiple objectives. *Optimization*, 38(1):73–84.

[53] Ehrgott, M. (2000). Hard to say it's easy - four reasons why combinatorial multiobjective programmes are hard. In Haimes, Y. Y. and Steuer, R. E., editors, *Research and Practice in Multiple Criteria Decision Making*, volume 487 of *Lecture Notes in Economics and Mathematical Systems*, pages 69–81. Springer, Berlin.

[54] Ehrgott, M. (2005). *Multicriteria Optimization*. Springer Verlag, Berlin, Heidelberg.

[55] Ehrgott, M. and Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22:425–460.

[56] Ehrgott, M. and Gandibleux, X. (2002). *Multicriteria Optimization: State of the Art Annotated Bibliographic Surveys*. kluwer Academic Publishers, Boston, Massachusetts.

[57] Ehrgott, M., Hamacher, H. W., and Maffioli, F. (1999). Fixed cardinality combinatorial optimization problems. Technical report, University of Kaiserslautern.

[58] Ehrgott, M. and Klamroth, K. (1997). Connectedness of efficient solutions in multiple criteria combinatorial optimization. *European Journal of Operational Research*, 97:159–166.

[59] Falk, J. and Soland, R. (1969). An algorithm for separable nonconvex programming problems. *Management Science*, 15(9):550–569.

[60] Fletcher, R. and Leyffer, S. (2002). Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–269.

[61] Floudas, C. (1995). *Nonlinear and Mixed Integer Optimization: Fundamentals and Applications*. Oxford Press, New York.

[62] Floudas, C. and Visweswaran, V. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: I. Theory. *Computers and Chemical Engineering*, 14(12):1397 – 1417.

[63] Floudas, C. and Visweswaran, V. (1993). A primal-relaxed dual global optimization approach. *Journal of Optimization Theory and Applications*, 78(2):187–225.

[64] Floudas, C. A. (2000). *Deterministic Global Optimization*. Kluwer Academic Publishers, Dordrecht, Netherlands, first edition.

[65] Forster, O. (2008). *Analysis 2*. Vieweg & Teubner.

[66] Frank, A. (1981). A weighted matroid intersection algorithm. *J. Algorithms*, 2(4):328–336.

[67] Gabow, H. N. and Tarjan, R. E. (1984). Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms*, 5:80–131.

[68] Gabow, H. N. and Tarjan, R. E. (1988). Algorithms for two bottleneck optimization problems. *Algorithms*, 9:411–417.

[69] Gandibleux, X., Beugnies, F., and Randriamasy, S. (2006). Multi-objective shortest path problems with a maxmin cost function. *4OR - Quarterly Journal of the Belgian,French and Italian Operations Research Societies*, 4:47–59.

[70] Gao, Y. and Xu, C. (2002). An outer approximation method for solving the biconcave programs with separable linear constraints. *Mathematica Applicata*, 15(3):42–46.

[71] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability*. Freeman, New York.

[72] Garfinkel, R. S. and Gilbert, K. C. (1978). The bottleneck traveling salesman problem: algorithms and probabilistic analysis. *Journal of the ACM*, 25:435–448.

[73] Geist, D. and Rodin, E. Y. (1992). Adjacency of the $0-1$ knapsack problem. *Computers and Operations Research*, 19(8):797–800.

[74] Gelbaum, B. and Olmsted, J. (2003). *Counterexamples in Analysis*. Dover Publications, Inc., Mineola, New York.

[75] Geng, Z. and Huang, L. (2000a). Robust stability of systems with both parametric and dynamic uncertainties. *Systems & Control Letters*, 39:87–96.

[76] Geng, Z. and Huang, L. (2000b). Robust stability of the systems with mixed uncertainties under the IQC descriptions. *International Journal of Control*, 73(9):776–786.

[77] Geoffrion, A. (1972). Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260.

[78] Geoffrion, A. M. (1968). Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22:618–630.

[79] Ghosh, A. and Dehuri, S. (2004). Evolutionary algorithms for multi-criterion optimization: A survey. *International Journal of Computing & Information Sciences*, 2(1):38–57.

[80] Glover, F. and Kochenberger, G. (1996). Critical event tabu search for multidimensional knapsack problems. In Osman, I. and Kelly, J., editors, *Metaheuristics: Theory and applications*, pages 407–427. Kluwer Academic Publishers.

[81] Goh, K., Safonov, M., and Papavassilopoulos, G. (1995). Global optimization for the biaffine matrix inequality problem. *Journal of Global Optimization*, 7:365–380.

[82] Goh, K., Turan, L., Safonov, M., Papavassilopoulos, G., and Ly, J. (1994). Biaffine matrix inequality properties and computational methods. In *Proceedings of the American Control Conference Baltimore, Maryland*, pages 850–855.

[83] Gorski, J. (2004). Untersuchungen zum Zusammenhang Pareto-optimaler lösungen in multikriteriellen, kombinatorischen Optimerungsproblemen. Master's thesis, Friedrich-Alexander-Universität of Erlangen-Nürnberg, Deutschland. In German.

[84] Gorski, J., Klamroth, K., and Ruzika, S. (2006a). Connectedness of efficient solutions in multiple objective combinatorial optimization. Technical Report 310, Friedrich-Alexander-Universität Erlangen-Nürnberg, Institut für Angewandte Mathematik.

[85] Gorski, J., Klamroth, K., and Ruzika, S. (2006b). Connectedness of efficient solutions in multiple objective combinatorial optimization. Technical Report 102/2006, Technische Universität Kaiserslautern.

[86] Gorski, J., Paquete, L., and Pedrosa, F. (2009). Greedy algorithms for a class of knapsack problems with binary weights. Technical Report BUW-AMNA-OPAP 09/02, Bergische Universität Wuppertal, Fachbereich Mathematik und Naturwissenschaften.

[87] Gorski, J., Pfeuffer, F., and Klamroth, K. (2007). Biconvex sets and optimization with biconvex functions - a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–408.

[88] Gorski, J. and Ruzika, S. (2009). On k-max optimization. *Operations Research Letters*, 37(1):23–26.

[89] Gupta, S. K. and Punnen, A. P. (1988). Minimum Deviation Problems. *Operations Research Letters*, 7(4):201–204.

[90] Gupta, S. K. and Punnen, A. P. (1990). k-sum Optimization Problems. *Operations Research Letters*, 9:121–126.

[91] Gusfield, D. (1984). Matroid optimization with the interleaving of two ordered sets. *Discrete Applied Mathematics*, 8(1):41–50.

[92] Hamacher, H. W. and Klamroth, K. (2006). *Lineare und Netzwerk-Optimierung*. Vieweg-Verlag.

[93] Hamacher, H. W., Labbe, M., Nickel, S., and Skriver, A. J. V. (2002). Multicriteria semi-obnoxious network location problems with sum and center objectives. *Annals of Operations Research*, 110:33–53.

[94] Hamacher, H. W., Pedersen, C. R., and Ruzika, S. (2007). Finding representative systems for discrete bicriteria optimization problems by box algorithms. *Operations Research Letters 35*, 3:336–344.

[95] Hamacher, H. W. and Rendl, F. (1991). Color constrained combinatorial optimization problems. *Operations Research Letters*, 10:211–219.

[96] Hamacher, H. W. and Ruhe, G. (1994). On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52:209–230.

[97] Handler, G. and Zang, I. (1980). A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–310.

[98] Hansen, P. (1980). Bicriterion path problems. In Fandel, G. and Gal, T., editors, *Multicriteria decision making: theory and applications*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 109–127. Springer, Heidelberg.

[99] Hartmann, M. and Olmstead, T. (1993). Solving sequential knapsack problems. *Operations Research Letters*, 13:225–232.

[100] Hausmann, D. (1980). *Adjacency on Polytopes in Combinatorial Optimization*. Verlag Anton Hain, Königstein/Ts.

[101] Helbig, S. (1990). On the connectedness of the set of weakly efficient points of a vector optimization problem in locally convex spaces. *Journal of Optimization Theory and Applications*, 65:257–271.

[102] Henn, S. (2007). The weight-constrained minimum spanning tree problem. Master's thesis, Universität Kaiserslautern, Deutschland.

[103] Hodgson, M., Rosing, K., and Shmulevitz (1993). A review of location-allocation applications literature. *Studies in Locational Analysis*, 5:3–29.

[104] Horst, R. and Thoai, N. (1996). Decomposition approach for the global minimization of biconcave functions over polytopes. *Journal of Optimization Theory and Applications*, 88(3):561–583.

[105] Horst, R. and Tuy, H. (1990). *Global Optimization, Deterministic Approaches*. Springer-Verlag, Berlin, Heidelberg.

[106] Huang, S. (2004). *The Connection-Location and Sizing Problem: Models, Methods and Applications to Supply Chain Design*. PhD thesis, The State University of New York and Buffalo.

[107] Huang, S., Batta, R., Klamroth, K., and Nagi, R. (2005). K-Connection location problem in a plane. *Annals of Operations Research*, 136:193–209.

[108] Isermann, H. (1974). Proper efficiency and the linear vector maximum problem. *Operations Research*, 22:189–191.

[109] Isermann, H. (1977). The enumeration of the set of all efficient solutions for a linear multiple objective program. *Operations Research Quarterly*, 28:711–725.

[110] Jahn, J. (1996). *Introduction to the Theory of Nonlinear Optimization*. Springer-Verlag, Berlin Heidelberg, second edition.

[111] Jorgensen, C. and Powell, S. (1987). Solving $0-1$ minimax problems. *Journal of the Operational Research Society*, 38(6):515–522.

[112] Jouak, M. and Thibault, L. (1985). Directional derivatives and almost everywhere differentiability of biconvex and concave-convex operators. *Mathematica Scandinavica*, 57:215–224.

[113] Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer Verlag, Berlin.

[114] Klamroth, K. and Tind, J. (2007). Constrained optimization using multiple objective programming. *Journal of Global Optimization*, 37:325–355.

[115] Klamroth, K., Tind, J., and Wiecek, M. M. (2002). Unbiased approximation in multicriteria optimization. *Mathematical Methods of Operations Research*, 56:413–437.

[116] Klamroth, K. and Wiecek, M. (2000). Dynamic programming approaches to the multiple criteria knapsack problem. *Naval Research Logistics*, 47:57–76.

[117] Knowles, J., Corne, D., and Deb, K. (2008). *Multiobjective Problem Solving from Nature*. Springer-Verlag, Berlin.

[118] Krarup, J. and Pruzan, P. M. (1981). Reducibility of minimax to minisum 0-1 programming problems. *European Journal of Operations Research*, 6:125–132.

[119] Kuhn, H. (1967). On a pair of dual nonlinear problems. In Abadie, T., editor, *Nonlinear Programming*, chapter 3, pages 38–54. North-Holland, Amsterdam.

[120] Kuhn, H. and Kuenne, R. (1962). An efficient algorithm for the numerical solution of the generalized Weber problem in spatial economies. *Journal of Regional Science*, 4:21–34.

[121] Kung, H. T., Luccio, F., and Preparata, F. P. (1975). On finding the maxima of a set of vectors. *Journal of the ACM*, 22(4):469–476.

[122] Kung, J.-P. S. (1986). *A source book in matroid theory*. Birkhäuser, Boston.

[123] Lee, J. M. (1993). On Burkholder's biconvex-function characterisation of Hilbert spaces. In *Proceedings of the American Mathematical Society*, volume 118, pages 555–559.

[124] Lieshout, P. M. D. and Volgenant, A. (2007). A branch-and-bound algorithm for the singly constrained assignment problem. *European Journal of Operational Research*, 176:151–164.

[125] Luenberger, D. (1989). *Linear and Nonlinear Programming*. Addison-Wesley, Reading, Massachusetts, second edition.

[126] Lust, T. and Teghem, J. (2009). Two-phase pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*. DOI: 10.1007/s10732-009-9103-9.

[127] Martello, S., Pulleyblank, W. R., Toth, P., and de Werra, D. (1984). Balanced Optimization Problems. *Operations Research Letters*, 3(5):275–278.

[128] Martins, E. Q. V. (1984a). On a multicriteria shortest path problem. *European Journal of Operational Research*, 16:236–245.

[129] Martins, E. Q. V. (1984b). On a special class of bicriterion path problems. *European Journal of Operational Research*, 17:85–94.

[130] Mattfield, D. (2006). *The Management of Transshipment Terminals*. Springer, New York.

[131] Mazzola, J. B. and Schantz, R. H. (1995). Single-facility resource allocation under capacity-based economies and diseconomies of scope. *Management Science*, 41(4):669–689.

[132] Melamed, I. I. (1996). The theory of linear parametrization of criteria in multicriteria optimization. *Doklady Mathematics*, 53(3):381–383.

[133] Melamed, I. I. and Sigal, P. I. K. (1995a). Computational study of linear convolution of criteria in discrete multicriteria programming. *Doklady Mathematics*, 52(3):469–472.

[134] Melamed, I. I. and Sigal, P. I. K. (1995b). An investigation of linear convolution of criteria in multicriteria discrete programming. *Computational Mathematics and Mathematical Physics*, 35(8):1009–1017.

[135] Melamed, I. I. and Sigal, P. I. K. (1996). A computational investigation of linear parametrization of criteria in multicriteria discrete programming. *Computational Mathematics and Mathematical Physics*, 36(10):1341–1343.

[136] Melamed, I. I. and Sigal, P. I. K. (1997). The linear convolution of criteria in the bicriteria traveling salesman problem. *Computational Mathematics and Mathematical Physics*, 37(8):902–905.

[137] Melamed, I. I. and Sigal, P. I. K. (1998). Numerical analysis of tricriteria tree and assignment problems. *Computational Mathematics and Mathematical Physics*, 38(10):1707–1714.

[138] Melamed, I. I. and Sigal, P. I. K. (1999). Combinatorial optimization problems with two and three criteria. *Doklady Mathematics*, 59(3):490–493.

[139] Melamed, I. I., Sigal, P. I. K., and Vladimirova, N. Y. (1999). Study of linear parametrization of criteria in the bicriteria knapsack problem. *Computational Mathematics and Mathematical Physics*, 39(5):721–726.

[140] Meyer, R. (1976). Sufficient conditions for the convergence of monotonic mathematical programming algorithms. *Journal of Computer and System Sciences*, 12:108–121.

[141] Mezura-Montes, E. and Coello Coello, C. A. (2008). *Multiobjective Problem Solving from Nature*, chapter Constrained Optimization via Multiobjective Evolutionary Algorithms, pages 53–75. Springer-Verlag, Berlin.

[142] Miehle, W. (1958). Link-length minimization in networks. *Operations Research*, 6:232–243.

[143] Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston.

[144] Minoux, M. (1989). Solving combinatorial problems with combined min-max-min-sum objective and applications. *Mathematical Programming*, 45:361–372.

[145] Mirchandani, P. and Francis, R. (1990). *Discrete Location Theory*. John Wiley & Sons, Inc., New York.

[146] Montreuil, B. and Ratliff, H. (1988). Optimizing the location of input/output stations with facilities layout. *Engineering Costs and Production Economics*, 14:177–187.

[147] Murthy, I. and Her, S. S. (1992). Solving min-max shortest-path problems on a network. *Naval Research Logistics*, 39:669–683.

[148] Murty, K. G. (1968). An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16(3):682–687.

[149] Naccache, P. (1978). Connectedness of the set of nondominated outcomes. *Journal of Optimization Theory and Applications*, 25:459–467.

[150] Nemhauser, G. L. and Wolsey, L. A. (1999). *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., New York.

[151] Neumann, F. and Wegener, I. (2006). Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5:305–319.

[152] Neumann, F. and Wegener, I. (2008). *Multiobjective Problem Solving from Nature*, chapter Can Single-Objective Optimization Profit from Multiobjective Optimization?, pages 115–130. Springer-Verlag, Berlin.

[153] Nickel, S. and Puerto, J. (2005). *Location Theory: A Unified Approach*. Springer.

[154] Ostrowski, A. (1966). *Solution of Equations and Systems of Equations*. Academic Press, New York and London, second edition.

[155] O'Sullivan, M. and Walker, C. (2004). Connecting efficient knapsacks - experiments with the equally-weighted bi-criteria knapsack problem. In *39th Annual ORSNZ Conference, University of Auckland, Auckland, New Zealand*, pages 198–207.

[156] Oxley, J. G. (1992). *Matroid Theory*. Oxford University Press, NJ.

[157] Papadimitriou, C. H. (1978). The adjacency relation on the traveling salesman polytope is NP-complete. *Mathematical Programming*, 14:312–324.

[158] Paquete, L., Chiarandini, M., and Stützle, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In Gandibleux, X., Sevaux, M., Sörensen, K., and T'kindt, V., editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*. Springer Verlag.

[159] Paquete, L., Schiavinotto, T., and Stützle, T. (2007). On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research*, 159(1):83–97.

[160] Paquete, L. and Stützle, T. (2003). A two-phase local search for the biobjective traveling salesman problem. In Fonseca, C. M., Fleming, P. J.and Zitzler, E., Deb, K., and Thiele, L., editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, Faro, Portugal, April 2003. Lecture Notes in Computer Science*, volume 2632, pages 479–493, berlin. Springer.

[161] Paquete, L. and Stützle, T. (2006). Clusters of non-dominated solutions in multiobjective combinatorial optimization. In *MOPGP'06: 7th International Conference on Multi-Objective Programming and Goal Programming, Loire valley, Tours*.

[162] Parker, R. G. (1984). Guaranteed performance heuristics for the bottleneck traveling salesman problem. *Operations Research Letters*, 84:269–272.

[163] Pedersen, C. R. (2006). *Multicriteria Discrete Optimization - and Related Topics*. PhD thesis, University of Aarhus.

[164] Pedersen, C. R., Nielsen, L. R., and Andersen, K. A. (2005). On the bicriterion multi modal assignment problem. Working paper WP-2005-3, Department of Operations Research, University of Aarhus.

[165] Pelegrin, B. and Fernandez, P. (1998). On the sum-max bicriterion path problem. *Computers and Operations Research*, 25(12):1043–1054.

[166] Pinto, L., Bornstein, C., and Maculan, N. (2009). The tricriterion shortest path problem with at least two bottleneck objectives. *European Journal of Operational Research*, 198:387–391.

[167] Plastria, F. (1995). Continuous location problems. In Drezner, Z., editor, *Facility Location*, pages 225–262. Springer Series in Operations Research.

[168] Przybylski, A., Gandibleux, X., and Ehrgott, M. (2006). The biobjective integer minimum cost flow problem - incorrectness of Sedeño-Noda and González-Martín's algorithm. *Computers & Operations Research*, 33(5):1459–1463.

[169] Przybylski, A., Gandibleux, X., and Ehrgott, M. (2008). Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185:509–533.

[170] Punnen, A. P. (1994). On combined minmax-minsum optimization. *Computers & Operations Research*, 21(21):707–716.

[171] Punnen, A. P. (1996). A fast algorithm for a class of bottleneck problems. *Computing*, 56:397–401.

[172] Punnen, A. P. and Aneja, Y. P. (1995). Minmax combinatorial optimization. *European Journal of Operational Research*, 81:634–643.

[173] Punnen, A. P. and Aneja, Y. P. (1996). On k-sum optimization. *Operations Research Letters*, 18:233–236.

[174] Punnen, A. P. and Aneja, Y. P. (2004). Lexicographic balanced optimization problems. *Operations Research Letters*, 32:27–30.

[175] Punnen, A. P. and Nair, K. P. K. (1994). A fast and simple algorithm for the bottleneck biconnected spanning subgraph problem. *Information Processing Letters*, 50:283–286.

[176] Punnen, A. P. and Nair, K. P. K. (1996). An $\mathcal{O}(m \log n)$ algorithm for the max + sum spanning tree problem. *European Journal of Operational Research*, 89:423–426.

[177] Punnen, A. P. and Nair, K. P. K. (1999). Constrained Balanced Optimization Problems. *Computers and Mathematics with Applications*, 37:157–163.

[178] Punnen, A. P., Nair, K. P. K., and Aneja, Y. P. (1995). Generalized bottleneck problems. *Optimization*, 35(2):159–169.

[179] Raidl, G. R. and Gottlieb, J. (2005). Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. *Evolutionary Computation Journal*, 13(4):441–475.

[180] Ravi, R. and Goesmans, M. (1996). The constrained minimum spanning tree problem. In *Proceedings of the 5th Scandinavian Workshop on Algorithmic Theory (SWAT)*, volume 1097 of *LNCS*, pages 66–75.

[181] Ravi, R., Sundaram, R., Marathe, M. V., Rosenkrantz, D. J., and Ravi, S. S. (1996). Spanning trees - short or small. *SIAM Journal of Discrete Mathematics*, 9(2):178–200.

[182] Rendl, F. and Leclerc, M. (1988/89). A multiply constrained matroid optimization problem. *Discrete Mathematics*, 73:207–212.

[183] Rockafellar, R. (1997). *Convex Analysis*. Princeton University Press, Princeton, New Jersey, first edition.

[184] Ruzika, S. (2008). *On Multiple Objective Combinatorial Optimization*. PhD thesis, Technische Universität Kaiserslautern.

[185] Ruzika, S. and Hamacher, H. (2009). A survey on multiple objective minimum spanning tree problems. In Lerner, J., Wagner, D., and Zweig, K. A., editors, *Algorithmics*, volume 5515/2009 of *Lecture Notes in Computer Science*, pages 104–116. Springer Berlin/Heidelberg.

[186] Sayin, S. and Kouvelis, P. (2005). The multiobjective discrete optimization problem: A weighted min-max two-stage optimization approach and a bicriteria algorithm. *Management Science*, 51(10):1572–1581.

[187] Schandl, B., Klamroth, K., and Wiecek, M. M. (2002a). Introducing oblique norms into multiple criteria programming. *Journal of Global Optimization*, 23:81–97.

[188] Schandl, B., Klamroth, K., and Wiecek, M. M. (2002b). Norm-based approximation in multicriteria programming. *Computers and Mathematics with Applications*, 44:925–942.

[189] Schrijver, A. (2003). *Combinatorial Optimization*. Springer Verlag, Berlin.

[190] Sedeño-Noda, A. and González-Martín, C. (2001). An algorithm for the biobjective integer minimum cost flow problem. *Computers and Operations Research*, 28:139–156.

[191] Sherali, H., Alameddine, A., and Glickman, T. (1994). Biconvex models and algorithms for risk management problems. *American Journal of Mathematical and Management Sciences*, 14(3 & 4):197–228.

[192] Sherali, H., Alameddine, A., and Glickman, T. (1995). Biconvex models and algorithms for risk management problems. *Operations Research / Management Science*, 35(4):405–408.

[193] Sokkalingam, P. T. and Aneja, Y. P. (1998). Lexicographic bottleneck combinatorial problems. *Operations Research Letters*, 23(1):27–33.

[194] Srinivas, M. A. (1995). Matroid optimization with generalized constraints. *Discrete Applied Mathematics*, 63:161–174.

[195] Srinivasan, V. and Thompson, G. L. (1976). Algorithms for minimazing total cost, bottleneck time and bottleneck shipment in transportation problems. *Naval Research Logistics Quarterly*, 23:567–595.

[196] Steiner, S. and Radzik, T. (2003). Solving the biobjective minimum spanning tree problem using k-best algorithm. Technical Report TR-03-06, Department of Computers Science, Kingś College, London, UK.

[197] Steuer, R. (1985). *Multiple criteria optimization. Theory, computation, and application.* John Wiley & Sons, Inc., New York.

[198] Steuer, R. and Choo, E. (1983). An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26:326–344.

[199] Stiglmayr, M., Pfeuffer, F., and Klamroth, K. (2008). A branch and bound algorithm for medical image registration. In Brimkov, V. E., Barneva, R. P., and Hauptman, H. A., editors, *Proceedings of the 12th International Workshop on Combinatorial Image Analysis (IWCIA 08)*, volume 4958, pages 218–227, Berlin/Heidelberg. Springer.

[200] Tavares, J., Pereira, F. B., and Costa, E. (2008). Multidimensional knapsack problem: A fitness landscape analysis. *IEEE Transaction on Systems, Man and Cybernetics - Part B*, 38(3):604–616.

[201] Teunissen, P. J. G. (2000). *Adjustment Theory: An Introduction.* VSSD.

[202] Thibault, L. (1984). Continuity of measurable convex and biconvex operators. In *Proceedings of the American Mathematical Society*, volume 90, pages 281–284.

[203] Turner, L. (2010). private communication.

[204] Turner, L. (forthcoming). Universal combinatorial optimization problems. Ph. D. Thesis.

[205] Tuyen, H. and Muu, L. (2001). Biconvex programming approach to optimization over the weakly efficient set of a multiple objective affine fractional problem. *Operations Research Letters*, 28:81–92.

[206] Ulungu, E. L. and Teghem, J. (1995). The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20:149165.

[207] Vasquez, M. and Vimont, Y. (2005). Improved results on the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, 165:70–81.

[208] Visée, M., Pirlot, M., and Ulungu, E. L. (1998). Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12:139–155.

[209] Visweswaran, V. and Floudas, C. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: II. Application of theory and test problems. *Computers and Chemical Engineering*, 14(12):1419 – 1434.

[210] Visweswaran, V. and Floudas, C. (1993). New properties and computational improvement of the GOP algorithm for problems with quadratic objective function and constraints. *Journal of Global Optimization*, 3(3):439–462.

[211] Warburton, A. (1983). Quasiconcave vector maximization: Connectedness of the sets of Pareto-optimal and weak Pareto-optimal alternatives. *Journal of Optimization Theory and Applications*, 40:537–557.

[212] Weber, A. (1909). *Über den Standort der Industrien, 1. Teil: Reine Theorie des Standortes*. J.C.B. Mohr, Tübingen.

[213] Weingartner, H. and Ness, D. N. (1967). Method for the solution for the multidimensional 0–1 knapsack problem. *Operations Research*, 15:83–103.

[214] Weiszfeld, E. (1937). Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal*, 43:335–386.

[215] Wendell, R. and Hurter Jr., A. (1976). Minimization of non-separable objective function subject to disjoint constraints. *Operations Research*, 24(4):643–657.

[216] Wesolowski, G. (1993). The Weber problem: History and perspectives. *Location Science*, 1:5–23.

[217] Whitney, H. (1935). On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533.

[218] Xu, J. C. and Liu, Y. M. (1993). All optimal solutions of the 0-1 bottleneck problem. *Chinese Journal of Numerical Mathematics and Applications*, 15:36–46.

[219] Xue, G. (2000). Primal-dual algorithms for computing weight-constrained shortest paths and weight-constrained minimum spanning trees. In *19th IEEE International Performance, Computing and Communications Conference (IPCCC)*, pages 271–777.

[220] Yu, P. (1985). *Multiple Criteria Decision Making: Concepts, Techniques and Extensions*. Plenum Press, New York.

[221] Zangwill, W. (1969). Convergence conditions for nonlinear programming algorithms. *Management Science*, 16(1):1–13.

[222] Ziegelmann, M. (2007). *Constrained Shortest Paths and Related Problems*. VDM Verlag Dr. Mueller e. K.

[223] Zitová, B. and Flusser, J. (2003). Image registration methods: A survey. *Image and Vision Computing*, 21:977–1000.

Acknowledgement

This work would not have been possible without the support of a lot of people. First of all, I would like to express my deepest gratitude to my supervisor Prof. Dr. Kathrin Klamroth who offered me the exciting opportunity to work in the field of multiple objective optimization. During endless discussions on various topics of this thesis, she gave me deeper insight into the most important aspects of single and multiple objective optimization. Furthermore, she was always receptive to my professional and private problems and encouraged me throughout my time at the university. The work in her group was a wonderful experience.

I owe a special debt of gratitude to my colleagues Jun.-Prof. Dr. Stefan Ruzika from the Technische Universität Kaiserslautern and Dr. Luis Paquete from the Universidade de Coimbra for the joint work and fruitful discussions on several topics of this thesis. It was a pleasure to meet and work with them throughout different places around the world. During my stays in Kaiserslautern and Coimbra, respectively, I always felt welcome. Thank you for the nice time we spent. In addition, I would like to thank Frank Pfeuffer and Fabio Pedrosa for the joint work on our publications.

I thank all former and present colleagues from the Chair of Applied Mathematics II at the Friedrich-Alexander-Universität Erlangen-Nürnberg and from the Research Group of Optimization and Approximation at the Bergische Universität Wuppertal for their support and the pleasant atmosphere. I am grateful to Doris Ederer and Kirsten Wilshaus for their support in all bureaucracy concerns, especially during the move of the complete working group from Erlangen to Wuppertal. In addition, I would like to thank Dr. Friedrich Graef for his advise and help concerning the computer system in Erlangen.

My special thanks goes to my colleagues and friends Kerstin Dächert, Markus Kaiser, Michael Stiglmayr, Alexander Thekale, Martin Wagner and to Dr. Barbara Pfeiffer and Dr. Martin Bischoff who both left the working group before we moved to Wuppertal. Thank you for many interesting discussions, helpful comments and the fun we had during our time in Erlangen and Wuppertal, respectively.

My sincere appreciation goes to my mother Renate and my father Peter Gorski, who unfortunately died before the completion of this thesis. Without their endless love and support during my studies, this dissertation would not have been possible.

I am very grateful to Judith Fleischmann for her love and patience during the work on my thesis. Especially her encouragement and understanding were a great help over the last years.

Furthermore, I gratefully acknowledge partial financial support by the German Research Foundation (DFG) and the German Academic Exchange Service (DAAD).