



**BERGISCHE
UNIVERSITÄT
WUPPERTAL**

Ordinal Costs in Multi-objective Combinatorial Optimization

Dissertation

zur Erlangung des Doktorgrades (Dr. rer. nat.)

Fakultät für Mathematik und Naturwissenschaften

Bergische Universität Wuppertal

vorgelegt von

Julia Sudhoff

Erstgutachterin: Prof. Dr. Kathrin Klamroth

Zweitgutachter: Prof. Dr. Alexander Engau

Drittgutachter: Prof. Dr. Stefan Ruzika

Wuppertal, September 2022

Acknowledgements

This thesis would not exist without the support of many people. First of all, I like to thank my supervisor Kathrin Klamroth for her support and her willingness to listen whenever I had any difficulties. Especially, I like to thank her for her encouragement regarding conferences and further training opportunities.

Furthermore, I like to thank her as well as Jochen Gorski and Michael Stiglmayr for the excellent cooperation during the writing process of our joint paper. Thank you for teaching me how to write a good scientific article in proper English and for many tips for the usage of \LaTeX .

Moreover, my thanks go to the Working Group of Optimization at the University of Wuppertal. Unfortunately, we did not spend much time in the office together due to the Corona pandemic, but nevertheless we had a great time and some wonderful events (like, e.g., Christmas parties or seminar trips) together. I would like to thank you all especially for helpful discussions and feedback to my work, like, e.g., presentations or this thesis.

I also like to thank everyone else who supported and inspired me. Sometimes the little things are the most important ones, but unfortunately I tend to forget them, so I can not list them all. I like to thank my family and friends for supporting me and also some students for inspiring me by asking questions or for infecting me with their motivation.

Finally, I like to thank the Deutsche Forschungsgemeinschaft for the partial financial support through project number KL 1076/11-1.

Contents

1. Introduction	7
2. Basic Concepts and Notation	11
2.1. Binary Relations and Cones	11
2.2. Optimization Problems and Optimality Concepts	13
2.3. Scalarizations	17
2.4. Graphs	27
2.5. Matroids	33
2.6. Single-objective Matroid Optimization and Multi-objective Minimum Spanning Tree Problem	36
2.7. Single-objective Shortest Path Problem	42
2.8. Single-objective Matroid Intersection Problem	50
2.9. Multi-objective Knapsack Problem	54
2.10. Multi-objective Assignment Problem	56
3. Bi-objective Matroid Optimization Problems with Binary Costs	59
3.1. Problem Formulation	60
3.2. Theoretical Results	62
3.3. Efficient Swap Algorithm	69
3.4. Numerical Results	74
3.5. Conclusion and Further Ideas	80
4. Single- and Multi-objective Matroid Optimization Problems with Ordinal Costs	83
4.1. Single-objective Matroid Optimization with Ordinal Costs	84
4.2. Bi-objective Matroid Optimization with Ordinal Costs	90
4.3. Matroid Intersection Algorithm for Ordinal Constraints	93
4.4. Multi-objective Matroid Optimization with Ordinal Costs	98
4.5. Numerical Results	100
4.6. Conclusion and Further Ideas	104
5. Single- and Multi-objective Combinatorial Optimization Problems with Ordinal Costs	107
5.1. Single-objective Combinatorial Optimization with Ordinal Costs	108
5.2. Ordinal Optimality versus Pareto Optimality: An Interpretation based on Ordering Cones	118
5.3. Solution Strategies	124
5.4. Excursus: Olympic Medals and Ordinal Weight Space Decomposition . . .	128
5.5. Numerical Results	130
5.6. Multi-objective Combinatorial Optimization with Ordinal Costs	136

5.7. Conclusion and Further Ideas	139
6. Conclusion and Further Ideas	141
Bibliography	143
A. A Story About Ordinal Optimization Through Multi-objective Reformulation	153

1. Introduction

Optimization appears in many aspects of our every day live. For example, we want to find the shortest path to work in the morning, at work we make decisions to improve the corporate profit, and after work we search for the cheapest products to buy. Often, we actually have more than one optimization goal. Consider, for example, the selection of the best bicycle path to work. Then, we are interested in finding short and safe paths. Also, the company has, besides economical interests, additional ecological goals as customers are more and more concerned about environmentally friendly products. And even if we go shopping, we want to buy cheap products but they should also be of good quality.

Obviously, it is in general not possible to realize the best value in several objectives at the same time. For example, the cheapest product often does not have the highest quality. Nevertheless, there may exist a company that produces a particular product at a better price and a higher quality than another company. In this case, we say that the product of the first company dominates the product of the second company. In general, a solution dominates another solution if it is at least as good in all objectives and strictly better in at least one objective. We are interested in those solutions for which there exist no other solutions which dominate them.

In this thesis we particularly focus on ordinal objective functions. Ordinal costs are used for objectives which cannot be measured by numerical values but which can be associated with ordered categories. As an example, we consider the shortest path problem for cyclists. Cyclists often want to find a short path to their destination, but they are also concerned about its safety, the inclines in the path and the smoothness of the surface. So they usually prefer, in terms of safety, roads with bicycle lanes over roads without bicycle lanes and a high traffic volume. In terms of the incline, cyclists prefer flat over hilly paths and, in terms of the surface conditions, they usually prefer asphalt over gravel, which is again preferred over sand. Rather than directly using an ordinal objective, Raith et al., 2009 suggests a method to quantify the attractiveness of a route for cyclists. This is determined by, e.g., the safety and the incline of a path. The quantified attractiveness is then used as a second objective function besides the travel time in a bi-objective shortest path problem. Raith et al., 2009 applies the resulting route-choice model to a case study in New Zealand. Furthermore, this method is used in Raith et al., 2011 for a new demand forecast methodology. This methodology is part of a prioritizing strategy to decide in which order infrastructure projects for bicycles are to be executed in Auckland, New Zealand. In this thesis we suggest solution methods for directly handling ordinal objective functions such that it is not necessary to replace the ordinal categories by numerical values.

Problems, like, e.g., the shortest path problem, that have feasible solutions defined on a discrete set, are called combinatorial optimization problems. In this thesis we investigate such problems with several objective functions, the so called *multi-objective combinatorial optimization* (MOCO) problems.

MOCO problems are known to be notoriously hard. We refer to Figueira et al., 2017 for

a recent discussion of the prevalent difficulties in MOCO problems. Even when considering “simple” problems like matroid optimization, where an optimal solution can be found with the greedy algorithm, the multi-objective variants tend to be very difficult. The decision problem associated with multi-objective matroid optimization problems is proven to be \mathcal{NP} -complete in general, see Ehrgott, 1996. A survey on multi-objective combinatorial optimization is given in Ehrgott and Gandibleux, 2000.

Due to the hardness of MOCO problems, many authors suggest heuristics or approximation methods as an alternative to exact algorithms to solve, or at least to compute feasible solutions, for those problems. Exact algorithms compute at least one efficient solution. Often, the complete non-dominated set, and one efficient solution for every non-dominated outcome vector are computed. In contrast, approximation methods return solutions that approximate optimal solutions and the returned solutions often satisfy a given error-bound, i.e., the outcome vectors found lie within a certain distance to the non-dominated points. Heuristics cannot give this guarantee in general and hence, we often only know that the returned solution is feasible, but we don’t know how close the solution is to the non-dominated set. Nevertheless, the huge advantage of heuristics is that they often need significantly less running time than exact or approximation methods.

An example for exact solution strategies are branch and bound algorithms, which can be used to solve MOCO problems, see, for example, Stidsen et al., 2014, Jesus et al., 2021 or for a survey Przybylski and Gandibleux, 2017. For a review of exact solution methods, see Ehrgott et al., 2016 and for a survey on approximation methods for MOCO problems, see Ehrgott and Gandibleux, 2004.

There exist different heuristic methods for MOCO problems. For a survey on genetic algorithms for multi-objective optimization, see Coello, 1998. Evolutionary algorithms have been presented, like the genetic local search algorithm from Jaskiewicz, 2002 and the particle swarm algorithm from Roy et al., 2011. A frequently used evolutionary algorithm is the non-dominated sorting genetic algorithm (NSGA-II). Verma et al., 2021 give a review on different results of other researchers, who used the NSGA-II to solve MOCO problems. Other heuristic strategies are local search techniques, see Blot et al., 2018 for a survey.

Moreover, there exist some methods that combine heuristics and exact methods. Such hybrid methods for MOCO problems are presented in Ehrgott and Gandibleux, 2008.

At the beginning of this thesis, we focus on multi-objective optimization problems on matroids, which are a special case of MOCO problems. The multi-objective spanning tree problem is a prominent special case of multi-objective optimization problems on matroids. For multi-objective spanning tree problems it was shown in Hamacher and Ruhe, 1994 that already in the bi-objective case the cardinality of the non-dominated set may grow exponentially with the size of the instance. This result holds analogously for general multi-objective optimization problems on matroids. As a consequence, for such instances the complete enumeration of the non-dominated set is often too time consuming since it requires an exponential amount of time in the problem size in the worst case. To distinguish algorithms by their efficiency for problems with exponentially large non-dominated sets, Bökler et al., 2017 recently suggested to consider the concept of output sensitive complexity in the context of MOCO problems and analyzed various problem classes. In the dissertation of Bökler, 2018 the output sensitive complexity of the bi-objective spanning tree problem is related to that of bi-objective unconstrained combinatorial optimization

(BUCO) which is, however, also still open. Despite the general intractability of multi-objective spanning tree problems, it is shown in the dissertation of Seipp, 2013 that the number of extreme supported non-dominated outcome vectors grows only polynomially with the size of the instance.

Outline of this thesis

MOCO problems are, in general, very hard to solve. In this thesis we focus on special cases of MOCO problems that allow for efficient, i.e., polynomial time solution methods despite these general difficulties. Decisive for the problem complexity are, on the one hand, the objective functions, and on the other hand the problem structure. We consider the novel and practically relevant class of ordinal objective functions, with binary (0-1) objectives being an important special case, and derive efficient solution methods that take advantage of the close relation to associated MOCO problems. For multi-objective matroid optimization problems with ordinal costs we derive a polynomial time algorithm based on matroid intersection. When considering a binary objective function instead of an ordinal objective function in bi-objective matroid problems, then a particularly efficient algorithm with a linear number of iterations is derived which allows the exact solution of even large instances within seconds.

From an overall perspective, we first consider a specific case of combinatorial optimization problems with a lot of structure and derive a very efficient solution algorithm. Afterwards, we relax the special structure step by step and present efficient algorithms for more and more general cases. Hence, we first consider matroids with binary costs, then matroids with ordinal costs, and finally we omit the restriction to matroids and consider general combinatorial optimization problems with ordinal costs. While investigating one problem type, we often start with the single-objective case and move on to the multi-objective case.

Chapter 2 We review some basic mathematical structures like graphs, matroids, binary relations and cones that are relevant for this thesis. Furthermore, we introduce a general optimization problem and possible optimality concepts for multi-objective problems. Moreover, we introduce scalarization techniques, which generate solutions of multi-objective problems by solving one, or a series of, associated single-objective optimization problems. The single-objective optimization problems are usually easier to solve and can be used to find solutions of the corresponding multi-objective problem. After considering such general strategies, we introduce different combinatorial optimization problems with special structures and some solution methods. Thereby, we focus on those problem types and solution strategies, that are used throughout this thesis.

Chapter 3 We consider bi-objective optimization problems on matroids where one of the objective functions is restricted to binary cost coefficients. We show that in this case the problem has a connected efficient set with respect to a natural definition of a neighborhood structure. This is, to the best of our knowledge, the first non-trivial problem on matroids where connectedness of the efficient set can be established. Due to the connectedness of the

efficient set, we can formulate a polynomial time algorithm based on a neighborhood search approach. The algorithm computes the complete non-dominated set in a linear number of iterations. The theoretical results are validated by numerical experiments for bi-objective minimum spanning tree problems (graphic matroids) and bi-objective knapsack problems with a cardinality constraint (uniform matroids).

Chapter 4 In the context of matroid optimization, we consider ordinal, i.e., non-additive, objective functions with more than two categories. We introduce several problem variants that can be distinguished w.r.t. their respective optimization goals, analyze their interrelations, and derive a polynomial time solution method that is based on the repeated solution of matroid intersection problems. Numerical tests on minimum spanning tree problems and on partition matroids confirm the efficiency of the approach.

Chapter 5 We analyze combinatorial optimization problems with ordinal objective functions that assign categories (like good, medium and bad) rather than cost coefficients to the elements of feasible solutions. We review different optimality concepts for ordinal optimization problems and discuss their similarities and differences. We then focus on two prevalent optimality concepts that are shown to be equivalent. Our main result is a bijective linear transformation that transforms ordinal optimization problems into associated standard multi-objective optimization problems with binary cost coefficients. Since this transformation preserves all properties of the underlying problem, problem-specific solution methods remain applicable. A prominent example is dynamic programming and Bellman's principle of optimality, that can be applied, e.g., to ordinal shortest path and ordinal knapsack problems. We extend our results to multi-objective optimization problems that combine ordinal and real-valued objective functions.

Chapter 6 The thesis concludes with a brief summary of the main results.

Contribution

Parts of the results of this thesis were already published in Gorski et al., 2022, Klamroth et al., 2022a and Klamroth et al., 2022b (note that, as common in this field, the authors are listed in alphabetical order). Chapter 1 and Chapter 2 contain parts from all three articles, while Chapter 3 is mainly based on Gorski et al., 2022. Chapter 4 and Chapter 5 are based on the articles Klamroth et al., 2022a and Klamroth et al., 2022b, respectively.

2. Basic Concepts and Notation

In this chapter we introduce some basic mathematical structures, starting with binary relations and cones in Section 2.1. These structures are used to define different optimality concepts for multi-objective optimization problems in Section 2.2, after providing a short general introduction to single-objective and multi-objective combinatorial optimization problems. Multi-objective optimization problems are often solved by generating and solving many corresponding single-objective optimization problems. The single-objective problems are constructed with scalarization techniques, see Section 2.3, which, for example, combine several objective functions to one new objective function or transform one or several objective functions into constraints. Combinatorial optimization problems are often defined on objects with specific structures, like graphs or matroids, which are introduced in Sections 2.4 and 2.5.

In the remaining sections, we consider specific combinatorial optimization problems and provide a short literature review regarding properties and solution strategies for those problems. Algorithms that are used throughout this thesis are explained in more detail and illustrated at small examples. We start with single-objective matroid optimization problems in Section 2.6 and recall the greedy algorithm, which solves such problems efficiently. As minimum spanning tree problems are a special case of matroid optimization problems we illustrate the greedy algorithm at this example. Then, we investigate the more difficult multi-objective minimum spanning tree problem. In Section 2.7 we recall two algorithms that solve the single-objective shortest path problem. These algorithms are used in a method, which solves single-objective matroid intersection problems, see Section 2.8. Finally, we review some literature regarding solution strategies for the multi-objective knapsack problem in Section 2.9 and the multi-objective assignment problem in Section 2.10.

2.1. Binary Relations and Cones

In this section we formally introduce binary relations as well as some orders and their properties, see, e.g., Ehrgott, 2005. Furthermore, we repeat the definition of cones and their properties as well as the interrelations between cones and binary relations.

A *binary relation* on a set R is a subset \mathcal{R} of $R \times R$. The following properties of binary relations are needed throughout this thesis. For a more comprehensive overview, see, e.g., Ehrgott, 2005. A binary relation \mathcal{R} on R is called

- *reflexive*, if $(u, u) \in \mathcal{R}$ for all $u \in R$,
- *irreflexive*, if $(u, u) \notin \mathcal{R}$ for all $u \in R$,
- *transitive*, if $(u, v) \in \mathcal{R}$ and $(v, w) \in \mathcal{R} \implies (u, w) \in \mathcal{R}$ for all $u, v, w \in R$,
- *asymmetric*, if $(u, v) \in \mathcal{R} \implies (v, u) \notin \mathcal{R}$ for all $u, v \in R$,

- *antisymmetric*, if $(u, v) \in \mathcal{R}$ and $(v, u) \in \mathcal{R} \implies u = v$ for all $u, v \in R$,
- *connected*, if $(u, v) \in \mathcal{R}$ or $(v, u) \in \mathcal{R}$ for all $u, v \in R$ with $u \neq v$.

A reflexive and transitive binary relation is called a *preorder*. A preorder which is also antisymmetric is called a *partial order*. An irreflexive and transitive binary relation is called *strict partial order*, which is always also asymmetric by definition. A connected preorder is called *total preorder* and a connected partial order, i.e., a reflexive, transitive, antisymmetric and connected binary relation, is called *total order*.

In the following we consider, among others, binary relations on the real-valued vectorspace \mathbb{R}^p . In this case, we call a binary relation \mathcal{R} *compatible with addition* if $(u, v) \in \mathcal{R}$ implies $(u + w, v + w) \in \mathcal{R}$ for all $u, v, w \in \mathbb{R}^p$ and *compatible with scalar multiplication* if $(u, v) \in \mathcal{R}$ implies $(\lambda u, \lambda v) \in \mathcal{R}$ for all $u, v \in \mathbb{R}^p$ and $\lambda > 0$.

Some of the most common binary relations on \mathbb{R}^p are the (weak (2.1), strict (2.3)) componentwise orders:

$$y' \leq \hat{y} : \iff y'_i \leq \hat{y}_i, \quad i = 1, \dots, p, \quad (2.1)$$

$$y' \leq \hat{y} : \iff y'_i \leq \hat{y}_i, \quad i = 1, \dots, p \text{ and } y' \neq \hat{y}, \quad (2.2)$$

$$y' < \hat{y} : \iff y'_i < \hat{y}_i, \quad i = 1, \dots, p. \quad (2.3)$$

Here we use the notation $u \mathcal{R} v$ for $u, v \in \mathbb{R}^p$ instead of $(u, v) \in \mathcal{R}$ for $\mathcal{R} \in \{\leq, \leq, <\}$. In the following, we use both notations equivalently for all binary relations. We define \geq, \geq and $>$ on \mathbb{R}^p analogously. Note that \leq defines a partial order on \mathbb{R}^p while \leq as well as $<$ define strict partial orders on \mathbb{R}^p . It is easy to see that all three orders (2.1), (2.2) and (2.3) are compatible with scalar multiplication and addition.

We denote the positive orthant of \mathbb{R}^p by $\mathbb{R}_{\geq}^p := \{x \in \mathbb{R}^p : x \geq 0\}$ when zero is included and by $\mathbb{R}_{>}^p := \{x \in \mathbb{R}^p : x > 0\}$ otherwise. Furthermore, we also consider the set of all rational numbers \mathbb{Q} , the set of all integers \mathbb{Z} , the set of all non-negative integers including zero $\mathbb{Z}_{\geq} := \{x \in \mathbb{Z} : x \geq 0\}$. The set of all strictly positive integers is denoted by $\mathbb{Z}_{>} := \{x \in \mathbb{Z} : x > 0\}$ and the set of all p -dimensional vectors with integer values greater or equal zero is denoted by $\mathbb{Z}_{\geq}^p := \{x \in \mathbb{Z}^p : x \geq 0\}$.

Other common orders on \mathbb{R}^p are the (weak (2.5)) lexicographic orders:

$$y' <_{\text{lex}} \hat{y} : \iff y' \neq \hat{y} \text{ and } y'_{k^*} < \hat{y}_{k^*} \text{ with } k^* := \min\{k : y'_k \neq \hat{y}_k\}, \quad (2.4)$$

$$y' \leq_{\text{lex}} \hat{y} : \iff y' <_{\text{lex}} \hat{y} \text{ or } y' = \hat{y}. \quad (2.5)$$

We define $>_{\text{lex}}$ and \geq_{lex} analogously. It is easy to show, that the relation $<_{\text{lex}}$ is a connected strict partial order while the relation \leq_{lex} is a total order.

Orders and cones are closely related. The following review of basic concepts regarding cones relevant in our context is based on Ehrgott, 2005; Engau, 2007; Ziegler, 1995.

A *cone* in \mathbb{R}^p is a subset $C \subseteq \mathbb{R}^p$ such that $\lambda u \in C$ for all $u \in C$ and for all $\lambda \in \mathbb{R}$ with $\lambda > 0$. A cone $C \subseteq \mathbb{R}^p$ is called *pointed* if $u \in C$ implies that $(-u) \notin C$ for all $u \neq 0$.

Moreover, a cone $C \subseteq \mathbb{R}^p$ is called a *polyhedral cone* if there exists a matrix $A \in \mathbb{R}^{m \times p}$ such that $C = \text{hcone}(A) := \{y \in \mathbb{R}^p : Ay \geq 0\}$. The rows of the matrix A are normal vectors of hyperplanes, and thus a polyhedral cone can be seen as the finite intersection of m (closed and linear) halfspaces. In general, see, for example, the textbook Matoušek,

2002, a *hyperplane* is a set $H \subset \mathbb{R}^p$ and is defined as $H = H(a, b) = \{y \in \mathbb{R}^p : a^\top y = b\}$, with the *normal vector* $a \in \mathbb{R}^p \setminus \{0\}$ and $b \in \mathbb{R}$. Furthermore, the *closed upper halfspace* induced by H is defined as $H_+ := \{y \in \mathbb{R}^p : a^\top y \geq b\}$. A *face* of a polyhedral cone C is a subset of C of the form $C \cap H$, with H a hyperplane such that $C \subseteq H_+$. A face of dimension 1 is called *extreme ray*. A set $S \subseteq \mathbb{R}^p$ is called *convex set* if for all $e_1, e_2 \in S$ and for every $\lambda \in [0, 1]$ it holds that $\lambda e_1 + (1 - \lambda)e_2 \in S$. It can be easily seen that every polyhedral cone is a convex set and hence also a *convex cone*.

Polyhedral cones can also be described by their extreme rays. This property is an immediate consequence of the well-known Weyl-Minkowski-Theorem:

Theorem 2.1 (Weyl-Minkowski-Theorem, cf. Ziegler 1995). *A cone $C \subseteq \mathbb{R}^p$ is finitely generated by n vectors in \mathbb{R}^p , i.e.,*

$$C = \text{vcone}(B) := \{B\lambda : \lambda \in \mathbb{R}^n, \lambda \geq 0\} \text{ for some } B \in \mathbb{R}^{p \times n}$$

if and only if it is a finite intersection of m halfspaces in \mathbb{R}^p , i.e.,

$$C = \text{hccone}(A) = \{y \in \mathbb{R}^p : Ay \geq 0\} \text{ for some } A \in \mathbb{R}^{m \times p}.$$

Now let $C \subset \mathbb{R}^p$ be a cone. Then the sets

$$\begin{aligned} C^* &:= \{d \in \mathbb{R}^p : d^\top c \geq 0 \text{ for all } c \in C\} \text{ and} \\ C_s^* &:= \{d \in \mathbb{R}^p : d^\top c > 0 \text{ for all } c \in C \setminus \{0\}\} \end{aligned}$$

are called the *dual cone* and the *strict dual cone* of C , respectively.

Every cone $C \subseteq \mathbb{R}^p$ induces a binary relation $\mathcal{R} \subseteq \mathbb{R}^p \times \mathbb{R}^p$ by defining that $(u, v) \in \mathcal{R}$ if and only if $(v - u) \in C$. Binary relations induced by cones are always compatible with addition and scalar multiplication.

Conversely, binary relations that are compatible with scalar multiplication induce cones that represent the respective relation. The following result is particularly useful in Chapter 5.

Lemma 2.2 (see, e.g., Ehrgott 2005). *Let $\mathcal{R} \subseteq \mathbb{R}^p \times \mathbb{R}^p$ be a binary relation on \mathbb{R}^p which is compatible with scalar multiplication. Then $C_{\mathcal{R}} := \{(v - u) \in \mathbb{R}^p : (u, v) \in \mathcal{R}\}$ is a cone, and $C_{\mathcal{R}}$ induces the binary relation \mathcal{R} . We call $C_{\mathcal{R}}$ ordering cone. If \mathcal{R} is additionally compatible with addition, then the following statements hold:*

1. $0 \in C_{\mathcal{R}}$ if and only if \mathcal{R} is reflexive.
2. $C_{\mathcal{R}}$ is pointed if and only if \mathcal{R} is antisymmetric.
3. $C_{\mathcal{R}}$ is convex if and only if \mathcal{R} is transitive.

2.2. Optimization Problems and Optimality Concepts

To introduce different optimality concepts, we consider first a general *optimization problem*:

$$\begin{aligned} \text{“minimize”} \quad & f(x) = (f_1(x), \dots, f_p(x))^\top \\ \text{s. t.} \quad & x \in X. \end{aligned} \tag{OP}$$

We consider throughout this thesis only combinatorial optimization problems. Hence we assume that the *feasible set* X is a subset of the power set of a finite discrete ground set E , i.e., $X \subseteq 2^E$. We call f the *objective function* of problem (OP), which usually maps feasible solutions $x \in X$ to an outcome space, which is often the \mathbb{R}^p . In the case of ordinal objective functions we have a different outcome space, see Chapter 4 and 5. In this section we assume $f : X \rightarrow \mathbb{R}^p$. For combinatorial optimization problems some weights $w(e)$ are often assigned to every element e of the ground set E , i.e., $w : E \rightarrow \mathbb{R}^p$ or $w : E \rightarrow \mathbb{Z}_{\geq}^p$, and the objective function is defined as the sum over all weights of the elements of a feasible solution $x \in X$, i.e., $f(x) = \sum_{e \in x} w(e)$. In this case we often denote the objective function $f : X \rightarrow \mathbb{R}^p$ as $w : X \rightarrow \mathbb{R}^p$, in slight abuse of the notation, and we call it *sum objective function* or *weight function*.

If $p = 1$, we call the problem (OP) a *single-objective optimization problem* and the term “minimize” means minimization in \mathbb{R} . For $p > 1$ the problem (OP) is a *multi-objective optimization problem* and there are different possibilities to define “minimize”. In the following, we introduce some optimality concepts for minimization problems. However, all of the presented concepts can be defined analogously for maximization problems. For an introduction into the field of multi-objective optimization, see, e.g., Ehrgott, 2005 and Miettinen, 1999.

Cone Optimality Due to the close relation between orders and cones, see Section 2.1, it is possible to define a general optimality concept based on cones. Lemma 2.2 implies that binary relations \mathcal{R} that are compatible with scalar multiplications can be equivalently represented by associated ordering cones $C_{\mathcal{R}}$. This interrelation was used, among others, in Engau, 2007 to define the concept of *cone-efficiency* (or $C_{\mathcal{R}}$ -efficiency). For a general introduction to ordering cones in the context of vector optimization see, e.g., Tammer and Göpfert, 2003; Jahn, 2011.

Definition 2.3 (c.f. Engau 2007). *Let $Y \subset \mathbb{R}^p$ be a nonempty set and let $C_{\mathcal{R}} \subset \mathbb{R}^p$ be a cone induced by a strict partial order $\mathcal{R} \subset \mathbb{R}^p \times \mathbb{R}^p$ (i.e., \mathcal{R} is irreflexive and transitive). Then the sets*

$$\begin{aligned} N(Y, C_{\mathcal{R}}) &:= \{y \in Y : (\{y\} \oplus (-C_{\mathcal{R}})) \cap Y = \emptyset\} \\ N_w(Y, C_{\mathcal{R}}) &:= \{y \in Y : (\{y\} \oplus (-\text{int}(C_{\mathcal{R}}))) \cap Y = \emptyset\} \end{aligned}$$

are called the $C_{\mathcal{R}}$ -non-dominated set and the weakly $C_{\mathcal{R}}$ -non-dominated set of Y , respectively. The corresponding pre-images $x \in X$ are called $C_{\mathcal{R}}$ -efficient and weakly $C_{\mathcal{R}}$ -efficient, respectively. Thereby, $\text{int}(C_{\mathcal{R}})$ denotes the interior of $C_{\mathcal{R}}$ and the Minkowski sum for two sets $S_1, S_2 \subseteq \mathbb{R}^p$ is denoted by $S_1 \oplus S_2 := \{s_1 + s_2 : s_1 \in S_1, s_2 \in S_2\}$.

Furthermore, we say that y' $C_{\mathcal{R}}$ -dominates \hat{y} for $y', \hat{y} \in Y$ if $y' \in (\{\hat{y}\} \oplus (-C_{\mathcal{R}}))$, and that y' strongly $C_{\mathcal{R}}$ -dominates \hat{y} if $y' \in (\{\hat{y}\} \oplus (-\text{int}(C_{\mathcal{R}})))$.

If cone optimality is used with a cone $C_{\mathcal{R}}$, we write $\min_{C_{\mathcal{R}}}$ instead of “minimize” in the optimization problem (OP).

Cone dominance is visualized in Figure 2.1 for the cone C_{\geq} induced by the component-wise order defined in (2.2).

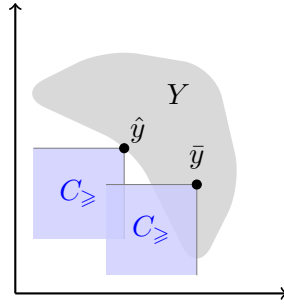


Figure 2.1.: Visualization of cone dominance for the cone C_{\succcurlyeq} . The outcome vector \bar{y} is C_{\succcurlyeq} -dominated while \hat{y} is C_{\succcurlyeq} -non-dominated.

Pareto Optimality One of the most common concepts for multi-objective optimization is the Pareto concept of optimality (see again, e.g., Ehrgott, 2005). Pareto optimality is based on the componentwise orders (2.1), (2.2) and (2.3) and a special case of cone optimality with the corresponding cones to the componentwise orders. The componentwise order (2.2) induces the *Pareto cone*, which is given by

$$C_P := \mathbb{R}_{\geq}^p = \{(v - u) \in \mathbb{R}^p : u \leq v\} = \{y \in \mathbb{R}^p : y \geq 0\}.$$

Similarly, the weak componentwise order (2.1) induces the *closure of the Pareto cone* given by $\mathbb{R}_{\geq}^p = \{y \in \mathbb{R}^p : y \geq 0\} = C_P \cup \{0\}$, which is a proper, pointed and convex cone (see Lemma 2.2). Moreover, it is a polyhedral cone that is defined by the identity matrix, i.e., $C_P \cup \{0\} = \text{hcone}(I) = \text{vcone}(I)$ (where I is the $p \times p$ identity matrix). Note also that the Pareto cone is self dual, i.e., $C_P^* = C_P$.

Efficient solutions and non-dominated points can be defined due to Definition 2.3 with respect to the Pareto cone, i.e., $N(Y, C_P)$ is the non-dominated set (or *Pareto set*) of Y , and $N_w(Y, C_P)$ is the weakly non-dominated set of Y . Equivalently, Pareto dominance can be defined by the underlying relation of the Pareto cone. As the definition based on the componentwise order is in some cases more useful we add this definition here. Hence, let $x', \hat{x} \in X$ be two feasible solutions and $y' := f(x'), \hat{y} := f(\hat{x}) \in \mathbb{R}^p$ be the corresponding outcome vectors. Then we say

- $y' (x')$ dominates $\hat{y} (\hat{x})$ if and only if $y' \leq \hat{y}$,
- $y' (x')$ strongly dominates $\hat{y} (\hat{x})$ if and only if $y' < \hat{y}$,
- $\hat{x} \in X$ is *efficient* or *Pareto optimal* if there does not exist another feasible solution $x' \in X$ that dominates \hat{x} , i.e., $\nexists x' \in X : f(x') \leq f(\hat{x})$,
- $\hat{x} \in X$ is called *weakly efficient* or *weakly Pareto optimal* if there does not exist $x' \in X$ that strongly dominates \hat{x} , i.e., for which $f(x') < f(\hat{x})$,
- the image $f(\hat{x})$ of an efficient solution \hat{x} is the *non-dominated* outcome vector or non-dominated point,

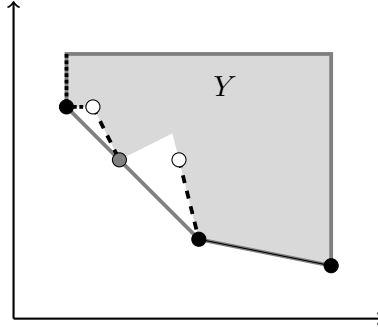


Figure 2.2.: Illustration of Pareto dominance.

- an efficient solution x is *supported efficient* if it is a minimizer of the non-trivial weighted sum problem $\min\{\sum_{i=1}^p \lambda_i f_i(x), x \in X\}$ for some $\lambda_i \in [0, 1]$, $i = 1, \dots, p$ and $\sum_{i=1}^p \lambda_i = 1$, see Section 2.3,
- the image $f(x)$ of a supported efficient solution x is a *supported non-dominated outcome vector or point* (it lies on the boundary of the convex hull $\text{conv}(Y)$ of the set $Y = f(X)$ of feasible outcome vectors in the objective space),
- $f(x)$ is an *extreme supported non-dominated outcome vector or point* if $f(x)$ is an supported non-dominated point and if $f(x)$ is an extreme point of $\text{conv}(Y)$.

In Figure 2.2 a light gray set Y of feasible outcome vectors and its convex hull, depicted by the dark gray line, is given. The black points are extreme supported non-dominated outcome vectors, the black points together with the dark gray point and all points on solid black lines are supported non-dominated points. The supported non-dominated points together with the points on the dashed black lines are non-dominated outcome vectors and all non-dominated outcome vectors together with the white points and all points on the dotted lines are weakly non-dominated points.

In the following we also write “Pareto-efficient”, “Pareto-dominates” and “Pareto-non-dominated” to make clear, that we consider the Pareto concept of optimality. If it is apparent from the context that Pareto optimality is meant, we omit the additional use of “Pareto”. If we use the Pareto concept of optimality, we write \min instead of “minimize” in the optimization problem (OP). As the Pareto optimality can be interpreted as a special case of cone optimality, we sometimes also write \min_{C_P} to emphasize that Pareto optimality is used.

Lexicographic Optimality The concept of lexicographic optimality assumes a specific ordering among the components of the given outcome vectors, i.e., the first component is more important than the second, and so on. We refer again to Ehrgott, 2005 for a more general introduction. Lexicographic optimality is based on the lexicographic orders 2.4 and 2.5, see Section 2.1. Let $x', \hat{x} \in X$ and $y' := f(x')$, $\hat{y} := f(\hat{x}) \in \mathbb{R}^p$. We say y' *lexicographically dominates* \hat{y} if and only if $y' <_{\text{lex}} \hat{y}$. Consequently, we call a feasible solution $\hat{x} \in X$ of problem (OP) *lexicographically optimal* if $f(\hat{x}) \leq_{\text{lex}} f(x)$ for all $x \in X$.

Note that the lexicographically optimal solutions of a problem are a subset of the Pareto efficient solutions of the same problem, see Ehrgott, 2005. To distinguish lexicographic optimization from other optimality concepts we write lexmin (and lexmax in the case of maximization problems, respectively).

Combined Orderings Some optimization problems have an objective function that maps feasible solutions to outcome vectors on which several of the above orderings are combined. This may be the case when, for example, the first p components of an outcome vector represent sum objective functions that are ordered w.r.t. Pareto dominance, while the following q objective values have to be minimized due to cone optimality with a different cone than the Pareto cone. We say that a feasible solution x' *dominates* a feasible solution \hat{x} , if all objective function values of x' are “at least as good” w.r.t. all components and if there exists at least one strict inequality in one of the respective ordering concepts. Similarly, we say x' *strongly dominates* a feasible solution \hat{x} if and only if all objective function values of x' strongly dominate all objective function values of \hat{x} w.r.t. the respective optimality concept.

We are interested in finding the *efficient set* (or the *weakly efficient set*) of problem (OP), possibly with combined orderings, given by

$$X_E := \{x \in X : \text{there exists no } x' \in X \text{ with } f(x') \text{ dominates } f(x)\} \text{ and}$$

$$X_{wE} := \{x \in X : \text{there exists no } x' \in X \text{ with } f(x') \text{ strongly dominates } f(x)\},$$

respectively. Given X_E and X_{wE} , the images of these two sets under the objective function f are called *non-dominated set* and *weakly non-dominated set*, respectively:

$$Y_N := f(X_E)$$

$$Y_{wN} := f(X_{wE})$$

If we consider efficient and non-dominated sets of different optimization problems, we indicate by superscripts to which problem the efficient or non-dominated set belong. A subset $X_{cE} \subseteq X_E$ satisfying $f(X_{cE}) = Y_N$ is called a *complete set of efficient solutions*. Note that in general $X_{cE} \subsetneq X_E$. If in addition $|f(X_{cE})| = |Y_N|$ holds true, we say that the set X_{cE} is of minimal cardinality or just minimal, for short. Here and in the following, $|S|$ denotes the cardinality of a set S . Note that in this case, X_{cE} contains exactly one efficient solution for each vector in the non-dominated set. Algorithms often aim to compute Y_N and X_{cE} rather than Y_N and X_E .

2.3. Scalarizations

In the following we review several frequently used scalarization techniques for a *multi-objective optimization problem* (MOP) where $C_{\mathcal{R}} \subseteq \mathbb{R}^p$ is an ordering cone, i.e.,

$$\begin{aligned} \min_{C_{\mathcal{R}}} \quad & f(x) = (f_1(x), \dots, f_p(x))^{\top} \\ \text{s. t.} \quad & x \in X. \end{aligned} \tag{MOP}$$

All scalarization techniques presented in the following transform the multi-objective problem (MOP) into a single-objective problem to which single-objective solution methods and available solvers can be applied. As the concept of Pareto optimality is mostly used, we investigate this specific case afterwards.

First, we introduce the Pascoletti-Serafini scalarization and show that the weighted Tchebycheff scalarization as well as a hybrid scalarization for (MOP) with Pareto optimality can be deduced from the Pascoletti-Serafini scalarization by an appropriate choice of the parameters. Moreover, from the hybrid scalarization we deduce the Benson scalarization, the ε -constrained scalarization and the weighted sum scalarization as special cases.

We refer to, e.g., Miettinen, 1999, Ehrgott, 2005 or Dächert, 2014 for a thorough introduction to scalarization techniques for Pareto optimality, covering also some additional scalarization methods like compromise programming. Some of the techniques are explained for general cones, for example, in Engau, 2007. Another scalarization technique for general cones is also, for example, described in Göpfert et al., 2003. Further references are given for each technique individually in the corresponding paragraph.

Pascoletti-Serafini Scalarization The Pascoletti-Serafini scalarization is introduced by Pascoletti and Serafini, 1984 for maximization problems w.r.t. cone optimality for closed convex cones. They formulate the scalarization for the general case where the objective space is any finite-dimensional real linear space.

The underlying idea is to start a search in the objective space from a reference point z into a search direction $d \in \text{int}(C)$, which points into the interior of a general convex cone C . The aim is to find the smallest possible step length $t \in \mathbb{R}$ such that a feasible outcome vector of problem (MOP), which corresponds to a weakly $C_{\mathcal{R}}$ -efficient solution, is obtained. Since t can also be negative, a feasible reference point can be improved.

In the following, we assume that the objective space is \mathbb{R}^p and that the ordering cone $C_{\mathcal{R}}$ is pointed, closed and convex, i.e., the underlying relation \mathcal{R} has to be antisymmetric and transitive. Furthermore, we assume that the interior of the ordering cone is non-empty, i.e., $\text{int}(C_{\mathcal{R}}) \neq \emptyset$. This case was investigated, e.g., by Eichfelder, 2007.

Now, let $z \in \mathbb{R}^p$ be the reference point and $d \in \text{int}(C_{\mathcal{R}})$ be a search direction pointing into the interior of the cone $C_{\mathcal{R}}$. Then the *Pascoletti-Serafini scalarization* is defined as

$$\begin{aligned} \min \quad & t \\ \text{s. t.} \quad & z + td - f(x) \in C_{\mathcal{R}} \\ & t \in \mathbb{R} \\ & x \in X. \end{aligned} \tag{PSMOP}(z,d)$$

An illustration of the Pascoletti-Serafini scalarization is given in Figure 2.3. The following result states that every optimal solution t^* , x^* of the Pascoletti-Serafini scalarization leads to a weakly $C_{\mathcal{R}}$ -efficient solution x^* for the corresponding multi-objective problem (MOP). Moreover, every weakly $C_{\mathcal{R}}$ -efficient solution of the multi-objective problem (MOP) can be computed with the Pascoletti-Serafini scalarization for an appropriate choice of the parameters z and d .

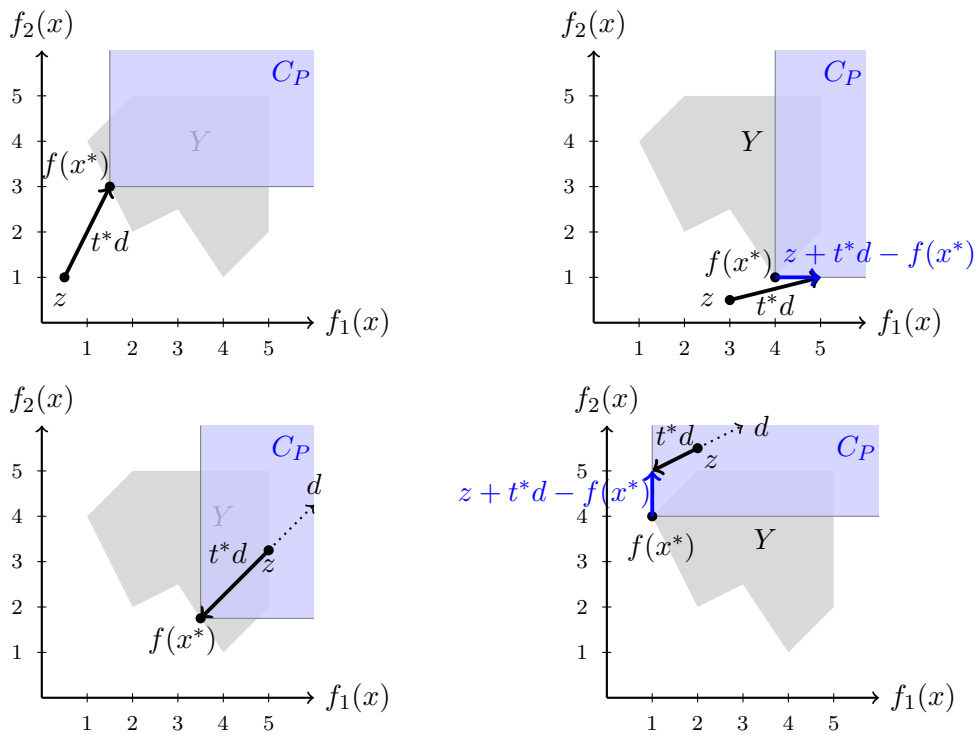


Figure 2.3.: Illustration of the Pascoletti-Serafini scalarization for the closure of the Pareto cone C_P and different choices of the reference point $z \in \mathbb{R}^2$ and the search direction $d \in \mathbb{R}^2_{>}$.

Theorem 2.4 (see, e.g., Eichfelder, 2008). *Let a problem (MOP) be given with a pointed, convex and closed cone $C_{\mathcal{R}}$ and a set of feasible outcome vectors Y .*

- *If t^* and x^* are optimal for (PSMOP(z,d)) with $z \in \mathbb{R}^p$ and $d \in \text{int}(C_{\mathcal{R}})$, then x^* is weakly $C_{\mathcal{R}}$ -efficient for problem (MOP).*
- *If x^* is a $C_{\mathcal{R}}$ -efficient solution of problem (MOP), then $t = 0$ and $x = x^*$ is an optimal solution of problem (PSMOP(z,d)) with $z := f(x^*)$ and arbitrary $d \in C_{\mathcal{R}} \setminus \{0\}$.*
- *If x^* is a weakly $C_{\mathcal{R}}$ -efficient solution of problem (MOP), then $t = 0$ and $x = x^*$ is an optimal solution of problem (PSMOP(z,d)) with $z := f(x^*)$ and arbitrary $d \in \text{int}(C_{\mathcal{R}})$.*

All scalarization techniques we present below for the Pareto cone can be interpreted as a special case of the Pascoletti-Serafini scalarization, as shown in Eichfelder, 2007.

Weighted Tchebycheff Scalarization The Tchebycheff scalarization is a special case of compromise programming, see, e.g., Ehrgott, 2005 for more details. The idea of compromise programming is to find a feasible point, that has the smallest possible distance to the ideal point z^I . The *ideal point* is given by

$$z_i^I := \min\{f_i(x) : x \in X\} \text{ for all } i = 1, \dots, p$$

and is in most applications infeasible, i.e., $z^I \notin Y$. Thereby, the distance between two points is often measured by a ℓ_p -norm with $p \in [1, \infty]$. If the ℓ_∞ -norm is used, we call this method Tchebycheff scalarization.

Bowman, 1976 developed this method further by including weights $\lambda \in \mathbb{Z}_{>}^p$ in the distance measure, resulting in the following weighted Tchebycheff scalarization of \bar{f} (MOP) with Pareto optimality:

$$\begin{aligned} \min \quad & \max_{i=1, \dots, p} \{\lambda_i \cdot |f_i(x) - z_i^U|\} \\ \text{s. t.} \quad & x \in X. \end{aligned} \tag{2.6}$$

Note that here an *utopian point* $z^U \in \mathbb{R}^p$, given by $z^U < z^I$, is used instead of the ideal point z^I . Otherwise it is not possible to compute the lexicographically optimal solutions with strictly positive weights $\lambda_i > 0$ for $i = 1, \dots, p$. The weighted Tchebycheff scalarization is illustrated in Figure 2.4(a). Every solution of problem (2.6) is weakly $C_{\mathcal{R}}$ -efficient for the corresponding problem (MOP) with Pareto optimality. It is $C_{\mathcal{R}}$ -efficient if the solution is unique, see Bowman, 1976. Steuer and Choo, 1983 investigated this scalarization technique in detail and they introduced two variants of this scalarization technique. The first variant is a two-stage method and the second variant includes augmentation terms.

Above we consider the case of Pareto optimality, but we are also interested in multi-objective optimization problems under general cone optimality. Thus, we want to find the weighted Tchebycheff scalarization for the general case. Towards this end, we need a reformulation of problem (2.6). First, we recognize that $z_i^U < f_i(x)$ for all $i = 1, \dots, p$ and for all $x \in X$. Hence, we can omit the absolute value sign $|\cdot|$. Furthermore, we introduce

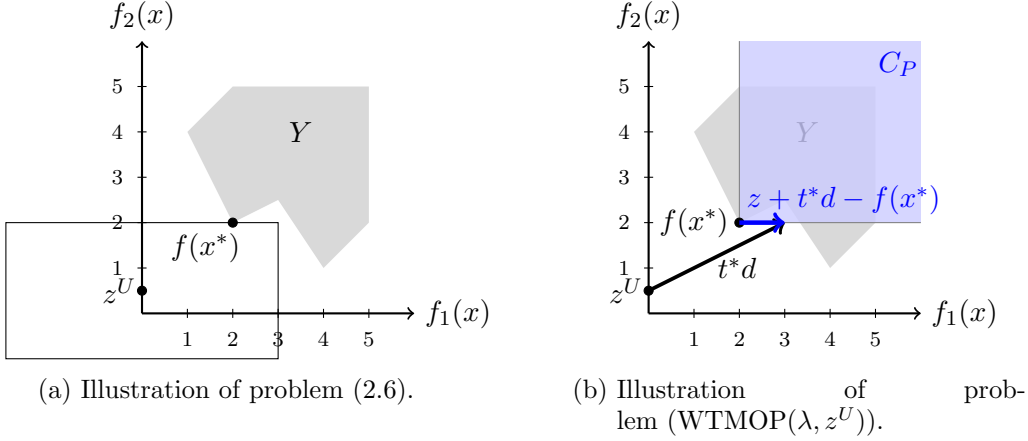


Figure 2.4.: Illustration of the weighted Tchebycheff scalarization w.r.t. the Pareto cone C_P for an utopian point $z^U \in \mathbb{R}^2$ and the weight $\lambda = (1, 2)^\top$.

a new variable $t \in \mathbb{R}$ to replace the objective function by adding some inequalities:

$$\begin{aligned}
 \min \quad & t \\
 \text{s. t.} \quad & t \geq \lambda_i (f_i(x) - z_i^U) \quad i = 1, \dots, p \\
 & t \in \mathbb{R} \\
 & x \in X.
 \end{aligned} \tag{2.7}$$

For strictly positive weights $\lambda_i > 0$ this formulation is equivalent to

$$\begin{aligned}
 \min \quad & t \\
 \text{s. t.} \quad & z_i^U + \frac{1}{\lambda_i} t - f_i(x) \geq 0 \quad i = 1, \dots, p \\
 & t \in \mathbb{R} \\
 & x \in X.
 \end{aligned} \tag{2.8}$$

This was shown, for example, in Dächert, 2014. Finally, we can formulate the equivalent *weighted Tchebycheff scalarization for general cones* and with strictly positive weights $\lambda \in \mathbb{R}_{>}^p$ as:

$$\begin{aligned}
 \min \quad & t \\
 \text{s. t.} \quad & z^U + t \left(\frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_p} \right)^\top - f(x) \in C_{\mathcal{R}} \\
 & t \in \mathbb{R} \\
 & x \in X.
 \end{aligned} \tag{WTMOP(λ, z^U)}$$

This formulation can be interpreted as a special case of the Pascoletti-Serafini scalarization with reference point z^U and search direction $d := \left(\frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_p} \right)^\top$, see Figure 2.4(b) for an illustration. Due to Theorem 2.4 every weakly $C_{\mathcal{R}}$ -efficient solution of problem (MOP) can be computed with the weighted Tchebycheff scalarization, as long as $\left(\frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_p} \right)^\top \in \text{int}(C_{\mathcal{R}})$ and $C_{\mathcal{R}}$ is a pointed, closed and convex cone.

Hybrid Scalarization Let $Y \subseteq \mathbb{R}^p$ be the set of feasible outcome vectors and let $C_{\mathcal{R}} \subseteq \mathbb{R}^p$ be a convex ordering cone of problem (MOP). Furthermore, let $\lambda \in (C_{\mathcal{R}})^* \setminus \{0\}$ and $\varepsilon \in \mathbb{R}^p$. Then the *hybrid scalarization* of problem (MOP), see, e.g., Engau, 2007, is defined as

$$\begin{aligned} \min \quad & \sum_{i=1}^p \lambda_i f_i(x) \\ \text{s. t.} \quad & \varepsilon - f(x) \in C_{\mathcal{R}} \\ & x \in X. \end{aligned} \quad (\text{HSMOP}(\lambda, \varepsilon))$$

In the following, we review a result on the connection between the solutions of problem (HSMOP(λ, ε)) and problem (MOP). Here we only need that the cone $C_{\mathcal{R}}$ is convex. The proof is included for the sake of completeness and to explain which assumptions have to be made at the ordering cone $C_{\mathcal{R}}$ in case of the weighted sum scalarization.

Theorem 2.5 (see, e.g., Engau, 2007). *Let $y' \in Y$ be an optimal outcome vector for the hybrid scalarization of problem (MOP) with a convex ordering cone $C_{\mathcal{R}}$, $\lambda \in (C_{\mathcal{R}})^* \setminus \{0\}$ and $\varepsilon \in \mathbb{R}^p$. Then it holds that y' is weakly $C_{\mathcal{R}}$ -efficient for the corresponding problem (MOP), i.e., $y' \in N_w(Y, C_{\mathcal{R}})$. If $\lambda \in (C_{\mathcal{R}})_s^*$, then y' is $C_{\mathcal{R}}$ -efficient for the corresponding problem (MOP), i.e., $y' \in N(Y, C_{\mathcal{R}})$.*

Proof. We first show that $\lambda \in (C_{\mathcal{R}})_s^*$ implies $y' \in N(Y, C_{\mathcal{R}})$ by contradiction. Hence, we assume $y' \in Y$ is optimal for (HSMOP(λ, ε)), $\lambda \in (C_{\mathcal{R}})_s^*$ and $y' \notin N(Y, C_{\mathcal{R}})$. Then there exists a $y \in Y$ that dominates y' , i.e., $y' = y + c$ for some $c \in C_{\mathcal{R}} \setminus \{0\}$. The outcome vector y is feasible for (HSMOP(λ, ε)), i.e., $\varepsilon - y = \varepsilon - y' + c \in C_{\mathcal{R}}$, due to the feasibility of y' for (HSMOP(λ, ε)) and the convexity of $C_{\mathcal{R}}$. Furthermore, it holds that $\lambda y' = \lambda y + \lambda c > \lambda y$ as $\lambda \in (C_{\mathcal{R}})_s^*$. This contradicts the optimality of y' for (HSMOP(λ, ε)).

It is left to show that for all $\lambda \in (C_{\mathcal{R}})^* \setminus \{0\}$ with an optimal outcome vector $y' \in Y$ for (HSMOP(λ, ε)) it holds that $y' \in N_w(Y, C_{\mathcal{R}})$. First, we observe that the assumption

$$\lambda \in (C_{\mathcal{R}})^* \setminus \{0\} = \{d \in \mathbb{R}^p : d^\top c \geq 0 \text{ for all } c \in C_{\mathcal{R}} \setminus \{0\}\}$$

is equivalent to

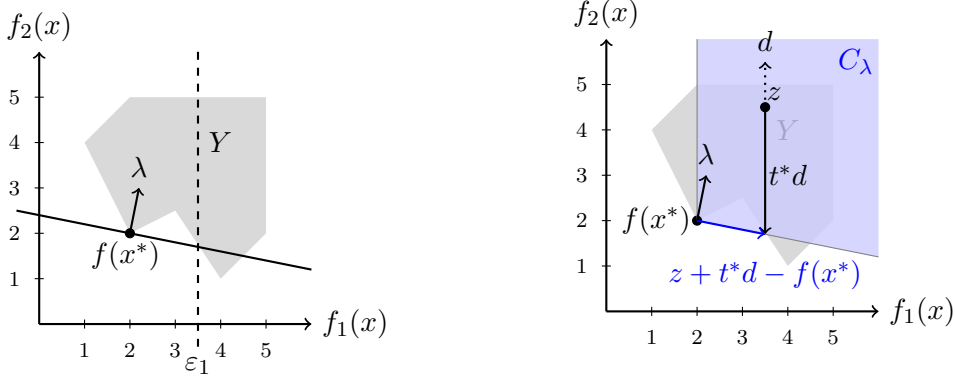
$$\lambda \in (\text{int}(C_{\mathcal{R}}))_s^* = \{d \in \mathbb{R}^p : d^\top c > 0 \text{ for all } c \in \text{int}(C_{\mathcal{R}}) \setminus \{0\}\}$$

as linear functions in \mathbb{R}^p are continuous. Hence, it follows that $y' \in N(Y, \text{int}(C_{\mathcal{R}}))$ which is equivalent to $y' \in N_w(Y, C_{\mathcal{R}})$. \square

We emphasize that we need a convex cone to guarantee the above optimality results. This means that the underlying binary relation \mathcal{R} has to be transitive, see Lemma 2.2.

Guddat et al., 1985 have proven this result for the special case of the Pareto cone $C_{\mathcal{R}} = C_P$, with the bound $\varepsilon = f(\hat{x})$ and with some feasible $\hat{x} \in X$. Note that in case of the Pareto cone $\lambda \in (C_P)_s^*$ is equivalent to $\lambda_i > 0$ for all $i = 1, \dots, p$ and $\varepsilon - f(x) \in C_P$ is equivalent to $f(x) \leq \varepsilon$.

In the case of the Pareto cone, sometimes only some of the constraints $f_i(x) \leq \varepsilon_i$, $i = 1, \dots, p$, from problem (HSMOP(λ, ε)) are used. Let $Q \subseteq \{1, \dots, p\}$ denote the set of



(a) Illustration of problem (HSMOP(λ, ε, Q)) with $\lambda = (0.2, 1)^\top$, $\varepsilon_1 = 3.5$ and $Q = \{1\}$.

(b) Illustration of problem (2.9) with $\lambda = (0.2, 1)^\top$, $z = (3.5, 4.5)$ and $d = (0, 1)^\top$.

Figure 2.5.: Illustration of the hybrid scalarization.

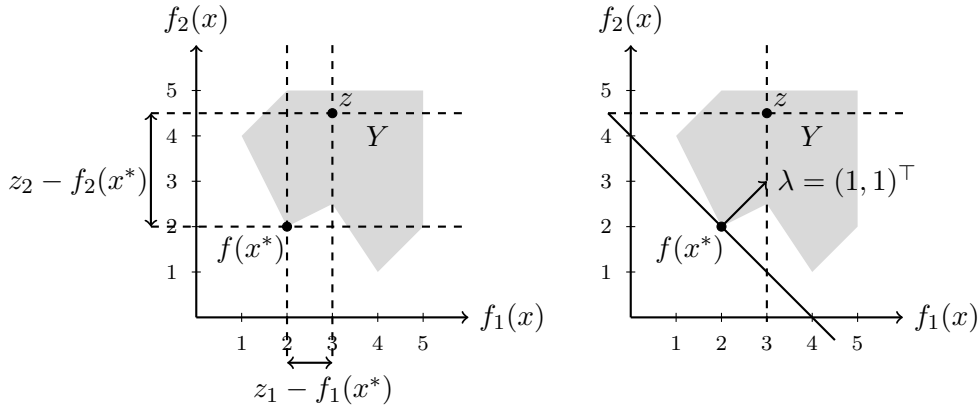
indices of the constraints that are present in (HSMOP(λ, ε)). Then we get the following variant of the hybrid scalarization

$$\begin{aligned} \min \quad & \sum_{i=1}^p \lambda_i f_i(x) \\ \text{s. t.} \quad & f_i(x) \leq \varepsilon_i, \quad \forall i \in Q \\ & x \in X. \end{aligned} \quad (\text{HSMOP}(\lambda, \varepsilon, Q))$$

An illustration of the hybrid scalarization with $\lambda = (1, 5)^\top$, $\varepsilon = 3.5$ and $Q = \{1\}$ is given in Figure 2.5(a). Note that problem (HSMOP(λ, ε, Q)) is equivalent to problem (HSMOP(λ, ε)) with the Pareto cone, i.e., $C_{\mathcal{R}} = C_P$, if $Q = \{1, \dots, p\}$. If this is not the case, i.e., if $Q \subsetneq \{1, \dots, p\}$, then the hybrid scalarization (HSMOP(λ, ε, Q)) is a special case of the Pascoletti-Serafini scalarization with the reference point $z = \varepsilon$ and a search direction d with $d_i = 0$ for all $i \in Q$ and $d_i = 1$ for all $i \in \{1, \dots, p\} \setminus Q$ under a specific ordering cone which depends on λ . Furthermore, if Q contains all indices except for one, i.e., $Q = \{1, \dots, j-1, j+1, \dots, p\}$ with $j \in \{1, \dots, p\}$, then d is equal to the j -th unit vector, i.e., $d = \mathbf{e}_j$. The following theorem shows the correspondence between the hybrid scalarization and the Pascoletti-Serafini scalarization, see, e.g., Eichfelder, 2007.

Theorem 2.6 (see, e.g., Eichfelder, 2007). *Let $Q \subsetneq \{1, \dots, p\}$ and $\lambda \in \mathbb{R}^p$ such that $\sum_{i \notin Q} \lambda_i > 0$. Then a solution $x^* \in X$ is an optimal solution of problem (HSMOP(λ, ε, Q)) if and only if there is a t^* such that (t^*, x^*) is an optimal solution of problem (PSMOP(z, d)) with $z_i = \varepsilon_i$ for $i \in Q$, z_i arbitrary for $i \in \{1, \dots, p\} \setminus Q$, $d_i = 0$ for all $i \in Q$, $d_i = 1$ for all $i \in \{1, \dots, p\} \setminus Q$ and the cone $C_\lambda := \{y \in \mathbb{R}^p : y_i \geq 0, \forall i \in Q, \lambda^\top y \geq 0\}$, i.e., of*

$$\begin{aligned} \min \quad & t \\ \text{s. t.} \quad & z + t d - f(x) \in C_\lambda \\ & t \in \mathbb{R} \\ & x \in X. \end{aligned} \quad (2.9)$$



(a) Illustration of problem (BMOP(z)) with $z = (3, 4.5)^\top$.

(b) Illustration of problem (2.10) with $z = (3, 4.5)^\top$.

Figure 2.6.: Illustration of the Benson scalarization for the Pareto cone C_P .

The Pascoletti-Serafini scalarization, which corresponds to the hybrid scalarization in Figure 2.5(a), is illustrated in Figure 2.5(b).

Benson Scalarization If we choose $\lambda = (1, \dots, 1)^\top$ and $z \in Y$ in the hybrid scalarization, we get the Benson scalarization, see, e.g., Engau, 2007:

$$\begin{aligned} \min \quad & \sum_{i=1}^p f_i(x) \\ \text{s. t.} \quad & z - f(x) \in C_{\mathcal{R}} \cup \{0\} \\ & x \in X. \end{aligned} \quad (2.10)$$

Usually, the Benson scalarization is formulated slightly differently to better emphasize the idea of this technique. The idea is to choose some feasible solution $x' \in X$ and maximize the sum of non-negative deviation variables $c_i = f_i(x') - f_i(x)$ over the feasible set. Here, we denote the objective function value of x' by $z := f(x') \in Y$. If x' is already an optimal solution, then the optimal objective function value is zero. Hence, we get the following version of the *Benson scalarization*

$$\begin{aligned} \max \quad & \sum_{i=1}^p z_i - f_i(x) \\ \text{s. t.} \quad & z - f(x) \in C_{\mathcal{R}} \cup \{0\} \\ & x \in X, \end{aligned} \quad (\text{BMOP}(z))$$

that is equivalent to (2.10) since z is constant. From Theorem 2.5 it follows that an optimal solution x^* of problem (BMOP(z)) with a convex cone $C_{\mathcal{R}}$ and $\lambda = (1, \dots, 1)^\top \in (C_{\mathcal{R}})^*$ is weakly $C_{\mathcal{R}}$ -efficient for problem (MOP). If even $\lambda = (1, \dots, 1)^\top \in (C_{\mathcal{R}})_s^*$ holds, then x^* is $C_{\mathcal{R}}$ -efficient for problem (MOP).

This method is first introduced by Benson, 1978 for Pareto optimality. In this case the constraints can be equivalently formulated as $z - f(x) = f(x') - f(x) \geq 0$. In Figure 2.6(a) is the Benson scalarization in the version of problem (BMOP(z)) illustrated. The corresponding version of the hybrid scalarization is given in Figure 2.6(b).

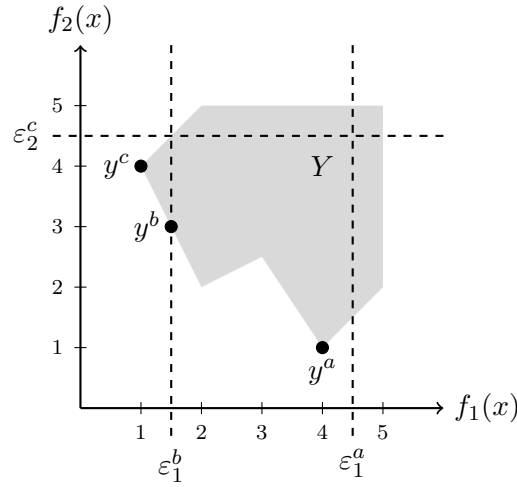


Figure 2.7.: Illustration of the ε -constraint scalarization for three different choices of ε , i.e., $\varepsilon_1^a = 4.5$, $\varepsilon_1^b = 1.5$ and $\varepsilon_2^c = 4.5$, with the corresponding optimal outcome vectors $y^a = (4, 1)^\top$, $y^b = (1.5, 3)^\top$ and $y^c = (1, 4)^\top$.

The ε -Constraint Scalarization If we choose $\lambda = \mathbf{e}_j$ the j -th unit vector, $j \in \{1, \dots, p\}$, together with $\varepsilon \in \mathbb{R}^p$ in the hybrid scalarization, we get the ε -constraint scalarization, see, e.g., Engau, 2007. Let $C_{\mathcal{R}}$ be a convex cone, $\mathbf{e}_j \in (C_{\mathcal{R}})^*$ and $\varepsilon \in \mathbb{R}^p$. Then

$$\begin{aligned} \min \quad & f_j(x) \\ \text{s. t.} \quad & \varepsilon - f(x) \in C_{\mathcal{R}} \\ & x \in X. \end{aligned} \quad (\varepsilon\text{-MOP})$$

From Theorem 2.5 it follows that an optimal solution x^* of problem (ε -MOP) with a convex cone $C_{\mathcal{R}}$ and $\lambda = \mathbf{e}_j \in (C_{\mathcal{R}})^*$ is weakly $C_{\mathcal{R}}$ -efficient for problem (MOP). If it even holds that $\lambda = \mathbf{e}_j \in (C_{\mathcal{R}})_s^*$, then x^* is $C_{\mathcal{R}}$ -efficient for problem (MOP).

If Pareto optimality is considered, then the constraint on $f_j(x)$, which defines the objective function, is often dropped. This is equivalent to choosing $\lambda = \mathbf{e}_j$, $\varepsilon \in \mathbb{R}^p$ and $Q = \{1, \dots, j-1, j+1, \dots, p\}$ in problem (HSMOP(λ, ε, Q)). Hence, we get the following version for Pareto optimality:

$$\begin{aligned} \min \quad & f_j(x) \\ \text{s. t.} \quad & f_i(x) \leq \varepsilon_i, \quad i = 1, \dots, p, \quad i \neq j \\ & x \in X. \end{aligned} \quad (2.11)$$

This version was introduced by Haimes et al., 1971 for a bi-objective optimization problem and further investigated, for example, by Chankong and Haimes, 1983. An illustration of the ε -constraint scalarization is given in Figure 2.7.

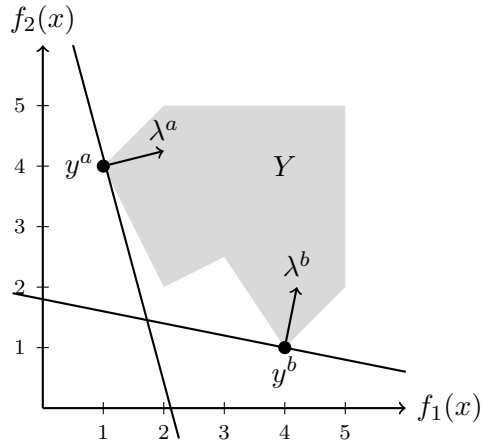


Figure 2.8.: Illustration of the weighted sum scalarization under the Pareto cone C_P for $\lambda^a = (1, 0.25)^\top$ and $\lambda^b = (0.2, 1)^\top$ with optimal outcome vectors $y^a = (1, 4)^\top$ and $y^b = (4, 1)^\top$.

Weighted Sum Scalarization If we drop the constraint $\varepsilon - f(x) \in C_{\mathcal{R}}$ in the hybrid method, we get the *weighted sum scalarization* of problem (MOP):

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

An illustration of the weighted sum scalarization of a problem (MOP) w.r.t Pareto optimality for $\lambda^a = (1, 0.25)^\top$ and $\lambda^b = (0.2, 1)^\top$ is given in Figure 2.8. From Theorem 2.5 it follows that an optimal solution x^* of problem (WSMOP(λ)) with a convex cone $C_{\mathcal{R}}$ and $\lambda \in (C_{\mathcal{R}})^*$ is weakly $C_{\mathcal{R}}$ -efficient for problem (MOP). If it even holds that $\lambda \in (C_{\mathcal{R}})_s^*$, then x^* is $C_{\mathcal{R}}$ -efficient for problem (MOP).

A similar result for closed convex cones can be found in Gearhart, 1983, Jahn, 1984 and Sawaragi et al., 1985, among others. In the proof of Theorem 2.5 the convexity of $C_{\mathcal{R}}$ is only needed to show that the outcome vector y satisfies the constraint $\varepsilon - f(x) \in C_{\mathcal{R}}$. As we dropped this constraint to obtain problem (WSMOP(λ)), Theorem 2.5 even holds for non-convex cones in the case of the weighted sum scalarization.

In the case of the Pareto cone, i.e., $C_{\mathcal{R}} = C_P$, it holds that only supported non-dominated outcome vectors can be found with the weighted sum scalarization. If the set of all feasible outcome vectors Y is convex, then all non-dominated points are supported and hence can be found by a weighted sum scalarization with an appropriate choice of $\lambda \in \mathbb{R}_{\geq}^p$. Furthermore, $\lambda \in (C_P)^*$ holds if and only if $\lambda \geq 0$ and $\lambda \in (C_P)_s^*$ holds if and only if $\lambda > 0$. We call the set of all appropriate values of λ for the computation of an Pareto efficient solution for problem (MOP) the *weight space* $\mathbb{R}_{>}^p := \{\lambda \in \mathbb{R}^p : \lambda > 0\}$.

Obviously, the optimal solution of problem (WSMOP(λ)) does not change, if λ is multiplied with a positive scalar. Hence, we can restrict the space of possible values of λ to the vectors of length one. This leads to the definition of the *normalized weight space* $\tilde{W} := \{\lambda \in \mathbb{R}_{>}^p : \sum_{i=1}^p \lambda_i = 1\}$, see, e.g., Schulze, 2017. As the sum over all weights

λ_i is fixed, we can compute $\lambda_p := 1 - \sum_{i=1}^{p-1} \lambda_i$ from the values of the other weights. Hence, every $\lambda \in \tilde{W}$ can be identified with exactly one vector in the *projected weight space* $W := \{(\lambda_1, \dots, \lambda_{p-1}) \in \mathbb{R}_{>}^{p-1} : \sum_{i=1}^{p-1} \lambda_i < 1\}$, see, e.g., Schulze, 2017. This projected weight space can be divided into subsets such that each subset has a one-to-one correspondence to supported non-dominated outcome vectors for multi-objective linear programming problems. This is shown by Benson and Sun, 2000. The subdivided projected weight space is called *weight space decomposition*. For more details on weight space decompositions, see, for example, Przybylski et al., 2010 or Schulze, 2017.

2.4. Graphs

In the following we give a brief introduction to graphs, which is based on Harris et al., 2008, Pitsoulis, 2014, Hamacher and Klamroth, 2006, Schrijver, 2003 and Ahuja et al., 1993. Note that in the literature some definitions might differ, as we see in the following. A *graph* consists of a set of vertices, sometimes also referred to as nodes, $V = \{v_1, \dots, v_n\}$ and a set of edges $E = \{e_1, \dots, e_m\}$. In this thesis we always consider graphs with a finite number $n = |V|$ of vertices and a finite number $m = |E|$ of edges. Graphs can either be *undirected*, then we write for an edge $e = [v_i, v_j] \in E$ or $e = [i, j]$ with $v_i, v_j \in V$ and it holds $[v_i, v_j] = [v_j, v_i]$ or the graph is *directed*, then we denote an edge by $e = (v_i, v_j) \in E$ or $e = (i, j)$ for $v_i, v_j \in V$ and call the graph *digraph*.

If there are multiple edges between two nodes, we call the graph a *multigraph*. Technically, we need to replace the set E by a multiset in this case. A *loop* is an edge that connects a vertex with itself. A graph is called *simple*, if it is no multigraph and does not contain loops. A *subgraph* $G' = (V', E')$ of a graph $G = (V, E)$ contains as set of vertices a subset of V , i.e., $V' \subseteq V$, and as edge set a subset of the edges of G on V' , i.e., for directed graphs $E' \subseteq \{(v_i, v_j) \in E : v_i, v_j \in V'\}$ and for undirected graphs $E' \subseteq \{[v_i, v_j] \in E : v_i, v_j \in V'\}$.

For directed and undirected graphs the vertices v_i and v_j of an edge $[i, j] \in E$, $(i, j) \in E$ or $(j, i) \in E$ are called *end-vertices*. For an edge $e = (i, j) \in E$ in a directed graph v_i is called *tail* $t(e)$ of e and v_j is called *head* $h(e)$ of e . Two vertices $v_i, v_j \in V$ are called *adjacent* if there is an edge $e \in E$ from v_i to v_j , i.e., $e = [i, j]$ or $e \in \{(i, j), (j, i)\}$, respectively. Furthermore, we say an edge e is *incident* to a vertex v if v is an end-vertex of e . Moreover, $\deg(v)$ denotes the number of edges incident to node v . We define the *adjacency matrix* $M^{AD}(G) = (m_{ij}^{AD}) \in \{0, 1\}^{n \times n}$ and the *incidence matrix* $M^{IN}(G) = (m_{ij}^{IN}) \in \{-1, 0, 1\}^{n \times m}$ for a *simple and directed graph* $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ and $E = \{e_1, \dots, e_m\}$ as follows:

$$m_{ij}^{AD} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$m_{ij}^{IN} = \begin{cases} 1 & \text{if } v_i = t(e_j) \\ -1 & \text{if } v_i = h(e_j) \\ 0 & \text{otherwise} \end{cases}$$

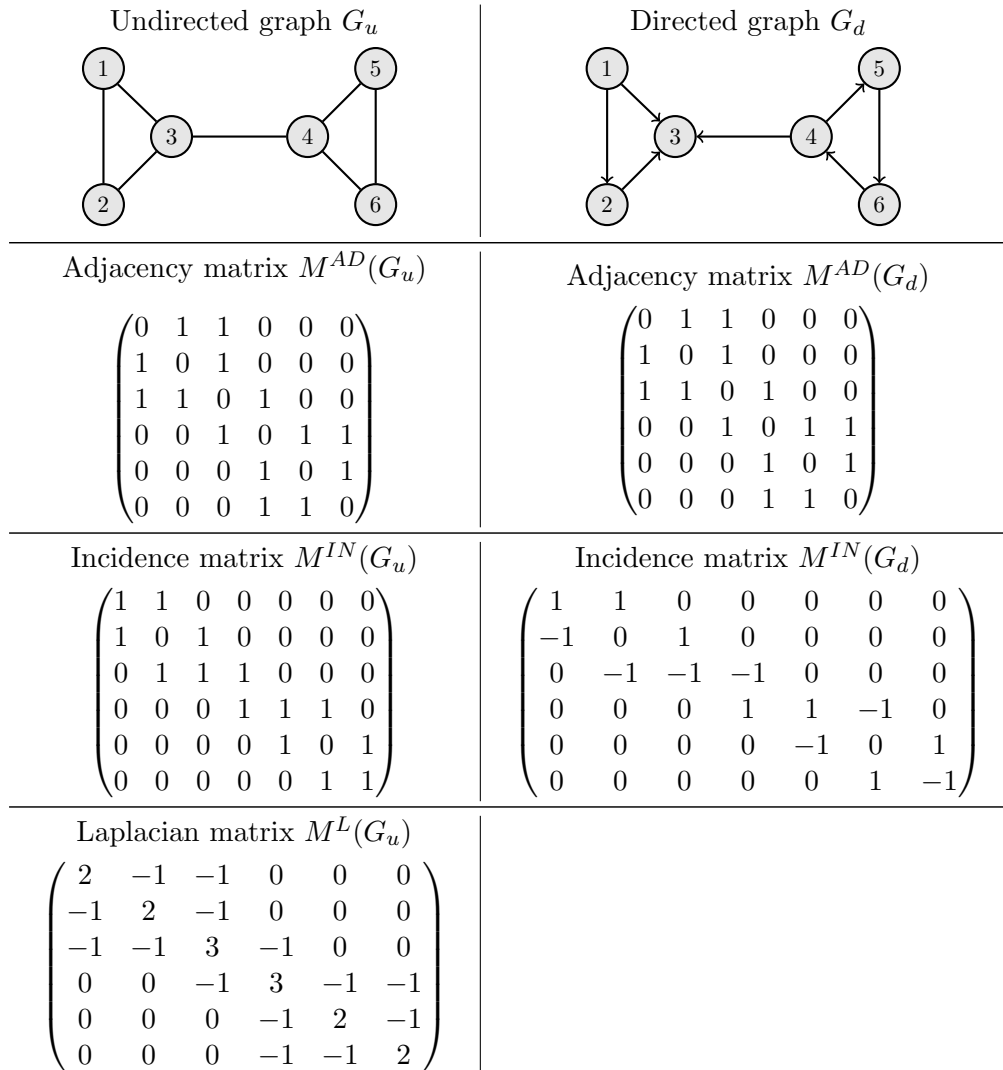


Figure 2.9.: An undirected graph G_u and a directed graph G_d with six vertices and seven edges. Their adjacency matrices, incidence matrices and the Laplacian matrix of G_u is given.

Similarly, the *adjacency matrix* $M^{AD}(G) = (m_{ij}^{AD}) \in \{0, 1\}^{n \times n}$ and the *incidence matrix* $M^{IN}(G) = (m_{ij}^{IN}) \in \{0, 1\}^{n \times m}$ can be defined for a *simple, undirected graph* $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ and $E = \{e_1, \dots, e_m\}$:

$$m_{ij}^{AD} = \begin{cases} 1 & \text{if } [i, j] \in E \\ 0 & \text{otherwise} \end{cases}$$

$$m_{ij}^{IN} = \begin{cases} 1 & \text{if } v_i \text{ is an end-vertex of } e_j \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, we define the *Laplacian matrix* $M^L(G) = (m_{ij}^L)_{i,j=1,\dots,n}$ of a *simple, undirected graph* $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ and $E = \{e_1, \dots, e_m\}$ as follows, see, for example, Stanley, 2013:

$$m_{ij}^L = \begin{cases} -m_{ij}^{AD} & \text{if } i \neq j \text{ and} \\ \deg(v_i) & \text{if } i = j \end{cases}$$

We usually visualize undirected edges by line segments, while we depict directed edges by an arrow. For example, consider a graph with six vertices, which are identified by the numbers $1, \dots, 6$ and seven undirected edges $[1, 2], [1, 3], [2, 3], [3, 4], [4, 5], [4, 6]$ and $[5, 6]$ and a directed graph on the same set of vertices with the edges $(1, 2), (1, 3), (2, 3), (3, 4), (4, 5), (4, 6)$ and $(5, 6)$. The corresponding illustrations for the undirected graph and the directed graph, as well as their adjacency matrices, incidence matrices and the Laplacian matrix of the undirected graph are shown in Figure 2.9.

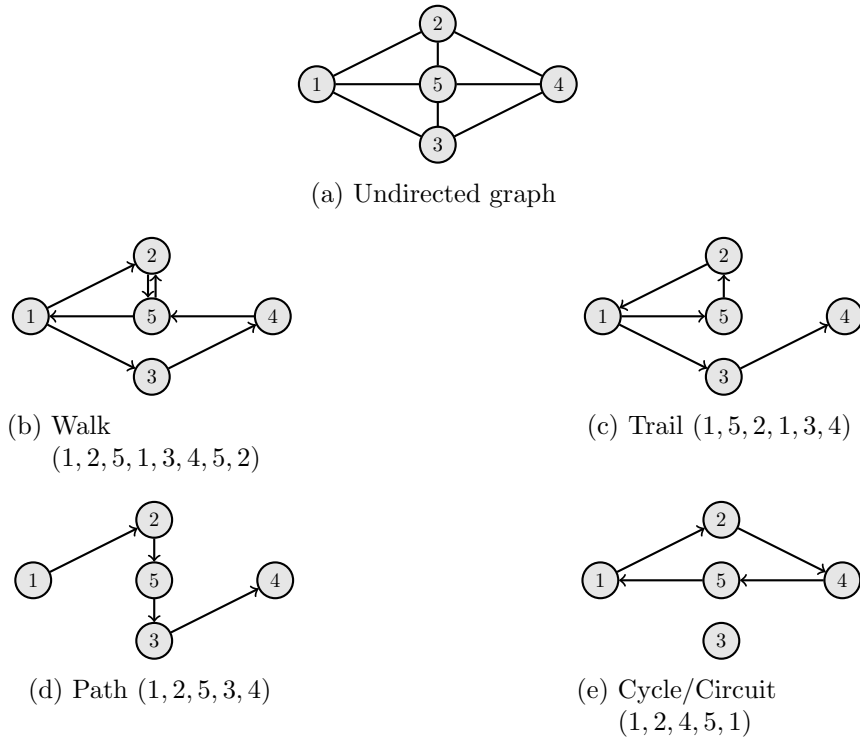


Figure 2.10.: An undirected graph and examples for a walk, a path, a trail and a cycle/circuit.

In the following we introduce some concepts, first for undirected graphs and afterwards for directed graphs. A sequence of vertices $(v_{i_1}, \dots, v_{i_\ell})$ is called a *walk* if $[v_{i_j}, v_{i_{j+1}}] \in E$ for all $j = 1 \dots, \ell - 1$. We call ℓ the *length* of the walk. A walk with distinct vertices, i.e., $v_{i_j} \neq v_{i_k}$ for all $1 \leq j < k \leq \ell$, is called a *path*, while a walk with distinct edges is called a *trail*, see Harris et al., 2008. A path $(v_{i_1}, \dots, v_{i_\ell})$, which starts at v_{i_1} and ends at v_{i_ℓ} is sometimes also called $v_{i_1} - v_{i_\ell}$ path. In general, every path is a trail but not every trail is a path. By adding the edge $[v_{i_\ell}, v_{i_1}] \in E$ to a path $(v_{i_1}, \dots, v_{i_\ell})$ we get a cycle, see Harris

et al., 2008. Similarly, a trail $(v_{i_1}, \dots, v_{i_\ell})$ with $v_{i_1} = v_{i_\ell}$ is called a circuit, see Harris et al., 2008. In the following, we use the terms *cycle* and *circuit* both for closed paths as in Schrijver, 2003¹. An example of a walk, a trail, a path and a cycle/circuit is given in Figure 2.10. We use arrows to show in which direction an edge is used. A walk can use edges multiple times, see Figure 2.10(b). A trail can use vertices multiple times but not edges, see Figure 2.10(c). Moreover, a path can use every edge and every vertex only once, see Figure 2.10(d). This path cannot be extended to a cycle/circuit, because the edge $[4, 1]$ is not in E . But we give an other example for a cycle/circuit in Figure 2.10(e).

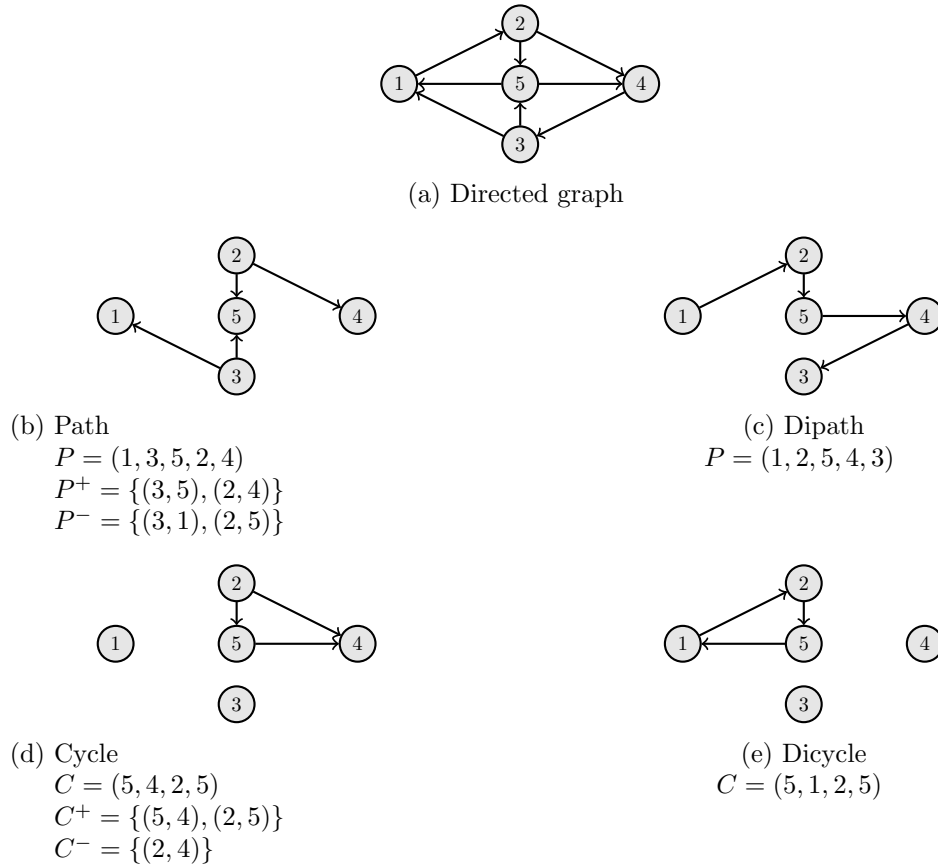


Figure 2.11.: A directed graph and examples for a path, a dipath, a cycle and a dicycle.

Now, we consider a directed graph. A first possibility is to ignore the directions and then we can define walks, paths, trails and cycles as in undirected graphs. For such a path or cycle $P = (v_{i_1}, \dots, v_{i_\ell})$, which uses the edges $(v_{i_j}, v_{i_{j+1}}) \in E$ or $(v_{i_{j+1}}, v_{i_j}) \in E$, $j = 1, \dots, \ell - 1$, we define in a directed graph the set of positive edges in the path or cycle by $P^+ := \{(v_{i_j}, v_{i_{j+1}}) \in P : (v_{i_j}, v_{i_{j+1}}) \in E\}$ and similarly we define the set of negative edges in the path or cycle by $P^- := \{(v_{i_j}, v_{i_{j+1}}) \in P : (v_{i_j}, v_{i_{j+1}}) \notin E \text{ and } (v_{i_{j+1}}, v_{i_j}) \in E\}$.

¹In other books only path and cycles are defined, see, e.g., Ahuja et al., 1993 and Pitsoulis, 2014. In Hamacher and Klamroth, 2006 a path is defined as a walk, which cannot use an edge and return on the same edge immediately after but in general the usage of edges multiple times is allowed. Moreover, a simple path is defined as a path and a cycle is defined as a circuit in Harris et al., 2008.

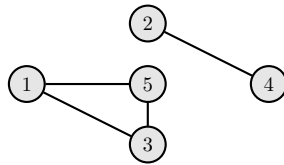
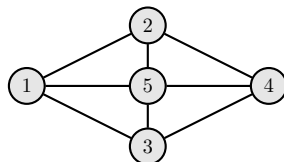


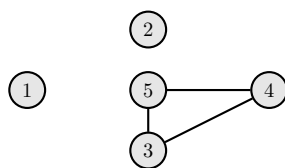
Figure 2.12.: An undirected graph with the connected components $G_1 = (\{2, 4\}, \{[2, 4]\})$ and $G_2 = (\{1, 3, 5\}, \{[1, 3], [1, 5], [3, 5]\})$.

If $P^- = \emptyset$, we call the path or cycle a dipath or dicycle, respectively. In Figure 2.11 an example for a path and a dipath as well as a cycle and a dicycle is given.

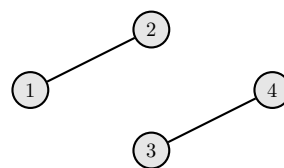
A graph is *connected* if there exists a path from every vertex to all other vertices. For an unconnected graph G every maximal connected subgraph is called *connected component*. For an example consider the graph in Figure 2.12.



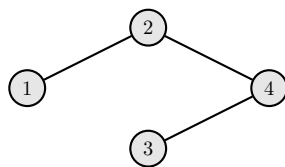
(a) Undirected graph



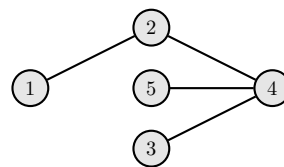
(b) Spanning Subgraph



(c) Forest



(d) Tree



(e) Spanning Tree

Figure 2.13.: An undirected graph and examples for a spanning subgraph, a forest, a tree and a spanning tree.

The following definitions are given only for undirected graphs. There are similar definitions for directed graphs, but we omit them here, as we do not need them throughout this thesis. A subgraph $G' = (V', E')$ of an undirected graph $G = (V, E)$ is called *spanning* if $V' = V$. Every subgraph without cycles is called a *forest* and if it is also connected it is called a *tree*. A tree which is a spanning subgraph is called a *spanning tree*. Obviously, spanning trees exist if and only if the graph is connected, and every spanning tree T has $n - 1$ edges, where n is the number of vertices in G , i.e., $|V| = n$, see Ahuja et al., 1993.

The number of spanning trees is called *complexity* of G and denoted by $\kappa(G)$. If a graph is not connected, then we call the union of spanning trees of all connected components a *spanning forest*, see Pitsoulis, 2014². In Figure 2.13 a graph and some examples for a spanning subgraph, a forest, a tree and a spanning tree are given.

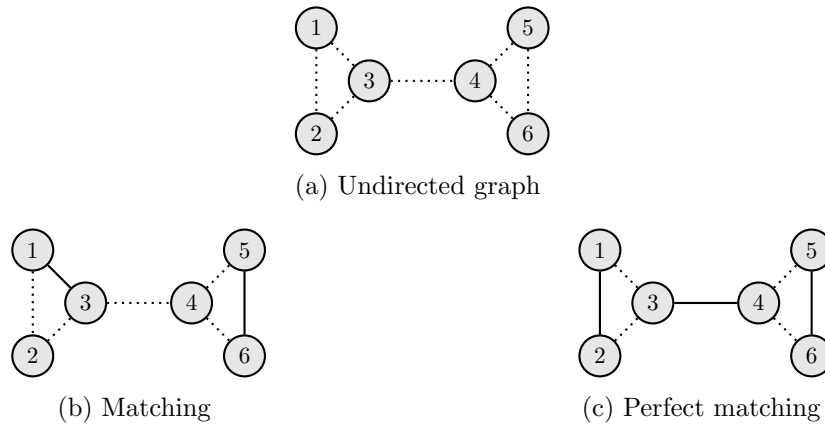


Figure 2.14.: An undirected graph and examples for a matching and a perfect matching. The edges of the matching are depicted by solid lines and all other edges of the graph are depicted by dotted lines.

Furthermore, we define a *matching* $M \subseteq E$ in a graph as a set of edges, such that no two edges in M have a common end-vertex, see Harris et al., 2008. A matching divides the set of vertices into two sets, first the set of *M -saturated* vertices which are the end-vertices of edges in M and second all other vertices, which form the set of *M -unsaturated* vertices. We call a matching *maximum matching* if it has the largest possible cardinality and we call it a *maximal matching* if it is not possible to add another edge to enlarge the matching. A matching M is called a *perfect matching*, if all vertices of the graph are M -saturated. An example for a matching and a perfect matching is given in Figure 2.14. The matching M in Figure 2.14(b) defines the set of M -saturated vertices $\{1, 3, 5, 6\}$ and the set of M -unsaturated vertices $\{2, 4\}$.

In the following, we give some examples for special types of graphs, see Harris et al., 2008. We call a graph *complete*, if there exists an edge between all pairs of nodes. A complete Graph with n vertices is denoted as K_n . Some examples are given in Figures 2.15(a), 2.15(b) and 2.15(c). An other type of graphs are the *bipartite graphs*. In this case the set of nodes V can be divided into two disjoint subsets, i.e., $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$ such that there exist only edges with one end-vertex in V_1 and the other end-vertex in V_2 , i.e., it holds for every edge $e = [v_i, v_j] \in E$ that $v_i \in V_1$ and $v_j \in V_2$. Such a graph is called *complete bipartite graph* if there exists an edge from every vertex in V_1 to all vertices in V_2 . We denote complete bipartite graphs by K_{n_1, n_2} with $n_1 = |V_1|$ and $n_2 = |V_2|$. An example of a bipartite graph is given in Figure 2.15(d), the corresponding complete bipartite graph is given in Figure 2.15(e) and an other complete bipartite graph is given in Figure 2.15(f).

²In contrast in Hamacher and Klamroth, 2006 a spanning forest is defined as a spanning subgraph without cycles, but we do not use this definition in the following.

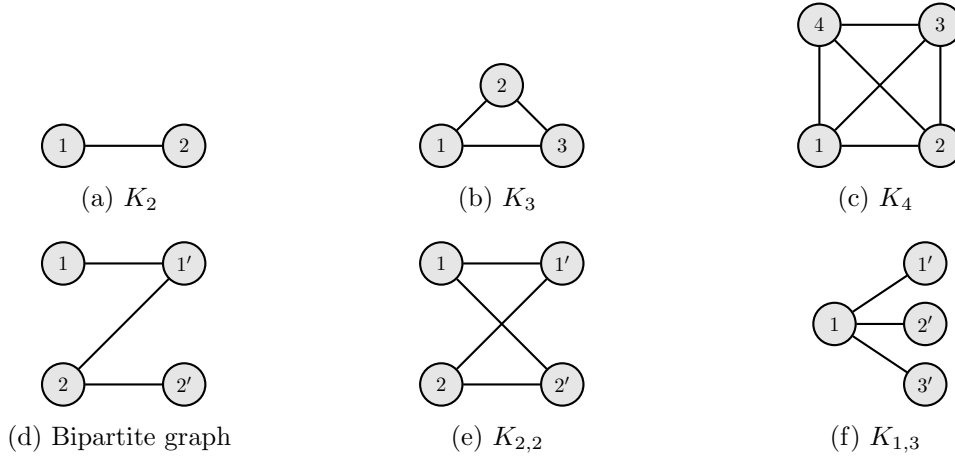


Figure 2.15.: Complete graphs and (complete) bipartite graphs.

2.5. Matroids

In the following, we summarize some basic definitions and results of matroid theory (for a self-contained introduction to matroid theory see, for example, Kung, 1986; Oxley, 2011; Schrijver, 2003; Edmonds, 1971; Schrijver, 2017). The whole section is a recombination of parts from the articles Gorski et al., 2022 and Klamroth et al., 2022a.

Let $E = \{e_1, \dots, e_n\}$ be a finite *ground set* with $n \in \mathbb{Z}_{>}$ elements and let \mathcal{I} be a subset of the power set 2^E of E . The ordered pair $\mathcal{M} = (E, \mathcal{I})$ is called a *matroid* if the following three conditions are satisfied:

$$\emptyset \in \mathcal{I} \tag{M1}$$

$$(I \in \mathcal{I} \wedge I' \subseteq I) \Rightarrow I' \in \mathcal{I} \tag{M2}$$

$$\forall I_1, I_2 \in \mathcal{I} \text{ with } |I_1| < |I_2| \exists e \in I_2 \setminus I_1 : I_1 \cup \{e\} \in \mathcal{I}. \tag{M3}$$

Moreover, if the tuple (E, \mathcal{I}) satisfies the conditions (M1) and (M2), then it is called an *independence system*. If $\mathcal{M} = (E, \mathcal{I})$ is a matroid or an independence system, then all sets $I \in \mathcal{I}$ are called *independent sets*. Conversely, a subset of E is called *dependent* if it is not contained in \mathcal{I} .

An independent set $I \in \mathcal{I}$ is called *maximal* if $I \cup \{e\} \notin \mathcal{I}$ for all $e \in E \setminus I$. Similarly, a dependent set $D \in 2^E \setminus \mathcal{I}$ is called *minimal* if $D \setminus \{e\} \in \mathcal{I}$ for all $e \in D$. Maximal independent sets are called *bases* of the matroid, and minimal dependent sets are called *circuits* of the matroid. To simplify the discussion, we use the following notation for set operations throughout this thesis: Let S denote a subset of a finite ground set E and let $e, f \in E$. We write $S + e$ to denote the set $S \cup \{e\}$ and $S - f$ to denote the set $S \setminus \{f\}$. Furthermore, let $S^c := E \setminus S$ denote the complement of S in E . To further simplify the notation we assume that set operations are executed from left to right.

In the following, we write $\mathcal{X} := \{\mathcal{B} \in \mathcal{I} : (\nexists I \in \mathcal{I} : I \supsetneq \mathcal{B})\}$ for the set of all bases of a matroid. All bases of a matroid have the same cardinality which is referred to as the *rank* of the matroid, see, for example, Oxley, 2011. Note, that this property does not hold in general for independence systems.

Given a matroid $\mathcal{M} = (E, \mathcal{I})$, a basis $\mathcal{B} \in \mathcal{X}$, and an element $e \in E \setminus \mathcal{B}$, then $\mathcal{B} \cup \{e\}$ contains a uniquely determined circuit $C(e, \mathcal{B})$ containing e . This circuit is also called the *fundamental circuit* of e w.r.t. \mathcal{B} . An important property of matroids is the *basis exchange property*:

$$\forall \mathcal{B}_1, \mathcal{B}_2 \in \mathcal{X} \quad \forall b_1 \in \mathcal{B}_1 \setminus \mathcal{B}_2 \quad \exists b_2 \in \mathcal{B}_2 \setminus \mathcal{B}_1 : (\mathcal{B}_1 \cup \{b_2\}) \setminus \{b_1\} \in \mathcal{X}. \quad (\text{B})$$

The following stronger version of the basis exchange property was proven in Brualdi, 1969.

Lemma 2.7 (Brualdi, 1969). *Let $\mathcal{B}_1, \mathcal{B}_2 \in \mathcal{X}$. For all $b_1 \in \mathcal{B}_1 \setminus \mathcal{B}_2$ there exists $b_2 \in \mathcal{B}_2 \setminus \mathcal{B}_1$ such that both $(\mathcal{B}_1 \cup \{b_2\}) \setminus \{b_1\}$ and $(\mathcal{B}_2 \setminus \{b_2\}) \cup \{b_1\}$ are bases in \mathcal{X} .*

In this context, two bases of a matroid are called *adjacent* if they have $r - 1$ elements in common, assuming that the matroid is of rank r . According to the basis exchange property (B) and Lemma 2.7, a given basis can be transformed into an adjacent basis by exactly one basis exchange. We refer to this as a *swap operation* in Chapter 3.

If a subset $S \subseteq E$ of the ground set E is deleted from E , we obtain the *restriction* of \mathcal{M} to $E \setminus S$. The ground set of this matroid is the set $E \setminus S$ and its independent sets are those independent sets of \mathcal{M} that are completely contained in $E \setminus S$, i.e., that do not contain any elements from S . We write $\mathcal{M} - S$ for short. Note that the independent sets of $\mathcal{M} - S$ correspond to the projection of all independent sets of \mathcal{M} to $E \setminus S$ due to property (M2).

Moreover, if an independent set I of \mathcal{M} is *contracted* we obtain the *contraction* of \mathcal{M} to I denoted by \mathcal{M}/I . The ground set of \mathcal{M}/I is given by $E \setminus I$, and its independent sets are the sets $I' \subseteq (E \setminus I)$ such that $I' \cup I$ is an independent set of \mathcal{M} .

A visualization for restriction and contraction can be found in Example 2.8, which is given after the introduction of some typical matroids.

For an extensive list of examples of matroids we refer to Oxley, 2011. The following matroids are frequently considered and are used for illustrations and numerical tests in this thesis.

- **graphic matroid:** Let $G = (V, E)$ be an undirected graph with finitely many edges and vertices, then $\mathcal{M}(G) = (E, \mathcal{I})$ with $\mathcal{I} := \{I \subseteq E : (V, I) \text{ contains no circuit}\}$ is a matroid. The independent sets of $\mathcal{M}(G)$ are all forests in G , and the bases of $\mathcal{M}(G)$ are all spanning forests, if we use the definition of spanning forests from Pitsoulis, 2014. If G is connected, then \mathcal{X} is the set of all spanning trees of G . In context of graphic matroids we use the terms circuits and cycles equivalently.
- **uniform matroid:** Let E be a finite set and $r \in \mathbb{Z}_{\geq}$, then $\mathcal{M} = (E, \mathcal{I})$ with $\mathcal{I} := \{I \subseteq E : |I| \leq r\}$ is a matroid with rank r , denoted by $\mathcal{U}_{r,n}$. The independent sets of $\mathcal{U}_{r,n}$ are all subsets of $E = \{e_1, \dots, e_n\}$ that have at most r elements, and the bases of $\mathcal{U}_{r,n}$ are all subsets of the set of edges E that have exactly r elements. A subset of E is a circuit of $\mathcal{U}_{r,n}$ if it contains exactly $r + 1$ elements of E .
- **partition matroid:** Let $E = E_1 \cup E_2 \cup \dots \cup E_p$ be the disjoint union of p finite sets and let $u_1, \dots, u_p \geq 0$ be non-negative integers. Then $\mathcal{M} = (E, \mathcal{I})$ with the set $\mathcal{I} := \{I \subseteq E : |I \cap E_i| \leq u_i \forall 1 \leq i \leq p\}$ is a matroid. Note that the uniform matroid is a special case of the partition matroid with $p = 1$ and $u_1 = r$.

- **matching matroid:** Let $G = (V, E)$ be a finite undirected graph. The ground set F of the matching matroid $\mathcal{M} = (F, \mathcal{I})$ is a subset of the vertices of G , i.e., $F \subseteq V$, and all subsets of F that can be covered by a matching of G are independent. Note, that in this case the ground set of the matroid \mathcal{M} is not equal to the set of edges E of the graph. Hence, we use in this case F to denote the ground set of the matroid.
- **transversal matroid:** A special case of the matching matroid is the *transversal matroid*, that is a matching matroid on a bipartite graph $G = (V_1 \cup V_2, E)$ with bipartition $V = V_1 \cup V_2$, where the ground set F of the matroid equals V_1 or V_2 .

Example 2.8. Now, we can visualize the concepts of restriction and contraction at the graphic matroid \mathcal{M} given in Figure 2.16(a). Let $S = \{[1, 2], [1, 3], [2, 3], [4, 5]\}$. The resulting restriction $\mathcal{M} - S$ is given in Figure 2.16(b). In this case, the set $\{[4, 6], [5, 6]\}$ is an independent set of $\mathcal{M} - S$. Now, we consider the independent set $I = \{[1, 2], [1, 3], [4, 5]\}$ to get the contracted matroid \mathcal{M}/I , which is given in Figure 2.16(c). Nodes 1, 2 and 3 of \mathcal{M} are contracted into node 3' and nodes 4 and 5 of \mathcal{M} are contracted into node 4'. In this case, the set $\{[4, 6], [5, 6]\}$ is **not** an independent set of \mathcal{M}/I as $\{[4, 6], [5, 6]\} \cup I$ is a dependent set of \mathcal{M} . Note, that the contraction of a graphic matroid may be a multigraph with loops.

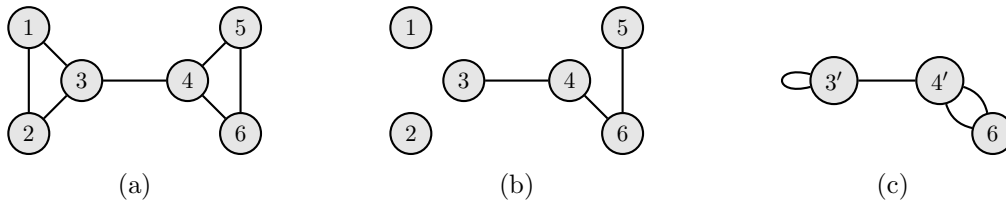


Figure 2.16.: Consider the graphic matroid \mathcal{M} on the graph given in (a). The restriction $\mathcal{M} - \{[1, 2], [1, 3], [2, 3], [4, 5]\}$ and the contraction $\mathcal{M}/\{[1, 2], [1, 3], [4, 5]\}$ are given in (b) and (c), respectively.

An important concept that is very useful in Chapter 4 is the intersection of two matroids. Consider two matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$ over the same ground set E . Then the *matroid intersection* of \mathcal{M}_1 and \mathcal{M}_2 is defined as the independence system $\mathcal{M}_1 \cap \mathcal{M}_2 := (E, \mathcal{I}_1 \cap \mathcal{I}_2)$.

It is important to note that a matroid intersection is not necessarily a matroid itself. A counter example is given in Figure 2.17. Let $G = (V, E_1 \cup E_2)$ be a graph with an edge set $E = E_1 \cup E_2$ that is partitioned into two subsets E_1, E_2 (e.g., green-dotted and red-solid edges, respectively). Moreover, let $\mathcal{M}_1 = (E, \mathcal{I}_1)$ be the graphic matroid on G and let $\mathcal{M}_2 = (E, \mathcal{I}_2)$ be a partition matroid with $E_1 := \{[1, 2], [4, 5], [4, 6]\}$ the set of all green-dotted edges, $E_2 := \{[1, 3], [2, 3], [3, 4], [5, 6]\}$ the set of all red-solid edges and $\mathcal{I}_2 := \{I \subseteq E : |I \cap E_1| \leq 3, |I \cap E_2| \leq 2\}$. Then the set $I := \{[1, 3], [2, 3], [4, 5], [4, 6]\}$ is an inclusion-wise maximal independent set of $\mathcal{M}_1 \cap \mathcal{M}_2$ since every additional edge $e \in E \setminus I$ makes $I \cup \{e\}$ dependent w.r.t. either \mathcal{M}_1 or \mathcal{M}_2 . However, the maximal independent set $J := \{[1, 2], [1, 3], [3, 4], [4, 5], [4, 6]\}$ of $\mathcal{M}_1 \cap \mathcal{M}_2$ has larger cardinality, i.e., $|J| > |I|$. Thus, $\mathcal{M}_1 \cap \mathcal{M}_2$ is not a matroid because all maximal independent sets of a matroid must

have the same cardinality. However, if at least one of the intersected matroids is a uniform matroid then the intersection is again a matroid, see Oxley, 2011.

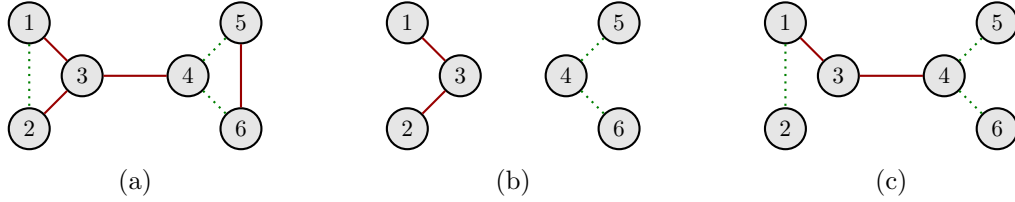


Figure 2.17.: The edges $E = E_1 \cup E_2$ of the graph $G = (V, E)$ in (a) are the ground set of a graphic matroid \mathcal{M}_1 and of a partition matroid \mathcal{M}_2 (where E_1 is the set of all green-dotted edges and E_2 is the set of all red-solid edges). The independent set of edges I illustrated in (b) is inclusion-wise maximal for $\mathcal{M}_1 \cap \mathcal{M}_2$, however, the alternative independent set J shown in (c) has larger cardinality.

2.6. Single-objective Matroid Optimization and Multi-objective Minimum Spanning Tree Problem

A specific type of combinatorial optimization problems are matroid optimization problems. In this case, the feasible set is the set of all bases of a matroid \mathcal{M} denoted by \mathcal{X} , see Section 2.5. Hence, a *matroid optimization problem* (MaO) with p sum objective functions can be written as

$$\begin{aligned} \min_{\mathcal{C}_P} \quad & w(\mathcal{B}) = (w_1(\mathcal{B}), \dots, w_p(\mathcal{B})) \\ \text{s. t.} \quad & \mathcal{B} \in \mathcal{X}. \end{aligned} \tag{MaO}$$

Note that we write in the special case of matroid optimization problems \mathcal{X} instead of X for the set of the feasible solutions and \mathcal{Y} instead of Y for the set of feasible points in the outcome space. Similarly, we adapt the notation of the (weak) efficient and (weak) non-dominated sets. In general, all matroid optimization problems are feasible, since the basis is only empty if the set of independent sets consist only of the empty set, i.e., $\mathcal{I} \neq \{\emptyset\}$.

One specific property of matroids is the basis exchange property, see Lemma 2.7, which allows us to show connectedness of the efficient solutions of bi-objective matroid optimization problems with ordinal costs, see Chapter 3. This connectedness is used in the Efficient Swap Algorithm presented in Chapter 3, which solves the problem in polynomial time. To define connectedness of the efficient set, we recall that two bases are called adjacent, if they have $r - 1$ elements in common for a matroid of rank r . Now, we define the *adjacency graph* $G = (V, E)$ of efficient bases of problem (MaO), see Gorski et al., 2011. The node set V of the adjacency graph consists of all efficient bases of (MaO). An (undirected) edge is introduced between all pairs of vertices corresponding to adjacent bases of the underlying problem. These edges form the set E . The set \mathcal{X}_E is said to be connected if its corresponding adjacency graph G is connected, i.e., if every pair of vertices in V is connected by a path. As shown in Gorski, 2010, the adjacency graph G is not connected

Algorithm 1: Greedy Algorithm for Single-objective Matroid Optimization Problems with a Sum Objective Function (see, e.g., Pitsoulis, 2014)

Input: A matroid $\mathcal{M} = (E, \mathcal{I})$ and a function $w : E \rightarrow \mathbb{R}$.

Output: Optimal basis $\mathcal{B} \in \mathcal{X}$.

```

1 Sort  $E$  such that  $w(e_1) \leq w(e_2) \leq \dots \leq w(e_{|E|})$ .
2  $X := \emptyset$ .
3 for  $i = 1, \dots, |E|$  do
4   if  $X \cup \{e_i\} \in \mathcal{I}$  then
5      $X := X \cup \{e_i\}$ .
6 return  $\mathcal{B} := X$ .
```

in general, even if it is extended to include weakly efficient bases. Nevertheless, the adjacency graph always contains a connected component given by the supported efficient bases of (MaO), see Ehrgott, 1996. Although the adjacency graph is not connected in general, many solution methods make use of the adjacency structure of matroids or of the adjacency structure of more general combinatorial optimization problems. Some examples for such solution strategies are described in Loera et al., 2010.

Single-objective matroid optimization problems, i.e., problems (MaO) with $p = 1$, with, for example, a sum objective function can be solved efficiently by the greedy algorithm, see Pitsoulis, 2014. The idea of the greedy algorithm is to sort the elements of the ground set E of a matroid by its weights in non-decreasing order and to start with the empty set, i.e., $X := \emptyset$. Then the next element of the sorted list is added to the set X , if the set X does not become dependent by the addition of this element. This is repeated until a basis of the matroid is generated. The greedy algorithm for minimization matroid problems is given in Algorithm 1. An example of this algorithm applied to an optimization problem on graphic matroids is given in Example 2.9. In this case the algorithm is the well-known Kruskal-Algorithm. The correctness of the greedy algorithm follows immediately from condition (M2) of matroids and line 4 of the algorithm. The runtime of the algorithm depends on the type of matroid, as the computationally expensive part is in most cases in line 4, where we have to check whether a set is dependent or independent, see Pitsoulis, 2014. In some cases, like, e.g., the matroid optimization problem on graphic matroids, the check whether a set is independent or not does not need much time and hence, the computationally expensive part is in this case the sorting of the elements in line 1.

The specific matroid optimization problem defined on graphic matroids is the *minimum spanning tree problem* (MST). The bases of graphic matroids are spanning trees and the minimum spanning tree problem is looking for the spanning tree with smallest costs, i.e., we consider a single-objective matroid optimization problem with a sum objective function. Hence, this problem can be solved by the greedy algorithm, which is known as algorithm of Kruskal for the minimum spanning tree problem.

Example 2.9. We consider the graph given in Figure 2.18(a) and identify the edges by their weights. Then in line 1 of Algorithm 1 we get the sorted edge list 1, 2, 3, 4, 5, 6, 10 and in line 2 we initialize $X = \emptyset$. The following steps are visualized in the Figures 2.18(b)

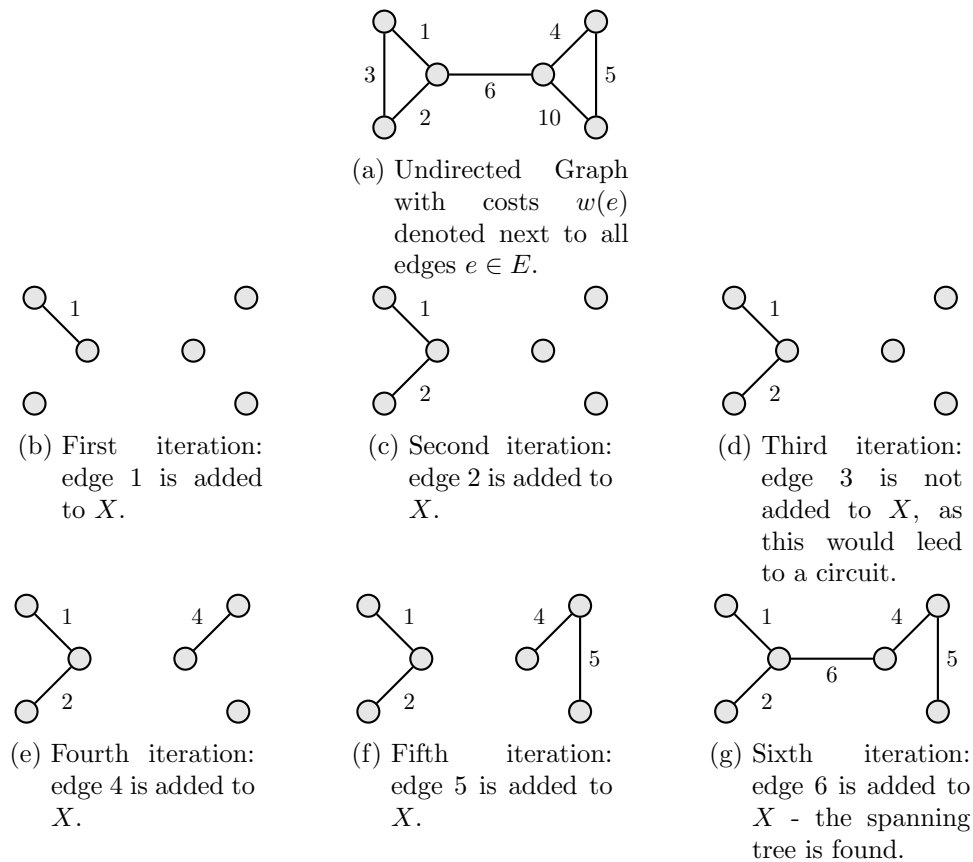


Figure 2.18.: Visualization of the algorithm of Kruskal (Algorithm 1 applied to a graphic matroid).

to 2.18(g). In the first two iterations of the for-loop the edges 1 and 2 are added to X , because $\{1,2\}$ does not contain a circuit. In the third iteration edge 3 is not added to X as this would lead to a circuit. In the following the edges 4, 5 and 6 are added. Then we have found a spanning tree and the algorithm could be stopped. Therefore, we could add to the algorithm the test whether $|X| = n - 1$. If this holds, we can stop the algorithm, otherwise we need to consider the next edge from the for loop. If we omit this additional test, then the algorithm would test all remaining edges, whether they can be added to X without generating a circuit. In our example, the edge 10 would be tested and then the algorithm of Kruskal returns the spanning tree $\{1,2,4,5,6\}$.

For the minimum spanning tree problem exist also other strategies to solve it, like, e.g., the algorithm of Sollin or the algorithm of Prim, see, for example, Ahuja et al., 1993. As we do not need those algorithms in the following we do not investigate them any further.

If we consider multi-objective spanning tree problems, we can not apply the greedy algorithm any more as we can not sort the edges in non-decreasing order with respect to the Pareto order. The greedy algorithm can only be applied if the underlying order is a total order, like, e.g., the lexicographic order. Nevertheless, the multi-objective

spanning tree problem with respect to the Pareto order has been investigated and there exist different solution approaches. For example, Hamacher and Ruhe, 1994 suggest a two phase algorithm, which computes first the extreme supported points and afterwards applies neighborhood search to determine a sequence non-supported outcome vectors such that the distance between two consecutive outcome vectors is bounded by a given accuracy. Some solutions strategies are designed to find only supported efficient solutions by using the connectedness of the solutions and a weight space decomposition, see Correia et al., 2021. Furthermore, see, for example, Ruzika and Hamacher, 2009 for a survey concerning solution strategies for multi-objective spanning tree problems and Benabbou and Perny, 2015 for a more recent reference on this topic. Evolutionary methods for multi-objective spanning tree problems were suggested, among others, in Zhou and Gen, 1999, Knowles and Corne, 2001, Neumann and Witt, 2010 and Bossek et al., 2019 as well as references therein. Loera et al., 2010 describe heuristic approaches to general multi-objective matroid optimization problems that rely on adjacency relations and nonlinear scalarizations. The methods are implemented in the MOCHA software package Loera et al., 2009. Approximation schemes were suggested, for example, in Grandoni et al., 2014 and Bazgan et al., 2019.

From a theoretical perspective exhaustive search is also a possible, however very time consuming, approach to solve multi-objective spanning trees problems. Exhaustive search generates all possible spanning trees, evaluates their objective function vector and filters them to obtain the non-dominated set. Such an strategy is, for example, presented in Shioura et al., 1997. In this thesis we use an algorithm, which is presented in Knuth, 2012, to compute all spanning trees of a graph G . The algorithm is based on very early results by Feussner, 1902 and we explain it in the following. We assume that G has n vertices and is a connected graph, because otherwise there do not exist any spanning trees. Furthermore, we assume w.l.o.g. that the graph does not contain any loops, because loops can not be part of spanning trees. Feussner's idea is to use a recursion to find all spanning trees. Let e be an edge of G then either e is contained in a spanning tree or it is not. All spanning trees that contain e can be computed by calculating all spanning trees of the contracted graph G/e , which contain $n - 2$ edges, and add to them the edge e . Illustrative the contracted graph G/e is obtained by shrinking the edge e to one vertex. To compute all spanning trees without e it is sufficient to compute all spanning trees of the smaller graph $G \setminus e$. We denote by $\text{ST}(G)$ the set of spanning trees in G . Then the recursion is given by

$$\text{ST}(G) = (\{e\} \sqcup \text{ST}(G/e)) \cup \text{ST}(G \setminus e). \quad (2.12)$$

Here $\{e\} \sqcup \text{ST}(G/e)$ denotes that we add to every tree $T = \{e_1, \dots, e_{n-2}\} \in \text{ST}(G/e)$ the edge e , i.e., $\{e\} \sqcup \text{ST}(G/e) := \{T \cup \{e\} : T \in \text{ST}(G/e)\}$.

Knuth, 2012 refers to Malcolm J. Smith, who introduced in his Master's thesis at the University of Victoria from 1997 a strategy to compute the recursion (2.12), which we explain in the following and which is given in Algorithm 2. To simplify we slightly abuse the notation and consider $\text{ST}(G)$ to be a sorted list, referring by $\text{ST}(G)[\text{end}]$ to the last element of this list. We assume that new elements of this list get added to the end of the list. The key point of the strategy of Smith is, that every new spanning tree is computed from its predecessor by removing one edge and replacing it by another in an efficient way.

Algorithm 2: Generation of all Spanning Trees $STG(G, T_{\text{near}})$

Input: A connected graph $G = (V, E)$ without loops and a corresponding near tree $T_{\text{near}} = \{e_1, \dots, e_{n-2}\}$, $n = |V|$.

Output: The set of spanning trees $ST(G)$ in G

```

1  $ST(G) = \emptyset$ ;
2 if  $n = 2$  then
3   | return  $ST(G) = E$ 
4 else
5   | Call  $STG(G/e_1, \{e_2, \dots, e_{n-2}\})$ .
6   | foreach  $T \in STG(G/e_1, \{e_2, \dots, e_{n-2}\})$  do
7     |  $ST(G)[\text{end} + 1] = T \cup \{e_1\}$ .
8   | if  $e_1$  is not a bridge then
9     | Call  $STG(G \setminus e_1, ST(G/e_1)[\text{end}])$ .

```

Smith's Algorithm starts to compute $ST(G)$ with a spanning tree, that includes a given *near tree*, i.e., a set of $n - 2$ edges without circuits. For $n = 2$ we only have to list all edges between those two nodes to get all spanning trees.

Now, we consider $n > 2$ and a given near tree $\{e_1, \dots, e_{n-2}\}$. First we compute G/e_1 and all of its spanning trees. If we add to those the edge e_1 , spanning trees of G are the result. We compute the spanning trees of G/e_1 by recursion with the near tree $\{e_2, \dots, e_{n-2}\}$.

To compute the spanning trees of $G \setminus e_1$ we use the last spanning tree found for G/e_1 , let's denote it by $\{\hat{e}_1, \dots, \hat{e}_{n-2}\}$, as near tree, which should be contained in the first spanning tree constructed for $G \setminus e_1$. If e_1 is a bridge, i.e., the graph G would become disconnected by removing e_1 , then we do not investigate $G \setminus e_1$ any further. For more details regarding the implementation of this algorithm see Knuth, 2012. Therein is also explained how the algorithm can be improved even further. The algorithm without further improvements can be downloaded from Hotz, 2016.

In the following, we give an example for this algorithm, which is similar to the example in Knuth, 2012.

Example 2.10. Consider the graph at the top of Figure 2.19, which consists of $n = 4$ vertices and 4 edges, which we identify by their number. We start with the near tree $\{1, 2\}$. Then we consider in the recursion first the graph $G' := G/\{1\}$, visualized as the left child of G in Figure 2.19. This graph consists of $n = 3$ nodes, and hence we need to do the recursion once again to compute its spanning trees. Therefore, we compute $G'/\{2\}$, i.e., we consider the contraction over the second edge of the near tree from the beginning. This graph has only two nodes and hence its spanning trees are all edges between those nodes, i.e., the spanning trees of $G'/\{2\}$ are $\{3\}$ and $\{4\}$. To get the spanning trees of G' and G we have to add the edges $\{1\}$ and $\{1, 2\}$, respectively, which were used for the contraction of the graphs. Therefore, we get the spanning trees $\{2, 3\}$ and $\{2, 4\}$ of G' as well as $\{1, 2, 3\}$ and $\{1, 2, 4\}$ of G . Now, we consider the graph $G'' := G' \setminus \{2\}$ to complete the computation of all spanning trees of G' . The corresponding graph can be found in Figure 2.19 as the right child of G' . To compute all its spanning trees, we need a near tree, which is given by

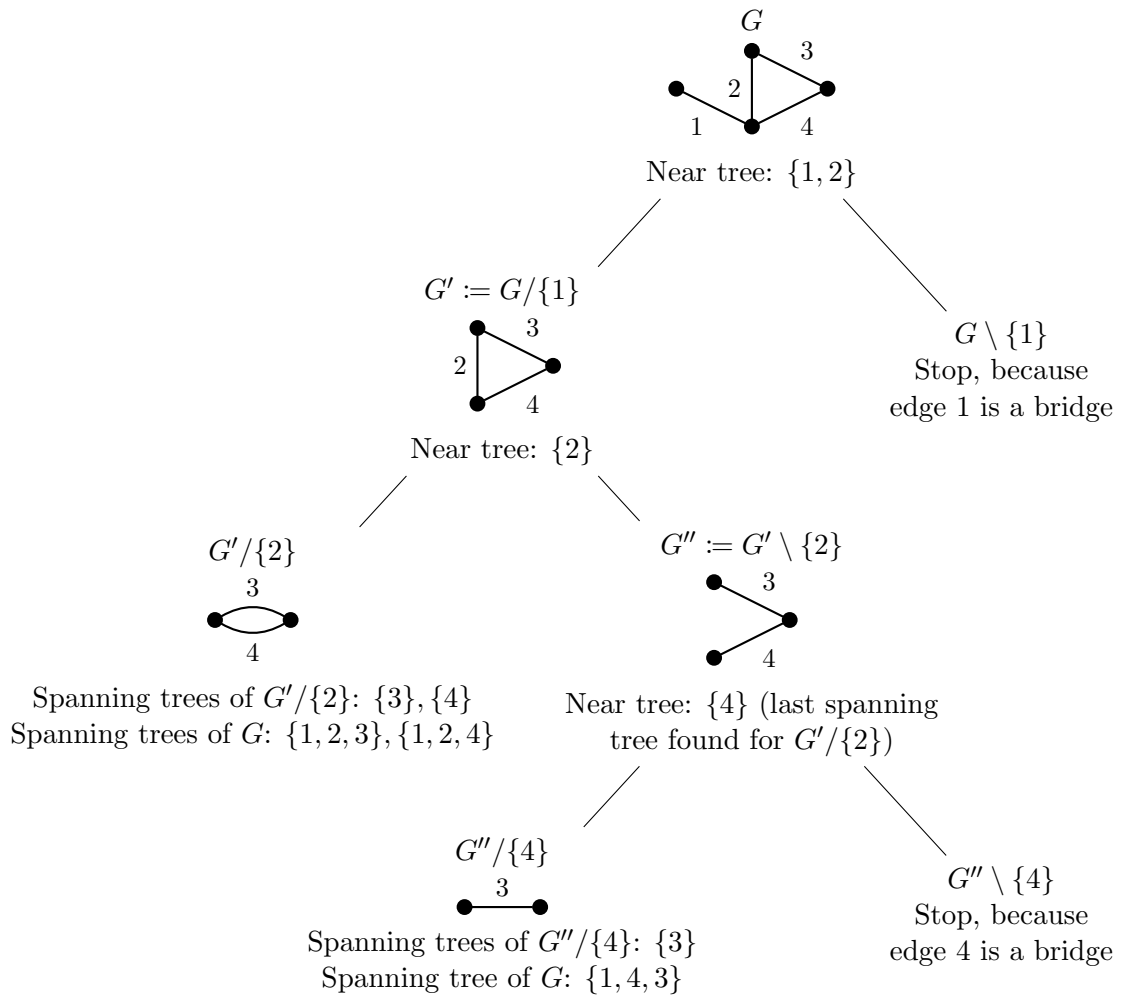


Figure 2.19.: Visualization of the recursion ST.

the last spanning tree found for $G'/\{2\}$, i.e., the near tree $\{4\}$. We apply again a recursion and compute $G''/\{4\}$ and as the resulting graph has only two nodes we can easily identify all of its spanning trees, which is here the edge 3. Hence we found $\{3, 4\}$ as spanning tree of G' and G'' as well as $\{1, 4, 3\}$ as spanning tree of G . It is left to compute $G'' \setminus \{4\}$, but the algorithm would recognize that the edge 4 is a bridge, and hence the algorithm stops the recursion here. As a result, we know that we have found all spanning trees of G'' and G' . Therefore, we go back to the graph G and have to investigate $G \setminus \{1\}$, but the edge 1 is again a bridge, hence the algorithm stops and returns all spanning trees of G , namely $\{1, 2, 3\}$, $\{1, 2, 4\}$ and $\{1, 3, 4\}$. We see, that the spanning trees are generated in such a way that we always have only swapped one element in a tree against another element to get a new spanning tree, which is the advantage of the strategy of Malcolm J. Smith.

The computation of all possible spanning trees is costly for large graphs, because the number of spanning trees grows exponentially in n , e.g., for complete graphs with n nodes. The exact number of spanning trees of a graph G , i.e., $\kappa(G)$, can be computed easily by Kirchhoff's Matrix-Tree-Theorem, see, e.g., the textbook Stanley, 2013.

Theorem 2.11 (Kirchhoff's Matrix-Tree Theorem, see, e.g., Stanley, 2013). *Let G be a finite connected graph without loops and with Laplacian matrix $M^L(G)$. We denote by $M_i^L(G)$ the matrix that results from matrix $M^L(G)$ by removing the i -th row and column, $i \in \{1, \dots, n\}$. Then the determinant of $M_i^L(G)$ equals the complexity $\kappa(G)$ for all $i \in \{1, \dots, n\}$.*

The complexity $\kappa(G)$ grows exponentially, in particular $\kappa(G) \in \mathcal{O}(n^n)$, and it can be computed by the determinant of $M_i^L(G)$, which results from the Laplacian matrix $M^L(G)$ by removing the i -th row and column for any $i \in \{1, \dots, n\}$.

2.7. Single-objective Shortest Path Problem

In this section we consider single-objective shortest path problems, see, for example, Ahuja et al., 1993. We assume that a directed, connected graph $G = (V, E)$ is given together with a weight function $w : E \rightarrow \mathbb{R}$. There exist different types of this problem, depending whether we are interested in finding a path with minimal total weight from a given start node s to an other node t or if we are interested in finding minimal paths from s to all other nodes or if we are even interested in finding shortest paths from all nodes to all other nodes. Furthermore, there exist different algorithms for non-negative weights and general weights. In the first case label-setting algorithms can be applied, which iteratively assign distance labels to nodes and in each iteration one label is fixed. Label-setting algorithms can also be applied, if there exist edges with negative weights, but there does not exist a dicycle. In the case, that there exist no dicycle the nodes of the graph can be ordered in linear time such that $i < j$ for every $(i, j) \in E$. This is called a topological ordering, which is used to solve the shortest path problem on graphs without dicycles. If all edges have non-negative weight, then the algorithm of Dijkstra can be applied. It is a typical label-setting algorithm and finds all shortest paths from a node s to all other nodes in the graph. To solve shortest path problems on graphs with dicycles and negative weights, there also exist label-correcting algorithms that do not fix any labels until the last iteration.

Algorithm 3: Dijkstra's Algorithm

Input: A digraph $G = (V; E)$, costs $w(e) \geq 0$ for all $e \in E$, start node $s \in V$.

Output: Shortest paths from start node s to all other nodes $V \setminus \{s\}$.

```

1  $L := \emptyset, \bar{L} := V.$ 
2  $d(i) := \infty$  for all  $i \in V \setminus \{s\}$  and  $d(s) := 0.$ 
3  $\text{pred}(i) := 0$  for all  $i \in V.$ 
4 while  $|L| < n$  do
5    $i' := \operatorname{argmin}_{i \in \bar{L}} d(i).$ 
6    $L := L \cup \{i'\}, \bar{L} := \bar{L} \setminus \{i'\}.$ 
7   foreach  $j \in \bar{L} : (i', j) \in E$  do
8     if  $d_j > d_{i'} + w((i', j))$  then
9        $d_j := d_{i'} + w((i', j)).$ 
10       $\text{pred}(j) := i'.$ 
11 return  $d_i, \text{pred}(i)$  for all  $i \in V.$ 

```

In the following, we introduce the algorithm of Floyd-Warshall, which finds all shortest paths between all nodes in a digraph with negative weights or returns a negative dicycle if it exists. There exist different implementation strategies of the algorithms we present here based on the data structures to store the current labels of the edges. Details on the different data structures as well as more label-setting algorithms and label-correcting algorithms are, for example, given in Ahuja et al., 1993.

First, we assume that a connected digraph $G = (V, E)$ with a non-negative weight function $w : E \rightarrow \mathbb{Z}_{\geq}$ and a start node s is given. Furthermore, we assume that there exists a path from s to all other nodes. Then Dijkstra's algorithm (Algorithm 3) computes all shortest paths from start node s to all other nodes. In each iteration we update

- a set L of permanently labeled vertices, a set \bar{L} of all other vertices,
- the actual distance $d(i)$ for all nodes $i \in V$ and
- the predecessor $\text{pred}(i)$ for every node $i \in V$.

The predecessor labels $\text{pred}(i)$, $i \in V$, are used for the reconstruction of the shortest paths. In the initialization step, we set $L := \emptyset$, $\bar{L} := V$, $d(i) := \infty$ for all $i \in V \setminus \{s\}$, $d(s) := 0$ and $\text{pred}(s) := 0$. As long as not all nodes are labeled permanently, we choose a not permanently labeled node i' with smallest distance $d(i')$ and mark this node as permanently labeled. Then we update the distance labels of all not permanently labeled nodes j , if there exist an edge $(i', j) \in E$ and if the costs $d(i') + w((i', j))$ are smaller than the current distance of node j . In this case we also update the predecessor label of node j and set it to $\text{pred}(j) := i'$. For a proof of the correctness of the algorithm see, for example, Ahuja et al., 1993.

In Figure 2.20 we illustrate Dijkstra's algorithm using a small example. Next to each edge e the weight $w(e)$ is given, and next to each node we denoted first the actual distance with the update and second the actual predecessor with the update. Grey nodes are the permanently labeled nodes, which are in set L , and all other nodes are colored in white.

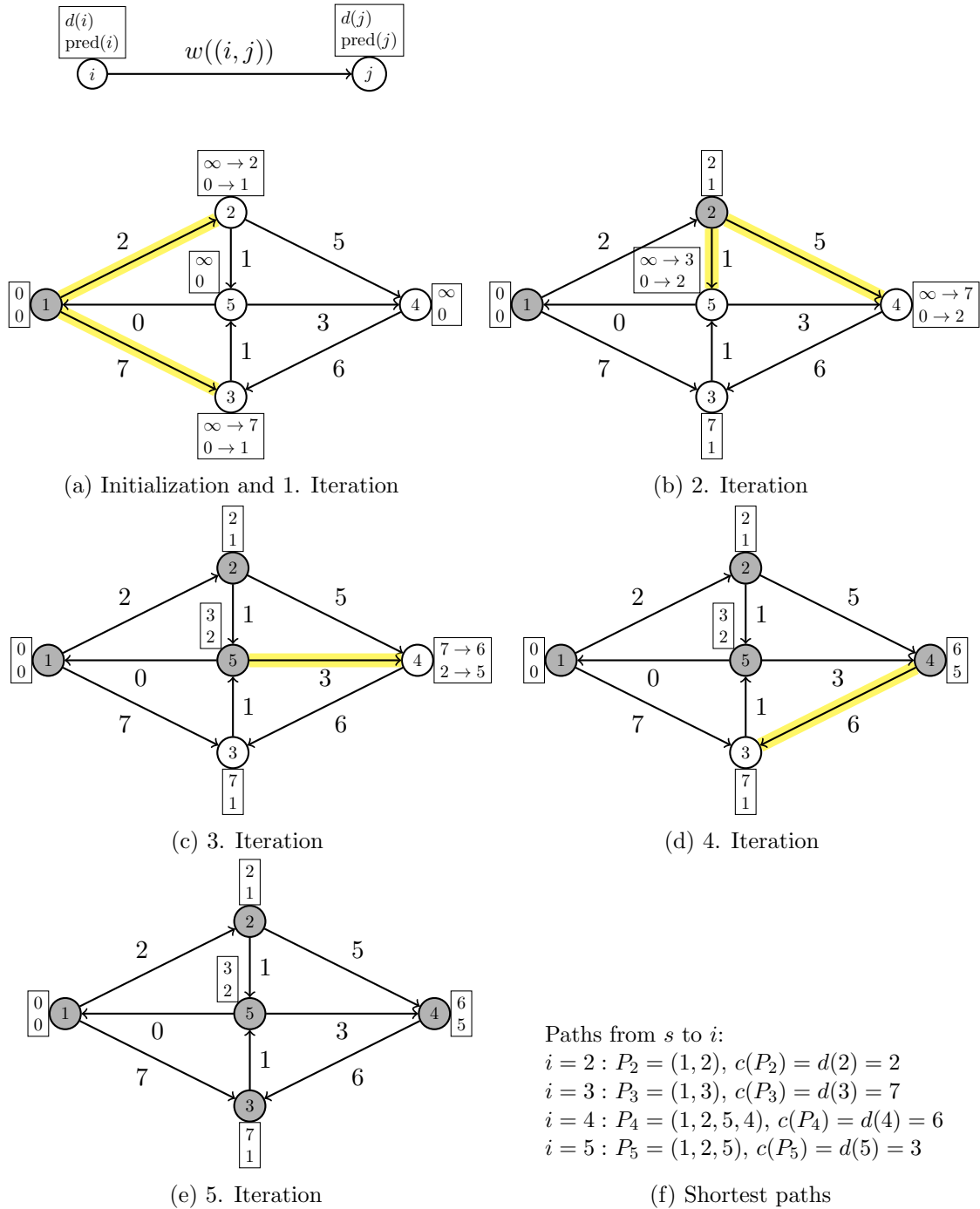


Figure 2.20.: An illustration of Dijkstra's algorithm.

Now, we explain the algorithm of Floyd-Warshall (Algorithm 4) as an example for a label-correcting algorithm. It computes shortest paths between all pairs of nodes or detects a negative dicycle. We assume that a connected digraph $G = (V, E)$ with $|V| = n$ and an arbitrary weight function $w : E \rightarrow \mathbb{Z}$ is given. The algorithm updates iteratively a matrix

Algorithm 4: Algorithm of Floyd-Warshall**Input:** digraph $G = (V, E)$, edge weights $w(e)$ for all $e \in E$ ($n = |V|$, $m = |E|$)**Output:** shortest path for all pairs of nodes

```

1 foreach  $(i, j) \in V \times V$  do
2    $d_{ij} := \begin{cases} w((i, j)) & \text{if } (i, j) \in E \\ 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}, \quad \text{pred}_{ij} := \begin{cases} i & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$ 
3 for  $k := 1, \dots, n$  do
4   for  $i := 1, \dots, n$  do
5     for  $j := 1, \dots, n$  do
6       if  $d_{ij} > d_{ik} + d_{kj}$  then
7          $d_{ij} := d_{ik} + d_{kj}$ 
8          $\text{pred}_{ij} := \text{pred}_{kj}$ 
9       if  $i = j$  and  $d_{ii} < 0$  then
10        return  $G = (V, E)$  contains a negative dicycle
11 return distance matrix  $d_{ij}$  and predecessor labels  $\text{pred}_{ij} \forall i, j \in V$ 

```

$D = (d_{ij})_{i,j=1,\dots,n}$ with all pairwise distances and a matrix $\text{Pred} = (\text{pred}_{ij})_{i,j=1,\dots,n}$, which saves the predecessors of node j of the path from i to j . The matrix D gets initialized with $d_{ij} = w((i, j))$ if $(i, j) \in E$, $d_{ij} = 0$ if $i = j$ and $d_{ij} = \infty$ if $i \neq j$ and $(i, j) \notin E$. The matrix Pred gets initialized with $\text{pred}_{ij} = i$ if $(i, j) \in E$ and $\text{pred}_{ij} = 0$ if $(i, j) \notin E$. Then we consider iteratively the k th row and column of D , $k = 1, \dots, n$. If $d_{ij} > d_{ik} + d_{kj}$ for $i, j = 1, \dots, n$ (which can only happen for $i, j \neq k$), we update the distance of the path from i to j , i.e., $d_{ij} := d_{ik} + d_{kj}$, and the predecessor of j in the corresponding path, i.e., $\text{pred}_{ij} := \text{pred}_{kj}$. This update implies that it is shorter to use the path from i to j passing node k , instead of taking the direct path from i to j . The algorithm stops immediately, if one of the diagonal entries of D turn negative, i.e., if $d_{ii} < 0$ for some $i \in \{1, \dots, n\}$, because then a negative dicycle has been found. For a proof of the correctness of the algorithm see, for example, Ahuja et al., 1993.

To illustrate how the Algorithm of Floyd-Warshall (Algorithm 4) works, we consider the digraph in Figure 2.21, which contains a negative dicycle, and the digraph without a negative dicycle in Figure 2.22.

In the next section we present an algorithm to solve weighted matroid intersection problems. As a subproblem a shortest path problem with respect to node weights has to be solved. This can be done by constructing a corresponding extended graph with edge weights. As the graph gets much larger, this solution method can be computationally expansive. Hence we use an adapted version of the Floyd-Warshall algorithm, see Algorithm 5. When we consider node weights instead of edge weights, the main differences are the following: We need to adapt the initialization, because for every edge $(i, j) \in E$ we get the distance $d_{ij} = w(i) + w(j)$. Furthermore, every node by itself has a weight, and hence we get entries on the diagonal of D . This requires to change the condition to identify

Algorithm 5: Algorithm of Floyd-Warshall for node weights**Input:** digraph $G = (V, E)$, node weights $w(i)$ for all $i \in V$ ($n = |V|$, $m = |E|$)**Output:** shortest path for all pairs of nodes

```

1 foreach  $(i, j) \in V \times V$  do
2    $d_{ij} := \begin{cases} w(i) + w(j) & \text{if } (i, j) \in E \\ w(i) & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}, \quad \text{pred}_{ij} := \begin{cases} i & \text{if } (i, j) \in E \text{ or } i = j \\ 0 & \text{otherwise} \end{cases}$ 
3 for  $k := 1, \dots, n$  do
4   for  $i := 1, \dots, n$  do
5     for  $j := 1, \dots, n$  do
6       if  $d_{ij} > d_{ik} + d_{kj} - w(k)$  then
7          $d_{ij} := d_{ik} + d_{kj} - w(k)$ 
8          $\text{pred}_{ij} := \text{pred}_{kj}$ 
9       if  $i = j$  and  $d_{ii} < w(i)$  then
10        return  $G = (V, E)$  contains a negative dicycle
11 return distance matrix  $d_{ij}$  and predecessor labels  $\text{pred}_{ij} \forall i, j \in V$ 

```

negative dicycles, because as we allow negative node weights, it can happen that there are negative entries on the diagonal of D . However, we can identify a negative dicycle, if we update an entry on the diagonal, because then we have found a cycle that reduces the costs. Furthermore, if we compare the length of the paths from i to j or from i to j via node k , we have to subtract once the node weight $w(k)$ of node k . Otherwise, we would count node k twice, because it is contained in the path from i to k and in the path from k to j . An illustration of Algorithm 5 is given in Figure 2.23.

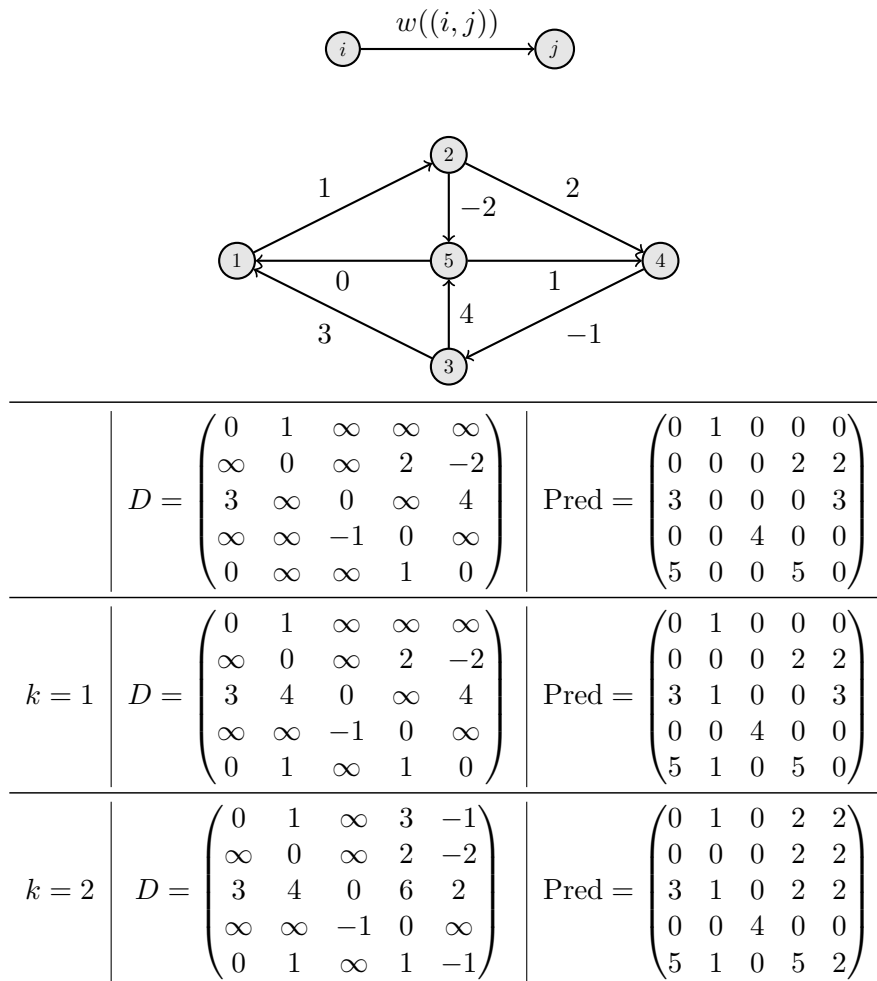


Figure 2.21.: An illustration of the algorithm of Floyd-Warshall. The algorithm stops for $k = 2$ and $i = j = 5$, because $d_{55} = -1 < 0$. The algorithm has found the negative dicycle $(5, 1, 2, 5)$.

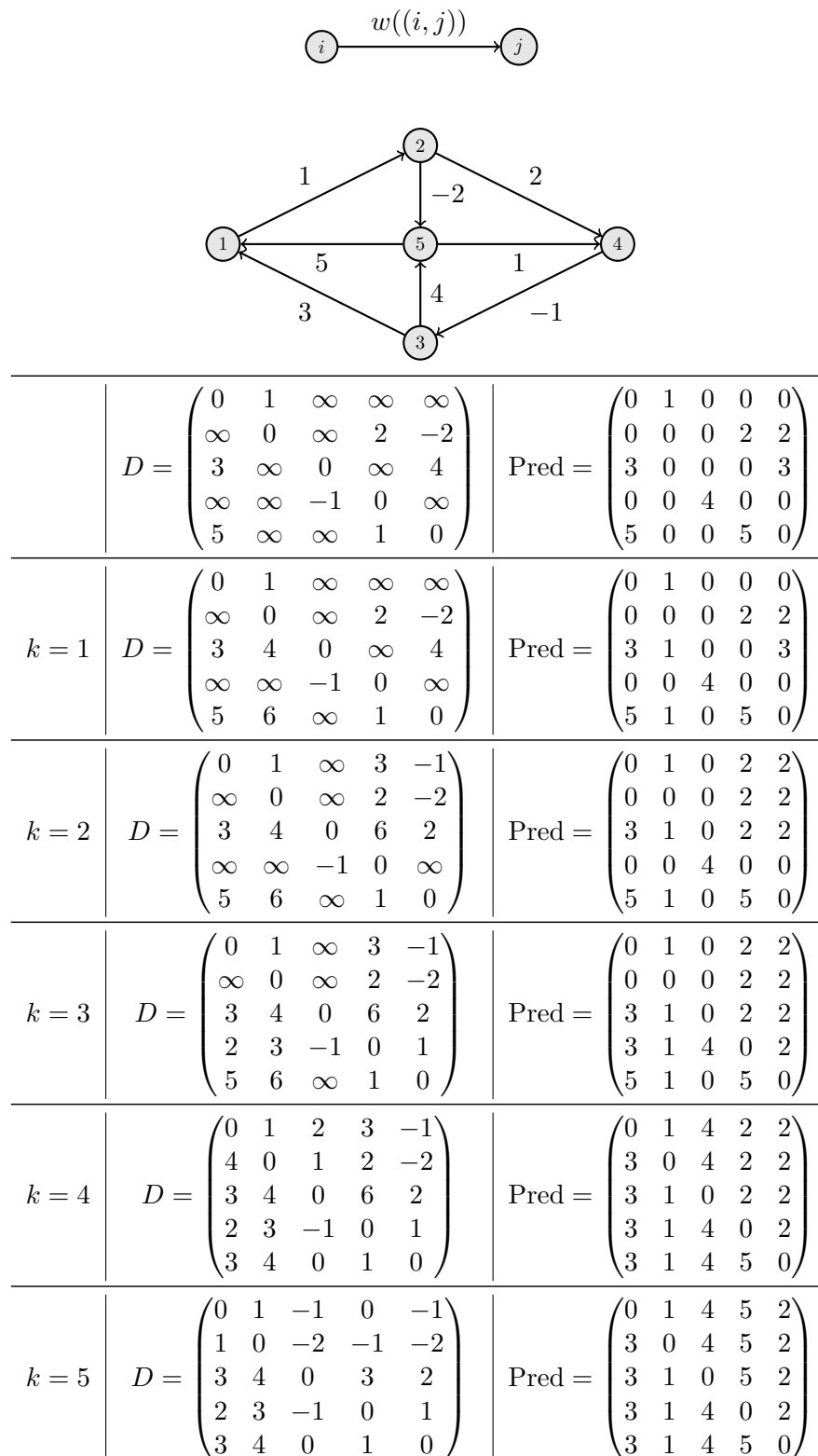


Figure 2.22.: An illustration of the algorithm of Floyd-Warshall for edge weights.

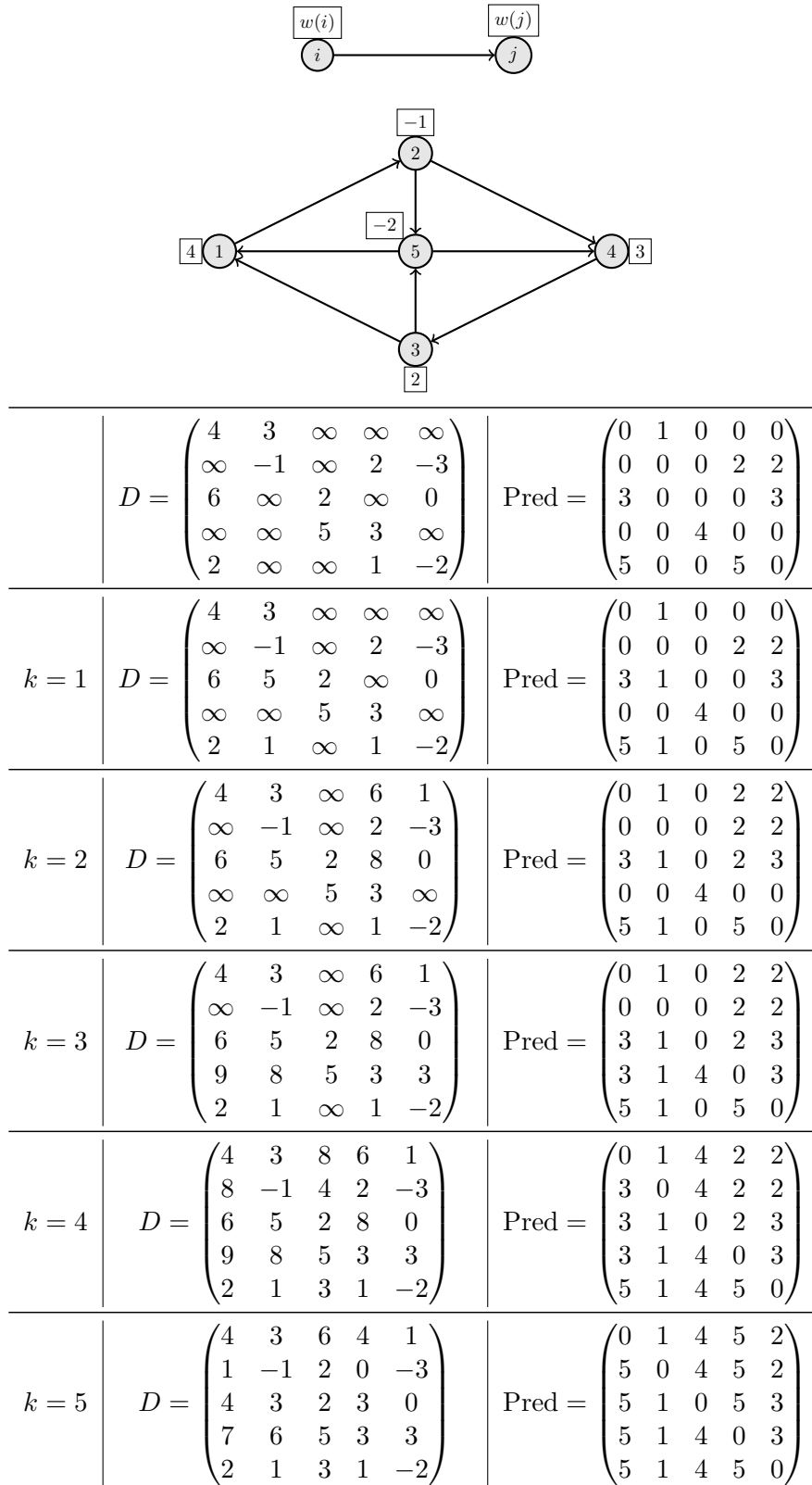


Figure 2.23.: An illustration of the algorithm of Floyd-Warshall for node weights.

2.8. Single-objective Matroid Intersection Problem

For the matroid intersection problem, we assume that two matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$ with the same ground set are given. Furthermore, let $w : E \rightarrow \mathbb{Q}$ be a weight function assigning to each element of the ground set a weight. We are looking for an independent set of the independence system, resulting from the intersection of those two matroids, which has maximal cardinality and minimal total weight. Hence, the *weighted matroid intersection problem* (MaIP) is defined as:

$$\begin{aligned} \min \quad & w(I) \\ \text{s. t.} \quad & I \in \mathcal{I}_1 \cap \mathcal{I}_2 \\ & |I| = \max\{|J| : J \in \mathcal{I}_1 \cap \mathcal{I}_2\}. \end{aligned} \tag{MaIP}$$

This problem can be solved by the *matroid intersection algorithm* (MI) of Edmonds, 1971. This algorithm is originally formulated for maximization problems, but it can also be applied to minimization problems by multiplying all weights $w(e)$ by -1 , $e \in E$, and hence maximizing $-w(e)$ rather than minimizing $w(e)$.

The idea of the algorithm is to start with an *extreme common independent set*, which is defined as follows:

Definition 2.12. *Let two matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$ over the same ground set be given and let $w : E \rightarrow \mathbb{Q}$ be a weight function. Then a common independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ is denoted as extreme (for minimization problems) if $w(I) \leq w(J)$ for each $J \in \mathcal{I}_1 \cap \mathcal{I}_2$ with $|J| = |I|$.*

Note that we can start with $I = \emptyset \in \mathcal{I}_1 \cap \mathcal{I}_2$. The matroid intersection algorithm iteratively extends the cardinality of extreme independent sets. For this purpose a directed graph is constructed, in which the nodes correspond to the elements of the matroid. Then, the nodes are divided into different, not necessarily disjoint subsets. The edges in the graph are chosen such that every path from a node in one subset to a node in an other subset defines an augmenting path of swaps increasing the cardinality of the common independent set by one. The precise definition is as follows:

Definition 2.13. *Let two matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$ over the same ground set be given and let $w : E \rightarrow \mathbb{Q}$ be a weight function. Furthermore, let $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ be a set that is independent w.r.t. both matroids \mathcal{M}_1 and \mathcal{M}_2 . Then we define the directed graph $G_{\mathcal{M}_1, \mathcal{M}_2, I} = (E, A_{\mathcal{M}_1, \mathcal{M}_2, I})$ with*

$$\begin{aligned} (i, j) \in A_{\mathcal{M}_1, \mathcal{M}_2, I} &\iff (I \setminus \{i\}) \cup \{j\} \in \mathcal{I}_1 \\ (j, i) \in A_{\mathcal{M}_1, \mathcal{M}_2, I} &\iff (I \setminus \{i\}) \cup \{j\} \in \mathcal{I}_2 \end{aligned}$$

for $i \in I$ and $j \in E \setminus I$. Each node of the graph has a weight v given by

$$v(i) := \begin{cases} -w(i) & i \in I \\ w(i) & i \in E \setminus I. \end{cases}$$

We define the sets $S := \{x \in E \setminus I : I \cup \{x\} \in \mathcal{I}_1\}$ and $T := \{x \in E \setminus I : I \cup \{x\} \in \mathcal{I}_2\}$.

Algorithm 6: Matroid Intersection Algorithm (MI($\mathcal{M}_1, \mathcal{M}_2, w, I$))

Input: $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$, $w : E \rightarrow \mathbb{Q}$ and an extreme common independent set I

Output: $I^* = \operatorname{argmin}\{\sum_{i \in I} w(i) : I \in \mathcal{I}_1 \cap \mathcal{I}_2 \text{ with } |I| = \max\{|J| : J \in \mathcal{I}_1 \cap \mathcal{I}_2\}\}$

1 Construct graph $G_{\mathcal{M}_1, \mathcal{M}_2, I} = (E, A_{\mathcal{M}_1, \mathcal{M}_2, I})$

2 $S := \{x \in E \setminus I : I \cup \{x\} \in \mathcal{I}_1\}$

3 $T := \{x \in E \setminus I : I \cup \{x\} \in \mathcal{I}_2\}$

4 **while** \exists a $S - T$ dipath in $G_{\mathcal{M}_1, \mathcal{M}_2, I}$ **do**

5 Determine a dipath P such that $v(P) = \sum_{i \in P} v(i)$ (counting multiplicities) is minimal and so that it has a minimum number of arcs among all minimum length $S - T$ paths, with

$$v(i) := \begin{cases} -w(i) & i \in I \\ w(i) & i \in E \setminus I. \end{cases}$$

$I := I \Delta P$

6 update $G_{\mathcal{M}_1, \mathcal{M}_2, I}$, S and T

7 **return** I

The set S contains those elements of E that can be added to the common independent set I , such that $I + e$ is still independent with respect to the matroid \mathcal{M}_1 . Analogously, the set T contains the same information for matroid \mathcal{M}_2 . Every edge in the graph $G_{\mathcal{M}_1, \mathcal{M}_2, I}$ either starts or ends in a node in I . Therefore, every path alternates between nodes in I and $E \setminus I$.

If a path starts in a node $e \in S$, then $I + e \in \mathcal{I}_1$. Nevertheless, $I + e \in \mathcal{I}_2$ is only true if e is also in T . In this case $I + e$ would be a common independent set additional with one more element. If $e \notin T$, then we have to follow an edge to a node i in I . If there exists such a node, we know that $I + e - i \in \mathcal{I}_1$, because $I + e \in \mathcal{I}_1$ and $I + e - i \in \mathcal{I}_2$ due to the definition of $A_{\mathcal{M}_1, \mathcal{M}_2, I}$. So, we found the new common independent set $I + e - i$ with the same cardinality as I . If we use further edges, for example, from i in I to $\tilde{e} \in S$, we can apply the same argumentation and get an other swap. Hence, the result is an other common independent set with the same cardinality as I . But finally the path terminates in a node $f \in T$. For easier notation we consider the case that the path is $P = \{e, i, f\}$. Then we know as explained above, that $I + e \in \mathcal{I}_1$ and $I + e - i \in \mathcal{I}_2$. Furthermore from $(i, f) \in A_{\mathcal{M}_1, \mathcal{M}_2, I}$ follows that $((I + e) - i + f) \in \mathcal{I}_1$ and from $f \in T$ we conclude that $((I + e - i) + f) \in \mathcal{I}_2$. Overall, we have found the common independent set $I + e - i + f$ with one more element than I .

Consequently, every $S - T$ path in $G_{\mathcal{M}_1, \mathcal{M}_2, I} = (E, A_{\mathcal{M}_1, \mathcal{M}_2, I})$ corresponds to a sequence of swaps such that the result is again a common independent set. Since vice versa every augmentation of the extreme common independent set I is represented by a $S - T$ path and the shortest path with minimal weight is chosen, we obtain an extreme common independent set.

The matroid intersection algorithm (Algorithm 6) repeats this for every updated graph

$G_{\mathcal{M}_1, \mathcal{M}_2, I}$ as long as there are paths from a node in S to a node in T . To formalize, we denote the augmentation of a common independent set I by swaps along a path P by $I \Delta P$.

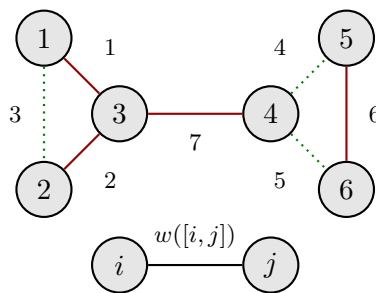
We refer to Schrijver, 2017 for a proof of its correctness and of its polynomial run time. Note that in Schrijver, 2017 a maximization problem is considered, while we consider a minimization problem.

In our implementation of the matroid intersection algorithm (Algorithm 6) we use a variant of the Floyd-Warshall algorithm (see Section 2.7) to compute the required shortest path. Note that, alternatively, the weights could be further transformed such that non-negative weight coefficients are obtained. Then, the algorithm of Dijkstra (see Section 2.7) could be applied for the relevant distance computations, see Frank, 1981 and Brezovec et al., 1986 for more details.

We illustrate the matroid intersection algorithm with the following example.

Example 2.14.

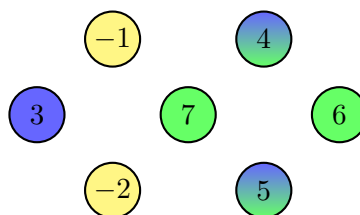
Input: Let $\mathcal{M}_1 = (E, V)$ a graphic matroid, given by the following graph:



Let \mathcal{M}_2 a partition matroid on the edges of the graph above and every subset of E with at most two red edges is an independent set.

A possible extreme common independent set as input for the algorithm is $I = \{1, 2\}$, where the edges are identified by their costs $w([i, j])$, $[i, j] \in V$.

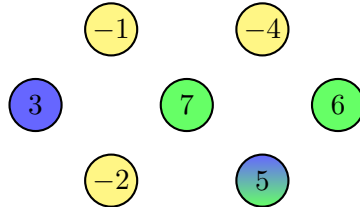
1. Iteration:



The nodes in I are colored in yellow, nodes in S are colored in green and nodes in T are colored in blue. Some nodes are both green and blue, when they belong to S and T simultaneously. In the nodes the weight $v(i)$ is written. The arcs of $G_{\mathcal{M}_1, \mathcal{M}_2, I}$ are missing, because the shortest path can be identified without them. The shortest path is obviously $P = (4)$, because every path in $G_{\mathcal{M}_1, \mathcal{M}_2, I}$ alternates between nodes in E/I and nodes in

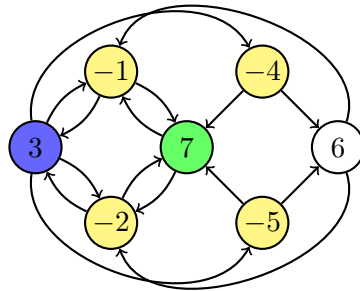
I. We notice that the costs of the nodes in E/I are larger than the negative costs of the nodes in I . Therefore, every path starting from a green node and ending in a blue one is longer than the path $P = (4)$. Hence, we get $I := I \triangle P = \{1, 2, 4\}$.

2. Iteration:



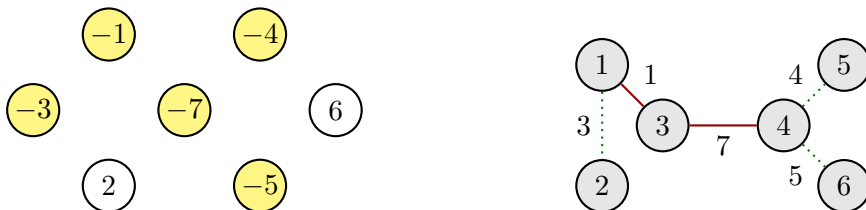
Here are the edges again missing, because we don't need them to identify the shortest path $P = (5)$. The only shorter path starting from a green node and ending with a blue node, with alternating nodes in I and in E/I would be the path $(5, -4, 3)$, but this is not a possible path, because $(5, -4) \notin A_{\mathcal{M}_1, \mathcal{M}_2, I}$. Hence, we get $I := I \triangle P = \{1, 2, 4, 5\}$.

3. Iteration:



Now the graph $G_{\mathcal{M}_1, \mathcal{M}_2, I}$ is given with its edges. The shortest path is $P = (7, -2, 3)$. Hence, we get $I := I \triangle P = \{1, 3, 4, 5, 7\}$.

4. Iteration:



We finally get $G_{\mathcal{M}_1, \mathcal{M}_2, I}$ as above on the left with $S = T = \emptyset$ and therefore, the algorithm terminates and we have found the optimal solution $I = \{1, 3, 4, 5, 7\}$, which leads to the spanning tree on the right.

2.9. Multi-objective Knapsack Problem

We first formally introduce the knapsack problem and review some solution methods afterwards. Thereby, we explain a dynamic programming approach in more detail, because it is used in some numerical tests of this thesis.

Problem Formulation In a knapsack problem we consider a set of n items or elements $E = \{e_1, \dots, e_n\}$ associated with non-negative weights $a(e_i) \in \mathbb{Z}_{\geq}$ for all $i = 1, \dots, n$. Additionally, every item e_i has a profit vector $w(e_i) \in \mathbb{Z}_{\geq}^p$, $i = 1, \dots, n$. Then, we aim to select items for our knapsack in order to maximize the total value vector w.r.t. Pareto optimality. Thereby, the total weight of the selected items must not exceed a given bound $\Omega \in \mathbb{Z}_{\geq}$. For every item $e_i \in E$, $i = 1, \dots, n$, we introduce a variable $x_i \in \{0, 1\}$ (or $x_i \in \mathbb{Z}_{\geq}$, respectively) that indicates whether we select item e_i (or even several copies of item e_i) or not. Depending on whether the variables are integer or binary, we obtain the *integer multi-objective knapsack problem*

$$\begin{aligned} \max_{C_P} \quad & \sum_{i=1}^n w(e_i) \cdot x_i \\ \text{s. t.} \quad & \sum_{i=1}^n a(e_i) \cdot x_i \leq \Omega \\ & x \in \mathbb{Z}_{\geq}^n \end{aligned}$$

or the *binary multi-objective knapsack problem*

$$\begin{aligned} \max_{C_P} \quad & \sum_{i=1}^n w(e_i) \cdot x_i \\ \text{s. t.} \quad & \sum_{i=1}^n a(e_i) \cdot x_i \leq \Omega \\ & x \in \{0, 1\}^n. \end{aligned}$$

Example 2.15. We consider the following integer multi-objective knapsack problem with three items, i.e., $n = 3$, and two objective functions, i.e., $p = 2$:

$$\begin{aligned} E &= \{e_1, e_2, e_3\} \\ w(e_1) &= (1, 5)^\top, \quad w(e_2) = (6, 3)^\top, \quad w(e_3) = (2, 2)^\top, \\ a(e_1) &= 3, \quad a(e_2) = 3, \quad a(e_3) = 2 \quad \text{and} \quad \Omega = 7. \end{aligned}$$

In the following all feasible solutions are specified, for which it holds that the addition of another item would make them infeasible:

$$\begin{aligned} x^1 &= (2, 0, 0)^\top, \quad x^2 = (1, 1, 0)^\top, \quad x^3 = (1, 0, 2)^\top, \\ x^4 &= (0, 2, 0)^\top, \quad x^5 = (0, 1, 2)^\top \quad \text{and} \quad x^6 = (0, 0, 3)^\top. \end{aligned}$$

The corresponding outcome vectors are

$$\begin{aligned} w(x^1) &= (2, 10)^\top, \quad w(x^2) = (7, 8)^\top, \quad w(x^3) = (5, 9)^\top, \\ w(x^4) &= (12, 6)^\top, \quad w(x^5) = (10, 7)^\top \quad \text{and} \quad w(x^6) = (6, 6)^\top. \end{aligned}$$

Outcome vector $w(x^6)$ is dominated by outcome vector $w(x^5)$ and all other outcome vectors do not dominate each other. Hence, the non-dominated set is $Y_N = \{w(x^1), \dots, w(x^5)\}$.

Solution Methods The decision problem associated with the knapsack problem is \mathcal{NP} -complete, see, for example, Kellerer et al., 2004. Nevertheless, knapsack problems can be solved in pseudopolynomial time by dynamic programming, which is a method based on Bellman's Principle of Optimality, see, for example, Lew and Mauch, 2007. The idea is to solve the original problem by iteratively solving subproblems. The principle says that, for appropriately defined subproblems, subsolutions of an optimal solution must be optimal for the associated subproblems. There exist different dynamic programming strategies for multi-objective knapsack problems, see, for example, Klamroth and Wiecek, 2000. We explain Model III from Klamroth and Wiecek, 2000 for the integer multi-objective knapsack problem since we use this method in Chapter 5. The model was developed for the single-objective knapsack problem by Garfinkel and Nemhauser, 1972 as well as Ibaraki, 1987 and for the multi-objective knapsack problem by Villarreal and Karwan, 1981. Eben-Chaime, 1996 proposed a similar model, which calculates all supported non-dominated outcome vectors for the multi-objective knapsack problem.

This dynamic programming strategy uses the set of states Q :

$$Q := \{q(k, j) : k = 0, 1, \dots, \Omega, j = 0, 1, \dots, n\}$$

with

$$q(k, j) := \{x \in \mathbb{Z}_{\geq}^n : \sum_{i=1}^j a(e_i) \cdot x_i = k, x_{j+1}, \dots, x_n = 0\},$$

i.e., in state $q(k, j)$ we assume that the total weight of the selected items has to be *exactly* k and assume that only the first j items can be considered. We define stage j as the union of all states $q(k, j)$ for $k = 0, \dots, \Omega$. Hence, stage j consists of all feasible solutions for which only the first j variables are considered.

We define the initial stage 0 by the states $q(k, 0) = \{0\}$ for $k = 0, \dots, \Omega$. We want to compute the final stage n with the states $q(k, n)$ for $k = 0, \dots, \Omega$. If we make the decision to fix variable x_j to the value $\alpha \in \mathbb{Z}_{\geq}$ in state $q(k, j-1)$, we move to the state $q(k + a(e_j) \cdot \alpha, j)$, since we fix one more variable and the total weight of the chosen items is increased by $a(e_j) \cdot \alpha$. We denote by $\mathcal{Y}_N(q(k, j))$ the set of all non-dominated outcome vectors of the multi-objective knapsack problem where the total weight of the chosen items is exactly k and $x_{j+1}, \dots, x_n = 0$. With the following recursive formula we can compute $\mathcal{Y}_N(q(k, j))$ for all $k = 0, \dots, \Omega$ and for all $j = 0, \dots, n$:

$$\begin{aligned} \mathcal{Y}_N(q(k, 0)) &= \{0\}, \text{ for all } k = 0, \dots, \Omega, \\ \mathcal{Y}_N(q(k, j)) &= \max_{C_P} \{\mathcal{Y}_N(q(k - a(e_j) x_j, j-1)) + x_j w(e_j) : x_j \in \mathbb{Z}_{\geq}, k - a(e_j) x_j \geq 0\}, \end{aligned}$$

for all $k = 0, \dots, \Omega$, and for all $j = 1, \dots, n$ where

$$\mathcal{Y}_N(q(k - a(e_j) x_j, j-1)) + x_j w(e_j) := \{v + x_j w(e_j) : v \in \mathcal{Y}_N(q(k - a(e_j) x_j, j-1))\} \subseteq \mathbb{Z}_{\geq}^n.$$

Finally, the set of non-dominated outcome vectors \mathcal{Y}_N of the original multi-objective knapsack problem is given as the set of Pareto non-dominated outcome vectors of the union of the sets $\mathcal{Y}_N(q(k, n))$, $k = 0, \dots, \Omega$, i.e.,

$$\mathcal{Y}_N = \max_{C_P} \bigcup_{k=0}^{\Omega} \mathcal{Y}_N(q(k, n)).$$

Example 2.16. We consider the knapsack problem defined in Example 2.15. We initialize $q(k, 0) = \{(0, 0, 0)^\top\}$ for $k = 0, \dots, 7$. If we are only allowed to choose the first item, we can select it either once, twice or not at all as it has the weight $a(e_1) = 3$. Hence, we get

$$q(k, 1) = \begin{cases} \{(1, 0, 0)^\top\} & \text{if } k = 3, \\ \{(2, 0, 0)^\top\} & \text{if } k = 6, \\ \{(0, 0, 0)^\top\} & \text{if } k \in \{1, 2, 4, 5, 7\}. \end{cases}$$

Suppose that we want to compute state $q(6, 2)$. The weight of item e_2 is $a(e_2) = 3$. Thus, we have the following options:

- move from state $q(0, 1) = \{(0, 0, 0)^\top\}$ to state $q(6, 2)$ by choosing element e_2 twice
- move from state $q(3, 1) = \{(1, 0, 0)^\top\}$ to state $q(6, 2)$ by choosing element e_2 once
- move from state $q(6, 1) = \{(2, 0, 0)^\top\}$ to state $q(6, 2)$ by deciding not to choose element e_2

Hence, we get $q(6, 2) = \{(0, 2, 0)^\top, (1, 1, 0)^\top, (2, 0, 0)^\top\}$. The corresponding outcome vectors are

$$w((0, 2, 0)^\top) = (12, 6)^\top, \quad w((1, 1, 0)^\top) = (7, 8)^\top \quad \text{and} \quad w((2, 0, 0)^\top) = (2, 10)^\top.$$

Those vectors do not dominate each other and we can conclude

$$Y_N(q(6, 2)) = \{(12, 6)^\top, (7, 8)^\top, (2, 10)^\top\}.$$

Similarly, all other states have to be computed.

There exist some improvements of standard dynamic programming algorithms for the bi-objective binary knapsack problem based on lower and upper bounds, see, for example, Figueira et al., 2013. Furthermore, Bazgan et al., 2009 improved a dynamic programming algorithm for binary multi-objective knapsack problems by using several complementary dominance relations to discard partial solutions that cannot result in a new non-dominated outcome vector. There exist also other solution strategies like the one proposed by Yuan and Li, 2021 for binary multi-objective knapsack problem, which modifies infeasible solutions such that they get feasible and improves feasible solutions by a greedy strategy. In some studies, evolutionary methods are applied to the multi-objective knapsack problem, see, for example, Mansour et al., 2018. In Schulze, 2017 efficient algorithms are presented to solve multi-objective unconstrained combinatorial optimization problems and multi-objective knapsack problems. Moreover, the special case of rectangular knapsack problems is investigated.

2.10. Multi-objective Assignment Problem

In this section we first formally introduce assignment problems and afterwards, we review some solution methods. We explain a generic objective space algorithm for general multi-objective optimization problems, which we use in this thesis to solve assignment problems. Nevertheless, the algorithm can solve all classes of MOCO problems.

Problem Formulation We consider a bipartite directed graph $G = (V_1 \cup V_2, E)$ with $|V_1| = |V_2| \geq 1$ and with edges that have tails in V_1 and heads in V_2 , i.e., $i \in V_1, j \in V_2$ for all $(i, j) \in E$. As we want to formulate a multi-objective assignment problem, we assume that a weight vector $w(e) \in \mathbb{R}^p$ is associated with every edge $e \in E$. The feasible set is given by all perfect matchings of the assignment problem. For the objective function we consider the minimization of the total weight vectors (w.r.t. Pareto optimality). The multi-objective assignment problem is \mathcal{NP} -complete, see, for example, Ehrgott, 2005.

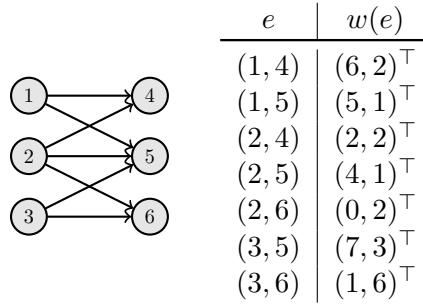


Figure 2.24.: Example of a bi-objective assignment problem with the disjoint node sets $V_1 = \{1, 2, 3\}$ and $V_2 = \{4, 5, 6\}$.

Example 2.17. We consider the bi-objective assignment problem defined by the graph in Figure 2.24. In the table next to the graph the edge weights are specified. The perfect matchings are

- $\{(1, 4), (2, 5), (3, 6)\}$ with outcome vector $(11, 9)^\top$,
- $\{(1, 4), (2, 6), (3, 5)\}$ with outcome vector $(13, 7)^\top$ and
- $\{(1, 5), (2, 4), (3, 6)\}$ with outcome vector $(8, 9)^\top$.

The third matching dominates the first one and thus, the resulting non-dominated set is $Y_N = \{(13, 7)^\top, (8, 9)^\top\}$.

Solution Methods We consider multi-objective assignment problems in Chapter 5 and solve them with a generic objective space algorithm presented in Klamroth et al., 2015. This algorithm is not restricted to solve multi-objective assignment problems and can also be used to solve general multi-objective optimization problems. The algorithm assumes that an upper bound and a lower bound for the outcome set is given. This is obviously the case for assignment problems on finite graphs. Then the whole volume between the lower and upper bound is seen as initial search region in the objective space. In this search region a non-dominated point is computed by an appropriate scalarization technique, e.g., the ε -constraint scalarization. The search region is updated and new local upper bounds are computed such that the search region corresponds exactly to the region where possibly further non-dominated points can exist. The search region consists of non-disjoint boxes, called search zones, each of which is defined by a local upper bound. Iteratively, the boxes are investigated to find new non-dominated points. If there is no non-dominated point in a search zone, the zone is discarded. Otherwise, if a new point is found, the overall search

region, i.e., the set of local upper bounds which define the search zones, is updated. See also Dächert et al., 2017 and Tamby and Vanderpooten, 2021 for variants of this method.

There exist a large variety of other exact and heuristic solution approaches like, e.g., heuristics based on evolutionary algorithms or two phase algorithms. For example, in Cubukcuoglu et al., 2019 a memetic algorithm, which is an evolutionary algorithm that uses a local search technique for bi-objective quadratic assignment problems, is presented. In Przybylski et al., 2008 two phase algorithms for the bi-objective assignment problems are presented and compared w.r.t. their computational performance.

In some references, e.g., a more general variant of the assignment problem is considered, the so called generalized assignment problem. In the generalized assignment problem every edge has, in addition to its weight, some costs, and every node of the first vertex set V_1 has a budget. In this case we are not looking for a perfect matching. Instead we want to maximize the total weight of chosen edges under the constraint that for every vertex in V_1 it has to hold that the total sum of the costs of the chosen incident edges is less or equal to the budget of this vertex. If all budgets and all costs are equal to 1, then we get the standard assignment problem (as a maximization problem). For the bi-objective generalized assignment problem Zhang and Ong, 2007 presented a heuristic based on linear programming methods.

There exist many results on other variants of the multi-objective assignment problem, see, for example, Mosheiov and Sarig, 2008 and Pramanik and Biswas, 2012. For a survey on variations of the single-objective and multi-objective assignment problem we refer to Pentico, 2007.

3. Bi-objective Matroid Optimization Problems with Binary Costs

The hardness results mentioned in Chapter 1 usually refer to MOCO instances with 'large' cost coefficients that may grow exponentially with the instance size. For problems with 'small' cost coefficients the situation is different. When coefficients are small, then the ranges of possible outcome values are bounded, which limits the size of the non-dominated set. For example, the bi-objective minimum spanning tree problem has only supported efficient solutions when all cost coefficients take only values from the set $\{0, 1, 2\}$, see Seipp, 2013. This implies that all efficient solutions of this problem are connected, i.e., the complete efficient set can be generated by only performing simple swap operations (e.g., pivot operations in an associated linear programming formulation) among efficient solutions. In the same work, Seipp, 2013 shows that tri-objective optimization problems on uniform matroids with one general cost function and two binary cost functions have a connected efficient set. However, in general even comparably simple problems like bi-objective unconstrained combinatorial optimization problems may possess a non-connected efficient set, see Gorski et al., 2011.

In this chapter we focus on bi-objective optimization problems on matroids that have binary coefficients in *one* of the objectives. While the first objective may take arbitrary non-negative integer values, we assume that the second objective takes only values from the set $\{0, 1\}$. The specific structure of this problem allows us to solve the problem efficiently, as the number of feasible outcome vectors is bounded. Note that binary coefficients allow for an alternative interpretation of the problem: When associating a cost of 0, for example, with the color 'green', and a cost of 1 with the color 'red', then we are interested in the simultaneous minimization of the cost of a solution (w.r.t. the first objective) *and* of the number of its red elements.

A related problem is the *multicolor matroid problem* that was discussed by Rendl and Leclerc, 1988-1989 and by Brezovec et al., 1988. In this problem, a minimum cost solution is sought that does not exceed a given bound on the number of elements from different colors. Srinivas, 1995 extended the results from Brezovec et al., 1988 to the case that the number of elements of different colors is constrained by linear inequalities. Hamacher and Rendl, 1991 generalized the multicolor matroid problem to combinatorial optimization problems, now allowing for elements having more than one color. Similar to Brezovec et al., 1988 the goal is to find minimum cost solutions not exceeding given bounds on the number of elements in each color. A different optimization objective was considered in Climaco et al., 2010, who discussed a bi-objective minimum cost / minimum label spanning tree problem in a graph where each edge is associated with a cost value and a label (i.e., a color). While the first objective is a classical cost objective that is to be minimized, the second objective is to find a solution with a minimal number of *different* labels (i.e., colors). 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, 1997, this problem is also \mathcal{NP} -hard.

From an application point of view, MOCO problems with one general objective function and one (or several) binary objectives are closely related to k – max optimization where the k th largest cost coefficient of an outcome vector is to be minimized. Such problems can be translated into a series of problems with binary sum objectives in a thresholding framework, see, Gorski and Ruzika, 2009 for more details.

Contribution. This work was motivated by discussions with Stefan Ruzika on spanning tree problems with two binary objective functions, which were presented at the 23rd European Conference on Operational Research, see Ruzika, 2009. The results of this chapter are published in Gorski et al., 2022. Parts of the results can be seen as an extension and generalization of Chapter 10 of the dissertation of Gorski, 2010. We show that the non-dominated set of bi-objective optimization problems on matroids with one general and one binary objective function contains only supported efficient solutions and is connected. This is the basis for an efficient exact algorithm, which has polynomial runtime, that enumerates the non-dominated set using a neighborhood search approach, i.e., using simple swaps between elements contained in different (efficient) bases of the problem. This *Efficient Swap Algorithm* ESA can be interpreted as an extension of the algorithm of Gabow and Tarjan, 1984 for a constrained version of the problem that is guaranteed to generate the complete non-dominated set. To the best of our knowledge, this is the first non-trivial optimization problem on matroids for which connectedness of the efficient set is established.

Organization of the chapter. The remainder of this chapter is organized as follows. The bi-objective matroid optimization problem with one general and one binary cost objective is introduced in Section 3.1. In Section 3.2 we present some theoretical results, that are needed to prove the correctness of the algorithm and to prove the connectedness of the efficient set. The neighborhood search algorithm ESA is presented in Section 3.3. The numerical results presented in Section 3.4 confirm the efficiency of the algorithm ESA. The chapter is concluded in Section 3.5 with some ideas for future research.

3.1. Problem Formulation

Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid and let \mathcal{X} denote the set of all bases of \mathcal{M} . We assume that $\text{rank}(\mathcal{M}) = r > 0$, i.e., the cardinality $|\mathcal{B}|$ of all bases $\mathcal{B} \in \mathcal{X}$ is equal to r . In the following we consider two different types of cost functions on the ground set E . While the first function $w : E \rightarrow \mathbb{Z}_{\geq}$ is given by arbitrary non-negative integer coefficients, we assume that the second cost function $b : E \rightarrow \{0, 1\}$ only takes binary values. According to these definitions the two different costs of a basis $\mathcal{B} \in \mathcal{X}$ are given by $w(\mathcal{B}) = \sum_{e \in \mathcal{B}} w(e)$ and $b(\mathcal{B}) = \sum_{e \in \mathcal{B}} b(e)$, respectively. The related *bi-objective matroid problem with binary costs* (BBMP) is given by

$$\begin{aligned} \min \quad & (w(\mathcal{B}), b(\mathcal{B})) \\ \text{s. t.} \quad & \mathcal{B} \in \mathcal{X}. \end{aligned} \tag{BBMP}$$

Since the second cost function b has binary coefficients for all elements $e \in E$, the corresponding objective function values of feasible bases $\mathcal{B} \in \mathcal{X}$ are lower bounded by zero and upper bounded by r . In other words, $b(\mathcal{X}) := \{b(\mathcal{B}) : \mathcal{B} \in \mathcal{X}\} \subseteq \{0, \dots, r\}$ is of size $\mathcal{O}(r)$, and thus the same bound also holds for \mathcal{Y}_N .

For solving problem (BBMP) we introduce the following two associated ε -constraint versions of the problem. The first is given by

$$\begin{aligned} \min \quad & w(\mathcal{B}) \\ \text{s. t.} \quad & b(\mathcal{B}) \leq \varepsilon \\ & \mathcal{B} \in \mathcal{X}, \end{aligned} \tag{BMP}_{\leq}$$

where $\varepsilon \in \{0, \dots, r\}$ is a fixed integer bound on the binary cost function b . From the theory of multiple criteria optimization (see, e.g., Chankong and Haimes, 1983) we know that each optimal solution of problem (BMP_{\leq}) is at least weakly efficient for problem (BBMP). Note that this is not true in general when the inequality constraint in problem (BMP_{\leq}) is replaced by an equality constraint. Indeed, given an optimal solution of the equality constrained problem

$$\begin{aligned} \min \quad & w(\mathcal{B}) \\ \text{s. t.} \quad & b(\mathcal{B}) = \varepsilon \\ & \mathcal{B} \in \mathcal{X}, \end{aligned} \tag{BMP}_{=}$$

this solution may be dominated in problem (BBMP). An example for this situation can be seen in Figure 3.3. There, each basis $\mathcal{B} \in \mathcal{X}$ that maps to the outcome vector $(w(\mathcal{B}), b(\mathcal{B})) = (18, 5)$ is optimal for problem $(\text{BMP}_{=})$ with $\varepsilon = 5$, while it is dominated by all bases that map to the outcome vector $(17, 4)$ for the bi-objective problem. Nevertheless, we use problem $(\text{BMP}_{=})$ to generate a sequence of optimal solutions by varying $\varepsilon \in \{0, \dots, r\}$ and show that there exists a critical index j such that for all $\varepsilon \leq j$ all generated bases that are optimal for problem $(\text{BMP}_{=})$ correspond to efficient bases of problem (BBMP).

Note that the binary cost function b introduced above also allows for another interpretation as used, for example, in Gabow and Tarjan, 1984 and Gusfield, 1984: Given a matroid $\mathcal{M} = (E, \mathcal{I})$ and a (first) cost function $w : E \rightarrow \mathbb{Z}_{\geq}$, one of the two colors red and green is assigned to each element of E . In Gabow and Tarjan, 1984 and Gusfield, 1984 algorithms are presented that determine a minimum cost basis $\mathcal{B} \in \mathcal{X}$ that contains exactly ε red elements from E (here, ε is a predetermined parameter). To establish a connection between the problem discussed in Gabow and Tarjan, 1984 and Gusfield, 1984 and the problems considered here, we simply identify the red elements $e \in E$ from the ground set E with the binary costs $b(e) = 1$, while all green elements $f \in E$ are considered to have binary cost $b(f) = 0$. Hence, determining a minimum cost basis $\mathcal{B} \in \mathcal{X}$ containing at most or exactly ε red elements from E corresponds to solving problem (BMP_{\leq}) and $(\text{BMP}_{=})$, respectively. In this context, especially problem $(\text{BMP}_{=})$ can be seen as a generalized version of a single-objective matroid problem with an additional constraint, where the original problem is obtained when E only consists of red elements and $\varepsilon = r$. Note that for a better illustration, we make use of the idea of red and green elements in the further sections.

Given an instance of problem (BBMP), we denote by $E_0 := \{e \in E : b(e) = 0\}$ the

subset of E containing all elements with binary cost 0 (green elements) while the set $E_1 := \{e \in E : b(e) = 1\} = E_0^c$ contains all elements with binary cost 1 (red elements). By definition, E_0 and E_1 form a partition of E .

For this purpose, let $i \in \{0, \dots, r\}$ and $\mathcal{X}_i := \{\mathcal{B} \in \mathcal{X} : |\mathcal{B} \cap E_0| = i\}$ be the set of all bases with exactly i green elements. Note that \mathcal{X}_i might be empty for low or high values of i , respectively. Furthermore, let

$$\mathcal{S}_i := \{\mathcal{B} \in \mathcal{X}_i : w(\mathcal{B}) \leq w(\mathcal{B}') \forall \mathcal{B}' \in \mathcal{X}_i\}$$

denote the set of all bases with minimal costs containing exactly i green elements from E_0 . By construction, $\mathcal{B} \in \mathcal{S}_i$ is an optimal basis of problem (BMP $_{=}$) with right hand side value $\varepsilon = r - i$.

3.2. Theoretical Results

In this section we present the general steps of an algorithm that computes the complete non-dominated set of problem (BBMP) in polynomial time. The algorithm is discussed in detail in Section 3.3. The method is based on the ideas stated in Gabow and Tarjan, 1984 and can be used to establish a connectedness result for the adjacency graph of problem (BBMP). In more detail, the algorithm generates a sequence of optimal solutions of problem (BMP $_{=}$) for decreasing right-hand side values ε . In Subsection 3.2.1 we formulate the theoretical results that are needed to prove the correctness of the method, and in Subsection 3.2.2 we show the connectedness of the efficient set.

Throughout this section, we consider problem (BBMP) on a given matroid $\mathcal{M} = (E, \mathcal{I})$ with the set of feasible bases \mathcal{X} .

3.2.1. Preliminaries

The idea of our approach to generate the complete non-dominated set of problem (BBMP) is based on the stronger version of the basis exchange property for matroids stated in Lemma 2.7 in Section 2.5. Given this property we define *swaps* between elements from E_0 and E_1 .

Definition 3.1. *Let $\mathcal{B} \in \mathcal{X}$. Then the swap (e, f) w.r.t. \mathcal{B} is an ordered pair of elements such that $e \in E_1 \cap \mathcal{B}$, $f \in E_0 \setminus \mathcal{B}$ and $\mathcal{B} - e + f \in \mathcal{X}$ is a basis. The cost of the swap (e, f) is defined as $w(e, f) := w(f) - w(e)$. A swap (e, f) is called minimal w.r.t. \mathcal{B} if $w(e, f) \leq w(e', f')$ for all $e' \in E_1 \cap \mathcal{B}$ and $f' \in E_0 \setminus \mathcal{B}$ with $\mathcal{B} - e' + f' \in \mathcal{X}$.*

By definition, a swap always improves the binary cost function by one unit since a red element from E_1 is replaced by a green element from E_0 . The idea of the *efficient swap algorithm* (ESA) is to generate a sequence of minimal swaps that yields all non-dominated outcome vectors of problem (BBMP), as outlined in Algorithm 7.

A detailed description of this approach is given in Algorithm 8 in Section 3.3, after a thorough analysis of the individual steps.

From Gabow and Tarjan, 1984 we recall that given an optimal solution $\mathcal{B} \in \mathcal{S}_{i-1}$, a minimal swap can be used to generate an optimal solution contained in \mathcal{S}_i whenever \mathcal{S}_i is non-empty.

Algorithm 7: Outline of the Efficient Swap Algorithm (ESA) for Bi-objective Matroid Problems with one Binary Cost Function

Input: An instance $((\mathcal{M}, \mathcal{X}, (c, b))$ of problem (BBMP).

Output: \mathcal{Y}_N and a complete set \mathcal{X}_{cE} of efficient solutions.

- 1 Determine a basis \mathcal{B}_j , which is optimal with respect to w , and a basis \mathcal{B}_u , which is optimal with respect to b , such that both bases have as many elements as possible in common.
 - 2 Compute a sequence of minimal swaps, which describes the necessary swaps to get from basis \mathcal{B}_j to basis \mathcal{B}_u .
 - 3 Sort the swaps in non-decreasing order with respect to their costs.
 - 4 Compute from the sorted swap sequence a complete set \mathcal{X}_{cE} of efficient solutions and the corresponding outcome vectors \mathcal{Y}_N .
 - 5 **return** \mathcal{X}_{cE} and \mathcal{Y}_N .
-

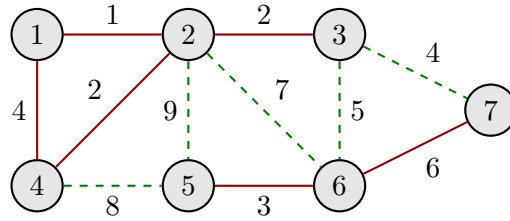


Figure 3.1.: Graph $G = (V, E)$ with costs $w(e)$, $e \in E$, for the graphic matroid considered in Example 3.4. Dashed green lines correspond to edges $e \in E$ with $b(e) = 0$ and solid red lines correspond to edges with $b(e) = 1$, respectively.

Theorem 3.2 (see Gabow and Tarjan, 1984, Augmentation Theorem 3.1). *Let $\mathcal{B} \in \mathcal{S}_{i-1}$ for an $i \in \{1, \dots, r\}$ and assume that $\mathcal{S}_i \neq \emptyset$. If the swap (e, f) is minimal w.r.t. \mathcal{B} , then $\mathcal{B} - e + f$ is contained in \mathcal{S}_i .*

The following result is an immediate consequence of Theorem 3.2.

Corollary 3.3. *Let $l, u \in \mathbb{Z}_{\geq}$ with $0 \leq l < u \leq r$ such that $\mathcal{S}_l \neq \emptyset \neq \mathcal{S}_u$. Then $\mathcal{S}_i \neq \emptyset$ for all $i \in \{l, \dots, u\}$.*

Note that Corollary 3.3 does not state that problem (BMP₌) is feasible for all right-hand side values $\varepsilon \in \{0, \dots, r\}$. However, it implies that there exist fixed lower and upper bounds $l, u \in \mathbb{Z}_{\geq}$ (satisfying $0 \leq l \leq u \leq r$) such that $\mathcal{S}_i \neq \emptyset$ for all $i \in \{l, \dots, u\}$ while $\mathcal{S}_j = \emptyset$ for all $j \in \{0, \dots, r\} \setminus \{l, \dots, u\}$.

The results of Theorem 3.2 and Corollary 3.3 imply a simple algorithm that allows to generate a superset of the non-dominated set for a given instance of problem (BBMP) by swapping between the optimal bases contained in \mathcal{S}_i for $i = \{l, \dots, u\}$. In this method, a sequence of minimal swaps has to be generated. The algorithm presented in Gabow and Tarjan, 1984 uses a recursive procedure to generate this sequence. Further details on the

generation of minimal swaps are given in Section 3.3 below. Example 3.4 illustrates the idea of sequential minimal swaps at a graphic matroid.

Example 3.4. We consider the graphic matroid induced by the graph $G = (V, E)$ given in Figure 3.1. Note that \mathcal{X} is the set of all spanning trees of G , and that the matroid has rank $r = 6$. The objective coefficients of the first objective function w are depicted next to each edge. For the second objective b , a solid red edge is used to indicate a cost of 1, while a dashed green edge indicates a cost of 0.

The spanning trees T_1, \dots, T_5 given in Figure 3.2 correspond to optimal solutions for problem (BMP₌) for the right-hand side values $\varepsilon \in \{1, \dots, 5\}$. We have that $T_i \in \mathcal{S}_i$, $i \in \{1, \dots, 5\}$ while $\mathcal{S}_0 = \mathcal{S}_6 = \emptyset$, i.e., $l = 1$ and $u = 5$. The objective vector $(w(T_i), b(T_i))$ of tree T_i , $i = 1, \dots, 5$, is stated in the first column, below the name of the respective tree. The corresponding trees are shown in the second column. The tables in the right-most column list relevant swaps w.r.t. the tree T_i , $i = 1, \dots, 5$, together with the respective cost, where minimal swaps are highlighted in bold. Here, the “in”-column goes through the list of all dashed green edges that are not yet contained in T_i and that may hence potentially be included. Adding the respective edges induces a unique cycle, and the best possible outgoing edge is shown in the “out”-column. It is selected as a solid red edge in this cycle with maximum cost. Since we exchange a red against a green edge, the swap with minimal cost $w(e, f)$ w.r.t. T_i leads to an optimal spanning tree $T_{i+1} \in \mathcal{S}_{i+1}$. While the spanning tree T_1 is dominated by T_2 (the implemented swap decreases each objective by one unit), the remaining trees form a complete set of efficient solutions and we conclude that $\mathcal{Y}_N = \{(17, 4), (22, 3), (27, 2), (34, 1)\}$.

Note that the procedure that is used to iteratively determine minimal swaps in Example 3.4 originates from Gusfield, 1984. In the following, we present an improved procedure that avoids the computation of many unnecessary swaps. Example 3.4 further shows that not all optimal spanning trees for problem (BMP₌) result in an efficient solution for problem (BBMP). However, we show in the following that there exists a fixed index $j \in \{l, \dots, u\}$ such that $\mathcal{B} \in \mathcal{S}_i$ is efficient whenever $i \geq j$. Having a closer look at the example, it can be recognized that the minimal swaps that lead from T_1 to T_5 have non-decreasing costs. To prove that this property holds in general, we need the following lemma from Gabow and Tarjan, 1984.

Lemma 3.5 (see Gabow and Tarjan, 1984, Lemma 3.2). *Let \mathcal{B} be a basis containing the element $e \in E_1 \cap \mathcal{B}$. Let (e, f) be a swap w.r.t. \mathcal{B} that has minimal cost among all swaps w.r.t. \mathcal{B} involving e , and set $\mathcal{B}' = \mathcal{B} - e + f$. Given $\tilde{e} \in E_1 \cap (\mathcal{B} - e)$ arbitrary but fixed, let (\tilde{e}, \tilde{f}) and (\tilde{e}, \tilde{f}') denote swaps w.r.t. \mathcal{B} and \mathcal{B}' , respectively, that have minimal costs w.r.t. \mathcal{B} and \mathcal{B}' , respectively, and that involve \tilde{e} . Then it holds that $w(\tilde{e}, \tilde{f}) \leq w(\tilde{e}, \tilde{f}')$.*

Using Lemma 3.5 it can now be shown that the sequence of costs induced by a sequence of minimal swaps is non-decreasing for increasing $i \in \{l, \dots, u\}$.

Theorem 3.6. *Let $u \geq l + 2$. For $i \in \{l, \dots, u - 1\}$ let $\mathcal{B}_i \in \mathcal{S}_i$ and let (e_i, f_i) denote a minimal swap w.r.t. \mathcal{B}_i leading to \mathcal{B}_{i+1} . Then the sequence of costs of minimal swaps $\{w(e_i, f_i)\}_{i=l}^{u-1}$ is non-decreasing, i.e., $w(e_i, f_i) \leq w(e_{i+1}, f_{i+1})$ for all $i \in \{l, \dots, u - 2\}$.*

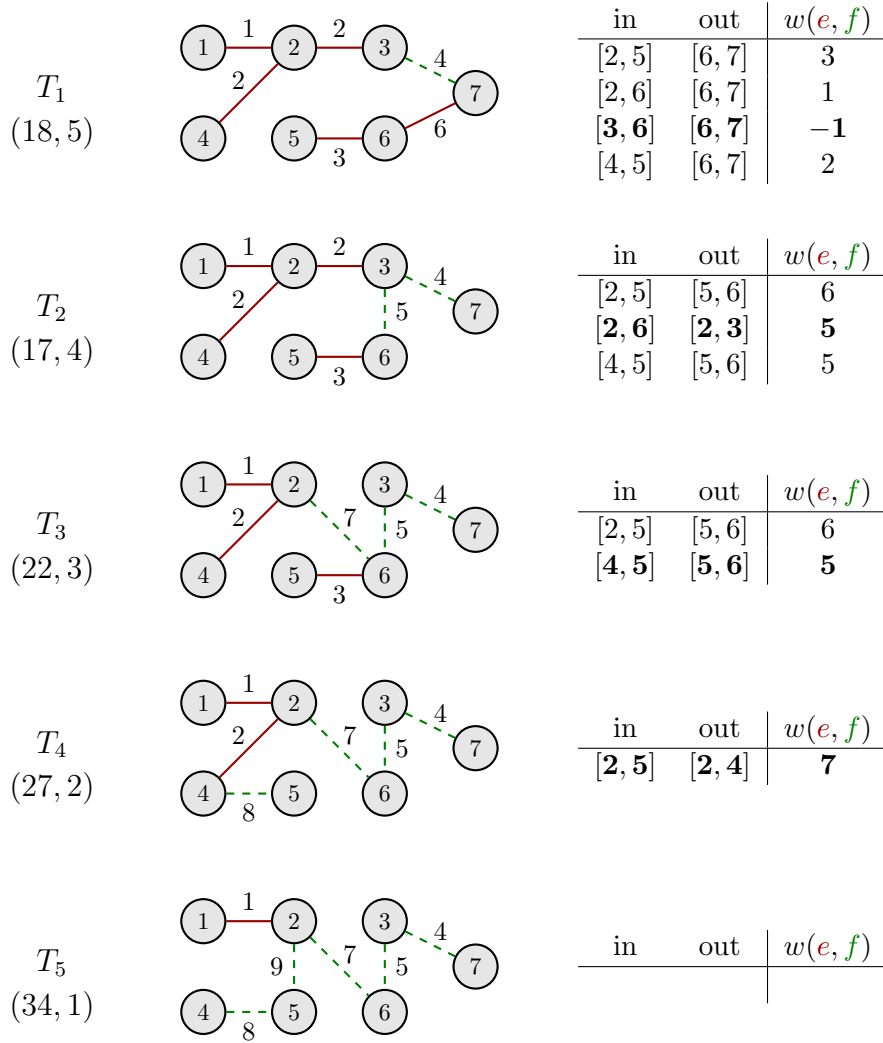


Figure 3.2.: Sequence of optimal spanning trees $\{T_1, \dots, T_5\}$ for $(\text{BMP}_=)$ for the graphic matroid defined in Example 3.4. Left column: tree T_i and corresponding objective vector. Center column: associated tree. Right column: computation of a minimal swap $w(e, f)$ w.r.t. T_i , $i = 1, \dots, 5$.

Proof. Let $\{w(e_i, f_i)\}_{i=l}^{u-1}$ be a cost sequence of minimal swaps and let $i \in \{l, \dots, u-2\}$ arbitrary but fixed. Note that $e_i \neq e_{i+1}$ since $e_i \in \mathcal{B}_i \setminus \mathcal{B}_{i+1}$. Moreover, $e_{i+1} \in \mathcal{B}_i \cap \mathcal{B}_{i+1}$ since otherwise e_{i+1} would be contained in $\mathcal{B}_{i+1} \setminus \mathcal{B}_i$, i.e., $e_{i+1} = f_i$. But since e_{i+1} is a red element of E while f_i is a green element, this is impossible.

Now consider a swap (e_{i+1}, f) w.r.t. \mathcal{B}_i that has minimal cost among all swaps w.r.t. \mathcal{B}_i that involve e_{i+1} . Note that the existence of a swap w.r.t. \mathcal{B}_i involving the edge e_{i+1} follows from the basis exchange property (B), see Section 2.5: For the two bases $\mathcal{B}_i, \mathcal{B}_{i+2}$ we have $e_{i+1} \in \mathcal{B}_i \setminus \mathcal{B}_{i+2}$ and hence there exists an element $f \in \mathcal{B}_{i+2} \setminus \mathcal{B}_i$ such that $\mathcal{B}_i - e_{i+1} + f \in \mathcal{X}$. Hence, (e_{i+1}, f) is a feasible swap w.r.t. \mathcal{B}_i involving e_{i+1} and with $f \in \mathcal{B}_{i+2} \setminus \mathcal{B}_i = \{f_i, f_{i+1}\}$.

Since (e_i, f_i) is a minimal swap w.r.t. \mathcal{B}_i it follows that $w(e_i, f_i) \leq w(e_{i+1}, f)$. If $f = f_{i+1}$, we are done. Otherwise, we conclude from Lemma 3.5 that the inequality $w(e_{i+1}, f) \leq w(e_{i+1}, f_{i+1})$ holds, since the swap (e_{i+1}, f_{i+1}) is minimal w.r.t. \mathcal{B}_{i+1} . Combining these results we get $w(e_i, f_i) \leq w(e_{i+1}, f_{i+1})$, which completes the proof. \square

Since we have that

$$w(\mathcal{B}_{i+1}) - w(\mathcal{B}_i) = w(f_i) - w(e_i) = w(e_i, f_i), \quad (3.1)$$

Theorem 3.6 implies that the minimum costs of bases $\mathcal{B} \in \mathcal{S}_i$ define a convex function for $i = |\mathcal{B} \cap E_0| \in \{l, \dots, u\}$. Furthermore, if w and b are conflicting, then there must exist an index $j \in \{l, \dots, u\}$ such that, starting from this index, all subsequent bases contained in the sequence $\{\mathcal{B}_i\}_{i=j}^u$ correspond to efficient solutions of problem (BBMP). This holds since the value of the binary objective function b is decreased by one unit when a swap from \mathcal{B}_i to \mathcal{B}_{i+1} is performed, while the corresponding value of the cost function w remains constant or is increased. By construction, the index j is the first index from $\{l, \dots, u-1\}$ for which $w(e_i, f_i) > 0$ holds true. This implies the following result.

Theorem 3.7. *Let $\{\mathcal{B}_i\}_{i=l}^u$ denote the sequence of minimum cost bases such that $\mathcal{B}_i \in \mathcal{S}_i$ for $i \in \{l, \dots, u\}$. Assume that $u \geq l+2$. If there exists an index $j \in \{l+1, \dots, u\}$ such that $w(\mathcal{B}_{j-1}) < w(\mathcal{B}_j)$, then $w(\mathcal{B}_i) < w(\mathcal{B}_{i+1})$ holds true for all $i \in \{j-1, \dots, u-1\}$.*

Proof. Let $j \in \{l+1, \dots, u\}$ denote the index where $w(\mathcal{B}_{j-1}) < w(\mathcal{B}_j)$ holds true for the first time. If $j = u$, then there is nothing to show. So, let $j < u$. It suffices to prove that $w(\mathcal{B}_j) < w(\mathcal{B}_{j+1})$ holds true. From equation (3.1) it follows that $w(e_{j-1}, f_{j-1}) > 0$. Furthermore, Theorem 3.6 implies that

$$w(\mathcal{B}_{j+1}) - w(\mathcal{B}_j) = w(e_j, f_j) \geq w(e_{j-1}, f_{j-1}) > 0,$$

which shows that $w(\mathcal{B}_j) < w(\mathcal{B}_{j+1})$ is valid. \square

Note that the basis \mathcal{B}_j , where j is the index such that $w(e_j, f_j) > 0$ holds true for the first time, is lexicographically optimal w.r.t. w (with secondary optimization w.r.t. b). This means that \mathcal{B}_j is optimal w.r.t. w and additionally satisfies $b(\mathcal{B}_j) \leq b(\mathcal{B})$ for all $\mathcal{B} \in \mathcal{X}$ with $w(\mathcal{B}) = w(\mathcal{B}_j)$. A lexicographically optimal basis \mathcal{B}_j can be computed efficiently using a greedy algorithm, see Subsection 2.6, by computing an optimal basis w.r.t. the costs $\tilde{w}(e) = (r+1) \cdot w(e) + b(e)$ for all $e \in E$, where r is the rank of \mathcal{M} .

Theorem 3.7 induces a method that generates a minimal complete set \mathcal{X}_{cE} of efficient bases. Starting from a lexicographically optimal basis contained in \mathcal{S}_j , we compute a sequence of minimal swaps $\{(e_i, f_i)\}_{i=j}^{u-1}$, which is called *swap sequence* in the following. By construction, we have that each of the generated bases \mathcal{B}_i is contained in \mathcal{S}_i for all $i \in \{j+1, \dots, u\}$. The basis \mathcal{B}_j as well as all subsequently generated bases correspond to efficient solutions of problem (BBMP). Note that starting with basis \mathcal{B}_j rather than with \mathcal{B}_l has the advantage that all generated bases are efficient. For example, for the graphic matroid from Example 3.4 the basis \mathcal{B}_l corresponds to T_1 in Figure 3.2 while basis \mathcal{B}_j is given by T_2 . Therefore, one unnecessary swap is omitted. Nevertheless, in the worst case $\mathcal{B}_l = \mathcal{B}_j$ holds and all swaps have to be calculated.

Since the binary objective b decreases by one unit in each iteration of this procedure, it is ensured that no non-dominated outcome vector is missed in the objective space and hence $\mathcal{B}_j, \dots, \mathcal{B}_u$ form a minimal complete set of efficient bases. Hence, we have proven the following result:

Theorem 3.8. *Let $\{\mathcal{B}_i\}_{i=j}^u$ denote a sequence of bases generated by a swap sequence. Then $\mathcal{X}_{\text{cE}} = \{\mathcal{B}_j, \dots, \mathcal{B}_u\}$ forms a minimal complete set of efficient solutions and the non-dominated set is given by $\mathcal{Y}_{\text{N}} = \{(w(\mathcal{B}_i), b(\mathcal{B}_i)), i = j, \dots, u\}$.*

3.2.2. Connectedness

In the following we show that the set of efficient bases \mathcal{X}_{E} for problem (BBMP) is always connected. We recall from Section 2.5 that the set \mathcal{X}_{E} is said to be connected if its corresponding adjacency graph is connected. Recall also that two efficient bases of a matroid of rank r are called adjacent if they have $r - 1$ elements in common. Our proof is based on the fact that the set of *supported* efficient bases is always connected. For more details on this topic we refer to Ehrgott, 1996. In the following we show that every efficient basis of problem (BBMP) is a supported efficient solution which implies that the adjacency graph of the problem is always connected.

To do so, we first state a sufficient condition that guarantees that the non-dominated set of a general bi-objective combinatorial minimization problem only consists of supported non-dominated outcome vectors. Given the non-dominated set $\mathcal{Y}_{\text{N}} = \{z_1, \dots, z_n\} \subset \mathbb{R}^2$ of the problem, where $n \geq 3$ and $z_i = (x_i, y_i) \in \mathbb{R}^2$, with $x_1 < \dots < x_n$ and $y_1 > \dots > y_n$, we define the *sequence of slopes* $\{m_i\}_{i=1}^{n-1}$ of subsequent points of \mathcal{Y}_{N} by setting

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad i = 1, \dots, n - 1.$$

Note that $m_i \in (-\infty, 0)$ holds for all $i \in \{1, \dots, n - 1\}$.

Lemma 3.9. *Consider a bi-objective combinatorial minimization problem and suppose that the sequence of slopes $\{m_i\}_{i=1}^{n-1}$ is non-decreasing. Then all non-dominated outcome vectors in the set \mathcal{Y}_{N} are supported.*

Proof. Suppose that, to the contrary, there is a non-supported non-dominated outcome vector $z_t \in \mathcal{Y}_{\text{N}}$, $t \in \{2, \dots, n - 1\}$. Since a non-dominated outcome vector is supported if

and only if it is an element of the convex hull of \mathcal{Y} , it follows that there exist supported non-dominated outcome vectors $z_i, z_j \in \mathcal{Y}_N$ and a weight $\lambda \in (0, 1)$ 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 can not be an element of \mathcal{Y}_N since otherwise it would dominate z_t . 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, it follows 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

$$m^\star := \frac{y_j - y_1}{x_j - x_1} = \frac{y_\lambda - y_1}{x_\lambda - x_1} < \frac{y_2 - y_1}{x_2 - x_1} = m_1.$$

This is impossible, since by assumption $m_1 \leq m_i$ for all $i \in \{1, \dots, n\}$, and hence

$$\begin{aligned} y_j &= y_1 + \sum_{i=1}^{j-1} (y_{i+1} - y_i) = y_1 + \sum_{i=1}^{j-1} m_i \cdot (x_{i+1} - x_i) \\ &\geq y_1 + m_1 \cdot \sum_{i=1}^{j-1} (x_{i+1} - x_i) = y_1 + m_1 \cdot (x_j - x_1). \end{aligned}$$

Therefore, it has to hold that $m^\star \geq m_1$, which is a contradiction. \square

We combine the results of Theorem 3.7, Theorem 3.8 and Lemma 3.9 to conclude that all non-dominated outcome vectors of bi-objective optimization problems on matroids with one binary objective function are supported.

Lemma 3.10. *Consider a feasible instance of problem (BBMP), i.e., assume that $\mathcal{Y} \neq \emptyset$. Then the non-dominated set \mathcal{Y}_N consists only of supported non-dominated outcome vectors.*

Proof. Using the notation introduced in Section 3.3, we denote by $\{(e_i, f_i)\}_{i=j}^{u-1}$ a swap sequence starting from a lexicographically optimal basis \mathcal{B}_j that induces a set of efficient bases $\mathcal{B}_i \in \mathcal{S}_i$ for problem (BBMP), $i = j, \dots, u$. According to Theorem 3.8 we have that $\mathcal{Y}_N = \{(w(\mathcal{B}_i), b(\mathcal{B}_i)), i = j, \dots, u\}$.

Note that the result is trivial when $|\mathcal{Y}_N| \leq 2$. When $|\mathcal{Y}_N| \geq 3$ we know from Lemma 3.9 that it suffices to show that the sequence of slopes $\{m_i\}_{i=j}^{u-1}$, where

$$m_i = \frac{b(\mathcal{B}_{i+1}) - b(\mathcal{B}_i)}{w(\mathcal{B}_{i+1}) - w(\mathcal{B}_i)} = \frac{-1}{w(\mathcal{B}_{i+1}) - w(\mathcal{B}_i)}$$

is non-decreasing. Since in this case $|\mathcal{Y}_N| \geq 3$ we have that $j \leq u - 2$. For an arbitrary but fixed index $i \in \{j, \dots, u - 2\}$ it follows from Theorem 3.6 and Theorem 3.7 that

$$w(\mathcal{B}_{i+2}) - w(\mathcal{B}_{i+1}) = w(e_{i+1}, f_{i+1}) \geq w(e_i, f_i) = w(\mathcal{B}_{i+1}) - w(\mathcal{B}_i) > 0.$$

This implies that

$$m_{i+1} = \frac{-1}{w(\mathcal{B}_{i+2}) - w(\mathcal{B}_{i+1})} \geq \frac{-1}{w(\mathcal{B}_{i+1}) - w(\mathcal{B}_i)} = m_i,$$

and hence the sequence of slopes $\{m_i\}_{i=j}^{u-1}$ is non-decreasing. This implies that \mathcal{Y}_N contains only supported non-dominated outcome vectors. \square

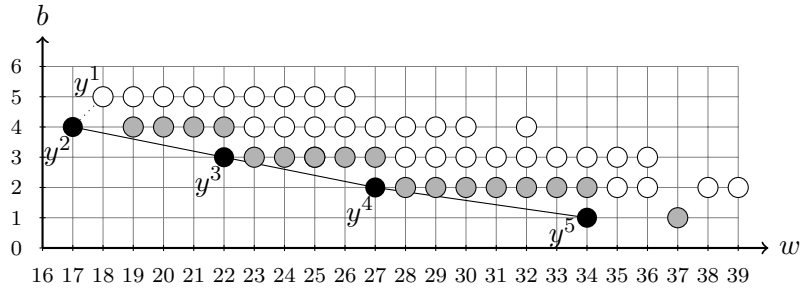


Figure 3.3.: Non-dominated set of the graphic matroid introduced in Figure 3.1. Non-dominated outcome vectors are shown in black and weakly non-dominated points are shown in grey. The remaining (white) points are all dominated outcome vectors. Note that the points y^2 , y^4 and y^5 are extreme supported non-dominated outcome vectors while y^3 is a non-extreme supported non-dominated outcome vector. The nodes y^i correspond to the trees T_i from Figure 3.2.

Note that not every supported non-dominated outcome vector must be extreme supported. Indeed, when the costs of two consecutive swaps (e_i, f_i) and (e_{i+1}, f_{i+1}) in a swap sequence are equal, then the point $(w(\mathcal{B}_{i+1}), b(\mathcal{B}_{i+1}))$ is not an extreme point of $\text{conv}(\mathcal{Y})$. To see this we consider again the graphic matroid introduced in Figure 3.1, c.f. Example 3.4. The set of feasible outcome vectors \mathcal{Y} in the objective space is shown in Figure 3.3. In this example, the supported non-dominated point y^3 (which is the image of the spanning tree T_3 from Figure 3.2) is not an extreme point of $\text{conv}(\mathcal{Y})$ and thus not extreme supported.

We finally conclude that the set of efficient bases is connected.

Theorem 3.11. *Consider a feasible instance of problem (BBMP). Then the set of efficient solutions \mathcal{X}_E is connected.*

Proof. Lemma 3.10 implies that all non-dominated outcome vectors in \mathcal{Y}_N are supported, and hence all efficient solutions in \mathcal{X}_E are supported. Using the fact that the (sub)graph of all supported efficient solutions is always connected (see Ehrgott, 1996) implies the result. \square

3.3. Efficient Swap Algorithm

The *Efficient Swap Algorithm* ESA presented in this section utilizes swap sequences to efficiently generate a minimal complete set of efficient solutions for problem (BBMP). Note that ESA can be interpreted as an extension of the algorithm stated in Gabow and Tarjan, 1984 for the solution of problem (BMP₌) for fixed ε . Indeed, it was shown in Gabow and Tarjan, 1984 that this algorithm generates a complete swap sequence $\{(e_i, f_i)\}_{i=l}^{\varepsilon-1}$ starting from $\mathcal{B}_l \in \mathcal{S}_l$ and leading to $\mathcal{B}_\varepsilon \in \mathcal{S}_\varepsilon$. Setting $\varepsilon = u$ and starting from a lexicographically optimal basis \mathcal{B}_j thus induces ESA, and hence we omit detailed proofs for the correctness

of this part of the algorithm. We rather focus on explaining how a complete swap sequence is generated without calculating a multiplicity of unnecessary swaps that do not lead to new efficient bases of the bi-objective problem (BBMP). At the end of this subsection we apply our algorithm to the graphic matroid from Example 3.4 to show how ESA works in practice.

We use the ideas from Theorem 3.2 and Corollary 3.3 to avoid the calculation of unnecessary swaps. In a first step, we generate bases $\mathcal{B}_j \in \mathcal{S}_j$ and $\mathcal{B}_u \in \mathcal{S}_u$ such that these two bases have as many elements as possible in common. The following two properties hold if and only if \mathcal{B}_j and \mathcal{B}_u coincide in a maximal number of elements:

- (a) $\mathcal{B}_j \cap E_0 \subseteq \mathcal{B}_u$, i.e., \mathcal{B}_u contains all green elements from \mathcal{B}_j .
- (b) $\mathcal{B}_u \cap E_1 \subseteq \mathcal{B}_j$, i.e., \mathcal{B}_j contains all red elements from \mathcal{B}_u .

Note that properties (a) and (b) imply that $U := \mathcal{B}_u \setminus \mathcal{B}_j \subseteq E_0$ and that $J := \mathcal{B}_j \setminus \mathcal{B}_u \subseteq E_1$, respectively, and that $|U| = |J|$.

If both properties (a) and (b) hold, then all elements of the matroid that are neither contained in \mathcal{B}_j nor in \mathcal{B}_u are redundant for ESA and can be removed from the ground set of the problem, i.e., we continue by considering the restriction $\mathcal{M} - (\mathcal{B}_j \cup \mathcal{B}_u)^c$. Furthermore, only those elements have to be swapped that are not contained in both bases simultaneously (see Gabow and Tarjan, 1984 for a detailed proof of this fact). This means that it is sufficient to consider the contraction of the matroid w.r.t. all elements that are contained in both bases. ESA works on this reduced problem $(\mathcal{M} - (\mathcal{B}_j \cup \mathcal{B}_u)^c) / (\mathcal{B}_j \cap \mathcal{B}_u)$ and uses a recursive swap sequence generation procedure (SSG) to generate a swap sequence. We illustrate the main aspects of this procedure in the following, assuming that \mathcal{B}_j and \mathcal{B}_u satisfy properties (a) and (b) above and that we start from \mathcal{B}_j .

If we add a green element f from $\mathcal{B}_u \setminus \mathcal{B}_j \subseteq E_0$ with minimal costs to \mathcal{B}_j , then a uniquely defined circuit $C(f, \mathcal{B}_j)$ is generated. Note that all elements of this circuit, with the only exception of f , are elements of \mathcal{B}_j . If these elements are all red, then a minimal swap (e^*, f) w.r.t. \mathcal{B}_j containing f , where $e^* \in C(f, \mathcal{B}_j) \setminus f$ and $w(e^*, f) \leq w(e, f)$ for all $e \in C(f, \mathcal{B}_j) \setminus f$, has to be contained in the swap sequence. The reason for this is that no other element of this circuit leads to a better swap than the swap (e, f) does, when f is added to \mathcal{B}_j . Otherwise, if the circuit $C(f, \mathcal{B}_j)$ contains red and green elements from \mathcal{B}_j , then a minimal swap w.r.t. \mathcal{B}_j containing f that is contained in a swap sequence cannot be deduced immediately. The idea in this case is to generate two smaller subproblems by contraction that do not intersect on the original ground set E . The reduction to two subproblems is repeated until adding f leads to a circuit with only red edges besides f .

As we explain in the following, problem splitting can be realised such that all swaps that are already guaranteed to be contained in a final swap sequence by the criterion given above are preserved (see Gabow and Tarjan, 1984 for further details). Moreover, the problem can be split until adding f leads to a circuit with only red edges besides f . Note that this is always satisfied when the respective ground sets of the contracted matroids consist of two elements $e \in \mathcal{B}_j$ and $f \in \mathcal{B}_u$ only. In this case, the swap (e, f) must be contained in a final swap sequence since this swap is minimal.

More formally, the split of the reduced matroid $(\mathcal{M} - (\mathcal{B}_j \cup \mathcal{B}_u)^c) / (\mathcal{B}_j \cap \mathcal{B}_u)$ into smaller parts is induced by a bisection of the sets $U = \mathcal{B}_u \setminus \mathcal{B}_j \subseteq E_0$ and $J = \mathcal{B}_j \setminus \mathcal{B}_u \subseteq E_1$. At first, the sets U and J are partitioned into two subsets U_1, U_2 and J_1, J_2 satisfying the

Algorithm 8: Efficient Swap Algorithm (ESA) for Bi-objective Matroid Problems with one Binary Cost Function

Input: An instance $((\mathcal{M}, \mathcal{X}, (c, b))$ of problem (BBMP).

Output: \mathcal{Y}_N and a complete set \mathcal{X}_{cE} of efficient solutions.

- 1 $\mathcal{X}_{cE} = \emptyset, \mathcal{Y}_N = \emptyset$.
 - 2 Determine a lexicographically optimal basis \mathcal{B}_j .
 - 3 Determine a minimum basis \mathcal{B}_u with respect to w such that \mathcal{B}_u contains a maximal number of elements from E_0 , all elements from $\mathcal{B}_j \cap E_0$ and only those elements from E_1 that are also contained in \mathcal{B}_j .
 - 4 Call $\text{SSG}((\mathcal{M} - (\mathcal{B}_j \cup \mathcal{B}_u)^c)/(\mathcal{B}_j \cap \mathcal{B}_u), \mathcal{B}_j \setminus \mathcal{B}_u, \mathcal{B}_u \setminus \mathcal{B}_j)$ to generate a swap sequence.
 - 5 Let $\{(e_i, f_i)\}_{i=j}^{u-1}$ denote the swap sequence found by Procedure SSG, where the swaps are sorted in non-decreasing order with respect to their costs.
 - 6 Set $\mathcal{B} = \mathcal{B}_j$, $\gamma = w(\mathcal{B}_j)$ and $\beta = b(\mathcal{B}_j)$.
 - 7 $\mathcal{X}_{cE} = \{\mathcal{B}\}$ and $\mathcal{Y}_N = \{(\gamma, \beta)\}$.
 - 8 **for** $i = j$ **to** $u - 1$ **do**
 - 9 Set $\mathcal{B} = \mathcal{B} - e_i + f_i$, $\gamma = \gamma + w(e_i, f_i)$ and $\beta = \beta - 1$.
 - 10 Set $\mathcal{X}_{cE} = \mathcal{X}_{cE} \cup \{\mathcal{B}\}$ and $\mathcal{Y}_N = \mathcal{Y}_N \cup \{(\gamma, \beta)\}$.
 - 11 **return** \mathcal{X}_{cE} and \mathcal{Y}_N .
-

following two conditions:

1. The set $U_1 \subseteq E_0$ consists of the $\lfloor |U|/2 \rfloor$ smallest elements of U with respect to w .
2. The set $\mathcal{B} = J_1 \cup U_1$ is a minimum basis for \mathcal{M} w.r.t. w satisfying $\mathcal{B} \cap E_0 = U_1$.

In a second step, the given problem is split into two different subproblems and the procedures $\text{SSG}((\mathcal{M} - U_2)/J_1, J_2, U_1)$ and $\text{SSG}((\mathcal{M} - J_2)/U_1, J_1, U_2)$ are executed.

Applying this procedure, it can be shown (cf. Gabow and Tarjan, 1984) that all involved matroid problems remain feasible and that all swaps contained in the final swap sequence are preserved. Furthermore, if a subproblem consists of exactly one red element $e \in J \subseteq (\mathcal{B}_j \setminus \mathcal{B}_u) \cap E_1$ and one green element $f \in U \subseteq (\mathcal{B}_u \setminus \mathcal{B}_j) \cap E_0$, it is guaranteed that the swap (e, f) is in the swap sequence.

The Efficient Swap Algorithm (ESA) for the solution of problem (BBMP) is summarized in Algorithm 8. The associated bisection procedure SSG that is recursively called during the course of ESA is outlined in Algorithm 9. At the beginning of Algorithm 8 the two bases \mathcal{B}_j and \mathcal{B}_u are calculated. Then, using Algorithm 9, a swap sequence for the (contracted) matroid $(\mathcal{M} - (\mathcal{B}_j \cup \mathcal{B}_u)^c)/(\mathcal{B}_j \cap \mathcal{B}_u)$ with ground set $(\mathcal{B}_j \cup \mathcal{B}_u) \setminus (\mathcal{B}_j \cap \mathcal{B}_u)$ is generated recursively. Finally, the generated swaps are sorted in non-decreasing order of their costs and, based on the result of Theorem 3.8, the non-dominated set \mathcal{Y}_N as well as a minimal complete set \mathcal{X}_{cE} of efficient solutions are determined.

Note that during the course of Algorithm 9 it may happen that swaps (or elements) with the same cost occur. So, a rule how to cope with ties in Line 4 of Algorithm 9 has to be given. We follow the approach suggested in Gabow and Tarjan, 1984: First assume that the elements of E_0 are sorted and indexed according to their costs w in non-decreasing

Algorithm 9: Swap Sequence Generation $\text{SSG}(\mathcal{M}, J, U)$ (Gabow and Tarjan, 1984)

Input: A matroid \mathcal{M} and two sets of elements $J \subseteq \mathcal{B}_j \setminus \mathcal{B}_u$ and $U \subseteq \mathcal{B}_u \setminus \mathcal{B}_j$,
 $|J| = |U|$.
Output: A minimal swap (e, f) or two recursive calls of the procedure SSG .

- 1 **if** $|U| = 1$ **then**
- 2 **return** the swap (e, f) , where $J = \{e\}$ and $U = \{f\}$.
- 3 **else**
- 4 Let U_1 be the set of $\lfloor |U|/2 \rfloor$ smallest elements with respect to w (contained in E_0) and set $U_2 = U \setminus U_1$.
- 5 Determine J_1 such that $\mathcal{B} = J_1 \cup U_1$ forms a minimal basis for \mathcal{M} with respect to w satisfying $\mathcal{B} \cap E_0 = U_1$ and set $J_2 = J \setminus J_1$.
- 6 Call $\text{SSG}((\mathcal{M} - U_2)/J_1, J_2, U_1)$ to find the swaps for the elements in U_1 .
- 7 Call $\text{SSG}((\mathcal{M} - J_2)/U_1, J_1, U_2)$ to find the swaps for the elements in U_2 .

order. Then, in Line 4 of Algorithm 9 we always choose the first $\lfloor |U|/2 \rfloor$ elements from U . When there are ties in the costs of the swap sequence, then the affected swaps are arranged in increasing order of the indices with respect to the elements that are contained in E_0 . The following theorem summarizes the results.

Theorem 3.12. *Algorithm 8 is correct and returns the non-dominated set and a minimal complete set of efficient solutions.*

Proof. The correctness of the algorithm follows from Theorem 3.8 and from the correctness of the algorithm for solving problem $(\text{BMP}_=)$ stated in Gabow and Tarjan, 1984. \square

Note that the complexity of Algorithm 8 depends on the considered matroid problem. For graphic matroids with $G = (V, E)$, for example, it is shown in Gabow and Tarjan, 1984 that their basic algorithm solves problem $(\text{BMP}_=)$ within $\mathcal{O}(m \log \log_{(2+m/n)} n + n \cdot \log(n))$ time, where $|V| = n$ and $|E| = m$. Hence, Algorithm 8 has the same time bound in this case, since the additional construction of \mathcal{X}_{cE} and \mathcal{Y}_N takes at most $\mathcal{O}(m)$ time. For a matching matroid and a transversal matroid the time bound is $\mathcal{O}(n \log n + m\ell)$, which follows again from a corresponding result in Gabow and Tarjan, 1984, where n is the number of vertices of a graph, ℓ is the number of edges in a maximum matching and m is the number of edges in the graph. Again, Algorithm 8 has the same time bound, since the additional construction of \mathcal{X}_{cE} and \mathcal{Y}_N takes at most $\mathcal{O}(\ell)$ time. Furthermore, it is proven in Gabow and Tarjan, 1984 that the problem $(\text{BMP}_=)$ can be solved in linear time, i.e., $\mathcal{O}(n)$, for a partition matroid (and therefore also for a uniform matroid) on a ground set E which consists of n elements. In this case the construction of \mathcal{X}_{cE} and \mathcal{Y}_N takes at most $\mathcal{O}(n)$ time and hence Algorithm 8 has the same time bound.

Example 3.13. *We apply ESA to the graphic matroid introduced in Example 3.4. To simplify the notation, the edges of the graph G (see Figure 3.1) are identified by their associated costs w rather than by their respective end nodes. This only induces ambiguity in the case of the edges $[2, 3]$ and $[2, 4]$ which both have cost 2, and in the case of the edges*

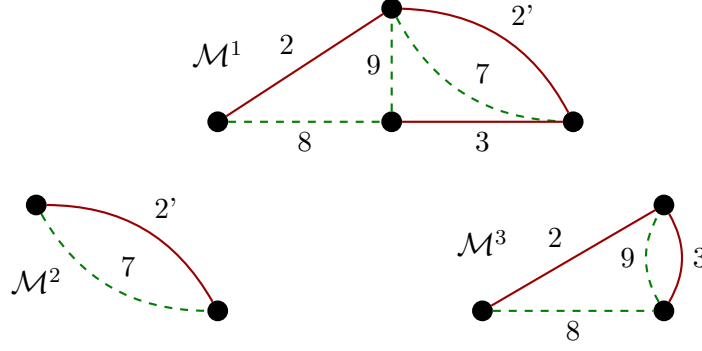


Figure 3.4.: Contracted graphic matroids \mathcal{M}^1 , \mathcal{M}^2 and \mathcal{M}^3 from Example 3.13. Solid red lines correspond to edges e with $b(e) = 1$ while the green dashed lines correspond to edges with $b(e) = 0$. The edges are identified by their associated cost value w , where ambiguities are resolved by using the notation 2 and 2' to refer to the edges $[2, 4]$ and $[2, 3]$, respectively.

$[1, 4]$ and $[3, 7]$ which both have cost 4. We refer to the edge $[2, 3]$ by writing 2' and to the edge $[1, 4]$ by writing 4' in the following to distinguish between these edges.

In a first step the optimal bases \mathcal{B}_j and \mathcal{B}_u are determined. This leads to the spanning trees T_2 and T_5 , respectively, shown in Figure 3.2, i.e., $\mathcal{B}_j = \{1, 2, 2', 3, 4, 5\}$ and $\mathcal{B}_u = \{1, 4, 5, 7, 8, 9\}$. Hence, $U = \mathcal{B}_u \setminus \mathcal{B}_j = \{7, 8, 9\} \subseteq E_0$, $J = \mathcal{B}_j \setminus \mathcal{B}_u = \{2, 2', 3\} \subseteq E_1$, and $\mathcal{B}_j \cap \mathcal{B}_u = \{1, 4, 5\}$. This implies that the edges 1, 4 and 5, i.e., the edges $[1, 2]$, $[3, 6]$ and $[3, 7]$, are contained in every efficient spanning tree in the set \mathcal{X}_{CE} generated by ESA, and the edges 4' and 6, i.e., the edges $[1, 4]$ and $[6, 7]$, can be removed from the problem since they are not contained in $\mathcal{B}_j \cup \mathcal{B}_u$. The contracted matroid $\mathcal{M}^1 := (\mathcal{M} - (\mathcal{B}_j \cup \mathcal{B}_u)^c) / (\mathcal{B}_j \cap \mathcal{B}_u)$ is shown in Figure 3.4. From now on, we enumerate (contracted) matroids and their respective subsets by superscripts, while referring to the corresponding subsets U_1, U_2, J_1, J_2 by subscripts, as before.

Then the procedure SSG is called with $\text{SSG}(\mathcal{M}^1, J^1, U^1)$, where $J^1 := J = \{2, 2', 3\}$ and $U^1 := U = \{7, 8, 9\}$. Since $|U^1| = 3 > 1$, the matroid \mathcal{M}^1 has to be split into two smaller matroids. We first determine the $\lfloor |U^1|/2 \rfloor$ smallest elements of U^1 as $U_1^1 = \{7\}$ and set $U_2^1 := U^1 \setminus U_1^1 = \{8, 9\}$. Now we determine J_1^1 such that $\mathcal{B}^1 = J_1^1 \cup U_1^1$ is a minimum basis for \mathcal{M}^1 with respect to w satisfying $\mathcal{B}^1 \cap E_0 = U_1^1$. This implies that $J_1^1 = \{2, 3\}$, $J_2^1 := J^1 \setminus J_1^1 = \{2'\}$ and $\mathcal{B}^1 = \{2, 3, 7\}$. Now the procedure SSG is called recursively with $\text{SSG}(\mathcal{M}^2, J^2, U^2)$ and $\text{SSG}(\mathcal{M}^3, J^3, U^3)$, respectively, where \mathcal{M}^2 and \mathcal{M}^3 correspond to the contracted matroids shown in Figure 3.4, and $J^2 = J_2^1$, $J^3 = J_1^1$, $U^2 = U_1^1$ and $U^3 := U^1 \setminus U_1^1 = \{8, 9\}$. $\text{SSG}(\mathcal{M}^2, J^2, U^2)$ returns immediately the swap $(2', 7)$ while $\text{SSG}(\mathcal{M}^3, J^3, U^3)$ needs another recursion to compute the swaps $(3, 8)$ and $(2, 9)$. Sorting these swaps in non-decreasing order of their costs leads to the swap sequence $\{(2', 7), (3, 8), (2, 9)\}$ with costs 5, 5, 7. This immediately leads to the final result $\mathcal{Y}_{\text{N}} = \{(17, 4), (22, 3), (27, 2), (34, 1)\}$ and $\mathcal{X}_{\text{CE}} = \{T_2, T_3, T_4, T_5\}$, see also Figure 3.2.

3.4. Numerical Results

The main advantage of ESA is its computational efficiency. In this section we present numerical results that validate this statement for the examples of graphic and uniform matroids. In addition we address the question whether, and if yes, how far the results on the connectedness of the efficient set can be extended to more general cases. Towards this end, we randomly generated instances of uniform matroids with more than two (integer) values for the coefficients in the second objective. For all instances we compute the complete efficient set \mathcal{X}_E and count the number of instances for which \mathcal{X}_E is non-connected.

We note that other generalizations have been investigated for specific matroids. Seipp, 2013, for example, analyzes uniform matroids with one general cost function and two binary cost functions in a tri-criteria model. They suggest an exact solution method that is, similar to ESA, based on neighborhood search. In contrast to ESA their algorithm may generate dominated solutions. Nevertheless, they show that a complete set of efficient solutions can be generated in polynomial time with this method and that the efficient set consists only of supported solutions and is thus connected.

3.4.1. Performance of the Efficient Swap Algorithm

In this section, we present numerical results on randomly generated instances of graphic matroids and of uniform matroids to validate the efficiency of ESA.

3.4.1.1. Graphic Matroids

For graphic matroids on undirected connected graphs $G = (V, E)$, i.e., for bi-objective minimum spanning tree problems with one general and one binary cost function, we evaluate the computational time needed by ESA to compute the non-dominated set \mathcal{Y}_N . To set this time in relation to the combinatorial complexity of the respective instances, we also provide the total number of feasible solutions, i.e., of spanning trees of the graph, and evaluate the time needed to determine all efficient trees from this set by total enumeration. This complete enumeration approach (CE) is implemented by using the matlab code by Hotz, 2016 for the generation of all spanning trees that is based on an algorithm described in Knuth, 2012, see Subsection 2.6. For a recent survey and numerical comparison of exact algorithms for general multi-objective minimum spanning tree problems we refer to Fernandes et al., 2020. Note that the problem could also be solved by $n = |V|$ restarts of the method of Gabow and Tarjan Gabow and Tarjan, 1984 with appropriately chosen constraints on the number of green edges. ESA avoids these restarts as well as the computation of dominated solutions by initializing the swap sequence with a lexicographically optimal basis. The induced savings depend on the considered instance and are most significant when the non-dominated set is rather small compared to $|V|$.

Note also that since ESA exploits the fact that the non-dominated set \mathcal{Y}_N solely consists of supported non-dominated outcome vectors when one of the objective functions has binary coefficients only (c.f. Lemma 3.10), a numerical comparison with general solvers for bi- and multi-objective minimum spanning tree problems is not meaningful. In two-phase methods, for example, the search for unsupported non-dominated outcome vectors

could be omitted, leading to an implementation that is somewhat similar to ESA. On the other hand, algorithms that generalize classical methods for the single-objective minimum spanning tree problem to the multi-objective case cannot be expected to be competitive with ESA since they generally enumerate far too many irrelevant trees.

The efficiency tests are run on a computer with an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz processor, 12MB Cache and 32 GB RAM. Both algorithms are implemented in MATLAB Version R2020a.

Recall from Section 3.1 that the objective values of the binary objective b can only take values between 0 and r , i.e., $b(\mathcal{B}) \in \{0, 1, \dots, r\}$ where r is the rank of the underlying matroid and \mathcal{B} is an arbitrary basis. As a consequence, we have that $|\mathcal{Y}_N| = \mathcal{O}(r)$. For an instance of the graphic matroid on a connected graph $G = (V, E)$ with n vertices and m edges, this implies that $b(T) \in \{0, 1, \dots, n-1\}$ for all spanning trees T of G . Note that due to the common notation that $|V| = n$ and $|E| = m$ for graphic matroids, the rank of a graphic matroid is thus $n-1$. The CE approach determines all efficient spanning trees by maintaining a list with n entries, one for each potential value of $b(T) \in \{0, 1, \dots, n-1\}$ that stores the currently best cost value $w(T)$ together with all corresponding trees that were enumerated so far.

Tables 3.1 and 3.2 summarizes the times needed to compute all non-dominated outcome vectors with ESA for instances with up to $n = 1000$ nodes and $m = 45\,000$ edges. To randomly generate connected graphs, we use a code from Schnepfer et al., 2021 that first constructs a random spanning tree for the required number of nodes and that afterwards adds randomly the remaining edges. For all instances, the cost coefficients of the first objective were uniformly distributed random integers between 1 and 50\,000 that were linearly transformed such that the smallest cost value is always equal to zero. For the second objective, the cost coefficients were uniformly distributed random integers from the set $\{0, 1\}$. To reduce the effect of fluctuations due to varying processor loads, all times are averaged over ten runs on the same instance. Despite the exponentially growing cardinality $|\mathcal{X}|$ of the feasible set, which was computed using Kirchhoff's matrix tree theorem (see, e.g., Merris, 1994) for instances up to $n = 100$, the computational time of ESA always remains below one minute. For the very large instances in Table 3.2 it was not possible to compute $|\mathcal{X}|$. The CE approach took even for smaller instances too long, denoted by $-$. In this case, we also do not know $|\mathcal{X}_E|$. It can be observed that both the number $|\mathcal{Y}_N|$ of non-dominated outcome vectors as well as the computational time needed by ESA grow mainly with n , and only marginally with m and with the number $|E_1|$ of red edges, i.e., which are in the set $E_1 = \{e \in E : b(e) = 1\}$.

The numerical results shown in Tables 3.1 and 3.2 confirm the expected efficiency of ESA. Indeed, since ESA computes the set of non-dominated outcome vectors \mathcal{Y}_N (which has at most n elements) rather than the set of all efficient solutions \mathcal{X}_E , the number of iterations of ESA is bounded by n . Moreover, each iteration requires a simple swap operation that can be implemented very efficiently.

Note that ESA generates only one pre-image, i.e., one feasible tree for each non-dominated outcome vector, while the number of efficient trees may be substantially larger. As an example, consider an instance where all edges have the same coefficients in both objectives. Then all spanning trees map to the same outcome vector in the objective space, i.e., $|\mathcal{Y}_N| = 1$, and are thus efficient, i.e., $|\mathcal{X}_E| = |\mathcal{X}|$. In order to test whether this

(n, m)	$ E_1 $	$ \mathcal{Y}_N $	$ \mathcal{X}_E $	$ \mathcal{X} $	ESA [s]	CE [s]
(7, 10)	2	1	1	76	0.065	0.010
(7, 10)	4	4	4	66	0.032	0.001
(7, 15)	8	2	2	1 615	0.010	0.013
(7, 15)	12	3	3	1 807	0.014	0.015
(7, 20)	8	4	4	12 005	0.019	0.080
(7, 20)	11	5	5	12 005	0.023	0.081
(7, 20)	12	5	5	12 005	0.022	0.081
(10, 20)	10	6	6	26 646	0.027	0.203
(10, 20)	11	5	5	21 560	0.023	0.167
(10, 20)	15	2	2	18 956	0.011	0.139
(10, 30)	15	3	3	$1.85 \cdot 10^6$	0.016	11.808
(10, 30)	17	2	2	$1.62 \cdot 10^6$	0.012	10.215
(10, 30)	20	5	5	$1.60 \cdot 10^6$	0.021	10.202
(10, 40)	17	6	6	$3.06 \cdot 10^7$	0.029	172.213
(10, 40)	18	7	7	$3.01 \cdot 10^7$	0.032	171.100
(10, 40)	20	3	3	$3.01 \cdot 10^7$	0.016	168.403
(15, 30)	13	5	5	$5.35 \cdot 10^6$	0.025	38.684
(15, 30)	15	5	5	$4.11 \cdot 10^6$	0.023	29.293
(15, 30)	16	5	5	$4.66 \cdot 10^6$	0.024	33.935
(15, 60)	27	5	-	$2.97 \cdot 10^{11}$	0.028	-
(15, 60)	28	7	-	$3.95 \cdot 10^{11}$	0.034	-
(15, 60)	34	10	-	$2.86 \cdot 10^{11}$	0.034	-
(15, 100)	46	6	-	$9.46 \cdot 10^{14}$	0.034	-
(15, 100)	48	8	-	$9.35 \cdot 10^{14}$	0.040	-
(15, 100)	51	7	-	$9.35 \cdot 10^{14}$	0.036	-
(20, 40)	19	7	-	$1.18 \cdot 10^9$	0.034	8 663.920
(20, 40)	20	10	-	$7.42 \cdot 10^8$	0.044	5 419.526
(20, 100)	47	12	-	$5.76 \cdot 10^{17}$	0.058	-
(20, 100)	48	13	-	$4.43 \cdot 10^{17}$	0.062	-
(20, 100)	52	13	-	$4.15 \cdot 10^{17}$	0.059	-
(20, 180)	83	7	-	$8.91 \cdot 10^{22}$	0.047	-
(20, 180)	84	10	-	$8.89 \cdot 10^{22}$	0.057	-
(20, 180)	103	11	-	$8.94 \cdot 10^{22}$	0.057	-
(100, 200)	93	36	-	$4.94 \cdot 10^{44}$	0.198	-
(100, 200)	101	32	-	$5.06 \cdot 10^{45}$	0.179	-
(100, 200)	102	36	-	$1.06 \cdot 10^{45}$	0.193	-
(100, 1 000)	476	39	-	$2.20 \cdot 10^{125}$	0.296	-
(100, 1 000)	488	48	-	$5.74 \cdot 10^{125}$	0.341	-
(100, 1 000)	494	50	-	$2.14 \cdot 10^{125}$	0.345	-
(100, 2 000)	981	48	-	$3.00 \cdot 10^{156}$	0.452	-
(100, 2 000)	988	43	-	$3.00 \cdot 10^{156}$	0.427	-
(100, 2 000)	1 047	52	-	$2.32 \cdot 10^{156}$	0.457	-
(100, 4 000)	1 960	49	-	$5.39 \cdot 10^{186}$	0.700	-
(100, 4 000)	1 976	46	-	$5.43 \cdot 10^{186}$	0.681	-
(100, 4 000)	1 998	55	-	$5.53 \cdot 10^{186}$	0.723	-

Table 3.1.: Computational results for randomly generated graphs with n vertices, m edges, and $|E_1|$ edges with cost 1 in the binary cost function b . The last two columns give the time in seconds for ESA and for CE, respectively.

(n, m)	$ E_1 $	$ \mathcal{Y}_N $	ESA [s]
(1 000, 2 000)	970	355	3.695
(1 000, 2 000)	976	320	3.445
(1 000, 2 000)	1 008	353	3.695
(1 000, 15 000)	7 419	489	8.101
(1 000, 15 000)	7 441	478	7.999
(1 000, 15 000)	7 541	511	8.325
(1 000, 30 000)	14 894	494	12.805
(1 000, 30 000)	14 947	482	12.416
(1 000, 30 000)	14 988	510	12.708
(1 000, 45 000)	22 430	470	17.375
(1 000, 45 000)	22 548	514	17.996
(1 000, 45 000)	22 632	517	18.011

Table 3.2.: Computational results for randomly generated graphs with n vertices, m edges, and $|E_1|$ edges with cost 1 in the binary cost function b . The last column give the time in seconds for ESA.

is a common situation also in randomly generated instances, we computed the complete set \mathcal{X}_E with the CE approach for the smaller instances from Table 3.1. It turns out that this is not the case for randomly generated instances on small graphs with a rather large range for the objective coefficients.

3.4.1.2. Uniform matroids

As a second test case we consider uniform matroids $\mathcal{U}_{r,n}$ on the ground set $E = \{e_1, \dots, e_n\}$, from which exactly r elements have to be selected in a basis. Rather than minimizing the cost of a basis we aim at maximizing its profit w.r.t. one general and one binary cost function to reflect the similarity of this problem to bi-objective knapsack problems with bounded cardinality.

To determine the profit vectors of each element $e_i \in E$, we generated n uniformly distributed random values from the set $\{0, 1, \dots, 10n\}$ (for the first objective) and n values from the set $\{0, 1\}$ (for the second objective). After sorting the values for the first objective in non-decreasing order and the values of the binary objective in non-increasing order, the coefficients were combined into profit vectors for the elements e_1, \dots, e_n .

For each instance on n elements, Table 3.3 shows the accumulated results over all values of $r \in \{1, \dots, \frac{n}{2}\}$. In order to analyse the relation between $|\mathcal{Y}_N|$ and $|\mathcal{X}_E|$, we applied a simple implementation of a dynamic programming algorithm (DP) for multi-objective knapsack problems as described, for example, in Klamroth and Wiecek, 2000 and in Section 2.9. Different from the bi-objective minimum spanning tree instances described above, we consistently observe that the number of efficient solutions exceeds the number of non-dominated outcome vectors, however, not by very much. As was to be expected, ESA easily solves larger instances within fractions of a second, while the computational time required by DP grows significantly with the size of the instance. The efficiency tests are run on a computer with an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz processor and 8 GB RAM. The algorithms are implemented in MATLAB, Version R2019b.

n	$ E_1 $	$ \mathcal{Y}_N $	$ \mathcal{X}_E $	ESA [s]	DP [s]
20	9	64	83	0.02	0.08
20	10	65	65	0.02	0.06
20	11	64	74	0.03	0.06
30	12	129	129	0.03	0.44
30	15	135	166	0.03	0.50
30	16	134	134	0.02	0.30
50	21	340	364	0.06	35.33
50	21	340	340	0.07	27.29
50	29	340	413	0.06	14.43
60	27	489	523	0.09	131.83
60	30	495	627	0.09	130.62
60	32	492	502	0.09	80.36
70	34	664	690	0.12	327.33
70	35	665	716	0.12	336.23
70	38	659	705	0.12	248.97
80	38	857	935	0.21	> 600
80	44	850	899	0.27	> 600
80	50	805	860	0.21	> 600
90	45	1 080	1 181	0.20	> 600
90	46	1 079	1 213	0.21	> 600
90	47	1 077	1 159	0.19	> 600
100	46	1 315	1 361	0.25	> 600
100	50	1 325	1 483	0.46	> 600
100	51	1 324	1 441	0.27	> 600

Table 3.3.: Computational results for randomly generated instances of uniform matroids $\mathcal{U}_{r,n}$. The two last columns show the accumulated average computation time over all $r = 1, \dots, n/2$ in seconds, rounded over 10 repetitions for each instance, for ESA (for the computation of \mathcal{Y}_N) and for DP (for the computation of \mathcal{X}_E), respectively. For instances with 80 or more elements DP needs more than 600 seconds.

\hat{b}	2	3	4	5	6	7	8	9	11	13	15	20	25	30
nc-instances	0	0	0	0	0	0	2	4	3	2	8	10	12	13

Table 3.4.: Number of observed nc-instances in 300 000 randomly generated instances of $\mathcal{U}_{r,20}$, cumulated for all $r \in \{1, \dots, 10\}$, for different values of \hat{b} .

3.4.2. Connectedness for more General Cost Functions

The proof of the connectedness of the efficient set of problem (BBMP) (c.f. Theorem 3.11) relies on two basic properties: On one hand, this is the matroid structure of the considered problem, and on the other hand it is the fact that one of the two objective functions has only binary cost coefficients. While the first property ensures the feasibility of elementary swap operations, the latter implies that two adjacent non-dominated outcome vectors always differ by exactly one unit in the binary objective function.

In general, i.e., when the objective coefficients can be chosen freely, bi-objective optimization problems on uniform matroids may have non-connected efficient sets. Corresponding examples are provided in Gorski et al., 2011 indicating that such non-connected instances (*nc-instances*) are very rare in randomly generated instances. The question remains whether non-connected instances already exist when the cost coefficients in the second objective are restricted to $\{0, 1, 2\}$ (rather than $\{0, 1\}$), or, more generally, to $\{0, 1, \dots, \hat{b}\}$ with $\hat{b} \geq 2$.

The frequency in which nc-instances occurred for different values of \hat{b} in a large numerical study are reported in Table 3.4. For each value of $\hat{b} \in \{2, \dots, 9, 11, 13, 15, 20, 25, 30\}$ we randomly generated 300 000 instances of uniform matroids $\mathcal{U}_{r,n}$ with $n = 20$ elements. The profit vectors were chosen as described in Section 3.4.1 above, where the coefficients for the second objective were now drawn from the set $\{0, 1, \dots, \hat{b}\}$. All instances were solved for all $r \in \{1, \dots, \frac{n}{2}\}$ using a DP approach for multi-objective knapsack problems, see Klamroth and Wiecek, 2000.

Table 3.4 indicates that it seems to become more likely to find nc-instances the larger the range for the coefficients in the second objective function is, i.e., the larger the value of \hat{b} is. Nevertheless, we suspect that nc-instances also exist for smaller values of \hat{b} but that such instances are extremely rare. While nc-instances may be more likely for larger values of n , analysing large data sets becomes more and more challenging since this requires the exact computation of the complete efficient set for each instance (without the possibility of using ESA). We note that preliminary tests with $n = 30$ and $n = 50$ did not provide further insight on this topic.

An other approach was to look after an example where the efficient set is not connected by using additionally to the first arbitrary objective function more than one binary objective function. So we first investigate in the case of two binary objective functions and an arbitrary one. Therefore, we generated for $n = 20$ 8 000 000, for $n = 50$ 600 000, for $n = 80$ 180 000 and for $n = 100$ 60 000 instances but we did not find an example. In the case of three binary objective functions and an arbitrary objective function we generated for $n = 20$ 500 000, for $n = 50$ 50 000 and for $n = 80$ 5 000 instances and did not find an example. It seems that there are no such examples, so it would be interesting to try to

e	$w(e)$	$b(e)$	\mathcal{B}	$w(\mathcal{B})$	$b(\mathcal{B})$
e_1	6	0	$\{e_1, e_2, e_3\}$	13	0
e_2	5	0	$\{e_1, e_2, e_4\}$	13	1
e_3	2	0	$\{e_1, e_2, e_5\}$	13	2
e_4	2	1	$\{e_1, e_4, e_5\}$	10	3
e_5	2	2	$\{e_1, e_5, e_6\}$	8	4
e_6	0	2	$\{e_4, e_5, e_6\}$	4	5

Table 3.5.: Left: Elements e_i , $i = 1, \dots, r$, of a knapsack $\mathcal{U}_{3,6}$ with their weights with $\hat{b} = 2$. Right: Weakly efficient bases for the knapsack.

find a proof for this theory in further research.

A necessary condition for the existence of nc-instances is the existence of non-dominated non-supported outcome vectors. But in contrast to nc-instances it is quite easy to generate knapsack problems with such outcome vectors. An example is given in Table 3.5. In the first column of the left table the number of an item, in the second column the first weight and in the third column the second weight with $\hat{b} = 2$ is given. In the right table, the efficient bases for $r = 3$ are given with their outcome vectors. Recall that we interpret the uniform matroid as a special case of a knapsack problem with bounded cardinality and thus consider both objective functions as maximization objectives. As can be seen in Figure 3.5, the basis $\{e_1, e_4, e_5\}$ is non-dominated and non-supported. Nevertheless, the efficient set of this instance is connected.

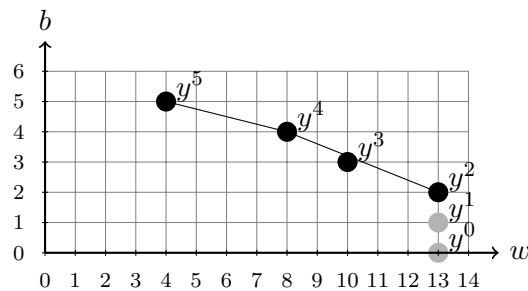


Figure 3.5.: Non-dominated set of the uniform matroid introduced in Table 3.5. Non-dominated outcome vectors are shown in black and weakly non-dominated points shown in grey. The point y^3 corresponds to a non-dominated and non-supported outcome vector.

3.5. Conclusion and Further Ideas

In this chapter we investigate bi-objective matroid problems involving one binary cost objective. We present the Efficient Swap Algorithm (ESA) that solves this special kind of bi-objective matroid problem in polynomial time, although the decision problem of the general version of this problem is known to be \mathcal{NP} -complete (cf. Ehrgott, 1996). The

idea of ESA is based on a method of Gabow and Tarjan, 1984 for a constrained version of single-objective matroid optimization problems. The complexity of ESA depends on the matroid type. For a graphic matroid on a graph $G = (V, E)$, for example, it is given by $\mathcal{O}(m \log \log_{(2+m/n)} n + n \cdot \log(n))$, where $|V| = n$ and $|E| = m$. Numerical experiments confirm the efficiency of this approach.

The efficient swap algorithm can be interpreted as a neighborhood search approach with an efficient strategy for the identification of relevant swaps. The correctness of this approach is based on the proof of the connectedness of the efficient set in this special case, which is in turn based on the insight that the non-dominated set consists only of supported non-dominated outcome vectors. This is surprising since it was shown in Gorski, 2010 that the efficient set of bi-objective matroid problems is in general non-connected. To the best of our knowledge this is the first class of problems where connectedness of \mathcal{X}_E can be established even though the non-dominated set is not contained in a hyperplane.

In future research we should investigate further, whether there exist non-connected efficient sets of matroid optimization problems with one sum objective function and either several bi-objective function or with a second objective function with non-negative integer costs, which are smaller or equal to 8 for each element of the ground set.

The binary objective function can be seen as a special case of an ordinal objective function. For an ordinal objective function we identify every element with one of K ordered categories. This is equivalent to the binary objective function if $K = 2$ categories are considered. Hence, we investigate in the next chapter matroid optimization problems with ordinal costs and introduce an efficient solution strategy even for this more general case.

4. Single- and Multi-objective Matroid Optimization Problems with Ordinal Costs

In this chapter, we consider multi-objective matroid optimization problems where, in addition to one sum objective function (with arbitrary non-negative integer coefficients), one or more ordinal objective functions are to be considered. This problem can be interpreted as a generalization of the bi-objective matroid optimization problem with binary costs, which was the topic of Chapter 3.



Figure 4.1.: Illustration of different edge or road categories for building new power lines or telephone cables.

Ordinal coefficients occur whenever there is no numerical value to reflect the quality or cost of an element. Consider, for example, the minimum spanning tree problem where we aim at finding connected networks with small overall costs. In addition to the (non-negative integer-valued and additive) length of an edge that directly represents the cost of, e.g., a telephone cable, the construction may involve major work that affects road traffic or even public transportation systems like tramways or trains. Roads that have already a cable canal allow for a cheap and easy addition of another cable. But if it is necessary to build a new cable canal under a street, this could lead to perturbations of the traffic or even affect public transportation systems. In this situation, it is useful to categorize each possible connection as “easy to build” (good), “leads to little problems with traffic jams” (medium), and “leads to major problems for public transportation” (bad), for the time of construction. For an illustration of these categories see Figure 4.1. It is then hard to compare, for example, a solution with two medium edges with a solution with one bad edge, since in general these categories can not be translated into monetary values.

Ordinal weights and ordinal objective functions have been, for example, introduced in Schäfer et al., 2020 for shortest path problems. Motivated by applications in civil security, edges are categorized, for example, as “secure”, “neutral”, or “insecure”. Schäfer et al., 2020 introduce an ordinal preorder based on ordinal weights, analyze the complexity of

the problem, and suggest a polynomial time labeling algorithm for its solution. Knapsack problems with ordinal weights are analyzed in Schäfer et al., 2021. They consider a general vector dominance and two lexicographic dominance concepts and suggest a dynamic programming based solution strategy and efficient greedy methods, respectively. Moreover, an outlook to multi-objective versions of ordinal problems is provided.

In this chapter we extend this concept by considering multi-objective problems that combine one “classical” sum objective function with possibly several additional ordinal objectives. We focus on multi-objective optimization problems with ordinal weights on matroids and relate the multi-objective formulation to a series of single-objective optimization problems on intersections of matroids. The latter can be efficiently solved by an algorithm from Edmonds, 2003. For the special case of bi-objective problems and only two ordinal categories, we compare this approach to the Efficient Swap Algorithm (ESA) presented in Chapter 3 which is even more efficient in this case since it exploits the specific problem structure of two ordinal categories (i.e., 0 and 1) more efficiently.

Contribution. The results of this chapter are published in Klamroth et al., 2022a. The algorithm we present in the following uses repeatedly the Matroid Intersection Algorithm from Edmonds, see Edmonds, 2003 and Section 2.8.

Organization of the chapter. Ordinal matroid optimization problems with only one ordinal objective function are discussed in Section 4.1. We show that ordinal matroid optimization problems can be solved by a greedy strategy that exploits their specific structure. Bi-objective matroid optimization problems with one sum objective and one ordinal objective are introduced in Section 4.2. Their relation to matroid intersection problems is analyzed in Section 4.3, yielding efficient polynomial time solution strategies for all considered problem variants. In Section 4.4 the results are extended to matroid optimization problems with several ordinal objective functions. The algorithms are numerically tested and compared at randomly generated instances of graphic matroids and of partition matroids in Section 4.5, and the chapter is concluded with a short outlook on future research topics in Section 4.6.

4.1. Single-objective Matroid Optimization with Ordinal Costs

First, we introduce an ordinal optimality concept that can be applied, whenever all solutions have the same cardinality. Hence, we assume that all solutions in the feasible set X_r have the same cardinality, i. e. $|x| = r \in \mathbb{Z}_>$ for all $x \in X_r$. This is, for example, satisfied in the case of a matroid optimization problem. Then an *ordinal optimization problem with feasible solutions of length r* (OOP_r) can be formulated as

$$\begin{aligned} \text{“ordinally minimize”} \quad & o(x) \\ \text{s. t.} \quad & x \in X_r. \end{aligned} \tag{\text{OOP}_r}$$

Intuitively, the ordinal objective function o assigns one out of K ordered categories to each element of the ground set E , and hence the ordinal objective of x is given by an r -dimensional ordinal vector. For example, in the case $K = 3$ we may think of good (green),

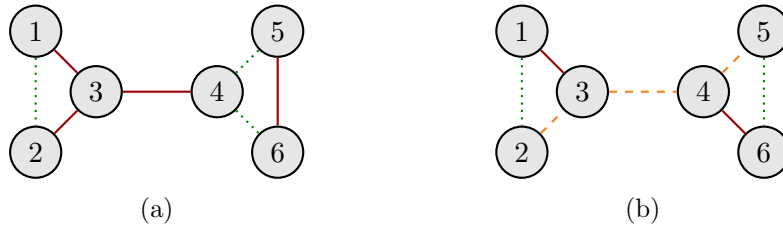


Figure 4.2.: The edges of the graph in (a) are categorized w.r.t. two categories (green-dotted and red-solid), c.f. Figure 2.17(a) above, while the edges of the same graph are categorized w.r.t. three categories in (b) (green-dotted, orange-dashed, and red-solid).

medium (orange) and bad (red) elements, where we prefer good over medium and medium over bad. Figure 4.2 shows two examples of the ground set E of a graphic matroid. While the edges in Figure 4.2(a) are assigned to only two categories (where green-dotted is better than red-solid), Figure 4.2(b) shows an example with three categories (where green-dotted is better than orange-dashed which is again better than red-solid).

Throughout this chapter we assume that the components of the ordinal vectors of feasible solutions are sorted in non-decreasing order w.r.t. the quality of the respective categories, see Figure 4.3 for an illustration. This sorting is useful when comparing different solutions in the following.

More formally, let $\mathcal{C} = \{\eta_1, \dots, \eta_K\}$ be an ordinal space consisting of K ordered categories, and let $o : E \rightarrow \mathcal{C}$ assign one ordinal category to each element of the ground set E . Moreover, (by slightly abusing the notation) let $o : X_r \rightarrow \mathcal{C}^r$ be a function mapping each feasible solution to an r -dimensional ordinal vector. We assume that category η_i with $i \in \{1, \dots, K-1\}$ is strictly preferred over all categories η_j with $i < j$, which is denoted by $\eta_i \prec \eta_j$. Similarly, we write $\eta_i \preceq \eta_j$ whenever $i \leq j$. Moreover, the components of the objective vector $o(x)$ of a feasible solution $x \in X_r$ are sorted in non-decreasing order, which is denoted by $o(x) := \text{sort}(o(x_1), \dots, o(x_r))$.

In order to define meaningful optimality concepts for problem (OOP_r) , we need to compare ordinal vectors in \mathcal{C}^r . The following definition is based on the concept first introduced in Schäfer et al., 2020. Let $y', \hat{y} \in \mathcal{C}^r$ be two ordinal vectors. Then we write

$$\begin{aligned} y' \preceq_o \hat{y} &: \iff y'_i \preceq \hat{y}_i, \quad i = 1, \dots, r, \\ y' \preceq_o \hat{y} &: \iff y'_i \preceq \hat{y}_i, \quad i = 1, \dots, r \text{ and } y' \neq \hat{y}, \\ y' \prec_o \hat{y} &: \iff y'_i \prec \hat{y}_i, \quad i = 1, \dots, r. \end{aligned}$$

When $y' = o(x')$ and $\hat{y} = o(\hat{x})$ are outcome vectors of problem (OOP_r) , then their components are sorted in non-decreasing order.

Definition 4.1. *Let $x', \hat{x} \in X_r$ and $y' = o(x')$ and $\hat{y} = o(\hat{x})$ outcome vectors of problem (OOP_r) , i.e., their components are sorted in non-decreasing order. We say that y' ordinally dominates \hat{y} whenever $y' \preceq_o \hat{y}$.*

We write “ \min_{\preceq_o} ”, for example, instead of “ordinally minimize” in problem (OOP_r) , to clarify that this optimality concept is used.

Note that what we consider here is a special case of the concept of ordinal dominance introduced in Schäfer et al., 2020 who considered the more general case when feasible solutions – and hence their outcome vectors – may differ w.r.t. their number of elements. Schäfer et al., 2020 showed that in this more general case, the binary relation \preceq_o is a preorder on the set of sorted outcome vectors of an ordinal optimization problem, i.e., it is reflexive and transitive. In the special case when all feasible solutions have the same number of elements, as considered in this paper, the binary relation \preceq_o is also antisymmetric and thus a partial order. See also Schäfer et al., 2021 for yet another perspective on ordinal efficiency for solutions with arbitrary length, which we investigate later in Chapter 5. If we restrict the definition of ordinal optimality by Schäfer et al., 2021, see Definition 5.2, to solutions with the same length, then these concepts are equivalent, see Theorem 5.13. Moreover, the binary relation \preceq_o is a strict partial order in our case, i.e., it is irreflexive, transitive and asymmetric. Thus, the concepts of (weak) ordinal efficiency and (weak) ordinal dominance can be defined in a similar way as for the case of Pareto optimality, see Section 2.2, by replacing \leq with \preceq_o and $<$ with \prec_o , respectively.

As a first step towards multi-objective ordinal optimization problems we investigate matroid optimization problems with only one ordinal objective function and show that such problems can be solved using a greedy algorithm. In slight abuse of the standard notation, we refer to the resulting problems as “single-objective optimization problems”, even though their objective functions are vector-valued. Similar results were obtained by Schäfer et al., 2020 and Schäfer et al., 2021 for shortest path and knapsack problems, respectively, however, for the case that a lexicographic optimization is employed on the ordinal outcome vectors. We show in the following that in the case of matroids, ordinal optimality actually coincides with lexicographic optimality, and hence a greedy algorithm always yields the ordinally non-dominated set in this case.

4.1.1. Ordinal and Lexicographic Optimality and their Interrelation

Let a matroid $\mathcal{M}_1 = (E, \mathcal{I}_1)$ with rank r be given and denote the set of its bases by \mathcal{X}_1 . As a first special case, consider the situation of only two categories, i.e., $K = 2$. W.l.o.g. we set $\eta_1 = 0$ (green) and $\eta_2 = 1$ (red). It is easy to see that in this case problem (OOP_r) is equivalent to a matroid optimization problem with a “classical” sum objective function with binary coefficients $b : E \rightarrow \{0, 1\}$ where the cost of a basis \mathcal{B} is the aggregated cost of all of its elements, i.e. $b(\mathcal{B}) := \sum_{e \in \mathcal{B}} b(e)$. Indeed, a basis $\mathcal{B}_1 \in \mathcal{X}_1$ ordinally dominates a basis $\mathcal{B}_2 \in \mathcal{X}_1$ whenever the number of one-entries in $o(\mathcal{B}_1)$ (i.e., red elements in \mathcal{B}_1) is smaller than that in $o(\mathcal{B}_2)$. This leads to a *matroid problem with a binary objective function* (BMP)

$$\begin{aligned} \min \quad & b(\mathcal{B}) \\ \text{s. t.} \quad & \mathcal{B} \in \mathcal{X}_1 \end{aligned} \tag{BMP}$$

as a particularly simple special case of problem (OOP_r) . When $K > 2$, i.e., when more than two ordinal categories have to be considered, a simple aggregation of all categories into one single aggregated objective value is no longer meaningful. However, we discuss two related optimization problems that are based on partial aggregation in the following. Towards this end, let problem (OOP_r) for the special case of matroid optimization be

defined as a *matroid problem with ordinal costs*, given by

$$\begin{aligned} \min_{\preceq_o} \quad & o(\mathcal{B}) \\ \text{s. t.} \quad & \mathcal{B} \in \mathcal{X}_1. \end{aligned} \tag{MPO}$$

Now consider a feasible basis $\mathcal{B} \in \mathcal{X}_1$. Then the information contained in the objective vector $o(\mathcal{B}) \in \mathcal{C}^r$ can equivalently be stored in an aggregated vector $c(\mathcal{B}) \in \mathbb{Z}_{\geq}^K$ with components $c_j(\mathcal{B}) := |\{e \in \mathcal{B}: o(e) = \eta_j\}|$ for $j = 1, \dots, K$ that count the number of elements in each category in \mathcal{B} . Indeed, there is a simple one-to-one correspondence between $o(\mathcal{B})$ and $c(\mathcal{B})$. We refer to c as a *counting objective function* in the following. This representation is often advantageous since in general K , i. e., the number of categories, is constant and much smaller than the dimension r , i. e., the number of elements in a basis. Note that since all bases have the same number of elements we have that $\sum_{j=1}^K c_j(\mathcal{B}) = r$, and hence all outcome vectors $c(\mathcal{B})$, $\mathcal{B} \in \mathcal{X}_1$, lie on the same hyperplane in \mathbb{R}^K . Moreover, one of the components of c can be omitted without losing any information.

This reformulation suggests two related lexicographic optimization problems: On the one hand, we may aim at lexicographically maximizing the number of elements in the “good” categories, and on the other hand, we may want to lexicographically minimize the number of elements in the “bad” categories. In order to clearly distinguish between these two optimization goals, we introduce two separate variants of the counting objective c denoted as c^{\max} and c^{\min} , respectively.

Maximizing the Number of Good Elements When aiming at the maximization of the number of elements in *good* categories, we can apply a lexicographic *maximization* to the counting objective c . Thus, in this case we set $c^{\max}(\mathcal{B}) := c(\mathcal{B})$ for $\mathcal{B} \in \mathcal{X}_1$ with $c_j^{\max}(\mathcal{B}) := |\{e \in \mathcal{B}: o(e) = \eta_j\}|$ for $j = 1, \dots, K$ as defined above, and formulate problem (MPCmax) as

$$\begin{aligned} \text{lexmax} \quad & c^{\max}(\mathcal{B}) \\ \text{s. t.} \quad & \mathcal{B} \in \mathcal{X}_1. \end{aligned} \tag{MPCmax}$$

Minimizing the Number of Bad Elements In order to lexicographically minimize the number of elements in *bad* categories, we first have to bring the corresponding entries of the counting objective c that represent the bad categories into the leading positions (which are considered first in lexicographic optimization). We hence define $c_j^{\min}(\mathcal{B}) := c_{K-j+1}(\mathcal{B})$ for $j = 1, \dots, K$ and for $\mathcal{B} \in \mathcal{X}_1$, i. e., $c_j^{\min}(\mathcal{B}) := |\{e \in \mathcal{B}: o(e) = \eta_{K-j+1}\}|$, and consider problem (MPCmin) given by

$$\begin{aligned} \text{lexmin} \quad & c^{\min}(\mathcal{B}) \\ \text{s. t.} \quad & \mathcal{B} \in \mathcal{X}_1. \end{aligned} \tag{MPCmin}$$

Figure 4.3 shows an example of a graphic matroid with all of its feasible bases and their respective objective vectors o , c^{\max} and c^{\min} , see also Example 4.7 below.

4.1.2. Interrelation Between (MPO), (MPCmin) and (MPCmax)

In general, the ordinally non-dominated set of problem (MPO) is different from the sets of lexicographically optimal outcome vectors of the associated formulations (MPCmin) and

(MPCmax), respectively. This can be seen, for example, at the cases of ordinal shortest path problems (see Schäfer et al., 2020) and ordinal knapsack problems (see Schäfer et al., 2021). In the special case of matroids, however, these three concepts are closely related and their respective efficient and non-dominated sets coincide.

Theorem 4.2. *Let $\mathcal{M}_1 = (E, \mathcal{I}_1)$ be a matroid, let \mathcal{X}_1 denote the set of bases of \mathcal{M}_1 , and let the functions o , c^{\min} and c^{\max} be given and defined as above. Moreover, let $\mathcal{B}_1, \mathcal{B}_2 \in \mathcal{X}_1$ be two bases of \mathcal{M}_1 . Then*

$$(o(\mathcal{B}_1) \preceq_o o(\mathcal{B}_2)) \Rightarrow (c^{\min}(\mathcal{B}_1) <_{\text{lex}} c^{\min}(\mathcal{B}_2) \text{ and } c^{\max}(\mathcal{B}_1) >_{\text{lex}} c^{\max}(\mathcal{B}_2)),$$

i. e., if $o(\mathcal{B}_1)$ ordinally dominates $o(\mathcal{B}_2)$, then $c^{\min}(\mathcal{B}_1)$ lexicographically dominates $c^{\min}(\mathcal{B}_2)$ and $c^{\max}(\mathcal{B}_1)$ lexicographically dominates $c^{\max}(\mathcal{B}_2)$.

Proof. We prove the result for c^{\min} . The corresponding result for c^{\max} follows analogously, noting that (MPCmin) involves lexicographic minimization while (MPCmax) involves lexicographic maximization.

Now let $o(\mathcal{B}_1) \preceq_o o(\mathcal{B}_2)$ and assume that $c^{\min}(\mathcal{B}_1)$ does not lexicographically dominate $c^{\min}(\mathcal{B}_2)$. First note that $o(\mathcal{B}_1) \preceq_o o(\mathcal{B}_2)$ implies $o(\mathcal{B}_1) \neq o(\mathcal{B}_2)$ and hence it holds that $c^{\min}(\mathcal{B}_1) \neq c^{\min}(\mathcal{B}_2)$. Let $\tau := \min\{i : c_i^{\min}(\mathcal{B}_1) \neq c_i^{\min}(\mathcal{B}_2)\}$ be the smallest index where $c^{\min}(\mathcal{B}_1)$ and $c^{\min}(\mathcal{B}_2)$ differ. Since we assumed that $c^{\min}(\mathcal{B}_1)$ does not lexicographically dominate $c^{\min}(\mathcal{B}_2)$, it follows that $c_\tau^{\min}(\mathcal{B}_1) > c_\tau^{\min}(\mathcal{B}_2)$. Thus, the vectors $o(\mathcal{B}_1)$ and $o(\mathcal{B}_2)$ are equal in the last $\ell := \sum_{i=1}^{\tau-1} c_i^{\min}(\mathcal{B}_1) = \sum_{i=1}^{\tau-1} c_i^{\min}(\mathcal{B}_2)$ components, i.e., $o_j(\mathcal{B}_1) = o_j(\mathcal{B}_2)$ for all $j = K - \ell + 1, \dots, K$. Furthermore, it holds that $o_{K-\ell}(\mathcal{B}_1) \succ o_{K-\ell}(\mathcal{B}_2)$, which contradicts the assumption that $o(\mathcal{B}_1)$ ordinally dominates $o(\mathcal{B}_2)$. \square

Note that while the proof of Theorem 4.2 relies on the fact that all feasible solutions have the same number of elements (and hence all outcome vectors have the same length), the matroid property is not used. Hence, Theorem 4.2 generalizes to all ordinal optimization problems with fixed length solutions. The following Corollary 4.3, that also follows from the results in Schäfer et al., 2020, is an immediate consequence of Theorem 4.2.

Corollary 4.3. *The set of efficient bases of (MPO) is a superset of the set of efficient bases of (MPCmin) and of (MPCmax).*

Proof. Theorem 4.2 implies that the efficient set of (MPCmin) can not contain any bases that are ordinally dominated w.r.t. o since this would imply that they are also lexicographically dominated w.r.t. c^{\min} . The same argument applies to (MPCmax). \square

Remark 4.4. *The reverse implication of Theorem 4.2 does not hold in general, neither for c^{\min} nor for c^{\max} . As a counter example consider the bases \mathcal{B}_4 and \mathcal{B}_6 from Figure 4.3. We have that*

$$\begin{aligned} c^{\min}(\mathcal{B}_6) &= c^{\max}(\mathcal{B}_6) = (2, 1, 2)^\top, c^{\min}(\mathcal{B}_4) = c^{\max}(\mathcal{B}_4) = (1, 3, 1)^\top, \\ o(\mathcal{B}_6) &= (1, 1, 2, 3, 3)^\top \text{ and } o(\mathcal{B}_4) = (1, 2, 2, 2, 3)^\top. \end{aligned}$$

Hence, $c^{\min}(\mathcal{B}_4)$ lexicographically dominates $c^{\min}(\mathcal{B}_6)$ and $c^{\max}(\mathcal{B}_6)$ lexicographically dominates $c^{\max}(\mathcal{B}_4)$, while $o(\mathcal{B}_4)$ and $o(\mathcal{B}_6)$ are ordinally incomparable.

We show in the following that in the case of matroids Corollary 4.3 can be strengthened. Indeed, the following result shows that the respective ordinal and lexicographic non-dominated sets are always equal and have cardinality one. This can also be observed in Example 4.7 below, where all three problems (MPO), (MPCmin) and (MPCmax) have the same efficient and non-dominated sets, namely the unique efficient basis is \mathcal{B}_9 .

The result can be briefly summarized as follows: Corollary 4.2 states that the efficient set of (MPO) is a superset of that of (MPCmin) and (MPCmax). If there were two non-dominated bases $\mathcal{B}_1, \mathcal{B}_2$ for (MPO) and only one of them, say, basis \mathcal{B}_1 , was optimal for problem (MPCmin), then the basis exchange property would imply that basis \mathcal{B}_2 could be improved w.r.t. c^{\min} by an appropriate swap operation. However, this would lead to a basis that also ordinally dominates \mathcal{B}_2 , contradicting the ordinal efficiency of \mathcal{B}_2 . This leads to the following result:

Theorem 4.5. *Let $\mathcal{M}_1 = (E, \mathcal{I}_1)$ be a matroid, let $\mathcal{X}_1 \neq \emptyset$ be the set of bases of \mathcal{M}_1 , and let the functions o , c^{\min} and c^{\max} be given as defined above. Then problems (MPO), (MPCmin) and (MPCmax) have the same efficient set, and the corresponding non-dominated sets have cardinality one.*

Proof. We show the equality of the efficient sets of (MPO) and (MPCmin). The equality of the efficient sets of (MPO) and (MPCmax) follows analogously.

First observe that the non-dominated set of problem (MPCmin) has cardinality one since the lexicographical order is a total order. Moreover, Theorem 4.2 implies that every efficient solution of (MPCmin) is also efficient for (MPO). Consequently, it is sufficient to show that all efficient solutions of (MPO) map to a unique non-dominated outcome vector. We prove this result by contradiction.

Suppose, to the contrary, that there are two efficient bases \mathcal{B}_1 and \mathcal{B}_2 for (MPO) with $o(\mathcal{B}_1) \neq o(\mathcal{B}_2)$ and hence also $c^{\min}(\mathcal{B}_1) \neq c^{\min}(\mathcal{B}_2)$. W.l.o.g. assume that $c^{\min}(\mathcal{B}_1)$ lexicographically dominates $c^{\min}(\mathcal{B}_2)$.

Let $e \in \mathcal{B}_2 \setminus \mathcal{B}_1$ be chosen such that $o(\hat{e}) \preceq o(e)$ for all $\hat{e} \in \mathcal{B}_2 \setminus \mathcal{B}_1$, i.e., e is an element of highest category among all elements in $\mathcal{B}_2 \setminus \mathcal{B}_1$. Then the basis exchange property (B), see Section 2.5, implies that there exists an element $e' \in \mathcal{B}_1 \setminus \mathcal{B}_2$ such that $\mathcal{B}^* := (\mathcal{B}_2 \cup \{e'\}) \setminus \{e\} \in \mathcal{X}_1$, and the choice of e and the fact that $c^{\min}(\mathcal{B}_1) <_{\text{lex}} c^{\min}(\mathcal{B}_2)$ imply that $o(e') \preceq o(e)$. Now, if $o(e') \prec o(e)$, then \mathcal{B}^* dominates \mathcal{B}_2 w.r.t. o , contradicting the assumption. Otherwise, i.e., if $o(e') = o(e)$, then \mathcal{B}^* has one more element in common with \mathcal{B}_1 than \mathcal{B}_2 , and iterating this procedure at most r times eventually yields a swap where $o(e') \prec o(e)$. \square

Corollary 4.6. *The ordinally non-dominated set of (MPO) can be computed by a greedy algorithm.*

Proof. This follows immediately from Theorem 4.5 and the matroid properties, see also Hamacher and Ruhe, 1994. \square

Note that, while Theorem 4.5 states that the non-dominated sets of problems (MPO), (MPCmin) and (MPCmax) have cardinality one, this does in general not transfer to the respective efficient sets. Indeed, the size of the efficient sets may grow exponentially with the problem size. As an example, consider instances with exponentially growing feasible

sets and assume that all elements of E are in the same ordinal category. Then, all feasible solutions of a considered problem are both ordinally and lexicographically efficient.

4.2. Bi-objective Matroid Optimization with Ordinal Costs

We extend the settings of the previous section and consider bi-objective matroid optimization problems (on a matroid $\mathcal{M}_1 = (E, \mathcal{I}_1)$ with rank r and set of bases \mathcal{X}_1) where we combine an ordinal objective with a sum objective function with non-negative integer coefficients $w : E \rightarrow \mathbb{Z}_{\geq}$. The cost of a basis $\mathcal{B} \in \mathcal{X}_1$ w.r.t. this sum objective is given by $w(\mathcal{B}) := \sum_{e \in \mathcal{B}} w(e)$.

4.2.1. Bi-objective Ordinal and Lexicographic Optimality and their Interrelation

If we add a sum objective function to the problems described in Section 4.1.1 above, we obtain the following four variants of bi-objective optimization problems involving additive as well as ordinal objective coefficients:

The *bi-objective matroid problem with a binary objective function* (BBMP)

$$\begin{array}{ll} \min & (w(\mathcal{B}), b(\mathcal{B})) \\ \text{s. t.} & \mathcal{B} \in \mathcal{X}_1, \end{array} \quad (\text{BBMP})$$

the *bi-objective matroid problem with an ordinal objective function* (BMPO)

$$\begin{array}{ll} \min & (w(\mathcal{B}), o(\mathcal{B})) \\ \text{s. t.} & \mathcal{B} \in \mathcal{X}_1, \end{array} \quad (\text{BMPO})$$

and two *bi-objective matroid optimization problems with a counting objective function* (BMPCmax) and (BMPCmin)

$$\begin{array}{ll} \min & w(\mathcal{B}) \\ \text{lexmax} & c^{\max}(\mathcal{B}) \\ \text{s. t.} & \mathcal{B} \in \mathcal{X}_1 \end{array} \quad (\text{BMPCmax})$$

$$\begin{array}{ll} \min & w(\mathcal{B}) \\ \text{lexmin} & c^{\min}(\mathcal{B}) \\ \text{s. t.} & \mathcal{B} \in \mathcal{X}_1. \end{array} \quad (\text{BMPCmin})$$

The problem (BBMP) is investigated in detail in Gorski, 2010, Gorski et al., 2022 and Chapter 3, where an efficient swap algorithm is presented that determines a minimal complete representation of the non-dominated set (i. e., all non-dominated points and one efficient solution for each of them) of (BBMP) in polynomial time. This assumes that an oracle can determine in polynomial time if a given subset $I \subseteq E$ is independent or not. This is, e.g., the case for graphic matroids, uniform matroids and partition matroids, see Gabow and Tarjan, 1984.

The similarities and differences between the bi-objective matroid problems (BMPO), (BMPCmax) and (BMPCmin) are illustrated at the following example of a graphic matroid:

Example 4.7. Consider the graphic matroid introduced in Figure 4.2(b). Its bases are enumerated and illustrated with their weight functions w , o , c^{\min} and c^{\max} in Figure 4.3. It is easy to see that, in accordance with Theorem 4.5, the unique efficient solution w.r.t. all of the individual objective functions o , c^{\min} and c^{\max} is the basis \mathcal{B}_9 .

The corresponding bi-objective problems that additionally consider the sum objective function w all have larger non-dominated sets in this example. The respective non-dominated outcome vectors of the bi-objective problems that combine w with the objective functions o , c^{\min} and c^{\max} , respectively, are highlighted in Figure 4.3 by printing the latter components, i.e., o , c^{\min} and c^{\max} , in bold. Note that the basis \mathcal{B}_1 is efficient in all three cases since it is the unique minimizer of w .

4.2.2. Interrelation Between (BMPO), (BMPCmin) and (BMPCmax)

When moving from optimization problems with only one ordinal objective function to bi-objective problems that additionally include a sum objective w , as in the bi-objective problems (BMPO), (BMPCmin) and (BMPCmax), the situation is much more complex than that described in Section 4.1.2 above. Indeed, while Corollary 4.3 can be adapted to the new situation, Theorem 4.5 does not transfer to the bi-objective case. A corresponding counter example is given below.

Theorem 4.8. The set of efficient bases of (BMPO) is a superset of the set of efficient bases of (BMPCmin) and of (BMPCmax).

Proof. We prove the result for c^{\min} . The corresponding result for c^{\max} follows analogously, noting that (BMPCmin) involves lexicographic minimization while (BMPCmax) involves lexicographic maximization.

We prove this result by contradiction. Hence, let $\bar{\mathcal{B}}$ be an efficient basis for (BMPCmin) but not for (BMPO). Then there exists a basis \mathcal{B}^* with $w(\mathcal{B}^*) \leq w(\bar{\mathcal{B}})$, $o(\mathcal{B}^*) \preceq_o o(\bar{\mathcal{B}})$, and $(w(\mathcal{B}^*), o(\mathcal{B}^*)) \neq (w(\bar{\mathcal{B}}), o(\bar{\mathcal{B}}))$. First note that $o(\mathcal{B}^*) \preceq_o o(\bar{\mathcal{B}})$ implies due to Theorem 4.2 that $c^{\min}(\mathcal{B}^*) \preceq_{\text{lex}} c^{\min}(\bar{\mathcal{B}})$. We distinguish two cases: Either $w(\mathcal{B}^*) < w(\bar{\mathcal{B}})$ and $c^{\min}(\mathcal{B}^*) \preceq_{\text{lex}} c^{\min}(\bar{\mathcal{B}})$, or $w(\mathcal{B}^*) = w(\bar{\mathcal{B}})$ and $c^{\min}(\mathcal{B}^*) <_{\text{lex}} c^{\min}(\bar{\mathcal{B}})$. However, both cases are in contradiction with the efficiency of $\bar{\mathcal{B}}$ for problem (BMPCmin). \square

However, as was to be expected, Theorem 4.5 does not generalize to the bi-objective case as is shown by the following counter example:

Example 4.9. Consider again the graphic matroid introduced in Example 4.7 and the set of all of its bases illustrated in Figure 4.3. The efficient bases for problem (BMPO) are the bases $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_4, \mathcal{B}_5, \mathcal{B}_6, \mathcal{B}_7, \mathcal{B}_9$, while the efficient bases for the problem (BMPCmin) are given by $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_4, \mathcal{B}_5, \mathcal{B}_7, \mathcal{B}_9$, and the efficient bases for problem (BMPCmax) are given by $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_6, \mathcal{B}_7, \mathcal{B}_9$. Hence, basis \mathcal{B}_6 is efficient for (BMPO) but not for (BMPCmin), and the two bases \mathcal{B}_4 and \mathcal{B}_5 are efficient for (BMPO) but not for (BMPCmax). Thus, Theorem 4.5 does not generalize to the bi-objective problems (BMPO), (BMPCmin) and (BMPCmax). Furthermore, it can be seen that neither (BMPCmax) is a superset of (BMPCmin) nor the other way around.

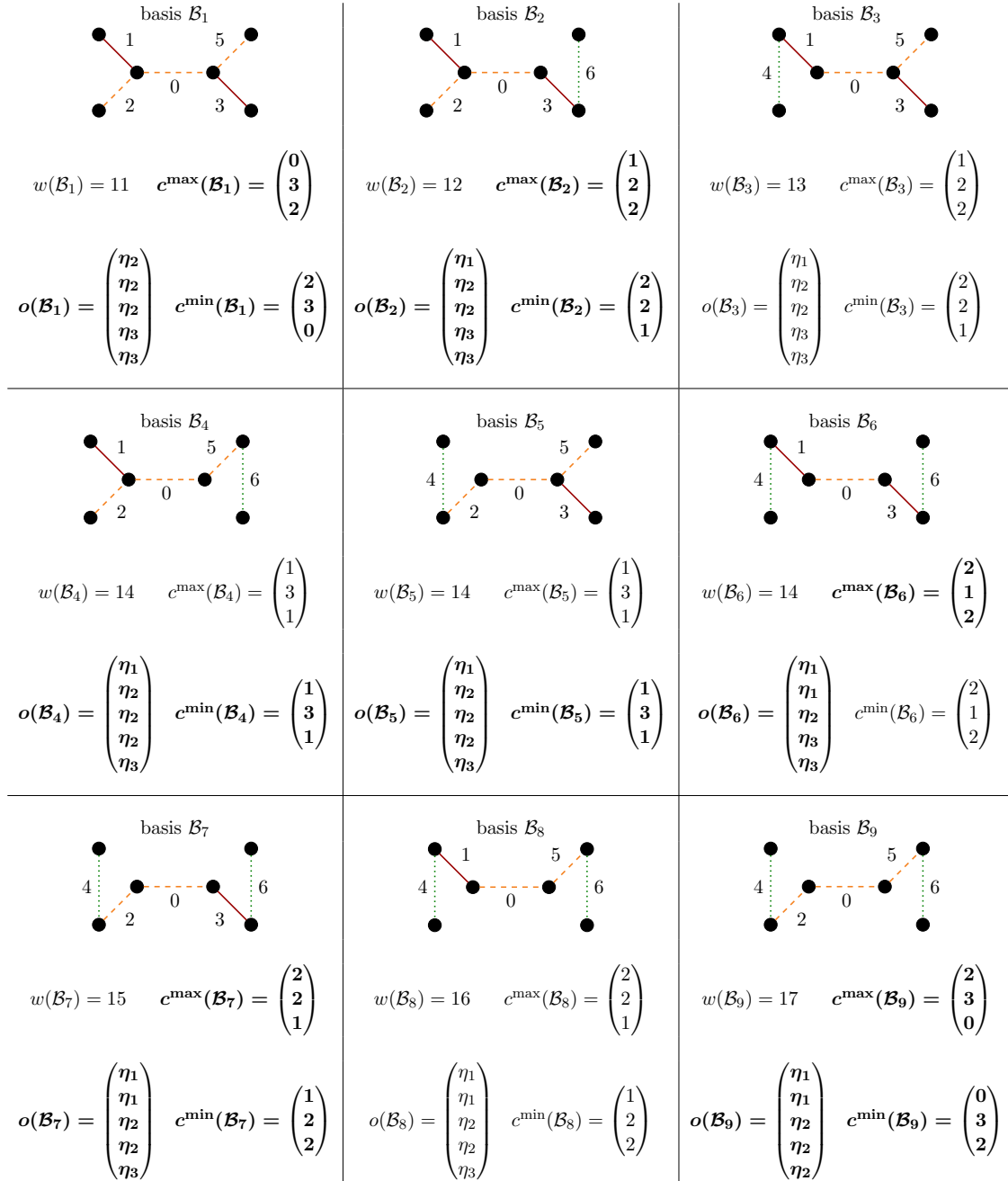


Figure 4.3.: All bases of the graphic matroid introduced in Figure 4.2(b) together with the objective values w , o , c^{\min} and c^{\max} , where we write η_1 for green-dotted, η_2 for orange-dashed, and η_3 for red-solid edges. When only considering the sum objective w , then \mathcal{B}_1 is optimal, and when only considering the objective functions o , c^{\min} or c^{\max} , respectively, then \mathcal{B}_9 is the unique efficient basis. For the problems (BMPO), (BMPCmin) and (BMPCmax) that combine w with o , c^{\min} and c^{\max} , respectively, the non-dominated outcome vectors are indicated by printing the partial objective vectors o , c^{\min} and c^{\max} in bold.

One could conjecture from Example 4.7 that every efficient basis for (BMPO) is efficient for at least one of the problems (BMPCmin) or (BMPCmax). However, this also does not hold in general as the following example shows.

Example 4.10. Consider the graphic matroid shown in Figure 4.4. We focus on all bases $\mathcal{B} \in \mathcal{X}_1$ that have an objective value of $w(\mathcal{B}) = 4$ in the sum objective. Note that these bases can only be dominated by other bases $\hat{\mathcal{B}}$ with $w(\hat{\mathcal{B}}) \leq w(\mathcal{B})$, and hence we restrict our analysis on those bases in Figure 4.4. First observe that all bases $\mathcal{B} \in \mathcal{X}_1$ with $w(\mathcal{B}) = 4$ map to one of the following three feasible outcome vectors

$$o(\mathcal{B}) \in \{(\eta_1, \eta_1, \eta_3, \eta_3)^\top, (\eta_1, \eta_2, \eta_2, \eta_3)^\top, (\eta_2, \eta_2, \eta_2, \eta_2)^\top\},$$

which are all non-dominated for (BMPO). Their corresponding counting vectors c^{\min} are $(2, 0, 2)^\top$, $(1, 2, 1)^\top$ and $(0, 4, 0)^\top$, where the last one is the only one that is lexicographically non-dominated. For c^{\max} the counting vectors are the same, but the first one is lexicographically non-dominated. Consequently, the counting vector $(1, 2, 1)^\top$ is neither lexicographically non-dominated for (BMPCmin) nor for (BMPCmax), but it is non-dominated for (BMPO).

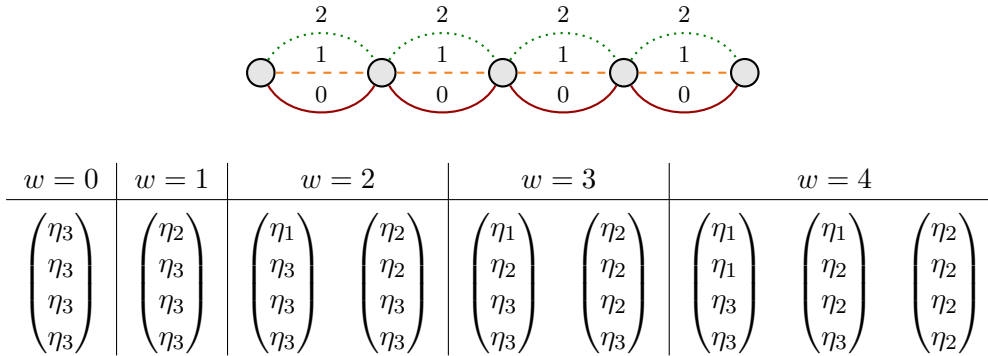


Figure 4.4.: All feasible outcome vectors $o(\mathcal{B})$ with $w(\mathcal{B}) \in \{0, \dots, 4\}$ for a graphic matroid with non-negative integer-valued costs w and three categories (η_1 :green-dotted, η_2 :orange-dashed and η_3 :red-solid).

4.3. Matroid Intersection Algorithm for Ordinal Constraints

In the following we show that the three problems (BMPO), (BMPCmin) and (BMPCmax) can be solved using a series of matroid intersection problems. The approach is based on variants of ε -constraint scalarizations of problem (BMPO) with appropriately selected optimization objective and constraints. Furthermore, we show that matroid intersection problems can be used to solve even problems with several ordinal objective functions and one sum objective function.

4.3.1. Variants of ε -Constraint Scalarizations

We consider an equality-constrained scalarization of (BMPO) (where equality constraints are used rather than inequality constraints as is commonly the case in ε -constraint scalarizations), given by

$$\begin{aligned} \min \quad & w(\mathcal{B}) \\ \text{s. t.} \quad & o_i(\mathcal{B}) = \varepsilon_i, \quad i = 1, \dots, r \\ & \mathcal{B} \in \mathcal{X}_1 \end{aligned} \tag{4.1}$$

with right-hand side vector $\varepsilon \in \mathcal{C}^r$. Intuitively, problem (4.1) specifies *exactly* how many elements of each category must be chosen, and hence each feasible basis $\mathcal{B} \in \mathcal{X}_1$ of (4.1) maps to the same ordinal vector $o(\mathcal{B})$. Depending on the choice of ε , problem (4.1) may be infeasible (if there is no $\mathcal{B} \in \mathcal{X}_1$ with $o(\mathcal{B}) = \varepsilon$), yield an efficient solution \mathcal{B}^* for (BMPO) (if there is *no* $\mathcal{B} \in \mathcal{X}_1$ with $w(\mathcal{B}) = w(\mathcal{B}^*)$ and $o(\mathcal{B}) \preceq_o o(\mathcal{B}^*)$), or yield a dominated solution $\hat{\mathcal{B}}$ for (BMPO) (if there *is* a $\mathcal{B} \in \mathcal{X}_1$ with $w(\mathcal{B}) = w(\hat{\mathcal{B}})$ and $o(\mathcal{B}) \preceq_0 o(\hat{\mathcal{B}})$). Note that suitable choices for ε satisfy $\varepsilon_1 \preceq \dots \preceq \varepsilon_r$ since the components of $o(\mathcal{B})$ are always in non-decreasing order and hence problem (4.1) is certainly infeasible otherwise. In the following, we denote all such *suitable right-hand-side vectors* by $\Upsilon := \{\varepsilon \in \mathcal{C}^r : \varepsilon_1 \preceq \dots \preceq \varepsilon_r\}$.

Since problem (4.1) can be interpreted as a variant of the “classical” ε -constraint scalarization in multi-objective optimization, see, e.g., Ehrgott, 2005 and Section 2.3, the following result is not surprising and follows basically by the same arguments.

Theorem 4.11. *The non-dominated set of problem (BMPO) can be determined by solving problem (4.1) for all suitable right-hand side vectors $\varepsilon \in \Upsilon$ and filtering out all dominated outcome vectors.*

Proof. Let $(w(\mathcal{B}^*), o(\mathcal{B}^*))$ be a non-dominated outcome vector for (BMPO) with pre-image $\mathcal{B}^* \in \mathcal{X}_1$. Then \mathcal{B}^* is optimal for problem (4.1) with $\varepsilon := o(\mathcal{B}^*) \in \Upsilon$. Thus, every non-dominated outcome vector of problem (BMPO) can be determined by solving an appropriate scalarization (4.1). The non-dominated set is then obtained by employing a dominance filtering to the set of all obtained outcome vectors. \square

Now let a suitable constraint vector $\varepsilon \in \Upsilon$ be given, i.e., ε satisfies $\varepsilon_1 \preceq \dots \preceq \varepsilon_r$. Then we define an associated *suitable counting vector* $u \in \mathbb{Z}_{\geq}^K$ by $u_i := |\{j \in \{1, \dots, r\} : \varepsilon_j = \eta_i\}|$ for all $i = 1, \dots, K$, where, by definition, we have that $\sum_{i=1}^K u_i = r$. We denote by $\mathfrak{U} := \{u \in \mathbb{Z}_{\geq}^K : \sum_{i=1}^K u_i = r\}$ the *set of all suitable counting vectors*.

Lemma 4.12. *There is a one-to-one correspondence between suitable right-hand-side vectors $\varepsilon \in \Upsilon$ and suitable counting vectors $u \in \mathfrak{U}$.*

Proof. First consider the case that a suitable constraint vector $\varepsilon \in \Upsilon$ is given. Then an associated suitable counting vector $u \in \mathfrak{U}$ can be determined from ε as described above, i.e., by setting $u_i := |\{j \in \{1, \dots, r\} : \varepsilon_j = \eta_i\}|$ for all $i = 1, \dots, K$. Conversely, if a suitable counting vector $u \in \mathfrak{U}$ is given, then we can determine associated suitable values for $\varepsilon \in \Upsilon$ by setting $\varepsilon_j := \eta_i$, where the ordinal level $i \in \{1, \dots, K\}$ is chosen such that $\sum_{l=1}^i u_l \leq j$ and $\sum_{l=1}^{i-1} u_l > j$, for all $j = 1, \dots, r$. \square

Lemma 4.12 implies that problem (4.1) can be equivalently written as

$$\begin{aligned} \min \quad & w(\mathcal{B}) \\ \text{s. t.} \quad & c_i(\mathcal{B}) = u_i, \quad i = 1, \dots, K \\ & \mathcal{B} \in \mathcal{X}_1, \end{aligned} \tag{4.2}$$

where the right-hand side vector $u \in \mathfrak{U}$ is chosen as a suitable counting vector, i.e., $u \in \mathbb{Z}_{\geq}^K$ and $\sum_{i=1}^K u_i = r$, and c is the counting objective introduced in Section 4.1.1. Moreover, since $\sum_{i=1}^K c_i(\mathcal{B}) = r = \sum_{i=1}^K u_i$ for all feasible bases $\mathcal{B} \in \mathcal{X}_1$, the equality constraints in (4.2) can be replaced by inequality constraints without changing the feasible set. Problem (4.2) is thus equivalent to the following variant of ε -constraint scalarization that relates to problem (BMPCmin)

$$\begin{aligned} \min \quad & w(\mathcal{B}) \\ \text{s. t.} \quad & c_i^{\min}(\mathcal{B}) \leq u_{K-i+1}, \quad i = 1, \dots, K. \\ & \mathcal{B} \in \mathcal{X}_1. \end{aligned} \tag{4.3}$$

Corollary 4.13. *The non-dominated set of problem (BMPO) can be determined by solving problem (4.2) (or problem (4.3)) for all suitable counting vectors $u \in \mathfrak{U}$, and filtering out all dominated outcome vectors. The non-dominated sets of problems (BMPCmin) and (BMPCmax) can be obtained from this set by further filtering out all lexicographically dominated outcome vectors.*

Proof. The result follows immediately from Theorems 4.8 and 4.11, using the equivalence of the formulations (4.1), (4.2) and (4.3). \square

We emphasize that problem (4.3) remains meaningful when a non-suitable counting vector $u \in \mathbb{Z}_{\geq}^K$ with $\sum_{i=1}^K u_i > r$ is used as right-hand-side vector. Indeed, when considering a suitable counting vector $u \in \mathfrak{U}$ and a non-suitable counting vector $\hat{u} \geq u$, then \hat{u} yields a relaxation of problem (4.3) with u as the right-hand-side vector. Nevertheless, the constraint $\mathcal{B} \in \mathcal{X}_1$ guarantees that only bases of \mathcal{M}_1 are returned, and hence $\sum_{i=1}^K c_i^{\min}(\mathcal{B}) = r$ remains satisfied also in this case. Moreover, using $\bar{u} := (r, \dots, r)^\top \in \mathbb{Z}_{\geq}^K$ yields a “complete” relaxation in the sense that constraints $c_i^{\min}(\mathcal{B}) \leq \bar{u}_{K-i+1} = r$, $i = 1, \dots, K$, are satisfied for *all* bases $\mathcal{B} \in \mathcal{X}_1$ and hence redundant in this case.

Using the above results, the cardinality of the non-dominated set of problem (BMPO) (and hence also of problems (BMPCmin) and (BMPCmax)) can be bounded. Note that this is in analogy to the results obtained in Schäfer et al., 2020 and Schäfer et al., 2021 for ordinal shortest path and ordinal knapsack problems, respectively. Indeed, the number of equality-constraint scalarizations (4.2) that need to be solved in order to guarantee that all non-dominated outcome vectors of problem (BMPO) are found is polynomially bounded. This can be seen from the fact that the number of suitable counting vectors $u \in \mathfrak{U}$, i.e., the number of K -dimensional non-negative integer vectors that satisfy $\sum_{j=1}^K u_j = r$, is given by $\binom{r+K-1}{K-1} = \mathcal{O}(r^{K-1})$ (assuming that K is constant), i.e., it is equal to the number of multisets of cardinality $K-1$ taken from a set of size $r+1$. This number is also known as occupancy number, see, e.g., Feller, 1968. We obtain the following result.

Theorem 4.14. *The cardinality of the non-dominated set of problem (BMPO) is bounded by $\mathcal{O}(r^{K-1})$, which is polynomial in r as long as K is constant.*

4.3.2. Matroid Intersection

We focus on the ε -constraint scalarization-variant (4.3) in the following and show that it can be equivalently formulated as a matroid intersection problem. Towards this end, let an arbitrary but fixed, suitable counting vector $u \in \mathfrak{U}$ be given as right-hand-side vector in problem (4.3).

Now consider the partition $E = E_1 \cup E_2 \cup \dots \cup E_K$ of the ground set E of \mathcal{M}_1 , where $E_j := \{e \in E : o(e) = \eta_j\}$ for $j = 1, \dots, K$, i.e., E_j contains all elements from E that are in category η_j . Given this partition of E , let $\mathcal{M}_2(u) = (E, \mathcal{I}_2(u))$ be an associated partition matroid with independent sets given by $\mathcal{I}_2(u) := \{J \subseteq E : |J \cap E_j| \leq u_j, 1 \leq j \leq K\}$. Then problem (4.3) can be solved using the matroid intersection problem

$$\begin{aligned} \min \quad & w(I) \\ \text{s. t.} \quad & I \in \mathcal{I}_1 \cap \mathcal{I}_2(u) \\ & |I| = \max\{|J| : J \in \mathcal{I}_1 \cap \mathcal{I}_2(u)\}. \end{aligned} \tag{4.4}$$

Note that the second constraint in (4.4) is needed since otherwise, $\mathcal{B} = \emptyset$ would always be optimal.

Theorem 4.15. *Let $u \in \mathbb{Z}_{\geq}^K$ be arbitrary but fixed. If problem (4.3) is feasible, then problems (4.3) and (4.4) are equivalent. Moreover, if problem (4.3) is infeasible, then every optimal solution \mathcal{B}^* of problem (4.4) satisfies $|\mathcal{B}^*| < r$.*

Proof. We first show that when problem (4.3) is feasible for the given suitable counting vector u , then problems (4.3) and (4.4) have the same feasible sets. Indeed, in this case there exists a basis $\hat{\mathcal{B}} \in \mathcal{X}_1$ that satisfies $c(\hat{\mathcal{B}}) \leq u$, and hence $\hat{I} := \hat{\mathcal{B}}$ with $|\hat{I}| = r$ is feasible for (4.4). This implies that all feasible solutions of (4.4) have cardinality r and are thus bases of \mathcal{M}_1 . In this situation, the constraints $c_i^{\min}(\mathcal{B}) \leq u_{K-i+1}$, $i = 1, \dots, K$ (for (4.3)) and $I \in \mathcal{I}_2(u)$ (for (4.4)) are equivalent. Since both problems also have the same objective function, they are clearly equivalent in this case. If, however, problem (4.3) is infeasible for the current choice of u , then the matroid intersection problem (4.4) is still feasible, but returns an optimal solution \mathcal{B}^* with $|\mathcal{B}^*| < r$. This situation can be easily recognized. \square

The advantage of this reformulation is that the matroid intersection problem (4.4) can be solved by the polynomial time *matroid intersection algorithm* (MI) of Edmonds, 1971, see Section 2.8.

Note that the cardinality constraint in the general formulation (4.4) of the matroid intersection problem can be omitted if \mathcal{I}_1 is replaced by \mathcal{X}_1 , i.e., the set of all bases of the matroid \mathcal{M}_1 . Hence, we can alternatively solve the problem

$$\begin{aligned} \min \quad & w(\mathcal{B}) \\ \text{s. t.} \quad & \mathcal{B} \in \mathcal{X}_1 \cap \mathcal{I}_2(u). \end{aligned} \tag{4.5}$$

This is realized in Algorithm 10 by only considering optimal solutions of problem (4.4) that are actually bases of \mathcal{M}_1 , see lines 4 and 5 in Algorithm 10 below.

4.3.3. Algorithmic Consequences

Theorems 4.11 and 4.14 imply that all non-dominated points of (BMPO), i.e., Y_N^O , can be determined by a polynomial number of matroid intersections applied on (4.4). The structure of this procedure is given in Algorithm 10. Note that this algorithm can be easily adapted to solve (BMPCmin) or (BMPCmax). Since the non-dominated set of (BMPO) is a superset of the corresponding non-dominated sets of (BMPCmin), i.e., $Y_N^{C\min}$, and (BMPCmax), i.e., $Y_N^{C\max}$, only a slight modification of the filtering step in line 6 is necessary.

Algorithm 10:	Matroid Intersection for Ordinal Constraints ($MIOC(\mathcal{M}_1, w, o)$)
----------------------	---

Input: Matroid $\mathcal{M}_1 = (E, \mathcal{I}_1)$, sum objective function w and ordinal objective function o

Output: Non-dominated set of problem (BMPO)

- 1 $X := \emptyset$
- 2 **foreach** $u \in \mathcal{U}$ **do**
- 3 Solve (4.4) with (MI) and save the obtained independent set I^*
- 4 **if** $|I^*| = r$ **then**
- 5 Set $X = X \cup \{I^*\}$
- 6 Filter the efficient independent sets of X w.r.t. (BMPO) and save the corresponding outcome vectors in Y_N^O
- 7 **return** Y_N^O

It is possible to improve the performance of Algorithm 10 by reducing the number of considered bounds $u \in \mathcal{U}$, i.e., the number of solved matroid intersections. This can be achieved by initially solving (4.4) with $u = (r, \dots, r)^\top \in \mathbb{R}^K$, which returns a weakly efficient basis \mathcal{B}^* (assuming $\mathcal{X}_1 \neq \emptyset$) with the smallest possible cost w^* . Consequently, only upper bounds $u \in \mathcal{U}$ such that $(u_K, \dots, u_1)^\top \leq_{\text{lex}} c^{\min}(\mathcal{B}^*)$ have to be considered. Thus, we modify lines 1–2 in Algorithm 10 accordingly and obtain Algorithm 11.

Note that, in the worst case, this initialization yields no reduction of the running time, since there might exist a basis \mathcal{B}' that minimizes w and for which it holds that $c^{\min}(\mathcal{B}') = (r, 0, \dots, 0)^\top$. However, in our numerical tests this procedure often leads to a significant reduction of the number of iterations, as described in Section 4.5. Note that the initial bound in Algorithm 11 is not a suitable counting vector as defined in Section 4.3.1, i.e., $u = (r, \dots, r)^\top \notin \mathcal{U}$. Since we consider in the following often relaxations of suitable subproblems, we use the notation $\bar{\mathcal{U}} := \{u \in \mathbb{Z}_{\geq}^K : \sum_{i=1}^K u_i \geq r, u_i \leq r, i = 1, \dots, K\}$ to denote the considered upper bound set.

Based on the fact that the lexicographic order is a total order, Algorithm 11 can be further improved when applied on problem (BMPCmin). In this case, the considered upper bound set can also be reduced during the course of the algorithm. The initialization of the bound set \mathcal{U} is analogous to Algorithm 11, i.e., we solve the matroid intersection problem for $u = (r, \dots, r)^\top \in \mathbb{R}^K$. Let \mathcal{B}^* be the obtained weakly efficient basis of (BMPCmin) minimizing the sum objective function w . Then, it is sufficient to solve subproblems with

Algorithm 11: Improved Initialization of Matroid Intersection for Ordinal Constraints ($MIOCO(\mathcal{M}_1, w, o)$)

Input: Matroid $\mathcal{M}_1 = (E, \mathcal{I}_1)$, sum objective function w and ordinal objective function o

Output: Non-dominated set of problem (BMPO)

```

1  $u = (r, \dots, r)$ 
2 Solve (4.4) with (MI) and save the obtained basis  $\mathcal{B}^*$ 
3 Set  $X = \{\mathcal{B}^*\}$ 
4 foreach  $u \in \{v \in \mathfrak{U}: (v_K, \dots, v_1)^\top \leq_{\text{lex}} c^{\min}(\mathcal{B}^*)\}$  do
5    $\lfloor$  run lines 3–5 of Algorithm 10
6 Filter the efficient independent sets of  $X$  w.r.t. (BMPO) and save the
   corresponding outcome vectors in  $Y_N^O$ 
7 return  $Y_N^O$ 

```

upper bounds $u \in \bar{\mathfrak{U}}$ such that $(u_K, \dots, u_1)^\top \leq_{\text{lex}} c^{\min}(\mathcal{B}^*)$. Due to the lexicographic order we can explicitly enumerate the new upper bounds u to be considered as

$$(u_K, \dots, u_1)^\top \in \{(c_1^{\min}(\mathcal{B}^*) - 1, r, \dots, r)^\top, (c_1^{\min}(\mathcal{B}^*), c_2^{\min}(\mathcal{B}^*) - 1, r, \dots, r)^\top, \dots, (c_1^{\min}(\mathcal{B}^*), \dots, c_{K-2}^{\min}(\mathcal{B}^*), c_{K-1}^{\min}(\mathcal{B}^*) - 1, r)^\top\}$$

such that $u \geq 0$. These upper bounds are added to the list $\bar{\mathfrak{U}}$ of open subproblems and sorted in lexicographically increasing order. Whenever a new candidate for an efficient basis is found, we update the list of open subproblems $\bar{\mathfrak{U}}$ and re-sort it. In Algorithm 12 this procedure is repeated until $\bar{\mathfrak{U}} = \emptyset$. To simplify the notation we slightly abuse the notation and consider $\bar{\mathfrak{U}}$ to be a sorted list, referring by $\bar{\mathfrak{U}}[1]$ to the first element of this list. Note that an analogous solution algorithm can be formulated for the corresponding lexicographic maximization problem (BMPCmax).

4.4. Multi-objective Matroid Optimization with Ordinal Costs

Problem (BMPO) can be generalized by considering $p \geq 2$ objective functions with ordinal weights. This can be illustrated at a graph whose edges are classified w.r.t. two types of categories, for example, colors (e.g., green, orange, red) and letters (e.g., A, B). Then every edge is in exactly one of the following categories: green-A, green-B, orange-A, orange-B, red-A or red-B. These combinations of categories are a-priori not completely ordered, since, in general, neither green-B is preferred over red-A, nor red-A is preferred over green-B. However, such problems can be considered in the context of combined orderings as introduced in Section 2.2.

Multi-objective matroid problems with one sum objective function and several ordinal objective functions can be handled analogously to bi-objective matroid problems with one ordinal objective function (BMPO). Without going much into detail, we shortly describe the formulation of an associated weighted matroid intersection problem that generalizes problem (4.4).

Let p denote the number of ordinal objective functions o^i , $i = 1, \dots, p$, let K_i denote the number of categories for the i -th ordinal objective, and let $\eta_{ij} \in \mathcal{C}$ denote the j -th category of the i -th ordinal objective, $j = 1, \dots, K_i$, where $\eta_{ij} \prec \eta_{ik}$ whenever $j < k$. Then we can define a partition matroid \mathcal{M}_3 (generalizing \mathcal{M}_2) by partitioning the ground set $E = \bigcup_{i=1}^p \bigcup_{j=1}^{K_i} E_{ij}$, where $E_{ij} := \{e \in E : o^i(e) = \eta_{ij}\}$ for $j = 1, \dots, K_i$ and $i = 1, \dots, p$. The set of independent sets of \mathcal{M}_3 is given by

$$\mathcal{I}_3 := \{J \subseteq E : |J \cap E_{ij}| \leq u_{ij}, j = 1, \dots, K_i \text{ and } i = 1, \dots, p\},$$

where u_{ij} denotes the number of elements that are allowed in category η_{ij} in the ordinal objective o^i . Note that again $0 \leq u_{ij} \leq r$ for all $j = 1, \dots, K_i$ and $i = 1, \dots, p$, and that $\sum_{j \in \{1, \dots, K_i\}} u_{ij} = r$ for all $i = 1, \dots, p$. Therefore, it is possible to solve this problem by solving all relevant weighted matroid intersection problems (4.4) (with \mathcal{M}_2 replaced by \mathcal{M}_3) and filtering out all dominated outcome vectors w.r.t. the combined ordering relation. In this case, the number of calls of problems (4.4) is bounded by $\mathcal{O}(p \cdot r^{\tilde{K}-1})$, where r still denotes the rank of the matroid \mathcal{M}_1 and $\tilde{K} = \max\{K_i : i = 1, \dots, p\}$. If p and K_i are fixed, $i = 1, \dots, p$, then the number of scalarized subproblems is polynomially bounded in the input size. Moreover, in this case every weighted matroid intersection problem can be solved in polynomial time if we assume that an oracle can determine in polynomial time whether a given subset $I \subseteq E$ is independent or not.

4.5. Numerical Results

The Efficient Swap Algorithm suggested in Chapter 3 and in Gorski et al., 2022 as well as the three versions of the Matroid Intersection Algorithm for Ordinal Constraints (Algorithms 10, 11 and 12 with the algorithm of Floyd-Warshall for node weights, see Section 2.7) are implemented and numerically tested. As test instances, we consider graphic matroids and partition matroids for \mathcal{M}_1 , where in the latter case the ground set is partitioned into three subsets. All computations were done on a computer with an Intel(R) Core(TM) i7-7500U CPU 2.70 GHz processor and 8 GB RAM. The algorithms were implemented and run in MATLAB, Version R2019b.

In the first experiment we compare the two types of algorithms on a graphic matroid with one sum objective function and one binary objective function. The instances were generated based on random connected undirected graphs $G = (V, E)$ with n nodes and m edges using the implementation of Schnepfer et al., 2021. The weight coefficients w of the sum objective are randomly chosen integer values in $\{1, \dots, 2m\}$, and the values of the binary objective b are random binary values. In both cases we used a uniform distribution. We solved the obtained instances of problem (BBMP) by the Efficient Swap Algorithm and by Algorithm 11.

The numerical results can be found in Table 4.1. In the first two columns the instance size is given by the number of nodes and edges (n, m) and the average number of non-dominated outcome vectors $|Y_N|$ over 100 random instances. The results show clearly that the average running of Algorithm 11 increases much faster with the instance size compared to the Efficient Swap Algorithm.

Instanze Size (n, m)	Y_N	$MIOC^O$		ESA
		<i>iter</i>	[s]	[s]
(7, 10)	2.39	3.26	0.08	0.03
(7, 15)	3.32	3.84	0.16	0.03
(7, 20)	3.87	3.97	0.24	0.03
(10, 20)	4.36	5.13	0.54	0.04
(10, 30)	5.17	5.44	0.97	0.05
(10, 40)	5.46	5.53	1.38	0.05
(15, 30)	6.05	6.94	2.43	0.05
(15, 60)	7.77	7.95	6.93	0.07
(15, 100)	7.90	7.99	12.57	0.07
(20, 40)	7.58	8.58	7.36	0.07
(20, 100)	10.49	10.60	30.26	0.10
(20, 180)	10.35	10.39	59.36	0.11

Table 4.1.: Average computation time in seconds to solve 100 instances of problem (BBMP) on a graphic matroid with the Efficient Swap Algorithm (ESA) and with Algorithm 11 ($MIOC^O$).

(n, m)	Y_N	$ESA[s]$
(100, 200)	36.01	0.38
(100, 1 000)	51.01	0.60
(100, 2 000)	50.41	0.77
(100, 4 000)	49.72	1.15
(1 000, 2 000)	348.75	6.28
(1 000, 15 000)	499.08	13.31
(1 000, 30 000)	501.31	20.28
(1 000, 45 000)	501.68	28.49

Table 4.2.: Average computation time in seconds to solve 100 instances of problem (BBMP) with the Efficient Swap Algorithm.

Therefore, you find in Table 4.2 even larger problem instances that are only solved with the Efficient Swap Algorithm. The largest instances with 1 000 nodes and 45 000 edges can be solved with the Efficient Swap Algorithm in round about 29 seconds, which is less then the mean time Algorithm 11 needs to solve instances with 20 nodes and 100 edges.

This huge difference in the computational time of the algorithms is not surprising, because the Efficient Swap Algorithm utilizes the specific problem structure, in particular the connectedness of the non-dominated set, as proven in Gorski et al., 2022 and in Chapter 3. Nevertheless, the number of problems that are solved with Algorithm 11 (*iter*) is quite close to the number of non-dominated outcome vectors ($|Y_N|$), which indicates that only few redundant problems were solved.

The strength of all three matroid intersection algorithms for ordinal constraints is that they can be applied to a broader class of problems than the Efficient Swap Algorithm, which is restricted to two ordinal categories. In the following tests we use again randomly generated graphs $G = (V, E)$ with n nodes and m edges with objective function coefficients w and o . The entries of w and o were generated randomly with uniform distribution in

(n, m)	Instance Size			$MIOC$		$MIOC^O$		$MIOC^{C_{\min}}$	
	$ Y_N^O $	$ Y_N^{C_{\min}} $	$ Y_N^{C_{\max}} $	<i>iter</i>	[s]	<i>iter</i>	[s]	<i>iter</i>	[s]
(7, 10)	3.90	3.65	3.75	28	0.48	17.10	0.31	6.30	0.16
(7, 15)	7.30	6.05	6.40	28	1.02	18.55	0.77	9.75	0.45
(7, 20)	6.20	5.40	5.65	28	1.67	14.45	0.87	8.00	0.56
(10, 20)	9.10	7.65	7.85	55	5.03	32.50	2.87	12.55	1.47
(10, 30)	12.55	9.60	11.35	55	9.63	32.80	5.67	13.60	2.61
(10, 40)	15.90	11.75	12.95	55	13.56	31.90	8.00	15.25	3.94
(15, 30)	15.20	11.60	11.40	120	36.16	70.40	21.00	18.60	6.43
(15, 60)	27.20	18.40	20.65	120	101.30	69.00	58.97	24.30	21.07
(15, 100)	27.90	18.20	22.65	120	184.26	66.95	105.23	23.10	36.19
(20, 40)	20.95	14.65	15.40	210	158.48	113.85	85.85	23.95	20.53
(20, 100)	38.45	24.55	27.00	210	578.96	101.00	282.43	30.95	87.42
(20, 180)	46.55	27.20	33.45	210	1162.27	115.35	648.08	33.70	190.49

Table 4.3.: Numerical results for a graphic matroid and $K = 3$ categories. For every problem size 20 instances were solved to obtain average results.

$\{1, \dots, 2m\}$ and in $\{1, \dots, K\}$ for w and o , respectively, were $K \in \{3, 4, 5\}$. The results for $K = 3, 4, 5$ can be found in Tables 4.3, 4.4 and 4.5, respectively. We observe that the number of solutions found for the different problems is quite similar for small problem sizes, but for larger instances and more categories the number of non-dominated points is much smaller for the lexicographic models as compared to the ordinal approach. The running time depends obviously on the instance size. However, the effect of an increasing number of edges m is rather limited. A significant influence can be seen by the number of nodes n , which determines the rank of the matroid. Furthermore, the number of categories K has an important effect on the running time.

As expected, reducing the number of considered upper bound vectors u for problem (BMPO) generally leads to fewer iterations. On average, only little more than half of the iterations are needed in this case. Nevertheless, note that in the worst case this strategy may not lead to an improvement. In the case of the lexicographic variant (BMPCmin) the potential reduction is much more significant. Indeed, the required computation time is drastically reduced in this case, especially for large K . For example, for $K = 5$ we have a reduction of the running time by a factor of around 20 in all cases with $n = 10$.

We get similar results when testing with partition matroids rather than graphic matroids. Here, we consider a ground set of n objects and restrict the analysis to partitions of the ground set into three subsets. The upper bounds on the number of elements from each subset are selected such that every basis consists of $\frac{n}{2}$ elements, and the problem is feasible. After defining an instance of a partition matroid \mathcal{M}_1 in this way, the objective functions are generated. Each object has an associated weight between 1 and $10 \cdot n$ and is assigned to one of K categories, where $K \in \{3, 4, 5\}$. The results for $K = 3, 4, 5$ can be found in Tables 4.6, 4.7 and 4.8, respectively. Again, the improved choice of u leads to significantly better running times. Moreover, the running time increases with the number of elements n and the number of categories K .

Note that further speed-ups can be expected by parallel implementations of the matroid intersection algorithms.

(n, m)	Instance Size			$MIOC$		$MIOC^O$		$MIOC^{C_{\min}}$	
	$ Y_N^O $	$ Y_N^{C_{\min}} $	$ Y_N^{C_{\max}} $	<i>iter</i>	[s]	<i>iter</i>	[s]	<i>iter</i>	[s]
(7, 10)	4.15	3.70	3.70	84	1.34	44.45	0.73	9.00	0.25
(7, 15)	7.50	5.85	6.25	84	2.83	43.85	1.61	12.10	0.56
(7, 20)	10.50	7.85	8.10	84	4.44	52.05	2.88	14.80	1.93
(10, 20)	12.15	8.65	8.60	220	18.15	120.25	9.38	18.75	2.01
(10, 30)	20.60	12.40	14.80	220	36.78	132.95	22.01	23.85	4.46
(10, 40)	23.70	14.35	17.35	220	53.79	118.00	28.27	23.90	5.95
(15, 30)	29.05	16.80	17.55	680	191.27	395.75	107.14	37.15	12.84
(15, 60)	56.00	24.95	34.10	680	560.72	350.15	291.72	42.40	35.97
(15, 100)	63.60	28.00	38.70	680	1042.33	377.45	597.78	44.15	69.33
(20, 40)	44.80	21.80	23.50	1540	1073.17	708.20	502.36	43.80	37.32

Table 4.4.: Numerical results for a graphic matroid and $K = 4$ categories. For every problem size 20 instances were solved to obtain average results.

(n, m)	Instance Size			$MIOC$		$MIOC^O$		$MIOC^{C_{\min}}$	
	$ Y_N^O $	$ Y_N^{C_{\min}} $	$ Y_N^{C_{\max}} $	<i>iter</i>	[s]	<i>iter</i>	[s]	<i>iter</i>	[s]
(7, 10)	4.20	3.85	3.85	210	3.10	106.75	1.57	10.95	0.27
(7, 15)	10.75	7.55	8.10	210	6.65	121.75	3.88	18.05	0.81
(7, 20)	14.90	10.00	11.25	210	10.56	124.30	6.14	20.35	1.36
(10, 20)	22.85	11.75	12.65	715	57.29	429.80	33.25	30.30	3.19
(10, 30)	24.70	14.55	15.95	715	114.37	396.00	62.16	32.00	5.57
(10, 40)	30.10	15.65	19.80	715	175.93	410.75	98.27	32.80	8.22

Table 4.5.: Numerical results for a graphic matroid and $K = 5$ categories. For every problem size 20 instances were solved to obtain average results.

n	Instance Size			$MIOC$		$MIOC^O$		$MIOC^{C_{\min}}$	
	$ Y_N^O $	$ Y_N^{C_{\min}} $	$ Y_N^{C_{\max}} $	<i>iter</i>	[s]	<i>iter</i>	[s]	<i>iter</i>	[s]
10	2.55	2.50	2.50	21	0.23	11.20	0.10	4.60	0.08
20	4.75	4.40	4.45	66	3.26	35.45	1.83	8.40	0.63
30	9.90	8.00	8.40	136	20.26	76.50	11.43	13.45	2.68
40	16.85	12.55	12.75	231	81.26	127.25	45.14	20.40	8.92
50	24.05	17.60	15.90	351	229.91	201.90	131.29	26.30	21.43
60	34.00	21.10	21.75	496	552.54	284.20	318.62	32.80	46.41
70	39.70	24.50	25.40	666	1177.52	375.90	662.34	37.30	83.06

Table 4.6.: Numerical results for a partition matroid with three subsets and $K = 3$ categories. For every problem size 20 instances were solved to obtain average results.

n	Instance Size			$MIOC$		$MIOC^O$		$MIOC^{C_{\min}}$	
	$ Y_N^O $	$ Y_N^{C_{\min}} $	$ Y_N^{C_{\max}} $	<i>iter</i>	[s]	<i>iter</i>	[s]	<i>iter</i>	[s]
10	2.55	2.45	2.40	56	0.41	31.10	0.23	6.15	0.10
20	7.10	5.80	5.85	286	12.60	173.90	7.43	14.05	0.94
30	15.15	9.75	10.05	816	117.69	451.70	62.22	22.85	4.33
40	40.40	20.95	21.25	1 771	594.17	967.80	317.36	42.25	18.51
50	44.90	22.50	22.20	3 276	2 081.56	1 946.25	1 191.88	49.70	41.17

Table 4.7.: Numerical results for a partition matroid with three subsets and $K = 4$ categories. For every problem size 20 instances were solved to obtain average results.

n	Instance Size			$MIOC$		$MIOC^O$		$MIOC^{C_{\min}}$	
	$ Y_N^O $	$ Y_N^{C_{\min}} $	$ Y_N^{C_{\max}} $	<i>iter</i>	[s]	<i>iter</i>	[s]	<i>iter</i>	[s]
10	3.65	3.15	3.25	126	0.84	71.55	0.47	8.20	0.14
20	10.65	8.20	7.75	1 001	43.34	539.15	22.62	23.20	1.67
30	28.25	15.35	16.35	3 876	534.59	2 293.50	308.62	41.50	7.83

Table 4.8.: Numerical results for a partition matroid with three subsets and $K = 5$ categories. For every problem size 20 instances were solved to obtain average results.

4.6. Conclusion and Further Ideas

In this chapter we consider single- and multi-objective matroid optimization problems that combine “classical” sum objective functions with one or several ordinal objective functions. Besides the concept of ordinal optimality, we consider two variants of lexicographic optimization that lexicographically maximize the number of “good” elements or minimize the number of “bad” elements, respectively. In the case of (single-objective) ordinal optimization, we show that these concepts are actually equivalent for matroids, and that optimal solutions can be found by a simple and efficient greedy strategy. In the bi-objective setting, we use variants of ε -constraint scalarizations to obtain a polynomial number of matroid intersection problems, from which the non-dominated sets of the respective problems can be derived by simple filtering operations. This yields an overall polynomial-time algorithm for multi-objective ordinal matroid optimization problems. Numerical tests on graphic matroids and on partition matroids validate the efficiency of this approach.

Future research should focus on a further analysis of the similarities and differences between multi-objective optimization problems with classical sum objectives and with ordinal objectives. Moreover, alternative (partial) orderings may be considered and analyzed in the light of different scalarization techniques.

In the next chapter we generalize the ordinal optimality concept to solutions with arbitrary length and hence investigate general combinatorial optimization problems with ordinal costs. Moreover, we describe the interrelation of ordinal combinatorial optimization problems and multi-objective combinatorial optimization problems. The interrelation

of those problems is based on the one to one correspondence between the objective functions o and c . Due to this equivalence there has to exist an optimality concept for c such that we get the same efficient set for both objective functions. We introduce the corresponding optimality concept in the next chapter.

5. Single- and Multi-objective Combinatorial Optimization Problems with Ordinal Costs

We investigate in this chapter again optimization problems with ordinal costs, but in contrast to Chapter 4, we do not restrict ourselves to matroid problems but consider general combinatorial optimization problems. To motivate the consideration of ordinal costs, we use an example of a shortest path problem with three categories. Note that in this case, feasible solutions, i.e., paths may be of different lengths. This example is used as an illustration throughout this chapter.

Consider the problem of finding optimized routes for cyclists in a road network: While edges may be associated with different categories like asphalt, gravel or sand—or, when related to safety considerations, very safe (there is a bicycle path), neutral (a quiet road) or unsafe (a main road without bicycle path)—such categories do not immediately translate into monetary or cost values. Bi-objective shortest path problems with route safety criteria are addressed, for example, in the web application *geovelo* and in the associated publications Kergosien et al., 2021; Sauvanet and Néron, 2010. In these references, only two categories are considered (safe or unsafe edges), and the safety criterion is translated into a cost function that evaluates the total length of unsafe route segments. In contrast, an ordinal shortest path problem is investigated in Schäfer et al., 2020. A major difficulty when considering ordinal objective functions is that “optimality” may be defined in many different ways. Schäfer et al., 2020 suggests an optimality concept that is based on sorted category vectors. A similar concept is used in Chapter 4, see also Klamroth et al., 2022a, where matroid optimization problems with one real-valued and one ordinal objective function are investigated. A different perspective is proposed in Schäfer et al., 2021 who define ordinal optimality for knapsack problems on the basis of numerical representations for the categories. Ordinal costs are considered in the literature since many years, see e.g. Barteo, 1971. Ordinal preferences have been studied since then in several further publications like e.g. Bossong and Schweigert, 1999, Bouveret and Endriss, 2010, Brams et al., 2003 and Delort et al., 2011. In Delort et al., 2011 a multi-objective problem is used to solve a problem with ordinal costs. In this chapter, we use an equivalent multi-objective problem while analyzing the interrelations and the transformation between ordinal and multi-objective formulations in much more detail. Since our problem formulation and notation is closely related to that of Schäfer et al., 2021, we mainly refer to this paper in the following.

In this chapter, we consider general combinatorial optimization problems and provide a new cone-based interpretation of the optimality concept for ordinal objectives suggested, for example, in Schäfer et al., 2021. In particular, we interrelate ordinal optimality with the classical concept of Pareto optimality for an associated multi-objective optimization problem. Since the underlying transformation of the objective function is linear and bijective and hence preserves the combinatorial structure of the respective problems, our results

immediately lead to efficient solution strategies as, for example, dynamic programming for shortest path problems. Moreover, all algorithms for specific problems can be used and thus, ordinal optimization problems with K categories can be solved in the same running time as the associated multi-objective optimization problem with K objective functions. The respective transformations are based on a representation of dominance relations by cones.

Contribution. Parts of the results in this chapter are published in Klamroth et al., 2022b. The work extends the discussion on optimality concepts for ordinal coefficients from Schäfer et al., 2021. The main results of this chapter were also presented at the international conference on Multicriteria Decision Making (MCDM) in June 2022 in Portsmouth in form of a comic. It can be found in the appendix in Chapter A.

Organization of the chapter. In Section 5.1 we review three optimality concepts for ordinal optimization. Two of them are based on Schäfer et al., 2021. These results and optimality concepts are then extended and re-interpreted as a special case of cone-optimality (see, e.g., Engau, 2007) in Section 5.2. A detailed analysis of the properties of the corresponding ordering cones then leads to a linear transformation of ordinal optimization problems to associated multi-objective optimization problems with the Pareto cone defining dominance and optimality. This transformation is used in Section 5.3 to formulate a general algorithm for solving ordinal optimization problems. Furthermore, we investigate the relation between the definition of ordinal optimality and the weight space decomposition of the associated multi-objective problem. Moreover, we compare in Section 5.4 the weight space decomposition for ordinal optimization problems in the context of olympic rankings with another illustration suggested in the New York Times, see Katz, 2022. In Section 5.5 we investigate the effect of the Pareto cone versus the ordering cone for ordinal objective functions on some test instances. In Section 5.6 we extend our results to more general problem types with additional real-valued objective functions. We conclude in Section 5.7 with a summary and an outlook on future research.

5.1. Single-objective Combinatorial Optimization with Ordinal Costs

In this section we describe two objective functions for ordinal optimization problems as well as different optimality concepts. Furthermore, we investigate the interrelation between those optimality concepts and the optimality concepts for ordinal optimization problems, where all solutions have the same length from the last chapter.

5.1.1. Problem Definition

We consider combinatorial optimization problems with an ordinal objective function, which is similar to the objective function introduced in Section 4.1. In general, an *or-*

dinal optimization problem (OOP) can be formulated as

$$\begin{aligned} \text{“ordinally minimize”} \quad & o(x) \\ \text{s. t.} \quad & x \in X, \end{aligned} \tag{OOP}$$

where X is the set of feasible solutions. We assume that X is a subset of the power set of a finite discrete set E , i.e., $X \subseteq 2^E$. The main difference to problem (OOP_r) from Section 4.1 is that the feasible solutions can have arbitrary length. Beside that we repeat that every element of E is assigned to one of K ordered categories. This assignment is encoded by a mapping $o : E \rightarrow \mathcal{C}$ with $\mathcal{C} = \{\eta_1, \dots, \eta_K\}$. We assume again that category η_i with $i \in \{1, \dots, K-1\}$ is strictly preferred over category η_{i+1} , written as $\eta_i \prec \eta_{i+1}$. The objective function of a feasible solution $x = \{e_1, \dots, e_n\}$ is given by the *ordinal vector* $o(x) = \text{sort}(o(e_1), \dots, o(e_n))$, where the operator $\text{sort}()$ means that the components of $o(x)$ are sorted w.r.t. non-decreasing preferences, i.e., $o_1(x) \preceq o_2(x) \preceq \dots \preceq o_n(x)$. Note, that different feasible solutions may have different numbers of elements, and hence the length of the ordinal vector $o(x)$ may vary for different $x \in X$.

Instead of using the un-aggregated, ordered ordinal vector $o(x)$ one can count the number of elements in a feasible solution per category. Accordingly, we recall the definition of the *counting vector* $c : X \rightarrow \mathbb{Z}_{\geq}^K$ with $\mathbb{Z}_{\geq}^K := \{y \in \mathbb{Z}^K : y_i \geq 0 \text{ for all } i = 1, \dots, K\}$ which we already introduced in Section 4.1. Thereby, the i -th component of $c(x)$ equals the number of elements in x which are in category η_i , i. e., $c_i(x) = |\{e \in x : o(e) = \eta_i\}|$. Obviously, there is a one to one correspondence between the vectors $o \in \mathbb{R}^n$ and $c \in \mathbb{R}^K$, since the ordinal vector o can be determined from a given counting vector c by $o_i(x) = \eta_j$ with $j = \text{argmin}\{j \in \{1, \dots, K\} : i \leq \sum_{l=1}^j c_l(x)\}$. Note again that the number of elements n of a feasible solution, and hence the length of the ordinal vectors $o \in \mathbb{R}^n$, may vary while the number of categories K and therefore the length of the counting vectors $c \in \mathbb{R}^K$ is fixed. Hence, we get the following formulation of an *ordinal counting optimization problem* (OCOP)

$$\begin{aligned} \text{“ordinally optimize”} \quad & c(x) \\ \text{s. t.} \quad & x \in X, \end{aligned} \tag{OCOP}$$

which will be shown to be equivalent to problem (OOP) for an appropriate definition of “ordinal minimization” and “ordinal optimization”. As we define also a concept that maximizes $c(x)$ in (OCOP), we use here the term “optimization” instead of “minimization”.

In the following, we also consider an *incremental tail counting vector* $\tilde{c} \in \mathbb{R}^K$ that counts, in its i -th component, the number of elements of a feasible solution x which are in category η_i or worse, i.e., $\tilde{c}_i(x) = |\{e \in x : \eta_i \preceq o(e)\}| = \sum_{j=i}^K c_j(x)$. In particular, the total number of elements of a solution x is given in the first component of \tilde{c} , i.e., $|x| = \tilde{c}_1(x) = \sum_{i=1}^K c_i(x)$.

As an example, consider the shortest path problem shown in Figure 5.1 together with the outcome vectors $o(x)$ (for problem (OOP)) and $c(x)$ (for problem (OCOP)) for all feasible solutions $x \in X$. In addition, the incremental tail counting vector $\tilde{c}(x)$ is given for all $x \in X$.

	o	c	\tilde{c}
$x^1 = \{e_1, e_2, e_5\}$	$\begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}$
$x^2 = \{e_4, e_5\}$	$\begin{pmatrix} \eta_1 \\ \eta_3 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$
$x^3 = \{e_1, e_3\}$	$\begin{pmatrix} \eta_2 \\ \eta_2 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix}$
$x^4 = \{e_6, e_8\}$	$\begin{pmatrix} \eta_2 \\ \eta_2 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix}$
$x^5 = \{e_4, e_7, e_8\}$	$\begin{pmatrix} \eta_1 \\ \eta_1 \\ \eta_2 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 1 \\ 0 \end{pmatrix}$
$x^6 = \{e_1, e_2, e_7, e_8\}$	$\begin{pmatrix} \eta_1 \\ \eta_1 \\ \eta_2 \\ \eta_2 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 2 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 4 \\ 2 \\ 2 \\ 0 \end{pmatrix}$

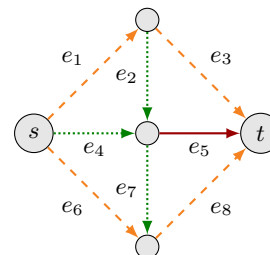


Figure 5.1.: Instance of an ordinal shortest path problem. A dotted-green edge is in the best category η_1 , a dashed-orange edge is in category η_2 and a solid-red edge is in the worst category η_3 . All feasible $s - t$ paths x^i , $i = 1, \dots, 6$ and their respective objective function vectors are given.

5.1.2. Optimality Concepts for Ordinal Objective Functions

In the following, we review three different concepts of optimality for ordinal optimization. All of them try to answer the question what minimization could mean for the problems (OOP) and (OCOP). The first concept is to use *numerical representations* that assign a numerical value to every category such that the order of the categories is respected. In this context, a numerical representation respects the order of the categories whenever the numerical value of a better category is strictly smaller than the numerical value of a less preferred category. If we take the sum over all numerical values of a vector $o(x')$ for a feasible solution x' , we obtain a unique numerical value that can be compared to the corresponding numerical value of another feasible solution \hat{x} . A feasible solution x' is called *efficient* if there is no other feasible solution \hat{x} which is better w.r.t. *all* numerical representations.

The second concept is to maximize the number of elements in the good categories, and the third concept is to minimize the number of elements in the bad categories. After the formal introduction of these three optimality concepts, we investigate their interrelation.

Optimality by Numerical Representations We consider the combinatorial optimization problems (OOP) and (OCOP). The concept of *optimality by numerical representation*

for ordinal objectives as introduced in Schäfer et al., 2021 is based on a previous and more general work of Fishburn, 1999. It assigns an order preserving numerical value to each category. Following Schäfer et al., 2021, we call a function $\nu : \mathcal{C} \rightarrow \mathbb{Z}_{\geq}$ a *numerical representation* if

$$\eta_i \prec \eta_j \iff \nu(\eta_i) < \nu(\eta_j) \text{ for all } i, j \in \{1, \dots, K\}.$$

Note that we assume strictly ordered categories, i.e., there are no categories that are indifferent. As a consequence, we do not allow $\nu(\eta_i) = \nu(\eta_j)$ for $i \neq j$ since this would make two different categories indistinguishable in the numerical representation. Let \mathcal{V} denote the set of all numerical representations for a given number of categories K .

For a given numerical representation ν , we define the numerical value of a feasible solution $x = \{e_1, \dots, e_n\} \in X$ w.r.t. ν (cf. Schäfer et al., 2021) as

$$\nu(x) := \sum_{i=1}^n \nu(o(e_i)) = \sum_{i=1}^K \nu(\eta_i) \cdot c_i(x).$$

The numerical value $\nu(x)$ of a feasible solution $x \in X$ can be evaluated in different ways by re-arranging the terms and using the counting vector c or the incremental tail counting vector \tilde{c} , respectively:

$$\begin{aligned} \nu(x) &= \sum_{i=1}^K \nu(\eta_i) \cdot c_i(x) \\ &= \sum_{i=1}^{K-1} \nu(\eta_i) \left(\sum_{j=i}^K c_j(x) - \sum_{j=i+1}^K c_j(x) \right) + \nu(\eta_K) c_K(x) \\ &= \nu(\eta_1) \cdot \sum_{i=1}^K c_i(x) + \sum_{i=2}^K (\nu(\eta_i) - \nu(\eta_{i-1})) \cdot \sum_{j=i}^K c_j(x) \\ &= \nu(\eta_1) \cdot \tilde{c}_1(x) + \sum_{i=2}^K (\nu(\eta_i) - \nu(\eta_{i-1})) \cdot \tilde{c}_i(x). \end{aligned}$$

An illustration for different ways to evaluate $\nu(x)$ is given in Figure 5.2.

Example 5.1. *We apply the concept of numerical representations to the shortest path problem given in Figure 5.1. As a motivation, suppose that there are two decision makers A and B who have to select a most preferred path. They would agree, for example, that path x^1 is worse than path x^2 , because x^1 has a dashed-orange edge more than x^2 and, other than that, their outcome vectors are the same. But they do not agree on the question whether x^2 or x^5 is preferred, because decision maker A chooses the numerical representation $\nu_A(\eta_1) = 1$, $\nu_A(\eta_2) = 2$ and $\nu_A(\eta_3) = 5$, while decision maker B chooses $\nu_B(\eta_1) = 2$, $\nu_B(\eta_2) = 3$ and $\nu_B(\eta_3) = 4$. Therefore, decision maker A would prefer path x^5 because $\nu_A(x^5) = 4 < 6 = \nu_A(x^2)$ while decision maker B would prefer path x^2 because $\nu_B(x^2) = 6 < 7 = \nu_B(x^5)$. Hence, the path x^2 does not ordinally dominate the path x^5 , i.e., x^2 is not better than x^5 for all numerical representations. Similarly, the path x^5 does not ordinally dominate the path x^2 since also x^5 is not better than x^2 for all numerical representations.*

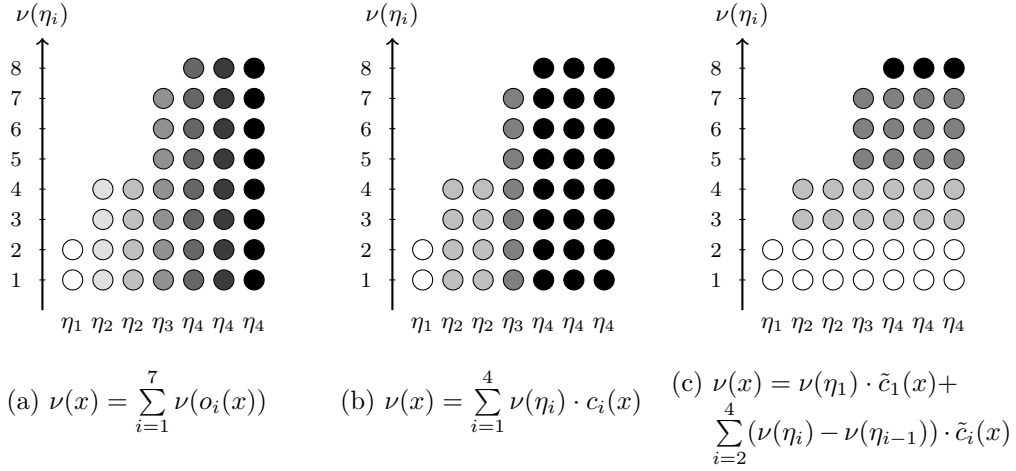


Figure 5.2.: Consider an example with $n = 7$ and $K = 4$. Different ways to compute the numerical value of a feasible solution x with $o(x) = (\eta_1, \eta_2, \eta_2, \eta_3, \eta_4, \eta_4, \eta_4)^\top$, $c(x) = (1, 2, 1, 3)^\top$, $\nu(\eta_1) = 2$, $\nu(\eta_2) = 4$, $\nu(\eta_3) = 7$ and $\nu(\eta_4) = 8$ are illustrated. The different colors represent the summands and visualize the different slicing strategies.

Definition 5.2 (cf. Schäfer et al., 2021). *Let $x', \hat{x} \in X$ be feasible solutions. Then,*

1. x' weakly ordinally dominates \hat{x} , $o(x')$ weakly ordinally dominates $o(\hat{x})$ and $c(x')$ weakly ordinally dominates $c(\hat{x})$, denoted by $x' \preceq_\nu \hat{x}$, $o(x') \preceq_\nu o(\hat{x})$, $c(x') \preceq_\nu c(\hat{x})$, respectively, if and only if for every $\nu \in \mathcal{V}$, it holds that $\nu(x') \leq \nu(\hat{x})$.
2. x' ordinally dominates \hat{x} , $o(x')$ ordinally dominates $o(\hat{x})$ and $c(x')$ ordinally dominates $c(\hat{x})$, denoted by $x' \prec_\nu \hat{x}$, $o(x') \prec_\nu o(\hat{x})$, $c(x') \prec_\nu c(\hat{x})$, respectively, if and only if x' weakly ordinally dominates \hat{x} and there exists $\nu^* \in \mathcal{V}$ such that $\nu^*(x') < \nu^*(\hat{x})$.
3. $x^* \in X$ is called ordinally efficient, if there does not exist an $x \in X$ such that $x \prec_\nu x^*$.
4. $o(x^*)$ and $c(x^*)$ are called ordinally non-dominated outcome vectors of the problems (OOP) and (OCOP), respectively, if x^* is ordinally efficient.

We write “ \min_{\preceq_ν} ”, for example, instead of “ordinally minimize” in problems (OOP) and (OCOP), to clarify that this optimality concept is used. Note that this concept of ordinal optimality is defined for general combinatorial optimization problems, i.e., we do not assume that all solutions have the same length as in Definition 4.1 of ordinal optimality in the last chapter. However, both definitions of ordinal optimality are equivalent, if we assume that all solutions have the same length, see Theorem 5.13.

Optimality by Maximization of Elements in Good Categories Another optimality concept in ordinal optimization is to maximize the number of elements in good categories. The intuition behind this concept is that solutions with many good elements are to be preferred over solutions with few good elements. The drawback, however, is that this concept rewards solutions with larger numbers of elements as long as these are in (relatively)

good categories, which may not be wanted in practice. This optimality concept is defined only for the problem (OCOP), as we need the counting vector c for its definition.

Definition 5.3. We say x' weakly head-dominates \hat{x} , denoted by $x' \geq_h \hat{x}$ or $c(x') \geq_h c(\hat{x})$, if and only if

$$\sum_{i=1}^j c_i(x') \geq \sum_{i=1}^j c_i(\hat{x}) \text{ for all } j = 1, \dots, K. \quad (5.1)$$

Furthermore, x' head-dominates \hat{x} , denoted by $x' \gg_h \hat{x}$ or $c(x') \gg_h c(\hat{x})$, if and only if (5.1) holds and $c(x') \neq c(\hat{x})$. Moreover, $x^* \in X$ is called head-efficient if there is no $x \in X$ such that $x \geq_h x^*$. The corresponding outcome vector $c(x^*)$ is called head-non-dominated.

Optimality by Minimization of Elements in Bad Categories The drawback that longer solutions may be preferred over shorter solutions, as long as the elements are in good categories, can be avoided by taking the converse perspective, i.e., when minimizing the number of elements in the bad categories. Again, this optimality concept is defined only for the problem (OCOP).

Definition 5.4. We say x' weakly tail-dominates \hat{x} , denoted by $x' \leq_t \hat{x}$ or $c(x') \leq_t c(\hat{x})$, if and only if

$$\tilde{c}_j(x') = \sum_{i=j}^K c_i(x') \leq \sum_{i=j}^K c_i(\hat{x}) = \tilde{c}_j(\hat{x}) \text{ for all } j = 1, \dots, K. \quad (5.2)$$

Again, x' tail-dominates \hat{x} , denoted by $x' \ll_t \hat{x}$ or $c(x') \ll_t c(\hat{x})$, if and only if (5.2) holds and $c(x') \neq c(\hat{x})$. Moreover, $x^* \in X$ is called tail-efficient if there is no $x \in X$ such that $x \leq_t x^*$. The corresponding outcome vector $c(x^*)$ is called tail-non-dominated.

Remark 5.5. Note that head-dominance as well as tail-dominance are equivalently defined on the feasible set $X \subseteq 2^E$ and on its image set $c(X) \subseteq \mathbb{R}^K$. The definitions immediately extend to the complete \mathbb{R}^K .

5.1.3. Properties of and Interrelations between Optimality Concepts for Ordinal Optimization

In addition to the concepts described above, there are further ways to define efficiency. This has been done, for example, in Schäfer et al., 2020 for the ordinal shortest path problem. Their definition has the disadvantage that *Bellman's principle of optimality* (see Bellman, 1957) does not hold in general, i.e., not every subpath of an efficient path is necessarily efficient w.r.t. this optimality concept. The definition of head-optimality has the same disadvantage, see Remark 5.10 below for more details. In contrast, the definitions of ordinal optimality and tail-optimality can be proven to be equivalent. Moreover, they are compliant with Bellman's principle of optimality. Note that, for the special case of a knapsack problem, this was shown in Schäfer et al., 2021.

Lemma 5.6. For feasible solutions $\bar{x} = \{\bar{e}_1, \dots, \bar{e}_n\}, x' = \{e'_1, \dots, e'_m\} \in X$ with $n < m$ and $c_i(\bar{x}) \leq c_i(x')$ for all $i = 1, \dots, K$ it holds that $\bar{x} \preceq_\nu x'$.

Proof. First let $\nu \in \mathcal{V}$ be an arbitrary numerical representation. Then it holds that $\nu(\bar{x}) = \sum_{i=1}^K \nu(\eta_i) \cdot c_i(\bar{x}) \leq \sum_{i=1}^K \nu(\eta_i) \cdot c_i(x') = \nu(x')$, i.e., $\bar{x} \preceq_{\nu} x'$. Note that since $n < m$, and due to the above assumptions, there must exist a category η_j , $j \in \{1, \dots, K\}$ such that $c_j(\bar{x}) < c_j(x')$. Therefore, there exists a numerical representation ν^* such that $\nu^*(\eta_j) > 0$ and thus $\nu^*(\bar{x}) < \nu^*(x')$, which concludes the proof. \square

Note that the condition of Lemma 5.6 is always satisfied if $\bar{x} = \{\bar{e}_1, \dots, \bar{e}_n\} \in X$ and $x' = \{e'_1, \dots, e'_m\} \in X$ with $\{o(\bar{e}_1), \dots, o(\bar{e}_n)\} \subsetneq \{o(e'_1), \dots, o(e'_m)\}$. Thus, in Example 5.1 the path x^2 is always preferred over the path x^1 . This is also the case for tail-efficiency, but not for head-efficiency, see Remark 5.10 below. In many application contexts this is a meaningful property, since adding additional elements to a solution (no matter from which category) does generally not improve the solution quality. As an example, we refer again to paths representing bicycle routes as in the app *geovelo*, where we are not interested in routes that are unnecessarily long.

Lemma 5.7 (Schäfer et al. 2021). *The ordinal dominance relation \preceq_{ν} defined on the feasible set X is a preorder, i.e., it is reflexive and transitive.*

Proof. Let $x', \hat{x}, \bar{x} \in X$. Obviously, $\nu(x') \leq \nu(x')$ holds for every $\nu \in \mathcal{V}$. Hence, the relation \preceq_{ν} is reflexive. If $\nu(x') \leq \nu(\hat{x})$ and $\nu(\hat{x}) \leq \nu(\bar{x})$ for every $\nu \in \mathcal{V}$, it follows that $\nu(x') \leq \nu(\bar{x})$ for every $\nu \in \mathcal{V}$ by definition and therefore, we have shown transitivity. \square

Note that the ordinal dominance relation \preceq_{ν} is in general not antisymmetric on the feasible set X since two different feasible solutions may have the same number of elements in each category like the paths x^3 and x^4 in Example 5.1.

The following results show that (weak) ordinal dominance and (weak) tail-dominance are actually equivalent on the feasible set X .

Lemma 5.8 (Schäfer et al. 2021). *Let $x', \hat{x} \in X$ be two feasible solutions. Then x' weakly ordinally dominates \hat{x} , i.e., $x' \preceq_{\nu} \hat{x}$ if and only if $x' \preceq_t \hat{x}$.*

Proof. The proof is a simplified variant of the proof in Schäfer et al., 2021. Note, that they consider maximization problems while we consider minimization problems.

First we show by contradiction that $x' \preceq_{\nu} \hat{x}$ implies $x' \preceq_t \hat{x}$. Let $x', \hat{x} \in X$ and let $j^* \in \{1, \dots, K\}$ with

$$\sum_{i=j^*}^K c_i(x') > \sum_{i=j^*}^K c_i(\hat{x}).$$

The idea of the proof is to make the bad categories $\eta_{j^*}, \dots, \eta_K$ very expensive, such that an element of this category can not be replaced by elements of the lower categories. Hence, we define the numerical representation

$$\nu(\eta_i) = \begin{cases} i, & \text{if } i < j^* \\ i + 2|\hat{x}|K, & \text{if } i \geq j^*. \end{cases}$$

This implies

$$\begin{aligned}
 \nu(x') &\geq 2|\hat{x}|K \cdot \sum_{i=j^*}^K c_i(x') \geq 2|\hat{x}|K \cdot \left(1 + \sum_{i=j^*}^K c_i(\hat{x})\right) \\
 &> |\hat{x}|K + 2|\hat{x}|K \cdot \sum_{i=j^*}^K c_i(\hat{x}) \geq \sum_{i=1}^K i c_i(\hat{x}) + 2|\hat{x}|K \cdot \sum_{i=j^*}^K c_i(\hat{x}) \\
 &= \sum_{i=1}^{j^*-1} i c_i(\hat{x}) + \sum_{i=j^*}^K i c_i(\hat{x}) + \sum_{i=j^*}^K 2|\hat{x}|K \cdot c_i(\hat{x}) = \nu(\hat{x}).
 \end{aligned}$$

For the other direction we use the reformulation of $\nu(x)$, which is visualized in Figure 5.2(c). It follows that for any $\nu \in \mathcal{V}$

$$\begin{aligned}
 \nu(x') &= \nu(\eta_1) \tilde{c}_1(x') + \sum_{i=2}^K (\nu(\eta_i) - \nu(\eta_{i-1})) \tilde{c}_i(x') \\
 &\leq \nu(\eta_1) \tilde{c}_1(\hat{x}) + \sum_{i=2}^K (\nu(\eta_i) - \nu(\eta_{i-1})) \tilde{c}_i(\hat{x}) = \nu(\hat{x}).
 \end{aligned}$$

The inequality holds because of the assumption $\tilde{c}_j(x') \leq \tilde{c}_j(\hat{x})$ for all $j = 1, \dots, K$ and $\nu(\eta_i) - \nu(\eta_{i-1}) > 0$ for all $\nu \in \mathcal{V}$ and $i = 2, \dots, K$. Hence, we have shown that $\sum_{i=j}^K c_i(x') \leq \sum_{i=j}^K c_i(\hat{x})$ for all $j = 1, \dots, K$ implies $x' \preceq_{\nu} \hat{x}$, which concludes the proof. \square

Lemma 5.9. *Let $x', \hat{x} \in X$ be two feasible solutions. Then x' ordinally dominates \hat{x} , i.e., $x' \preceq_{\nu} \hat{x}$ if and only if $x' \leq_t \hat{x}$.*

Proof. We first show that $x' \preceq_{\nu} \hat{x}$ implies $x' \leq_t \hat{x}$. If x' ordinally dominates \hat{x} , then $x' \preceq_{\nu} \hat{x}$ which implies $x' \leq_t \hat{x}$ due to Lemma 5.8. It remains to show that $c(x') \neq c(\hat{x})$. As x' ordinally dominates \hat{x} it holds that there is a numerical representation ν^* such that $\nu^*(x') < \nu^*(\hat{x})$. Hence,

$$\begin{aligned}
 0 &< \nu^*(\hat{x}) - \nu^*(x') \\
 \iff 0 &< \sum_{i=1}^K \nu^*(\eta_i) (c_i(\hat{x}) - c_i(x')).
 \end{aligned}$$

Since $\nu(\eta_i) > 0$ for all $i = 1, \dots, K$, it holds $c(x') \neq c(\hat{x})$. Consequently, we have shown that when x' ordinally dominates \hat{x} , then $x' \leq_t \hat{x}$ holds and $c(x') \neq c(\hat{x})$.

For the other direction it is sufficient to show that $c(x') \neq c(\hat{x})$ implies that there exists a numerical representation $\nu^* \in \mathcal{V}$ such that $\nu^*(x') < \nu^*(\hat{x})$. Let j^* be the largest category such that $c_{j^*}(x') \neq c_{j^*}(\hat{x})$. This implies

$$\sum_{i=j^*}^K c_i(x') < \sum_{i=j^*}^K c_i(\hat{x}).$$

Now the result follows analogously to the proof of Lemma 5.8 with exchanged roles of x' and \hat{x} . \square

Remark 5.10. Lemma 5.8 (and thus also Lemma 5.9) does not hold in general for the relation \succsim_h . As a counter example, consider the paths x^1 and x^2 with $c(x^1) = (1, 1, 1)^\top$ and $c(x^2) = (1, 0, 1)^\top$ from Figure 5.1. Obviously, x^1 head-dominates x^2 . But for every numerical representation it follows $\nu(x^1) > \nu(x^2)$, which contradicts $x^1 \preceq_\nu x^2$.

Note that the crucial point in the counter example given in Remark 5.10 is the different cardinality of the solutions, $|x^1| \neq |x^2|$. For ordinal optimization problems with fixed cardinality, for which matroid optimization problems as studied in Klamroth et al., 2022a and in both last chapters are an example, it can be shown that head- and tail-dominance are equivalent.

Lemma 5.11. *If all feasible solutions have the same cardinality, i.e., if $|x'| = |\hat{x}|$ for all $x', \hat{x} \in X$, then head- and tail-dominance as defined in Definitions 5.3 and 5.4, respectively, are equivalent.*

Proof. First assume that x' head-dominates \hat{x} , i.e., inequality (5.1) is satisfied. We show that then x' also tail-dominates \hat{x} , i.e., inequality (5.2) holds. Towards this end, let $j \in \{2, \dots, K\}$. Then

$$\begin{aligned} \sum_{i=j}^K c_i(x') &= \sum_{i=1}^K c_i(x') - \sum_{i=1}^{j-1} c_i(x') \stackrel{(5.1)}{\leq} \sum_{i=1}^K c_i(x') - \sum_{i=1}^{j-1} c_i(\hat{x}) \\ &\stackrel{|x'|=|\hat{x}|}{=} \sum_{i=1}^K c_i(\hat{x}) - \sum_{i=1}^{j-1} c_i(\hat{x}) = \sum_{i=j}^K c_i(\hat{x}), \end{aligned}$$

which implies (5.2).

Now let x' tail-dominate \hat{x} , i.e., (5.2) is satisfied. We show that then also x' head-dominates \hat{x} , i.e., (5.1) holds. Hence, let $j \in \{1, \dots, K-1\}$. Then

$$\begin{aligned} \sum_{i=1}^j c_i(x') &= \sum_{i=1}^K c_i(x') - \sum_{i=j+1}^K c_i(x') \stackrel{(5.2)}{\geq} \sum_{i=1}^K c_i(x') - \sum_{i=j+1}^K c_i(\hat{x}) \\ &\stackrel{|x'|=|\hat{x}|}{=} \sum_{i=1}^K c_i(\hat{x}) - \sum_{i=j+1}^K c_i(\hat{x}) = \sum_{i=1}^j c_i(\hat{x}), \end{aligned}$$

which implies (5.1). □

Now, we can easily show that ordinal dominance as defined in this chapter in Definition 5.2 is indeed equivalent to the ordinal dominance defined in Chapter 4 in Definition 4.1 if all feasible solutions have the same cardinality.

Lemma 5.12. *If all feasible solutions have the same cardinality, i.e., if $|x| = r \in \mathbb{Z}_{>}$ for all $x \in X$, then ordinal dominance and head-dominance as defined in Definitions 4.1 and 5.3, respectively, are equivalent.*

Proof. Let $x', \hat{x} \in X$ and let r be the cardinality of all feasible solutions.

We first show that $x' \preceq_o \hat{x}$ implies $x' \succeq_h \hat{x}$. By definition, $x' \preceq_o \hat{x}$ is equivalent to $o_i(x') \preceq o_i(\hat{x})$ for all $i = 1, \dots, r$ and $o(x') \neq o(\hat{x})$. Hence, it follows that $\nu(x') = \sum_{i=1}^r \nu(o_i(x')) < \sum_{i=1}^r \nu(o_i(\hat{x})) = \nu(\hat{x})$ for all numerical representations $\nu \in \mathcal{V}$ as numerical representations preserve the order of the categories. We have shown that $x' \preceq_o \hat{x}$ implies $x' \preceq_\nu \hat{x}$ and due to Lemmas 5.9 and 5.11 $x' \preceq_o \hat{x}$ implies $x' \succeq_h \hat{x}$.

For the other direction we assume $x' \succeq_h \hat{x}$ and we show that $o_i(x') \preceq o_i(\hat{x})$ for all $i = 1, \dots, \sum_{k=1}^j c_k(x')$ and for all $j = 1, \dots, K$ by induction on j . Note that $\sum_{k=1}^j c_k(x')$ corresponds to the number of elements in category η_j or better and $\sum_{k=1}^K c_k(x') = r$. Moreover, note that this proposition relies on the fact that the vector o is sorted in non-decreasing order w.r.t. the categories.

Initial case ($j = 1$): If x' head-dominates \hat{x} , it holds that $c_1(x') \geq c_1(\hat{x})$. Hence, it holds that $o_i(x') = \eta_1$ and $o_i(\hat{x}) \in \mathcal{C}$ for all $i = 1, \dots, c_1(x')$. Thus, $o_i(x') \preceq o_i(\hat{x})$ for all $i = 1, \dots, \sum_{k=1}^1 c_k(x')$ holds.

Induction step ($j \rightarrow j+1$): We deduce from $\sum_{k=1}^j c_i(x') \geq \sum_{k=1}^j c_i(\hat{x})$ that $o_i(x') = \eta_{j+1}$ and $o_i(\hat{x}) \in \{\eta_{j+1}, \dots, \eta_K\}$ for all $i = \left(\sum_{k=1}^j c_k(x')\right) + 1, \dots, \sum_{k=1}^{j+1} c_k(x')$. Together with the *induction hypothesis*, i.e., $o_i(x') \preceq o_i(\hat{x})$ for all $i = 1, \dots, \sum_{k=1}^j c_k(x')$, it follows that $o_i(x') \preceq o_i(\hat{x})$ for all $i = 1, \dots, \sum_{k=1}^{j+1} c_k(x')$.

Hence, we have shown that $x' \succeq_h \hat{x}$ implies $x' \preceq_o \hat{x}$. \square

Theorem 5.13. *The definitions of ordinal dominance as well as head- and tail-dominance, i.e., Definitions 4.1, 5.2, 5.3 and 5.4, respectively, are equivalent if the cardinality of feasible solutions is fixed.*

Proof. This follows immediately from Lemmas 5.9, 5.11 and 5.12. \square

Lemma 5.14. *The relation \leq_t is a partial order on \mathbb{R}^K , i.e., it is reflexive, transitive and antisymmetric. Moreover, the relation \leq_t is a strict partial order on \mathbb{R}^K , i.e., it is irreflexive and transitive.*

Proof. Let $u \in \mathbb{R}^K$. Then $u \leq_t u$, i.e., \leq_t is reflexive. Furthermore, for $u, v, w \in \mathbb{R}^K$ such that $u \leq_t v$ and $v \leq_t w$ it follows that $\sum_{i=j}^K u_i \leq \sum_{i=j}^K v_i \leq \sum_{i=j}^K w_i$ for all $j = 1, \dots, K$, i.e., $u \leq_t w$ which means \leq_t is transitive. To show that the relation \leq_t is antisymmetric, consider two vectors $u, v \in \mathbb{R}^K$ with $u \leq_t v$ and $v \leq_t u$. Then $\sum_{i=j}^K u_i = \sum_{i=j}^K v_i$ for all $j = 1, \dots, K$. This implies that $u = v$ and hence \leq_t is antisymmetric. Therefore, \leq_t is a partial order.

Now consider the relation \leq_t . Since $u \not\leq_t u$ for all $u \in \mathbb{R}^K$, it holds that \leq_t is irreflexive. It remains to show that \leq_t is transitive. Towards this end, consider three vectors $u, v, w \in \mathbb{R}^K$ such that $u \leq_t v$ and $v \leq_t w$. This implies $\sum_{i=j}^K u_i \leq \sum_{i=j}^K v_i \leq \sum_{i=j}^K w_i$ for all $j = 1, \dots, K$, and there exist indices $s, t \in \{1, \dots, K\}$ such that $\sum_{i=s}^K u_i < \sum_{i=s}^K v_i$ and $\sum_{i=t}^K v_i < \sum_{i=t}^K w_i$. Hence, we can conclude that $\sum_{i=j}^K u_i \leq \sum_{i=j}^K w_i$ for all $j = 1, \dots, K$ and $u \neq w$, i.e., $u \leq_t w$. Consequently, we have shown that \leq_t is irreflexive and transitive which concludes the proof. \square

As a consequence of the discussion in Sections 5.1.2 and 5.1.3, we focus in the following on the ordinal optimization problem (OOP) w.r.t. optimality by numerical representations,

or equivalently, on the ordinal counting optimization problem (OCOP) w.r.t. the concept of tail-dominance.

5.2. Ordinal Optimality versus Pareto Optimality: An Interpretation based on Ordering Cones

In this section we first introduce the cone which belongs to tail-dominance. Afterwards, we use this cone to transform ordinal optimization problems into multi-objective combinatorial optimization problems with binary costs.

5.2.1. The Ordinal Cone

In the following we show that tail-dominance, represented by the binary relation \leq_t , is induced by a polyhedral cone in \mathbb{R}^K . We refer to this cone as the *ordinal cone*. As a first step towards this goal, we prove that \leq_t is compatible with scalar multiplication and addition. As a second step, the associated ordinal cone is constructed and analyzed. Afterwards, we can reinterpret problem (OCOP) based on cone optimality.

Lemma 5.15. *The relation \leq_t on \mathbb{R}^K is compatible with scalar multiplication and with addition.*

Proof. To show that \leq_t is compatible with scalar multiplication, let $\lambda > 0$ and $u, v \in \mathbb{R}^K$ with $u \leq_t v$. It follows that $\lambda \sum_{i=j}^K u_i \leq \lambda \sum_{i=j}^K v_i$ which implies $\sum_{i=j}^K \lambda u_i \leq \sum_{i=j}^K \lambda v_i$ for all $j = 1, \dots, K$. Furthermore, it holds that $\lambda u \neq \lambda v$, and hence $\lambda u \leq_t \lambda v$, which implies that \leq_t is compatible with scalar multiplication.

It remains to show that \leq_t is also compatible with addition. Let $u, v, w \in \mathbb{R}^K$ with $u \leq_t v$, i.e., $\sum_{i=j}^K u_i \leq \sum_{i=j}^K v_i$ for all $j = 1, \dots, K$ and $u \neq v$. This implies that $\sum_{i=j}^K (u_i + w_i) \leq \sum_{i=j}^K (v_i + w_i)$ for all $j = 1, \dots, K$ and $(u+w) \neq (v+w)$, i.e., $u+w \leq_t v+w$. Hence we have proven the compatibility with addition. \square

It can be proven analogously that \leq_t on \mathbb{R}^K is also compatible with scalar multiplication and addition.

Due to Lemma 2.2 and Lemma 5.15 the ordering cone $C_{\leq_t} := \{(v - u) \in \mathbb{R}^K : u \leq_t v\}$ induced by the strict partial order \leq_t is pointed, convex and it does not contain 0. We call this cone the *ordinal cone* to emphasize that C_{\leq_t} equivalently represents ordinal dominance and show that its closure, the cone $C_{\leq_t} \cup \{0\}$, is a polyhedral cone that can be described as the intersection of K halfspaces.

Theorem 5.16. *The closure of the ordinal cone is a polyhedral cone. In particular, it holds that $C_{\leq_t} \cup \{0\} = \text{hcone}(A_{\leq_t})$ with $A_{\leq_t} \in \mathbb{R}^{K \times K}$ given by*

$$A_{\leq_t} = (a_{ij})_{i,j=1,\dots,K} \text{ with } a_{ij} = \begin{cases} 1, & \text{if } i \leq j \\ 0, & \text{otherwise} \end{cases}, \quad \text{i.e., } A_{\leq_t} = \begin{pmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Proof. First note that $0 \in (C_{\leq t} \cup \{0\}) \cap \text{hcone}(A_{\leq t})$. It thus remains to show that for all $\tilde{u} \in \mathbb{R}^K \setminus \{0\}$, it holds that $\tilde{u} \in C_{\leq t}$ if and only if $A_{\leq t} \tilde{u} \geq 0$.

Now let $\tilde{u} \in \mathbb{R}^K \setminus \{0\}$ with $A_{\leq t} \tilde{u} \geq 0$. We define $u := 0 \in \mathbb{R}^K$ and $v := \tilde{u}$. Hence, it holds $v - u = \tilde{u}$ and

$$\begin{aligned} & A_{\leq t} \tilde{u} \geq 0 \text{ and } \tilde{u} \neq 0 \\ \iff & \sum_{i=j}^K \tilde{u}_i \geq 0 \quad \text{for all } j = 1, \dots, K, \text{ and } \tilde{u} \neq 0 \\ \iff & \sum_{i=j}^K v_i \geq \sum_{i=j}^K u_i \quad \text{for all } j = 1, \dots, K, \text{ and } v \neq u \\ \iff & u \leq_t v \\ \iff & \tilde{u} = v - u \in C_{\leq t}. \end{aligned}$$

Thus, we obtain $\text{hcone}(A_{\leq t}) \setminus \{0\} = C_{\leq t}$, which concludes the proof. \square

Theorem 2.1 implies that the closure of the ordinal cone $C_{\leq t} \cup \{0\}$, which is a polyhedral cone by Theorem 5.16, must also have a description based on a finite number of extreme rays. Indeed, the following result provides such a description based on exactly K extreme rays.

Theorem 5.17. *It holds that $\text{hcone}(A_{\leq t}) = \text{vcone}(B_{\leq t})$ for $A_{\leq t}$ defined according to Theorem 5.16 and $B_{\leq t} \in \mathbb{R}^{K \times K}$ given by $B_{\leq t} = (b_{ij})_{i,j=1,\dots,K}$ with*

$$b_{ij} = \begin{cases} 1, & \text{if } i = j \\ -1, & \text{if } i = j - 1, \\ 0, & \text{otherwise} \end{cases} \quad \text{i.e., } B_{\leq t} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}.$$

Proof. We first show that $\text{hcone}(A_{\leq t}) \subseteq \text{vcone}(B_{\leq t})$. Let $d \in \text{hcone}(A_{\leq t})$. Hence, it holds $A_{\leq t} d \geq 0$ which is equivalent to $\sum_{i=j}^K d_i \geq 0$ for all $j = 1, \dots, K$. Set $\lambda_j := \sum_{i=j}^K d_i \geq 0$ and let $B_{j\bullet}$ denote the j -th row of $B_{\leq t}$, for $j = 1, \dots, K$. Then $B_{j\bullet} \lambda = \lambda_j - \lambda_{j+1} = d_j$ for $j = 1, \dots, K - 1$ and $B_{K\bullet} \lambda = \lambda_K = d_K$. Consequently, we have shown that $d \in \text{vcone}(B_{\leq t})$.

For the other direction, let $d \in \text{vcone}(B_{\leq t})$, i.e., $d = B_{\leq t} \lambda$ for some $\lambda \geq 0$. The definition of $B_{\leq t}$ implies that $\sum_{i=j}^K B_{i\bullet} \lambda = \sum_{i=j}^{K-1} (\lambda_i - \lambda_{i+1}) + \lambda_K = \lambda_j$ for all $j = 1, \dots, K - 1$ and $B_{K\bullet} \lambda = \lambda_K$. Hence, it follows that $A_{\leq t} \cdot d = A_{\leq t} \cdot (B_{\leq t} \lambda) = \lambda \geq 0$ and thus $d \in \text{hcone}(A_{\leq t})$, which concludes the proof. \square

Note that these descriptions of the closure of the ordinal cone $C_{\leq t} \cup \{0\}$ are not unique. Indeed, both the normal vectors in $A_{\leq t}$ as well as the extreme rays in $B_{\leq t}$ could be reordered, and they could be multiplied by arbitrary positive scalars without changing the cone that they define. In the particular description given in Theorems 5.16 and

5.17, however, we observe that the matrix $B_{\leq t}$ is the inverse of the matrix $A_{\leq t}$, i.e., $(A_{\leq t})^{-1} = B_{\leq t}$.

Remark 5.18. *Obviously, it holds that $(\text{hcone}(A_{\leq t}))^* = \text{hcone}((B_{\leq t})^\top)$ and, similarly, that $(\text{vcone}(B_{\leq t}))^* = \text{vcone}((A_{\leq t})^\top)$ since the normal vectors of the halfspaces given in $A_{\leq t}$ are orthogonal to the extreme rays contained in $B_{\leq t}$.*

It is easy to see that the Pareto cone C_P is a subset of the ordinal cone, $C_{\leq t}$. Moreover, the dual cone of the ordinal cone is a subset of the Pareto cone, i.e., $(C_{\leq t})^* \subseteq C_P \subseteq C_{\leq t}$. This holds since $z \in C_P$ implies that $z \geq 0$ and hence $A_{\leq t} \cdot z \geq 0$, i.e., $z \in C_{\leq t}$. Moreover, $z^* \in (C_{\leq t})^*$ is equivalent to $(z^*)^\top c \geq 0$ for all $c \in C_{\leq t}$ which implies $z_i^* \geq 0$, because the i -th unit vector $e_i \in \mathbb{R}^K$ is contained in $C_{\leq t}$ for all $i = 1, \dots, K$. These cones and their duals are visualized in Figure 5.3.

Remark 5.19. *Note that head-dominance (c.f. equation (5.1)), represented by the binary relation \geq_h , also induces a polyhedral cone. For $A_{\geq h} = (A_{\leq t})^\top$ and $B_{\geq h} = (B_{\leq t})^\top$ we have that $C_{\geq h} \cup \{0\} = \text{hcone}(A_{\geq h}) = \text{vcone}(B_{\geq h})$.*

Now we can use the ordinal cone $C_{\leq t}$ as described in Definition 2.3 to reformulate the optimization problem (OCOP) as follows:

$$\begin{array}{ll} \min_{C_{\leq t}} & c(x) \\ \text{s. t.} & x \in X. \end{array} \quad (\text{OCOP})$$

Here, $\min_{C_{\leq t}}$ denotes the minimization in the sense of Definition 2.3 for the ordinal cone $C_{\leq t} = \text{hcone}(A_{\leq t}) \setminus \{0\}$. In other words, the $C_{\leq t}$ -non-dominated set of problem (OCOP) is given by $N(Y, C_{\leq t})$, where $Y = c(X)$. In the following, we use this notation to clearly distinguish between the optimization w.r.t. different ordering cones.

5.2.2. Bijective Linear Transformation Between Ordinal and Pareto Optimization

In the previous subsection we showed that tail-dominance, and hence also ordinal dominance due to Lemma 5.9, can be equivalently described by the ordinal cone $C_{\leq t}$. Moreover, the closure $C_{\leq t} \cup \{0\}$ of the ordinal cone is the polyhedral cone $\text{hcone}(A_{\leq t}) = \text{vcone}(B_{\leq t})$ that is spanned by K linearly independent extreme rays in \mathbb{R}^K , c.f. Theorems 5.16 and 5.17. Since the closure of the Pareto cone $C_P \cup \{0\}$ is also a polyhedral cone that is spanned by K linearly independent extreme rays in \mathbb{R}^K (namely the K unit vectors in \mathbb{R}^K), there exists a bijective linear transformation that maps the (closure of the) ordinal cone onto the (closure of the) Pareto cone.

We thus define the following *transformed Pareto cone optimization problem* (TOP)

$$\begin{array}{ll} \min_{C_P} & A_{\leq t} \cdot c(x) \\ \text{s. t.} & x \in X, \end{array} \quad (\text{TOP})$$

where \min_{C_P} denotes the optimization w.r.t. the Pareto cone C_P according to Definition 2.3. Note that the objective vector of problem (TOP) corresponds to the incremental tail counting vector $\tilde{c}(x) = A_{\leq t} \cdot c(x) \in \mathbb{R}^K$ introduced in Section 5.1.1, that counts in its j th

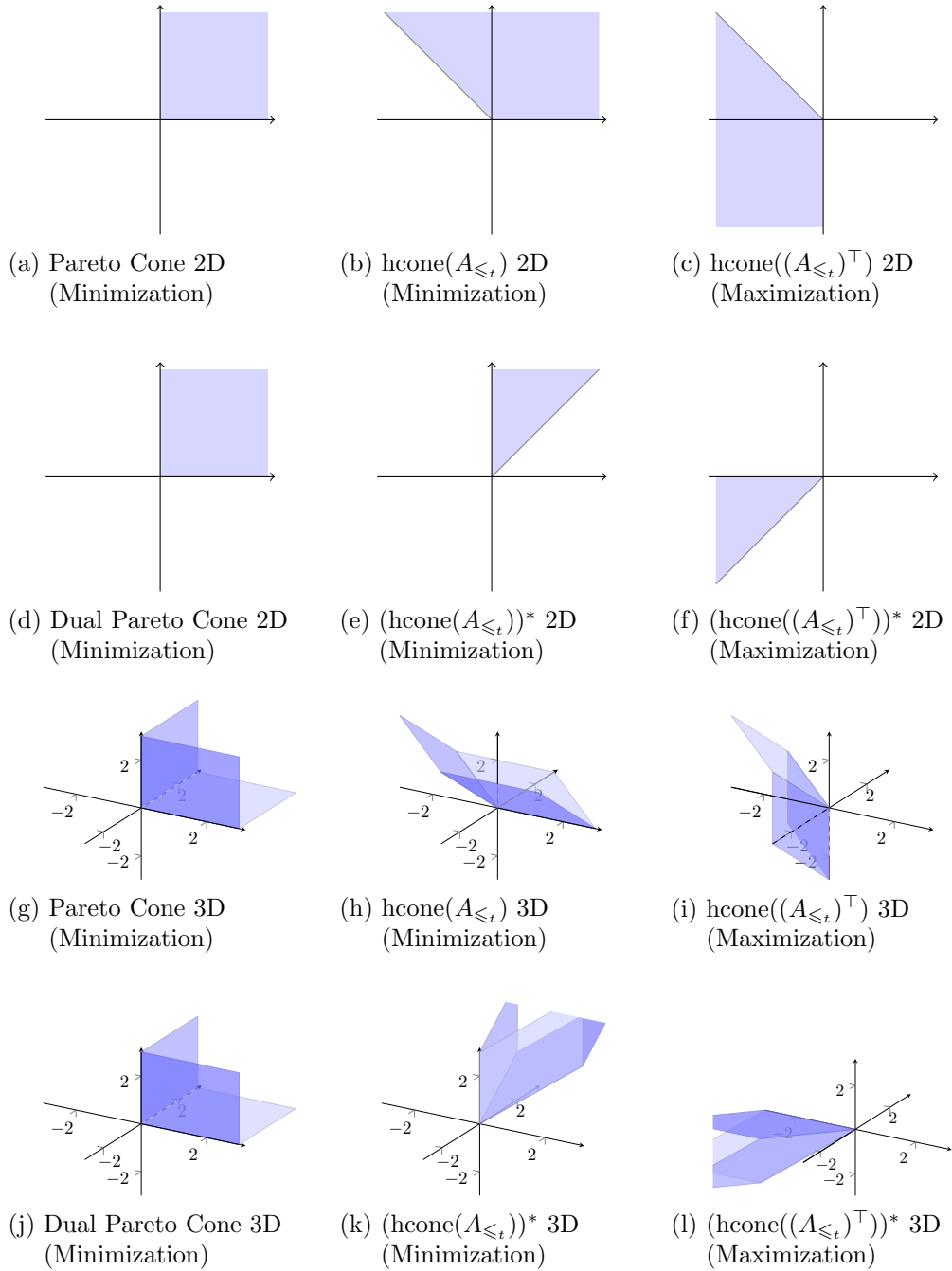


Figure 5.3.: Cones and their dual cones

component the number of elements of x that are in category η_j or worse. Indeed, for a feasible solution $x = \{e_1, \dots, e_n\} \in X$ we get $\tilde{c}(x) = \sum_{i=1}^n \tilde{c}(e_i)$, where

$$\tilde{c}_j(e_i) = \begin{cases} 1, & \text{if } \eta_j \preceq o(e_i) \\ 0, & \text{otherwise} \end{cases} \text{ for all } j = 1, \dots, K.$$

Thus, problem (TOP) is actually a multi-objective optimization problem with K binary objective functions $\tilde{c}_1, \dots, \tilde{c}_K$ defined on the ground set E , and with feasible set $X \subseteq 2^E$. Recall from Section 5.1.1 that $\tilde{c}_1(e) = 1$ for all $e \in E$ and hence $\tilde{c}_1(x)$ simply counts the number of elements in a feasible solution $x \in X$. Moreover, the vector $\tilde{c}(e)$ has the consecutive ones property in the sense that whenever a component of $\tilde{c}(e)$ is zero, then all subsequent components of $\tilde{c}(e)$ are also zero.

As an example, consider the ordinal shortest path problem introduced in Example 5.1. The path x^1 consists of the green-dotted edge e_2 with $\tilde{c}(e_2) = (1, 0, 0)^\top$, the orange-dashed edge e_1 with $\tilde{c}(e_1) = (1, 1, 0)^\top$, and the red-solid edge e_5 with $\tilde{c}(e_5) = (1, 1, 1)^\top$. Hence, we compute $\tilde{c}(x^1) = \tilde{c}(e_2) + \tilde{c}(e_1) + \tilde{c}(e_5) = (3, 2, 1)^\top$, see also Figure 5.1.

In order to show that the ordinal counting optimization problem (OCOP) (and hence the ordinal optimization problem (OOP)) can be solved by using the above transformation to the “standard” multi-objective optimization problem (TOP), we use a classical non-dominance mapping result for polyhedral cones. This result can be found in Engau, 2007 and the references therein among several others. We include a proof, which is similar to the more general proof in Hunt and Wiecek, 2003, for the sake of completeness.

Theorem 5.20 (see, e.g., Engau, 2007). *Let $Y \subset \mathbb{R}^K$ be a nonempty set and let $\text{hcone}(A)$ be a cone induced by a matrix $A \in \mathbb{R}^{m \times K}$. Then it holds*

$$A \cdot \text{N}(Y, \text{hcone}(A) \setminus \{0\}) \subseteq \text{N}(A \cdot Y, C_P).$$

If $\text{rank}(A) = K$, then equality holds, i.e., $A \cdot \text{N}(Y, \text{hcone}(A) \setminus \{0\}) = \text{N}(A \cdot Y, C_P)$. Here, the multiplication of a matrix A with a set Y is defined as $A \cdot Y := \{A \cdot y : y \in Y\}$.

Proof. Suppose that $\bar{y} \in Y$ such that $\bar{y} \in \text{N}(Y, \text{hcone}(A) \setminus \{0\})$ and $A \cdot \bar{y} \notin \text{N}(A \cdot Y, C_P)$. Then, by Definition 2.3, there exists a $\hat{y} \in Y \setminus \{\bar{y}\}$ such that $A \cdot \hat{y} \in (A \cdot \bar{y} - C_P)$, i.e., there exists $d \in C_P$ such that $A \cdot \hat{y} = A \cdot \bar{y} - d$. Hence, it follows that $d = A \cdot \bar{y} - A \cdot \hat{y} = A \cdot (\bar{y} - \hat{y}) \geq 0$ and thus $\bar{d} := \bar{y} - \hat{y} \in \text{hcone}(A) \setminus \{0\}$. Finally, we can deduce that $\hat{y} \in (\bar{y} - \text{hcone}(A) \setminus \{0\})$, with $\hat{y} \in Y$. But then $\bar{y} \notin \text{N}(Y, \text{hcone}(A) \setminus \{0\})$, which contradicts the assumption.

It remains to show that $A \cdot \text{N}(Y, \text{hcone}(A) \setminus \{0\}) \supseteq \text{N}(A \cdot Y, C_P)$ if $\text{rank}(A) = K$. Towards this end, suppose that $\bar{y} \in Y$ such that $A \cdot \bar{y} \in \text{N}(A \cdot Y, C_P)$ and $\bar{y} \notin \text{N}(Y, \text{hcone}(A) \setminus \{0\})$. Hence, there exists a $d \in \text{hcone}(A) \setminus \{0\}$ such that $\hat{y} = \bar{y} - d \in Y$. This implies that $A \cdot \hat{y} = A \cdot \bar{y} - A \cdot d \in A \cdot Y$. From $\text{rank}(A) = K$ and $d \neq 0$ we deduce that $A \cdot d \neq 0$ and thus $A \cdot d \geq 0$. Consequently, $(A \cdot \bar{y} - C_P) \cap A \cdot Y \neq \emptyset$ which contradicts the assumption that $A \cdot \bar{y} \in \text{N}(A \cdot Y, C_P)$. \square

Theorem 5.21. *The set of ordinally efficient solutions for problem (OOP), the set of tail-efficient (ordinally efficient) solutions of problem (OCOP) and the set of Pareto-efficient solutions of problem (TOP) are equal.*

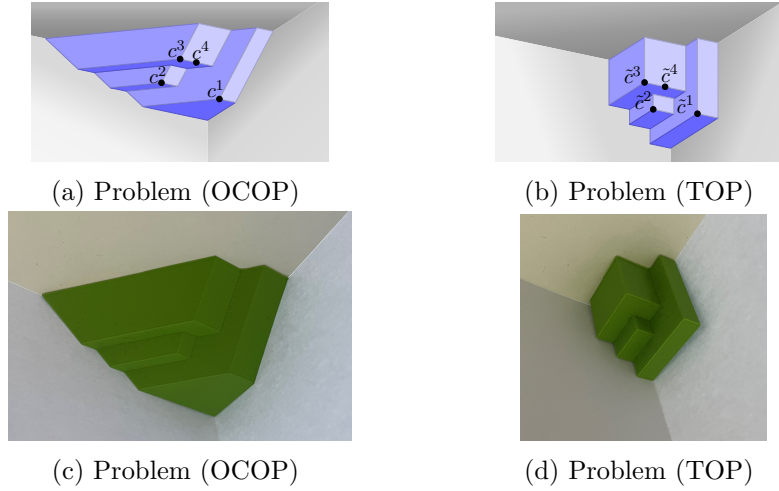


Figure 5.4.: Illustration of the tail-non-dominated outcome vectors of the instance of (OCOP) introduced in Example 5.22 (left) and of the respective Pareto-non-dominated outcome vectors of the transformed problem (TOP) (right). The figures at the top are generated with tikz, while the lower figures are photos of 3D-objects, which were generated by Rabea Freese during here bachelor thesis, see Freese, 2022. The dominated areas are shown up to the reference points $(4, 4, 4)^\top$ (left) and $(5, 5, 5)^\top$ (right).

Proof. This follows immediately from Lemma 5.9, the relation between orders and cones, see Section 2.1, and Theorem 5.20. \square

Example 5.22. Consider an instance of problem (OCOP) with $K = 3$ categories that has the following four feasible counting vectors $c^1 = (3, 1, 0)^\top$, $c^2 = (0, 2, 1)^\top$, $c^3 = (0, 0, 2)^\top$ and $c^4 = (1, 0, 2)^\top$. The transformation to problem (TOP) yields the corresponding incremental tail counting vectors as $\tilde{c}^1 = (4, 1, 0)^\top$, $\tilde{c}^2 = (3, 3, 1)^\top$, $\tilde{c}^3 = (2, 2, 2)^\top$ and $\tilde{c}^4 = (3, 2, 2)^\top$. The outcome spaces of both formulations are depicted in Figure 5.4 together with the ordering cones C_{\leq_t} and C_P , respectively.

In Figure 5.4(c) and 5.4(d) photographs of the 3D-objects of the dominated areas of Example 5.22 are given. The 3D-objects are printed by Rabea Freese during here bachelor thesis, see Freese, 2022. She showed, that the dominated area of problems (OCOP) and (TOP) with three categories can be printed with a 3D-printer, as the objects have three dimensions and they can be transformed into finite objects by defining an appropriate upper bound $u \in \mathbb{R}^3$. Freese, 2022 suggests to choose the same upper bound for the corresponding problems (OCOP) and (TOP). Therefore, she computes first for every component the maximal value of the non-dominated set of problems (OCOP) Y_N^{OCOP} and (TOP) Y_N^{TOP} and increases the value by one, i.e., $u_i := (\max\{y_i : y \in \mathcal{Y}_N^{\text{OCOP}} \cup \mathcal{Y}_N^{\text{TOP}}\}) + 1$. The addition of a positive value (here chosen as 1) is necessary as the upper bound has to be strictly larger than the largest value of all non-dominated points in each component.

Moreover, for 3D-printing, the objects have to be transformed, such that the upper bound u is turned into the origin and such that the object lies completely in the positive

orthant. This transformation is applied to all non-dominated points of problems (OCOP) and (TOP). Let $y \in \mathcal{Y}_N^{\text{OCOP}} \cup Y_N^{\text{TOP}}$. Then the corresponding transformed point \hat{y} is given by

$$\hat{y} := \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \cdot (y - u),$$

see Freese, 2022. This transformation has the advantage that the resulting object has no overhang, a property that significantly simplifies the printing procedure. In this context, we say that an object has an overhang whenever the angle α between the ground and an outside wall of the object is less than 45 degrees, i.e., if $\alpha \leq 45^\circ$. Freese, 2022 shows that the transformed dominated areas of problems (OCOP) and (TOP) have no overhang, which has the advantage that no support structure is needed for printing. Support structure is needed to print overhangs with a 3D-printer and it has to be removed after the print. This would increase the costs for the print as additional material is needed for the support structure. In Freese, 2022 a start manual for the 3D-printing of dominated areas of problems (OCOP) and (TOP) is provided.

5.3. Solution Strategies

In this section, we discuss a generic algorithmic framework for ordinal optimization problems that takes advantage of the close relationship to multi-objective optimization problems. Since the weighted sum scalarization is a popular approach in multi-objective optimization, the interpretation of weights in the context of ordinal optimization and Pareto optimization is analyzed in more detail, and their relation to numerical representations is discussed.

5.3.1. Ordinal Optimization by Pareto Transformation

From the theory above it follows that we can solve the problems (OCOP) and (OOP) by solving the transformed problem (TOP), which is a standard multi-objective combinatorial problem w.r.t. Pareto optimality. After the computation of the Pareto-efficient set of problem (TOP), or of a minimal complete Pareto-efficient set, respectively, it is necessary to re-compute the corresponding outcome vectors of either problem (OCOP) or (OOP). In this context, a minimal complete Pareto-efficient set of (TOP) is a subset of the Pareto-efficient set that contains one Pareto-efficient solution for each Pareto-non-dominated outcome vector. We refer to Serafini, 1987 for different solution concepts in multi-objective optimization. The efficient sets of the problems (TOP), (OCOP) and (OOP) are equal and they are denoted by X_E in the following. The respective non-dominated sets are denoted by $Y_N^{\text{TOP}} := N(\tilde{c}(X), C_P)$, $Y_N^{\text{OCOP}} := N(c(X), C_{\leq t})$ and Y_N^{OOP} , respectively. A procedure for the computation of the efficient set and the respective non-dominated sets based on this Pareto transformation is outlined in Algorithm 13.

Note that the structural properties of the problems (OCOP) and (OOP) are preserved by the transformation as we do not change the feasible set and as the transformation of the objective function is linear and bijective. In particular, combinatorial solution strategies,

Algorithm 13: Ordinal optimization by Pareto transformation (OOPT)

Input: feasible set $X \subseteq 2^E$ and ordinal function $o : E \rightarrow \mathcal{C}$
Output: efficient set X_E and non-dominated sets Y_N^{OCOP} and Y_N^{OOP}

- 1 Compute $c(x)$ for all $x \in X$ // compute counting objective c
- 2 $X_E := \min_{C_P} \{A_{\leq t} \cdot c(x) : x \in X\}$ // solve lin. transf. (TOP)
- 3 $Y_N^{\text{OCOP}} := c(X_E)$ // map efficient set to ...
- 4 $Y_N^{\text{OOP}} := o(X_E)$ // ... resp. obj. spaces
- 5 **return** efficient set X_E and non-dominated sets Y_N^{OCOP} and Y_N^{OOP}

like, e.g., Bellman's principle of optimality for knapsack problems, can be applied in step 2 of Algorithm 13 to efficiently compute X_E . Ordinal optimization is thus in general no more complex than standard multi-objective optimization.

5.3.2. Weighted Sum Scalarization and Ordinal Weight Space Decomposition

In the following we investigate the interrelation between weighted sum scalarizations, see Section 2.3, for (TOP) and (OCOP) and numerical representations for (OOP). Thereby we rely on the concept of weight space decompositions, which were introduced by Benson and Sun, 2000 for multi-objective linear programming and extended to integer linear problems in Przybylski et al., 2010.

The *weighted sum scalarization for (TOP)* is

$$\begin{aligned} \min \quad & \sum_{i=1}^K \lambda_i \tilde{c}_i(x) \\ \text{s. t.} \quad & x \in X \end{aligned} \tag{WSTOP(\lambda)}$$

with $\lambda_i > 0$ for $i = 1, \dots, K$ and $\sum_{i=1}^K \lambda_i = 1$. Analogously, the *weighted sum scalarization for (OCOP)* can be formulated as

$$\begin{aligned} \min \quad & \sum_{i=1}^K \mu_i c_i(x) \\ \text{s. t.} \quad & x \in X \end{aligned} \tag{WSOCOP(\mu)}$$

with $\mu \in (C_{\leq t})_s^*$, where $(C_{\leq t})_s^*$ is the strict dual cone of the ordinal cone $C_{\leq t}$, and $\sum_{i=1}^K \mu_i = 1$. Recall that the strict dual cone $(C_{\leq t})_s^*$ is the interior of the dual cone $(C_{\leq t})^*$, which is visualized in Figures 5.3(e) and 5.3(k).

It is a well-known fact that when considering a multi-objective optimization problem, then optimal solutions of weighted sum scalarizations with weighting vectors $\lambda \in \mathbb{R}_{>}^K$ are always Pareto-efficient (see, e.g., Ehrgott, 2005) and Section 2.3. Recall that such solutions are called *supported* efficient solutions. Thus, problem (WSTOP(λ)) always yields Pareto-efficient solutions for problem (TOP). Since (OCOP) can be interpreted as a multi-objective optimization problems w.r.t. the ordering cone $C_{\leq t}$, every optimal solution of the associated weighted sum problem (WSOCOP(μ)) with weights in the strict dual cone $(C_{\leq t})_s^*$ of $C_{\leq t}$ is ordinally efficient for (OCOP) (see, e.g., Engau, 2007).

The supported efficient solutions of problems (TOP) and (OCOP) are the same, hence there is a one-to-one correspondence between appropriate weighting vectors λ and μ . For a given $\lambda \in \mathbb{R}_{>}^K$ and $x \in X$ we define $\mu_i = \sum_{j=1}^i \lambda_j$ for all $i = 1, \dots, K$. Then it holds that

$$\sum_{i=1}^K \lambda_i \tilde{c}_i(x) = \sum_{i=1}^K \lambda_i \sum_{j=i}^K c_j(x) = \sum_{i=1}^K c_i(x) \sum_{j=1}^i \lambda_j = \sum_{i=1}^K c_i(x) \mu_i,$$

which shows that problems (WSOCOP(μ)) and (WSTOP(λ)) have the same objective functions in this case.

Note that $\mu_i = \sum_{j=1}^i \lambda_j$ and $\lambda_i > 0$ for all $i = 1, \dots, K$ implies that $\mu_i < \mu_j$ for all $i < j$, as required. Conversely, weighting vectors $\mu \in (C_{\leq t})_s^*$ satisfy $\mu_i < \mu_j$ for all $i < j$ and hence yield associated weighting vectors $\lambda \in \mathbb{R}_{>}^K$ by setting $\lambda_1 := \mu_1 > 0$ and $\lambda_i := \mu_i - \mu_{i-1} > 0$ for all $i = 2, \dots, K$. Note also that while the values of $\mu_i = \sum_{j=1}^i \lambda_j$, $i = 1, \dots, K$ (for given $\lambda \in \mathbb{R}_{>}^K$) are in general not normalized to satisfy $\sum_{i=1}^K \mu_i = 1$, such weighting vectors μ can be easily normalized by setting

$$\mu_i := \frac{\sum_{j=1}^i \lambda_j}{\sum_{\ell=1}^K \sum_{j=1}^{\ell} \lambda_j} = \frac{\sum_{j=1}^i \lambda_j}{\sum_{j=1}^K (K - j + 1) \lambda_j}.$$

Note that this normalization is applicable since $(C_{\leq t})_s^* \subset \mathbb{R}_{>}^K$. As a consequence, a weight space decomposition for the multi-objective problem (TOP) can be translated into an associated *ordinal weight space decomposition* for the ordinal counting optimization problem (OCOP). In this context, a weight space decomposition subdivides the space of relevant weighting vectors $\lambda \in \mathbb{R}_{>}^K$ with $\sum_{i=1}^K \lambda_i = 1$ into polyhedral cells such that all weighting vectors from the same cell generate the same efficient solution(s), see Section 2.3.

Example 5.23. *In the shortest path problem of Example 5.1 the solutions x^2 , x^3 , x^4 and x^5 are efficient. In Figure 5.5 the corresponding weight space decomposition is depicted showing the values of λ and μ for which the respective efficient solution is obtained.*

From yet another perspective, weighting vectors $\mu \in (C_{\leq t})_s^*$, i.e., weighting vectors $\mu \in \mathbb{R}^K$ satisfying $0 < \mu_i < \mu_j$ for all $i < j$, are related to numerical representations as introduced in Section 5.1.2. Indeed, numerical representations assign a numerical value $\nu(\eta_i)$ to every ordinal category η_i , $i = 1, \dots, K$, such that $\nu(\eta_i) < \nu(\eta_j)$ whenever $i < j$. Hence we can choose the values μ_i , $i = 1, \dots, K$, equal to the values $\nu(\eta_i)$ of any numerical representation that satisfies $\nu(\eta_1) > 0$. These values can again be normalized without changing the optimal solutions of (WSOCOP(μ)) by setting

$$\mu_i := \frac{\nu(\eta_i)}{\sum_{j=1}^K \nu(\eta_j)}, \quad i = 1, \dots, K.$$

It is important to note that this does not imply that numerical representations and weighted sum scalarizations are equivalent. Similarly, it is in general not possible to compute all ordinally efficient solutions of problem (OOP) by solving (WSOCOP(μ)) for an appropriate μ . To see this, recall that a solution $x' \in X$ is called ordinally efficient for problem (OOP) if and only if there is no $\hat{x} \in X$ that ordinally dominates x' , i.e., if for

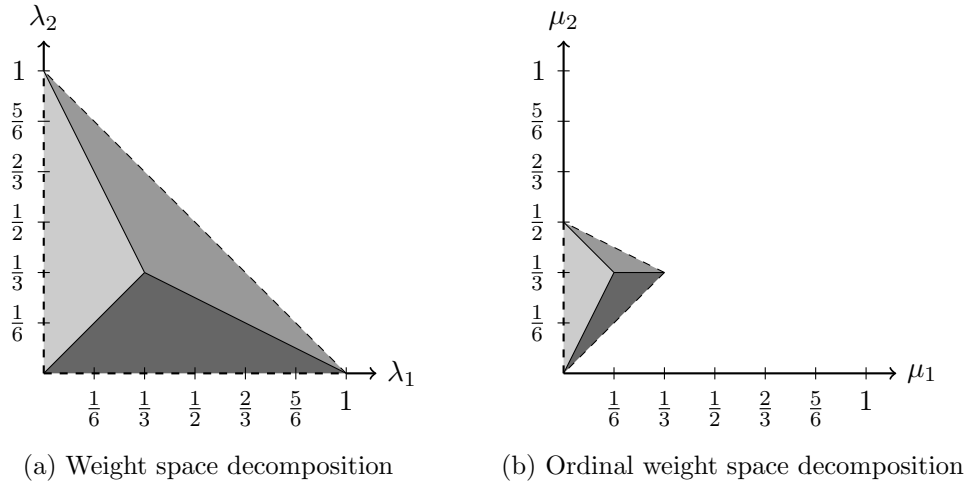


Figure 5.5.: Weight space decomposition and corresponding ordinal weight space decomposition for the shortest path problem given in Example 5.1. The efficient solution x^2 corresponds to the light grey triangle, both x^3 and x^4 correspond to the middle grey triangle and x^5 corresponds to the dark grey triangle. The values on dashed lines may not be chosen for λ and μ , because for the weight space decomposition we assume that $\lambda \in \mathbb{R}_{>}^K$ and $\sum_{i=1}^3 \lambda_i = 1$, and for the ordinal weight space decomposition we require $0 < \mu_1 < \mu_2 < \mu_3$ and $\sum_{i=1}^3 \mu_i = 1$.

every $\hat{x} \in X$ there exists a numerical representation $\nu^{\hat{x}} \in \mathcal{V}$ such that $\nu^{\hat{x}}(x') \leq \nu^{\hat{x}}(\hat{x})$. In contrast, a solution $x' \in X$ is optimal for problem (WSOCOP(μ)) with appropriate μ if and only if there exists a numerical representation $\nu^* \in \mathcal{V}$ such that $\nu^*(x') \leq \nu^*(\hat{x})$ for all $\hat{x} \in X$.

Remark 5.24. Ordinal optimization problems may have non-supported efficient solutions. This is illustrated in Example 5.25. Hence, we can not expect to determine all efficient solutions with the weighted sum method.

Example 5.25. Consider an instance with two categories and three efficient solutions x', \hat{x}, \bar{x} with counting vectors $c(x') = (3, 1)^\top$, $c(\hat{x}) = (5, 0)^\top$ and $c(\bar{x}) = (0, 2)^\top$. The incremental tail counting vectors in the transformed problem (TOP) are $\tilde{c}(x') = (4, 1)^\top$, $\tilde{c}(\hat{x}) = (5, 0)^\top$ and $\tilde{c}(\bar{x}) = (2, 2)^\top$, respectively. Obviously, $\tilde{c}(x')$ is non-dominated in (TOP) but unsupported, and thus x' is not optimal for (WSTOP(λ)), irrespective of the choice of $\lambda \in \mathbb{R}_{>}^K$. Similarly, there is no numerical representation such that x' is simultaneously better than \hat{x} and \bar{x} , i.e., there is no numerical representation ν such that $\nu(x') \leq \nu(\hat{x})$ and $\nu(x') \leq \nu(\bar{x})$. Indeed, the numerical values of the solutions are $\nu(x') = 3\nu(\eta_1) + \nu(\eta_2)$, $\nu(\hat{x}) = 5\nu(\eta_1)$ and $\nu(\bar{x}) = 2\nu(\eta_2)$. $\nu(x') \leq \nu(\hat{x})$ implies $\nu(\eta_2) \leq 2\nu(\eta_1)$ and $\nu(x') \leq \nu(\bar{x})$ implies $3\nu(\eta_1) \leq \nu(\eta_2)$, which is a contradiction to $\nu(\eta_1) < \nu(\eta_2)$ and $\nu(\eta_i) \geq 0$ for $i = 1, 2$. However, neither \hat{x} nor \bar{x} ordinally dominate x' , i.e., neither \hat{x} nor \bar{x} yield a better objective value for every numerical representation.

5.3.3. Other Scalarization Techniques

It is also possible to apply the Pascoletti-Serafini scalarization and the weighted Tchebycheff scalarization directly to problem (OCOP). Both techniques require a pointed, closed and convex cone and thus we have to consider the ordering cone C_{\leq_t} in problem (OCOP) as the relation \leq_t is reflexive, transitive and asymmetric while the relation \leq_t is irreflexive and transitive, see Lemma 5.14. The Pascoletti-Serafini scalarization can be applied to problem (OCOP) with the ordering cone C_{\leq_t} , because every weakly C_{\leq_t} -efficient solution x^* of problem (OCOP) can be computed by choosing $z := f(x^*)$ and an arbitrary search direction $d \in \text{int}(C_{\leq_t})$. In case of the weighted Tchebycheff scalarization this holds as we can choose z^U small enough in each component such that $Y^{\text{OCOP}} \subseteq \{z^U\} \oplus \text{int}(C_{\leq_t})$. This is only correct, because of the finiteness of the outcome set Y^{OCOP} due to the finite ground set E .

Note, that it is not possible to apply all scalarization techniques directly to problem (OCOP) which can be applied to problem (TOP). Consider, for example, the ε -constraint scalarization for general cones, see Section 2.3. There, the assumption is made that the j -th unit vector \mathbf{e}_j is contained in the dual cone of the ordering cone. This does not hold in the case of the ordinal ordering cone, i.e., $\mathbf{e}_j \notin (C_{\leq_t})^*$ for $j = 1, \dots, K - 1$. Similarly, the hybrid scalarization cannot be applied to problem (OCOP). The Benson scalarization can be applied to problem (OCOP) as C_{\leq_t} is a convex cone and $(1, \dots, 1)^\top \in (C_{\leq_t})^*$ and thus, all weakly C_{\leq_t} -efficient solutions of problem (OCOP) can be computed. But $(1, \dots, 1)^\top \notin (C_{\leq_t})_s^*$ and therefore, it is not possible to compute all C_{\leq_t} -efficient solutions of problem (OCOP).

Overall, the transformation of problem (OCOP) into problem (TOP) supports the application of more scalarization techniques.

5.4. Excursus: Olympic Medals and Ordinal Weight Space Decomposition

One example for ordinal categories are olympic medals. There is always the question how to rank the different countries which take part at the olympic games. The following two methods are commonly used in press and media:

The first is used by most countries and the International Olympic Committee (IOC) and it sorts the countries lexicographically, i.e., first only gold medals are considered, if two countries have the same number of gold medals, then the silver medals are considered and only if two countries have equally many gold and silver medals the bronze medals are coming into account.

In contrast, in the United States most newspapers make the ranking according to the total number of medals, i.e., it is assumed that gold medals have the same value as silver and bronze medals.

Furthermore, there also have been used other ranking methods in the past. For example, in the official report from the olympic games in 1908, see British Olympic Council, 1909, the countries are ranked by giving the gold medals the value 5, silver medals the value 3 and bronze medals the value 1. In the report from the olympic games in 1912, see International

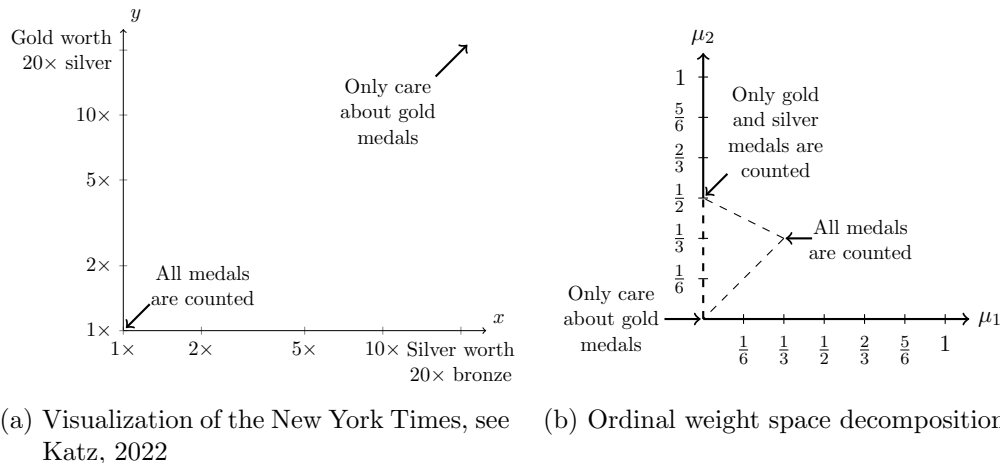


Figure 5.6.: Different visualizations of weight spaces for olympic rankings.

Olympic Committee, 1913, the ranking got changed and the value of a gold medal was 3, of a silver medal 2 and of a bronze medal 1. There have been even more different ranking strategies, but none of them has been used in large scale. Nevertheless, also scientist have investigated the question of a fair ranking strategy in the olympic games, see for example Du, 2018 and Gomes Júnior et al., 2014. Sitarz, 2013 suggests another value distribution for gold, silver and bronze medals based on the incenter of a convex cone. In Perini et al., 2022 it is considered that different rankings are given and aggregated to a final ranking. As the final ranking loses information, they apply weight set decomposition to the set of convex multipliers and investigate this decomposition. Furthermore, they suggest a heuristic and an exact algorithm to compute the weight set decomposition.

In the New York Times an article was published in 2022 regarding the olympic games in Beijing, see Katz, 2022. There, a weight space decomposition is suggested, which visualizes for every country the possible placement in the ranking regarding to the value of the medals. They assume that bronze medals have value 1 and they depict on the x -axis the value of the silver medal in quantities of the value of the bronze medal. On the y -axis the value of the gold medal is denoted in quantities of the value of the silver medal, see Figure 5.6(a). Note that the value in the lower left corner is $(1, 1)$ and that logarithmic axes are used. Therefore, in the lower left corner the rank of the country is given for the case that all medals have the same value, i.e., the total number of medals is considered. Moreover, the area is unbounded in the upper right direction. The further we go into this direction the more important become the gold medals. Hence, if we go far enough into this direction we get the rank of a country if we only care about gold medals.

We would like to compare this visualization with a variant of the weight space decomposition presented in Section 5.3. We are interested in the ordinal weight space decomposition, which tells us for a fixed country the possible placements of it, but not in the ordinal weight space decomposition, which tells us which country is the best for which values. The corners of this ordinal weight space decomposition correspond to giving all medals equal values, considering the lexicographic order, i.e., only caring about gold medals, and

counting silver and gold medals with equal weight while ignoring the number of bronze medals, see Figure 5.6(b).

In contrast to the visualization in Figure 5.6(a) the area of possible weights is bounded. Nevertheless, there exists a one to one correspondence, i.e., a bijective function, between both visualizations. A weight (x, y) from Figure 5.6(a) corresponds to the point $(\frac{1}{1+x+x \cdot y}, \frac{1}{1+x+x \cdot y})$ in Figure 5.6(b) and a weight (μ_1, μ_2) from Figure 5.6(b) corresponds to the point $(\frac{\mu_2}{\mu_1}, \frac{\mu_3}{\mu_2})$ with $\mu_3 = 1 - \mu_1 - \mu_2$ in Figure 5.6(a). The ordinal weight space decomposition as in Figure 5.6(b) can be computed by the algorithms presented in Perini et al., 2022 as it can be interpreted as the aggregation of the rankings in the corners.

5.5. Numerical Results

In this section, we compare the Pareto cone C_P with the ordinal cone C_{\leq_t} . Obviously, the ordinal cone includes the Pareto cone and hence, if we apply the different cones on the same outcome set Y , we get $|\mathcal{N}(Y, C_{\leq_t})| \leq |\mathcal{N}(Y, C_P)|$. It is easily possible to construct outcome sets, such that the non-dominated outcome vectors are the same for both cones or such that all outcome vectors are non-dominated for the Pareto cone, but only one point is non-dominated for the ordinal cone. This is illustrated in Figure 5.7. In the following, we investigate the differences between the non-dominated sets $\mathcal{N}(Y, C_{\leq_t})$ and $\mathcal{N}(Y, C_P)$ for assignment problems and knapsack problems. This analysis has to be interpreted with care since we compare both cones on the same outcome set Y , which is not necessarily a possible outcome set of problem (TOP) as we do not transform the set. For a possible outcome set of problem (TOP) it has to hold that $y_1 \geq y_2 \geq \dots \geq y_K$ for every $y \in Y$.

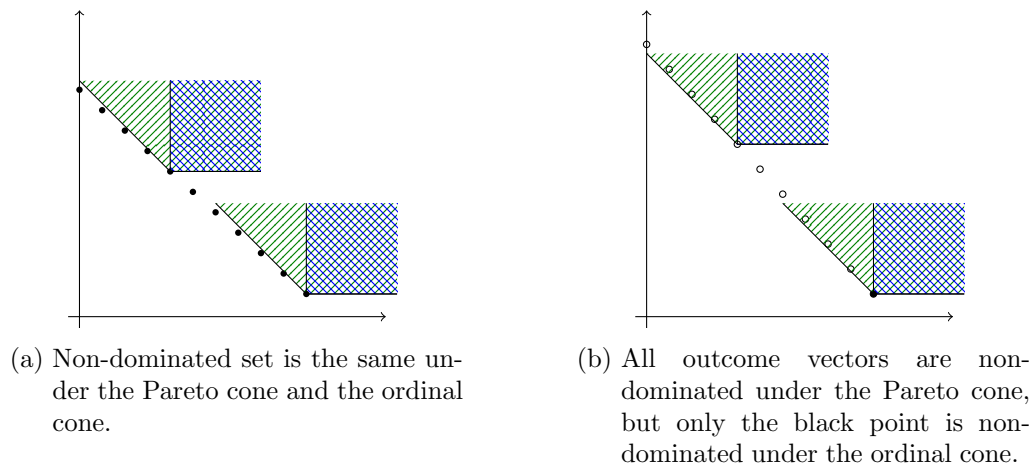


Figure 5.7.: Illustration of two different sets of outcome vectors and the non-dominated points under the blue Pareto cone and the green ordinal cone, respectively.

5.5.1. Assignment Problems

We consider multi-objective assignment problems, see Section 2.10, on complete bipartite graphs $G = (V_1 \cup V_2)$, which have the same number of vertices $n \in \{10, 20, 30, 40\}$ in V_1 and V_2 . We assign a random integer value between 1 and $K = 3$ to every edge in the graph, which can be interpreted as different categories. The objective functions are given by the counting vector c , i.e., the number of edges in each category.

Note, that only the ordinal cone reflects the ordering of the different categories, while if we apply the Pareto cone, the different categories are all interpreted as independent categories. The number of possible values for the outcome vectors can be computed as $\binom{n+K-1}{K-1} = \mathcal{O}(n^{K-1})$ (assuming that K is constant), i.e., it is equal to the number of multisets of cardinality $K - 1$ taken from a set of size $n + 1$.

For every possible value of n we generate 10 random instances and compare the cardinality of the non-dominated sets. Towards this end, we solve the problems with the objective space method presented in Klamroth et al., 2015, see Section 2.10 for a brief introduction.

The result is given in Table 5.1. Obviously, it is always possible to find an assignment such that all edges have the value 1, and hence the cardinality of the non-dominated set under the ordinal cone is always one. In contrast, in case of the Pareto cone all possible outcome vectors are non-dominated as the number of edges in each solution of the assignment problem is fixed. In most cases it is possible to find a feasible solution of the assignment problem for every possible distribution of the edges in the different categories, i.e., we found $\binom{n+K-1}{K-1}$ outcome vectors. This result is not surprising as the number of edges in the graph is much larger than the number of edges in a feasible solution of the assignment problem and in most cases a feasible solution can be constructed for all possible distributions of the edges in the different categories. The larger n is, the larger is the difference between all edges in the graph and the edges needed for a feasible solution. Therefore, we can see in Table 5.1 that for larger n the number of non-dominated outcome vectors under the Pareto cone gets closer to $\binom{n+K-1}{K-1}$. Overall, for the non-dominated sets of the considered assignment problems it makes a huge difference which cone is considered, and hence we are in a similar situation as in Figure 5.7(b).

K	n	$ \emptyset N(Y, C_P) $	$\min N(Y, C_P) $	$\max N(Y, C_P) $	$\binom{n+K-1}{K-1}$	$ \emptyset N(Y, C_{\leq t}) $
3	10	65.4	64	66	66	1
3	20	230.9	230	231	231	1
3	30	496	496	496	496	1
3	40	861	861	861	861	1

Table 5.1.: Average cardinality of the non-dominated sets of assignment problems under the Pareto cone $|\emptyset|N(Y, C_P)|$ and the ordinal cone $|\emptyset|N(Y, C_{\leq t})|$, respectively.

5.5.2. Knapsack Problems

We consider binary knapsack problems with n items, each of which is assigned to one of $K \in \{4, \dots, 6\}$ categories. Again, if we solve the problem w.r.t. the ordinal cone, the categories are interpreted as ordered, while if we apply the Pareto cone, the categories are interpreted as independent categories. Furthermore, every item e has a weight value

$w(e) = \max\{o(e) \cdot 100 + \text{round}(\text{normrnd}(0, \sigma)), 0\}$. Here, $o(e)$ denotes the category of item e , round is the rounding operation to the next integer and $\text{normrnd}(0, \sigma)$ computes a random number from the normal distribution with mean parameter 0 and standard deviation parameter σ . The bound on the total weight of the knapsack is chosen as ρ percent of the sum over the weight of all items. For every instance type, which are defined by n , K and ρ , we generate 10 random instances and solve them with the dynamic programming algorithm described in Section 2.9 and in Klamroth and Wiecek, 2000.

The numerical results can be found for $\sigma = 20$ in Table 5.2, for $\sigma = 30$ in Table 5.3 and for $\sigma = 40$ in Table 5.4. The choice of σ does not seem to have a large influence. Nevertheless, the numerical results show, that the number of non-dominated outcome vectors is often significantly reduced if the ordinal cone is considered instead of the Pareto cone. This has also impact on the running time of the algorithm, which gets also reduced.

Instance Size			Pareto Cone C_P				Ordinal Cone $C_{\leq t}$			
n	K	ρ	$\emptyset \mathcal{Y}_N $	$\min \mathcal{Y}_N $	$\max \mathcal{Y}_N $	$\emptyset[s]$	$\emptyset \mathcal{Y}_N $	$\min \mathcal{Y}_N $	$\max \mathcal{Y}_N $	$\emptyset[s]$
20	3	25%	12.1	11	14	0.57	8.7	8	10	0.48
20	3	50%	20.3	15	23	4.70	13.3	8	19	3.63
20	3	75%	13.4	12	15	12.49	8.1	7	10	9.47
20	4	25%	22.6	15	32	0.99	13.8	9	19	0.86
20	4	50%	41.6	21	62	10.00	23.2	12	41	7.82
20	4	75%	24.8	16	38	27.81	13.1	7	19	21.47
20	5	25%	42.7	23	54	1.54	28.4	19	38	1.27
20	5	50%	97.8	37	152	19.48	56.0	26	90	14.22
20	5	75%	51.6	28	67	55.41	29.3	14	40	41.49
20	6	25%	64.7	21	90	2.13	37.7	15	61	1.68
20	6	50%	158.5	26	259	32.25	82.5	20	133	21.48
20	6	75%	78.1	22	113	92.95	36.1	16	57	65.63
30	3	25%	24.7	21	27	10.28	18.7	15	22	8.35
30	3	50%	41.6	26	49	72.02	28.9	19	34	54.27
30	3	75%	23.9	22	26	181.59	17.0	14	19	134.02
30	4	25%	69.3	64	76	30.61	47.2	34	53	22.54
30	4	50%	141.7	80	182	293.33	87.5	57	111	192.04
30	4	75%	68.4	63	80	781.90	43.0	36	50	513.89
30	5	25%	151.6	106	192	60.75	101.3	66	129	41.29
30	5	50%	375.4	239	513	782.55	232.5	158	366	473.14
30	5	75%	161.3	116	193	2065.89	96.8	51	137	1311.52
40	3	25%	41.9	38	45	63.87	30.5	25	34	51.09
40	3	50%	73.4	47	82	457.59	51.3	40	59	332.81
40	3	75%	38.8	36	42	1162.01	27.7	23	31	837.81
40	4	25%	142.3	122	160	237.84	102.3	94	113	168.14
40	4	50%	317.7	268	377	2279.51	204.0	178	234	1472.95
40	4	75%	136.0	121	152	5875.43	90.1	77	101	3855.20
50	3	25%	62.4	57	67	253.74	46.6	39	55	204.93
50	3	50%	103.9	81	119	1763.07	76.1	63	86	1327.45
50	3	75%	56.5	51	60	4386.35	43.4	38	48	3282.83

Table 5.2.: Average number $\emptyset|\mathcal{Y}_N|$, minimal number $\min|\mathcal{Y}_N|$ and maximal number $\max|\mathcal{Y}_N|$ of non-dominated outcome vectors of different knapsack instances with $\sigma = 20$ and the average running time $\emptyset[s]$ for optimization under the Pareto cone and the ordinal cone, respectively.

Instance Size			Pareto Cone C_P				Ordinal Cone $C_{\leq t}$			
n	K	ρ	$\emptyset \mathcal{Y}_N $	$\min \mathcal{Y}_N $	$\max \mathcal{Y}_N $	$\emptyset[s]$	$\emptyset \mathcal{Y}_N $	$\min \mathcal{Y}_N $	$\max \mathcal{Y}_N $	$\emptyset[s]$
20	3	25%	12.0	8	15	0.60	7.3	5	11	0.48
20	3	50%	16.9	8	25	4.69	9.6	6	12	3.20
20	3	75%	11.9	8	15	12.28	6.1	4	8	8.06
20	4	25%	28.3	23	32	1.02	16.6	11	20	0.78
20	4	50%	55.5	36	64	11.24	27.5	21	34	7.18
20	4	75%	30.5	24	33	32.10	16.4	13	20	20.25
20	5	25%	48.6	34	63	1.70	24.6	13	34	1.24
20	5	50%	103.9	61	128	21.95	46.4	27	66	12.76
20	5	75%	53.8	35	66	63.27	23.1	17	28	36.86
20	6	25%	70.2	42	98	2.26	38.4	22	62	1.66
20	6	50%	166.9	55	294	33.58	76.8	35	118	20.01
20	6	75%	81.6	48	114	96.89	37.2	22	57	59.51
30	3	25%	25.5	23	28	9.10	17.2	14	20	6.65
30	3	50%	46.8	41	51	65.87	26.3	21	31	42.75
30	3	75%	23.7	22	26	168.01	14.4	11	17	106.28
30	4	25%	69.3	38	86	29.77	40.2	25	51	17.66
30	4	50%	140.0	36	186	283.45	75.2	26	103	143.94
30	4	75%	62.8	35	74	754.78	35.1	26	45	381.59
30	5	25%	172.1	118	222	87.98	96.2	77	111	47.45
30	5	50%	392.1	280	533	1 104.55	198.2	140	234	508.03
30	5	75%	167.1	115	213	2 796.90	85.7	50	115	1 383.28
40	3	25%	42.7	36	46	68.84	25.8	21	30	46.05
40	3	50%	65.7	51	87	452.23	39.9	31	48	272.28
40	3	75%	36.5	32	39	1 131.42	22.7	19	25	671.09
40	4	25%	147.7	105	176	259.94	84.4	65	100	149.09
40	4	50%	285.4	194	368	2 318.47	147.9	91	179	1 190.18
40	4	75%	126.2	93	155	5 818.78	69.3	57	87	3 040.92
50	3	25%	64.7	55	72	255.13	41.9	37	48	167.04
50	3	50%	101.5	91	114	1 716.07	62.1	54	73	1 040.41
50	3	75%	52.4	49	58	4 226.12	34.6	28	43	2 579.64

Table 5.3.: Average number $\emptyset|\mathcal{Y}_N|$, minimal number $\min|\mathcal{Y}_N|$ and maximal number $\max|\mathcal{Y}_N|$ of non-dominated outcome vectors of different knapsack instances with $\sigma = 30$ and the average running time $\emptyset[s]$ for optimization under the Pareto cone and the ordinal cone, respectively.

Instance Size			Pareto Cone C_P				Ordinal Cone $C_{\leq t}$			
n	K	ρ	$\emptyset \mathcal{Y}_N $	$\min \mathcal{Y}_N $	$\max \mathcal{Y}_N $	$\emptyset[s]$	$\emptyset \mathcal{Y}_N $	$\min \mathcal{Y}_N $	$\max \mathcal{Y}_N $	$\emptyset[s]$
20	3	25%	13.5	12	15	0.65	8.0	5	10	0.50
20	3	50%	20.6	12	28	4.88	10.8	5	14	3.25
20	3	75%	12.6	9	16	12.78	7.9	4	10	8.25
20	4	25%	27.2	18	33	1.13	11.7	8	14	0.80
20	4	50%	48.0	31	63	11.81	17.9	8	25	6.31
20	4	75%	26.7	19	32	32.92	10.8	6	14	16.83
20	5	25%	43.6	30	57	1.61	18.9	10	28	1.13
20	5	50%	94.4	57	130	21.27	33.2	17	47	11.02
20	5	75%	46.5	30	58	62.27	17.4	12	24	31.22
20	6	25%	78.3	41	103	2.48	32.8	21	51	1.64
20	6	50%	171.0	70	283	38.37	59.9	28	104	18.74
20	6	75%	83.0	39	111	112.25	28.3	13	47	55.62
30	3	25%	26.8	22	31	10.93	13.4	10	16	6.73
30	3	50%	41.7	29	50	74.54	20.1	15	25	40.08
30	3	75%	21.3	18	24	187.70	11.4	7	15	98.44
30	4	25%	69.8	53	85	25.78	33.2	20	42	13.32
30	4	50%	127.5	74	168	242.99	50.6	34	81	100.87
30	4	75%	59.7	45	78	652.05	26.6	19	33	267.13
30	5	25%	156.6	65	235	70.73	64.1	21	99	29.61
30	5	50%	350.2	93	555	886.40	127.7	45	193	284.69
30	5	75%	136.7	59	209	2316.92	57.1	23	83	801.80
40	3	25%	43.6	33	49	67.66	21.7	16	27	39.64
40	3	50%	69.3	29	86	451.10	33.7	19	44	229.39
40	3	75%	34.5	28	39	1125.02	18.8	16	25	553.99
40	4	25%	152.4	105	179	248.36	67.8	55	81	110.74
40	4	50%	307.9	193	369	2310.61	132.2	93	184	902.99
40	4	75%	119.3	82	135	5786.81	59.3	49	74	2387.01
50	3	25%	69.0	57	77	292.65	33.9	29	38	161.66
50	3	50%	100.8	81	121	1923.12	51.0	44	68	949.48
50	3	75%	51.1	39	62	4650.32	28.7	23	34	2320.62

Table 5.4.: Average number $\emptyset|\mathcal{Y}_N|$, minimal number $\min|\mathcal{Y}_N|$ and maximal number $\max|\mathcal{Y}_N|$ of non-dominated outcome vectors of different knapsack instances with $\sigma = 40$ and the average running time $\emptyset[s]$ for optimization under the Pareto cone and the ordinal cone, respectively.

5.6. Multi-objective Combinatorial Optimization with Ordinal Costs

In the following we extend our results to ordinal combinatorial optimization problems with additional real-valued objective functions and discuss two different modeling possibilities for this case, depending on whether the objectives are conflicting or coherent.

5.6.1. Conflicting Real-valued Objectives and Ordinal Objectives

The results of Section 5.2 can be extended to multi-objective optimization problems that combine a finite number of p “standard” real-valued objective functions $w^j : X \rightarrow \mathbb{R}$, $j = 1, \dots, p$ with a finite number of q ordinal objective functions $o^l : X \rightarrow \mathcal{C}^l$, $l = 1, \dots, q$, that are in mutual conflict. Note, that we use superscripts to identify the different real-valued objective functions, although we introduced multiple real-valued objective functions with subscripts and interpreted them as one vector-valued objective function. We do not use this notation here to emphasize the the p real-valued objective functions can be seen as different objectives and to make the notation consistent with the notation of several ordinal objective functions. The ordinal objective functions have to have a superscript to distinguish between them, because every single ordinal objective function has already a vector of varying length as outcome. The number of categories in the l -th ordinal objective function is denoted by K_l , i.e., $\mathcal{C}^l = \{\eta_1^l, \dots, \eta_{K_l}^l\}$ for $l = 1, \dots, q$.

For a feasible solution $x = \{e_1, \dots, e_n\} \in X$, we assume that $w^j(x) := \sum_{i=1}^n w^j(e_i)$, $j = 1, \dots, p$. Moreover, $o^l(x) = \text{sort}(o^l(e_1), \dots, o^l(e_n))$ for $l = 1, \dots, q$. This leads to the *multi-objective ordinal optimization problem with additional cost functions* (MOOP):

$$\begin{aligned}
 & \min_{C_P} (w^1(x), \dots, w^p(x))^\top \\
 & \min_{\preceq_\nu} o^1(x) \\
 & \quad \vdots \\
 & \min_{\preceq_\nu} o^q(x) \\
 & \text{s. t.} \quad x \in X.
 \end{aligned} \tag{MOOP}$$

By replacing the ordered vectors $o^l(x)$ for $l = 1, \dots, q$ by the counting vectors $c^l(x)$ for $l = 1, \dots, q$ we get a corresponding *multi-objective ordinal counting optimization problem with additional cost functions* (MCOP):

$$\begin{aligned}
 & \min_{C_P} (w^1(x), \dots, w^p(x))^\top \\
 & \min_{C_{\leq t}} c^1(x) \\
 & \quad \vdots \\
 & \min_{C_{\leq t}} c^q(x) \\
 & \text{s. t.} \quad x \in X.
 \end{aligned} \tag{MCOP}$$

We denote the concatenated outcome vectors of (MCOP) as

$$v(x) := (w^1(x), \dots, w^p(x), (c^1(x))^\top, \dots, (c^q(x))^\top)^\top \in \mathbb{R}^{p+\tilde{q}},$$

where $\tilde{q} := \sum_{l=1}^q K_l$. Then problem (MCOP) can be transformed into an equivalent standard multi-objective optimization problem w.r.t. Pareto dominance using a linear transformation that is defined by the block diagonal matrix

$$\tilde{A} := \begin{pmatrix} I_{p \times p} & & & \\ & A_{\leq t}^1 & & \\ & & \ddots & \\ & & & A_{\leq t}^q \end{pmatrix}.$$

Here, $I_{p \times p} \in \mathbb{R}^{p \times p}$ denotes the identity matrix and $A_{\leq t}^l$ is the transformation matrix corresponding to the objective c^l for $l = 1, \dots, q$, c.f. Theorem 5.16. Thus, we get the *multi-objective transformed Pareto cone optimization problem* (MTOP)

$$\begin{aligned} \min_{C_P} \quad & \tilde{A} \cdot v(x) \\ \text{s. t.} \quad & x \in X. \end{aligned} \tag{MTOP}$$

Now, problem (MOOP) or, equivalently, problem (MCOP) can be solved by using a simple adaptation of Algorithm 13, c.f. Section 5.3.

Example 5.26. *We consider a problem of type (MCOP) with one real-valued objective w and one counting objective c with $K = 2$ categories (i.e., $p = q = 1$). Consider an instance with four feasible outcome vectors $v = (w, c_1, c_2)^\top$ given by $v^1 = (4, 1, 0)^\top$, $v^2 = (3, 2, 1)^\top$, $v^3 = (2, 0, 2)^\top$ and $v^4 = (3, 0, 2)^\top$. Then the corresponding outcome vectors of problem (MTOP), $\tilde{v}^i = \tilde{A} v^i$ for $i = 1, \dots, 4$, are obtained as $\tilde{v}^1 = (4, 1, 0)^\top$, $\tilde{v}^2 = (3, 3, 1)^\top$, $\tilde{v}^3 = (2, 2, 2)^\top$ and $\tilde{v}^4 = (3, 2, 2)^\top$. In this case, the transformation matrix is given by*

$$\tilde{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

The feasible outcome vectors and the dominated volumes in the respective outcome spaces are depicted in Figure 5.8 for both problems, (MCOP) and (MTOP).

5.6.2. Coherent Real-valued Objectives and Ordinal Objectives

In some practical applications, the elements of E have a real-valued cost (e.g., the length of an edge) and an associated category (e.g., the safety of the corresponding road segment for a cyclist) such that the real-valued cost is, rather than in conflict, coherent with the respective category. This situation is illustrated at the following example:

Example 5.27. *Consider the shortest path problem shown in Figure 5.9. Let $w(e)$ denote the length of an edge e and let $o(e)$ denote its safety: green-dotted edges are safe and in category η_1 , while red-solid edges are insecure and in category η_2 . Then, irrespective of the number of edges contained in the respective paths, the path $x^1 = \{e_1, e_2, e_3\}$ should be preferred over the path $x^2 = \{e_4, e_5, e_6\}$ since the total weights are equal $w(x^1) = w(x^2) = 10$, and the red sub-path in x^1 has a smaller weight than that of x^2 . In this sense, the weight or length of an edge can be interpreted as an attribute of its respective category. However, x^1 is dominated by x^2 w.r.t. problem (MOOP).*

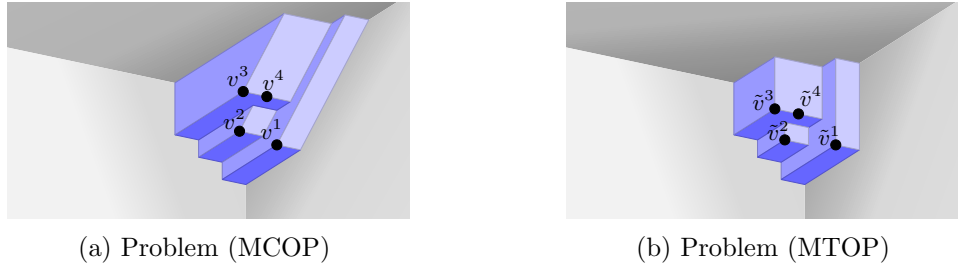


Figure 5.8.: Original and transformed outcome space for the multi-objective problem with one real-valued and one ordinal objective function introduced in Example 5.26. In both figures the upper corner of the bounding box is located at the point $(5, 5, 5)^\top$.

To model the situation where a real-valued objective function $w : E \rightarrow \mathbb{R}$ is in accordance with an ordinal objective function $o : E \rightarrow \mathcal{C}$ with K categories, i.e., the situation where the weight $w(e)$ reflects the multiplicity with which the category $o(e)$ of the element e is to be counted, we introduce a *weighted counting vector*

$$c_i^w(e) := \begin{cases} w(e) & \text{if } o(e) = \eta_i \\ 0 & \text{otherwise.} \end{cases}$$

The basic idea of this concept is also used in the risk-aware bicycle routing application *geovelo*, which takes, besides the route length, also the total length of unsafe route segments into account. For example, when $K = 4$, $w(e) = 7$ and $o(e) = \eta_2$, then the weighted counting vector is given by $c^w(e) = (0, 7, 0, 0)^\top$. The weighted counting objective of a feasible solution $x = \{e_1, \dots, e_n\} \in X$ equals the sum of the weighted counting vectors of all elements in x , i.e., $c^w(x) = \sum_{i=1}^n c^w(e_i)$. Thereby, the i -th component of $c^w(x)$ corresponds to the total weight of the elements in x that are in category η_i , $i = 1, \dots, K$. Now the weighted counting vector can be handled analogously to the counting vector. Indeed, as in the previous chapter we consider the transformation $\tilde{c}_i^w(x) := \sum_{j=i}^K c_j^w(x) = A_{\leq t} \cdot c^w(x)$ to obtain the *weighted transformed Pareto cone optimization problem*

$$\begin{aligned} \min_{C_P} \quad & \tilde{c}^w(x) \\ \text{s. t.} \quad & x \in X \end{aligned} \tag{WTOp}$$

w.r.t. the concept of Pareto optimality, that can be solved with the methods developed in the preceding sections.

5.6.3. Modelling Aspects

We emphasize that it depends on the context of the respective application whether the multi-objective model (MTOp) or the aggregated model (WTOp) is more suitable. The following example illustrates that the aggregated model (WTOp) is meaningful whenever w and c are interrelated and coherent objectives, while the multi-objective model (MTOp) is particularly useful for unrelated or incompatible objectives.

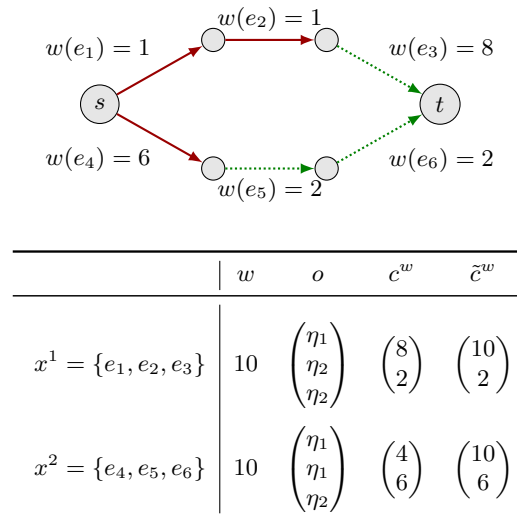


Figure 5.9.: Instance of a shortest path problem. A green-dotted edge is in the best category η_1 and the red-solid edges are in the worst category η_2 . The feasible s - t -paths x^1 and x^2 and their different objective function vectors are given.

Example 5.28. Consider again the shortest path problem depicted in Figure 5.9. Obviously, the path $x^1 = \{e_1, e_2, e_3\}$ is the unique efficient solution for problem (WTOP), while the path $x^2 = \{e_4, e_5, e_6\}$ is the unique efficient solution for problem (MOOP).

Whether x^1 or x^2 are actually preferred thus depends on the interpretation of the weights and of the ordinal categories. Towards this end, suppose that, as in Example 5.27, the category of an edge corresponds to its security level.

First, consider the case that the real-valued objective $w(e)$ represents the length of the edge e as in Example 5.27. Then both objectives are interrelated and hence model (WTOP) is appropriate.

If, on the other hand, $w(e)$ represents the toll of the edge or road e , then this real-valued objective is not an attribute of the corresponding category. In other words, both objectives are potentially conflicting and not coherent. Hence, in this case the path x^2 is preferred since the total amount of toll is the same for both paths, but the second path has more green and fewer red edges.

5.7. Conclusion and Further Ideas

In this chapter we investigate ordinal combinatorial optimization problems. We describe different optimality concepts for ordinal objective functions, namely ordinal optimality as well as tail- and head-optimality. We prove that all three concepts are equivalent if all feasible solutions have the same length. In general, only ordinal optimality and tail-optimality are equivalent.

We provide alternative descriptions of these three optimality concepts based on associated ordering cones. Using the fact that ordinal optimality and tail-optimality are

equivalent, and that tail-optimality can be represented by a polyhedral cone with K extreme rays in \mathbb{R}^K , we show that ordinal optimization problems can be transformed into equivalent multi-objective optimization problems with K objective functions and with binary cost coefficients. The transformation is realized by a bijective linear mapping. The resulting problem can be solved with standard methods from multi-objective optimization, and hence ordinal optimization is as easy or hard as the associated, “standard” multi-objective problems with K objective functions, as long as K is constant. For example, ordinal knapsack problems and ordinal shortest path problems can be solved by multi-objective dynamic programming, using Bellman’s principle of optimality.

The results can be extended to problems with more than one objective function. We suggest two modelling approaches to combine an ordinal objective function with a real-valued objective function. While in the first approach all objectives are considered in a standard multi-objective setting, the second approach allows to model interrelated and coherent objective functions, where the real-valued objective is interpreted as an attribute of the respective category in the ordinal objective.

Future work should focus on the development of tailored optimization algorithms for the associated multi-objective optimization problems that exploit the fact that these problems have binary cost coefficients. Moreover, specific combinatorial problems like, for example, shortest path, knapsack, assignment and general routing and network flow problems should be analyzed both with coherent and with conflicting ordinal and real-valued objective functions.

6. Conclusion and Further Ideas

In this thesis we investigate multi-objective matroid optimization problems with binary and ordinal costs as well as multi-objective combinatorial optimization problems with ordinal costs. An example for applications of matroid optimization problems are network design problems, which often rely on the computation of optimal spanning trees. General combinatorial optimization problems have many further applications like, for example, the selection of investments, which can be modeled by a knapsack problem, or like the traveling salesman problem. We investigate ordinal costs because in practical applications often objectives should be optimized, which cannot be measured by numerical values but which can be associated with ordered categories. An example for this is the safety of a street for a cyclist as there is no possibility to measure the safety numerically. Nevertheless, streets with a bike lane are safer than streets without a bike lane and little traffic which are again safer than streets without a bike lane and a lot of traffic. This can be modeled by assigning the streets to ordered categories.

One of our main results is that ordinal combinatorial optimization problems, given K (ordinal) quality categories, can be equivalently formulated as multi-objective combinatorial optimization problems with K binary objective functions and a specific structure through a bijective linear transformation. Towards this end, we define different concepts of ordinal optimality for matroid problems and for general combinatorial optimization problems. We show that all ordinal optimality concepts are equivalent for problems with solutions of equal length. Furthermore, we prove that an ordinal optimality concept based on numerical representations for combinatorial optimization problems is equivalent to the novel concept of tail-efficiency, which is defined on \mathbb{R}^K . Due to this result ordinal combinatorial optimization problems can be solved through the computation of the efficient solutions of multi-objective combinatorial optimization problems with binary costs. Thereby, all standard methods, like, e.g., dynamic programming for the knapsack problem, can be applied as the problem structure does not change. The weighted sum scalarization is a standard method to compute supported non-dominated points of multi-objective optimization problems and hence, we investigate the interrelation between those and numerical representations. Furthermore, we explain the relation between ordinal objective functions and weight space decompositions and relate our findings to olympic medal tables. In the numerical tests, we investigate the influence of the chosen optimality concept on the number of non-dominated points on assignment and knapsack problems. Moreover, we extend our results to ordinal combinatorial optimization problems with one additional sum objective function. We explain two possibilities to model such an additional function depending on the application at hand.

For the more specific optimization problems on matroids with binary or ordinal costs, we present algorithms to solve such problems in polynomial time. This is surprising as the decision problem corresponding to multi-objective matroid optimization is known to be \mathcal{NP} -complete in general. We investigate single-objective and bi-objective matroid

optimization problems with ordinal costs and their interrelation to corresponding problems with a lexicographic optimality concept. An important result is that single-objective matroid optimization problems can be solved efficiently, i.e., in polynomial time, with the greedy algorithm. For bi-objective matroid optimization problems with one ordinal objective function we show that they can be solved in polynomial time by solving a series of matroid intersection problems. The matroid intersection problems solve a corresponding ε -constraint scalarization of the original problem. Again, we validate the efficiency of the resulting algorithm by numerical tests. Moreover, we extend our results to multi-objective matroid optimization problems with more than one objective function with ordinal costs.

For bi-objective matroid optimization problems with one binary objective function, we introduce the Efficient Swap Algorithm, which solves the problem in polynomial time and in a linear number of iterations. The algorithm is very fast, because it uses the connectedness of the efficient set. While connectedness of the efficient set is a rare property in general MOCO problems, we prove that the efficient set of bi-objective matroid optimization problems with one binary objective function is connected. The efficiency of the algorithm even for large problem instances is verified by numerical tests. The numerical tests confirm that solving the bi-objective matroid optimization problems with one binary objective function with the Efficient Swap Algorithm is much faster than solving them with the Matroid Intersection Algorithm for Ordinal Constraints. Solving a series of matroid intersection problems takes also only polynomial time, but the Efficient Swap Algorithm uses the specific problem structure. As a side result, we search for counter examples to show that the connectedness of the efficient set does not hold if we consider more than two objective functions or more than two values, but still finitely many values for the formerly binary objective function. We found such examples for objective functions with values between 0 and 8.

Future work should investigate, whether the existing solution methods for specific multi-objective combinatorial optimization problems, like, e.g., shortest paths or knapsack problems, can be improved for the multi-objective combinatorial optimization problems with the specific structure, which results from the transformation from ordinal combinatorial optimization problems. Furthermore, alternative definitions of ordinal optimality and the similarities as well as differences between various ordinal and standard optimality concepts should be further investigated. Moreover, we should investigate further under which assumptions matroid optimization problems have connected efficient sets. A special case is the question, whether matroid optimization problems with one sum objective function and several binary objective functions have connected efficient sets in general.

Bibliography

- Ahuja, R. K., T. L. Magnanti, and J. B. Orlin (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall. DOI: 10.5555/137406.
- Bartee, E. M. (1971). “Problem Solving with Ordinal Measurement”. In: *Management Science* 17.10, B622–B633. URL: <http://www.jstor.org/stable/2628998> (visited on 12/21/2022).
- Bazgan, C., H. Hugot, and D. Vanderpooten (2009). “Solving efficiently the 0–1 multi-objective knapsack problem”. In: *Computers & Operations Research* 36.1, pp. 260–279. DOI: 10.1016/j.cor.2007.09.009.
- Bazgan, C., S. Ruzika, C. Thielen, and D. Vanderpooten (2019). “The Power of the Weighted Sum Scalarization for Approximating Multiobjective Optimization Problems”. In: *CoRR* abs/1908.01181. DOI: 10.1007/s00224-021-10066-5.
- Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.
- Benabbou, N. and P. Perny (2015). “On Possibly Optimal Tradeoffs in Multicriteria Spanning Tree Problems”. In: *Algorithmic Decision Theory*. Ed. by T. Walsh. Cham: Springer International Publishing, pp. 322–337. DOI: 10.1007/978-3-319-23114-3_20.
- Benson, H. P. (1978). “Existence of efficient solutions for vector maximization problems”. In: *Journal of Optimization Theory and Applications* 26.4, pp. 569–580. DOI: 10.1007/bf00933152.
- Benson, H. P. and E. Sun (2000). “Outcome space partition of the weight set in multi-objective linear programming”. In: *Journal of Optimization Theory and Applications* 105.1, pp. 17–36. DOI: 10.1023/A:1004605810296.
- Blot, A., M.-É. Kessaci, and L. Jourdan (2018). “Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation”. In: *Journal of Heuristics* 24.6, pp. 853–877. DOI: 10.1007/s10732-018-9381-1.
- Bökler, F. (2018). “Output-sensitive Complexity of Multiobjective Combinatorial Optimization With an Application to the Multiobjective Shortest Path Problem”. PhD thesis. TU Dortmund. DOI: 10.17877/DE290R-19130.
- Bökler, F., M. Ehrgott, C. Morris, and P. Mutzel (2017). “Output-sensitive complexity of multiobjective combinatorial optimization”. In: *Journal of Multi-Criteria Decision Analysis* 24, pp. 25–36. DOI: 10.1002/mcda.1603.
- Bossek, J., C. Grimme, and F. Neumann (2019). “On the Benefits of Biased Edge-Exchange Mutation for the Multi-Criteria Spanning Tree Problem”. In: *Proceedings of the 21th Genetic and Evolutionary Computation Conference (GECCO)*. Prague, Czech Republic: ACM, pp. 516–523. DOI: 10.1145/3321707.3321818.
- Bossong, U. and D. Schweigert (1999). “Minimal paths on ordered graphs”. URL: <http://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-4666>.

- Bouveret, S. and U. Endriss (June 2010). “Fair Division under Ordinal Preferences: Computing Envy-Free Allocations of Indivisible Goods”. In: vol. 215, pp. 387–392. DOI: 10.3233/978-1-60750-606-5-387.
- Bowman, V. J. (1976). “On the Relationship of the Tchebycheff Norm and the Efficient Frontier of Multiple-Criteria Objectives”. In: *Lecture Notes in Economics and Mathematical Systems*. Springer Berlin Heidelberg, pp. 76–86. DOI: 10.1007/978-3-642-87563-2_5.
- Brams, S. J., P. H. Edelman, and P. C. Fishburn (2003). “Fair division of indivisible items”. In: *Theory and Decision* 55.2, pp. 147–180.
- Brezovec, C., G. Cornuéjols, and F. Glover (1988). “A matroid algorithm and its application to the efficient solution of two optimization problems on graphs”. In: *Mathematical Programming* 42, pp. 471–487. DOI: 10.1007/BF01589417.
- Brezovec, C., G. Cornuéjols, and F. Glover (1986). “Two algorithms for weighted matroid intersection”. In: *Mathematical Programming* 36.1, pp. 39–53. DOI: 10.1007/BF02591988.
- British Olympic Council (Mar. 1909). *Fourth Olympiad; Being the Official Report of the Olympic Games of 1908 Celebrated in London Under the Patronage of His Most Gracious Majesty King Edward VII and by the Sanction of the International Olympic Committee*. Digitally published by the LA84 Foundation. URL: <https://digital.la84.org/digital/collection/p17103coll18/id/8217/rec/6>.
- Brualdi, R. A. (1969). “Comments on bases in dependence structures”. In: *Bulletin of the Australian Mathematical Society* 1.2, pp. 161–167. DOI: 10.1017/S000497270004140X.
- Casas, P. M. de las, A. Sedeño-Noda, and R. Borndörfer (2021). “An Improved Multiobjective Shortest Path Algorithm”. In: *Computers & Operations Research* 135, p. 105424. DOI: 10.1016/j.cor.2021.105424.
- Chang, R. and S.-J. Leu (1997). “The minimum labeling spanning trees”. In: *Information Processing Letters* 63.6, pp. 277–282. DOI: 10.1016/S0020-0190(97)00127-0.
- Chankong, V. and Y. Y. Haimes (1983). *Multiobjective Decision Making: Theory and Methodology*. Elsevier Science Publishing, New York.
- Climaco, J. C. N., M. E. Captivo, and M. M. B. Pascoal (2010). “On the bicriterion - minimal cost/minimal label - spanning tree problem”. In: *European Journal of Operational Research* 204, pp. 199–205. DOI: 10.1016/j.ejor.2009.10.013.
- Coello, C. A. (1998). “An Updated Survey of GA-Based Multiobjective Optimization Techniques”. In: *ACM COMPUTING SURVEYS* 32, pp. 109–143. DOI: 10.1145/358923.358929.
- Correia, P., L. Paquete, and J. R. Figueira (2021). “Finding multi-objective supported efficient spanning trees”. In: *Computational Optimization and Applications* 78.2, pp. 491–528. DOI: 10.1007/s10589-020-00251-6.
- Cubukcuoglu, C., M. F. Tasgetiren, I. S. Sariyildiz, L. Gao, and M. Kucukvar (2019). “A Memetic Algorithm for the Bi-Objective Quadratic Assignment Problem”. In: *Procedia Manufacturing* 39, pp. 1215–1222. DOI: 10.1016/j.promfg.2020.01.348.
- Delort, C., O. Spanjaard, and P. Weng (2011). “Committee Selection with a Weight Constraint Based on a Pairwise Dominance Relation”. In: *Algorithmic Decision Theory*. Ed. by R. I. Brafman, F. S. Roberts, and A. Tsoukiàs. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 28–41.

-
- Du, J. (2018). “Modifying Olympics Medal Table via a Stochastic Multicriteria Acceptability Analysis”. In: *Mathematical Problems in Engineering* 2018, pp. 1–11. DOI: 10.1155/2018/8729158.
- Dächert, K. (2014). “Adaptive Parametric Scalarizations in Multicriteria Optimization”. PhD thesis. University of Wuppertal.
- Dächert, K., K. Klamroth, R. Lacour, and D. Vanderpooten (2017). “Efficient computation of the search region in multi-objective optimization”. In: *European Journal of Operational Research* 260.3, pp. 841–855. DOI: 10.1016/j.ejor.2016.05.029.
- Eben-Chaïme, M. (1996). “Parametric Solution for Linear Bicriteria Knapsack Models”. In: *Management Science* 42.11, pp. 1565–1575. DOI: 10.5555/2777472.2777479.
- Edmonds, J. (1971). “Matroids and the greedy algorithm”. In: *Mathematical Programming* 1.1, pp. 127–136. DOI: 10.1007/bf01584082.
- Edmonds, J. (2003). “Submodular Functions, Matroids, and Certain Polyhedra”. In: *Combinatorial Optimization — Eureka, You Shrink!: Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*. Ed. by M. Jünger, G. Reinelt, and G. Rinaldi. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 11–26. DOI: 10.1007/3-540-36478-1_2.
- Ehrgott, M. (1996). “On matroids with multiple objectives”. In: *Optimization* 38.1, pp. 73–84. DOI: 10.1080/02331939608844238.
- Ehrgott, M. (2005). *Multicriteria Optimization*. Berlin, Heidelberg: Springer Verlag. DOI: 10.1007/3-540-27659-9.
- Ehrgott, M. and X. Gandibleux (2000). “A survey and annotated bibliography of multiobjective combinatorial optimization”. In: *OR Spektrum* 22.4, pp. 425–460. DOI: 10.1007/s002910000046.
- Ehrgott, M. and X. Gandibleux (2004). “Approximative solution methods for multiobjective combinatorial optimization”. In: *Top* 12.1, pp. 1–63. DOI: 10.1007/bf02578918.
- Ehrgott, M. and X. Gandibleux (2008). “Hybrid Metaheuristics for Multi-objective Combinatorial Optimization”. In: *Hybrid Metaheuristics: An Emerging Approach to Optimization*. Ed. by C. Blum, M. J. B. Aguilera, A. Roli, and M. Sampels. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 221–259. DOI: 10.1007/978-3-540-78295-7_8.
- Ehrgott, M., X. Gandibleux, and A. Przybylski (2016). “Exact Methods for Multi-Objective Combinatorial Optimisation”. In: *Multiple Criteria Decision Analysis*. Springer New York, pp. 817–850. DOI: 10.1007/978-1-4939-3094-4_19.
- Eichfelder, G. (2007). “Scalarizations for adaptively solving multi-objective optimization problems”. In: *Computational Optimization and Applications* 44.2, pp. 249–273. DOI: 10.1007/s10589-007-9155-4.
- Eichfelder, G. (June 2008). *Adaptive Scalarization Methods in Multiobjective Optimization*. Springer-Verlag GmbH. DOI: 10.1137/060672029.
- Engau, A. (2007). “Domination and decomposition in multiobjective programming”. PhD thesis. Clemson University.
- Feller, W. (1968). *An Introduction to Probability Theory and its Applications*. Vol. I. Wiley.
- Fernandes, I., E. Goldbarg, S. Maia, and M. Goldbarg (2020). “Empirical study of exact algorithms for the multi-objective spanning tree”. In: *Computational Optimization and Applications* 75, pp. 561–605. DOI: 10.1007/s10589-019-00154-1.

- Feussner, W. (1902). “Über Stromverzweigung in netzförmigen Leitern”. In: *Annalen der Physik* 314.13, pp. 1304–1329. DOI: 10.1002/andp.19023141320.
- Figueira, J. R., C. M. Fonseca, P. Halffmann, K. Klamroth, L. Paquete, S. Ruzika, B. Schulze, M. Stiglmayr, and D. Willems (2017). “Easy to say they’re hard, but hard to see they’re easy - Toward a categorization of tractable multiobjective combinatorial optimization problems”. In: *Journal of Multi-Criteria Decision Analysis* 24, pp. 82–98. DOI: 10.1002/mcda.1574.
- Figueira, J. R., L. Paquete, M. Simões, and D. Vanderpooten (2013). “Algorithmic improvements on dynamic programming for the bi-objective $\{0,1\}$ knapsack problem”. In: *Computational Optimization and Applications* 56.1, pp. 97–111. DOI: 10.1007/s10589-013-9551-x.
- Fishburn, P. (1999). “Preference structures and their numerical representations”. In: *Theoretical Computer Science* 217.2, 359–383. DOI: 10.1016/S0304-3975(98)00277-1.
- Frank, A. (1981). “A weighted matroid intersection algorithm”. In: *Journal of Algorithms* 2.4, pp. 328–336. DOI: 10.1016/0196-6774(81)90032-8.
- Freese, R. (2022). “Dreidimensionale Repräsentation von ordinalen und klassischen Ordnungskegeln”. Bachelors Thesis. University of Wuppertal.
- Gabow, H. N. and R. E. Tarjan (1984). “Efficient Algorithms for a Family of Matroid Intersection Problems”. In: *Journal of Algorithms* 5, pp. 80–131. DOI: 10.1016/0196-6774(84)90042-7.
- Garfinkel, R. and G. L. Nemhauser (1972). *Integer Programming*. New York: John Wiley & Sons Inc.
- Gearhart, W. B. (1983). “Characterization of properly efficient solutions by generalized scalarization methods”. In: *Journal of Optimization Theory and Applications* 41.3, pp. 491–502. DOI: 10.1007/bf00935368.
- Geovelo, an application for short and safe bicycle route computations (2022). <https://geovelo.fr>. Accessed: 2022-01-28.
- Gomes Júnior, S., J. Mello, and L. Angulo-Meza (Apr. 2014). “Sequential use of ordinal multicriteria methods to obtain a ranking for the 2012 Summer Olympic Games”. In: *WSEAS Transactions on Systems* 13, pp. 223–230.
- Gorski, J. (2010). “Multiple Objective Optimization and Implications for Single Objective Optimization”. PhD thesis. University of Wuppertal.
- Gorski, J., K. Klamroth, and S. Ruzika (2011). “Connectedness of Efficient Solutions in Multiple Objective Combinatorial Optimization”. In: *Journal of Optimization Theory and Applications* 150, pp. 475–497. DOI: 10.1007/s10957-011-9849-8.
- Gorski, J. and S. Ruzika (2009). “On k-max Optimization”. In: *Operations Research Letters* 37.1, pp. 23–26. DOI: 10.1016/j.orl.2008.09.007.
- Gorski, J., K. Klamroth, and J. Sudhoff (2022). “Biobjective optimization problems on matroids with binary costs”. In: *Optimization* 0.0, pp. 1–30. DOI: 10.1080/02331934.2022.2044479.
- Grandoni, F., R. Ravi, M. Singh, and R. Zenklusen (2014). “New approaches to multi-objective optimization”. In: *Mathematical Programming* 146, pp. 525–554. DOI: 10.1007/s10107-013-0703-7.

-
- Guddat, J., F. Guerra Vasquez, K. Tammer, and K. Wendler (1985). *Multiobjective and Stochastic Optimization Based on Parametric Optimization*. Vol. 26. Mathematical Research. Akademie-Verlag, Berlin.
- Gusfield, D. (1984). “Matroid Optimization with the interleaving of two ordered sets”. In: *Discrete Applied Mathematics* 8.1, pp. 41–50. DOI: 10.1016/0166-218X(84)90077-5.
- Göpfert, A., H. Riahi, C. Tammer, and C. Zălinescu (2003). *Variational Methods in Partially Ordered Spaces*. CMS Books in Mathematics. New York, NY: Springer-Verlag. DOI: 10.1007/b97568.
- Haimes, Y., L. Lasdon, and D. Wismer (1971). “On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization”. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-1.3, pp. 296–297. DOI: 10.1109/tsmc.1971.4308298.
- Hamacher, H. W. and F. Rendl (1991). “Color constrained combinatorial optimization problems”. In: *Operations Research Letters* 10, pp. 211–219. DOI: 10.1016/0167-6377(91)90061-S.
- Hamacher, H. W. and G. Ruhe (1994). “On spanning tree problems with multiple objectives”. In: *Annals of Operations Research* 52, pp. 209–230. DOI: 10.1007/BF02032304.
- Hamacher, H. W. and K. Klamroth (Apr. 2006). *Lineare Optimierung und Netzwerkoptimierung*. Vieweg+Teubner Verlag. 256 pp. DOI: 10.1007/978-3-8348-9031-3.
- Harris, J., J. L. Hirst, and M. Mossinghoff (2008). *Combinatorics and Graph Theory*. Springer New York. DOI: 10.1007/978-0-387-79711-3.
- Hotz, M. (2016). *generateSpanningTrees(A)*. Matlab implementation for MST computation, MATLAB Central File Exchange, downloaded on February 19, 2020. URL: <https://www.mathworks.com/matlabcentral/fileexchange/53787-generatespanningtrees-a>.
- Hunt, B. J. and M. M. Wiecek (2003). “Cones to Aid Decision Making in Multicriteria Programming”. In: *Multi-Objective Programming and Goal Programming*. Springer Berlin Heidelberg, 153–158. DOI: 10.1007/978-3-540-36510-5.
- Ibaraki, T. (May 1987). “Enumerative approaches to combinatorial optimization - part II”. In: *Annals of Operations Research* 11. Ed. by P. Hammer, pp. 343–602.
- International Olympic Committee (1913). *Fifth Olympiad: the Official Report of the Olympic Games of Stockholm, 1912 Swedish Olympic Committee*. Digitally published by the LA84 Foundation. URL: <https://digital.la84.org/digital/collection/p17103coll18/id/11660/rec/7#page=1178>.
- Jahn, J. (1984). “Scalarization in vector optimization”. In: *Mathematical Programming* 29.2, pp. 203–218. DOI: 10.1007/bf02592221.
- Jahn, J. (2011). *Vector optimization: Theory, Applications, and Extensions*. 2nd ed. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-642-17005-8.
- Jaszkiwicz, A. (2002). “Genetic local search for multi-objective combinatorial optimization”. In: *European Journal of Operational Research* 137.1, pp. 50–71. DOI: 10.1016/s0377-2217(01)00104-7.
- Jesus, A. D., L. Paquete, B. Derbel, and A. Liefoghe (2021). “On the design and anytime performance of indicator-based branch and bound for multi-objective combinatorial optimization”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM. DOI: 10.1145/3449639.3459360.

- Katz, J. (Feb. 2022). *Beijing Olympics: Who leads the Medal Count?* New York Times, online. Retrieved 14.02.2022. URL: <https://www.nytimes.com/interactive/2022/02/07/upshot/which-country-leads-olympic-medal-count.html>.
- Kellerer, H., U. Pferschy, and D. Pisinger (2004). *Knapsack Problems*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-24777-7.
- Kergosien, Y., A. Giret, E. Néron, and G. Sauvanet (2021). “An Efficient Label-Correcting Algorithm for the Multiobjective Shortest Path Problem”. In: *INFORMS Journal on Computing*. DOI: 10.1287/ijoc.2021.1081.
- Klamroth, K. and M. Wiecek (2000). “Dynamic Programming Approaches to the Multiple Criteria Knapsack Problem”. In: *Naval Research Logistics* 47, pp. 57–76. DOI: 10.1002/(SICI)1520-6750(200002)47:13.0.CO;2-4.
- Klamroth, K., R. Lacour, and D. Vanderpooten (2015). “On the representation of the search region in multi-objective optimization”. In: *European Journal of Operational Research* 245.3, pp. 767–778. DOI: 10.1016/j.ejor.2015.03.031.
- Klamroth, K., M. Stiglmayr, and J. Sudhoff (2022a). “Multi-objective Matroid Optimization with Ordinal Weights”. In: *Discrete Applied Mathematics*. DOI: <https://doi.org/10.1016/j.dam.2022.07.017>.
- Klamroth, K., M. Stiglmayr, and J. Sudhoff (2022b). *Ordinal Optimization Through Multi-objective Reformulation*. DOI: 10.48550/ARXIV.2204.02003.
- Knowles, J. D. and D. W. Corne (2001). “A comparison of encodings and algorithms for multiobjective minimum spanning tree problems”. In: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '01)*. IEEE Press, pp. 544–551. DOI: 10.1109/CEC.2001.934439.
- Knuth, D. E. (2012). *The Art of Computer Programming*. Vol. 4A (Combinatorial Algorithms, Part 1). Boston: Pearson Education, Inc.
- Kung, J.-P. S. (1986). *A source book in matroid theory*. Boston: Birkhäuser. DOI: 10.1007/978-1-4684-9199-9.
- Lew, A. and H. Mauch (2007). *Dynamic Programming*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-37014-7.
- Loera, J. A. D., D. C. Haws, J. Lee, and A. O’Hair (2009). *MOCHA – Matroids Optimization Combinatorics Heuristics and Algorithms*. <https://github.com/coin-or/MOCHA>. URL: <https://github.com/coin-or/MOCHA>.
- Loera, J. A. D., D. C. Haws, J. Lee, and A. O’Hair (2010). “Computation in Multicriteria Matroid Optimization”. In: *J. Exp. Algorithmics* 14. DOI: 10.1145/1498698.1658383.
- Mansour, I. B., M. Basseur, and F. Saubion (2018). “A multi-population algorithm for multi-objective knapsack problem”. In: *Applied Soft Computing* 70, pp. 814–825. DOI: 10.1016/j.asoc.2018.06.024.
- Martins, E. Q. V. (1984). “On a multicriteria shortest path problem”. In: *European Journal of Operational Research* 16.2, pp. 236–245. DOI: 10.1016/0377-2217(84)90077-8.
- Matoušek, J., ed. (2002). *Lectures on Discrete Geometry*. Springer New York. DOI: 10.1007/978-1-4613-0039-7.
- Merris, R. (1994). “Laplacian Matrices of Graphs: A Survey”. In: *Linear Algebra and its Applications* 197-198, pp. 143–176. DOI: 10.1016/0024-3795(94)90486-3.
- Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Boston: Kluwer Academic Publishers. DOI: 10.1007/978-1-4615-5563-6.

-
- Mosheiov, G. and A. Sarig (2008). “A multi-criteria scheduling with due-window assignment problem”. In: *Mathematical and Computer Modelling* 48.5-6, pp. 898–907. DOI: 10.1016/j.mcm.2007.08.018.
- Neumann, F. and C. Witt (2010). “Multi-objective Minimum Spanning Trees”. In: *Bioinspired Computation in Combinatorial Optimization*. Ed. by F. Neumann and C. Witt. Natural Computing Series. Berlin, Heidelberg: Springer, pp. 149–159. DOI: 10.1007/978-3-642-16544-3_10.
- Oxley, J. G. (2011). *Matroid Theory*. Oxford University Press, NJ.
- Pascoletti, A. and P. Serafini (1984). “Scalarizing vector optimization problems”. In: *Journal of Optimization Theory and Applications* 42.4, pp. 499–524. DOI: 10.1007/bf00934564.
- Pentico, D. W. (2007). “Assignment problems: A golden anniversary survey”. In: *European Journal of Operational Research* 176.2, pp. 774–793. DOI: 10.1016/j.ejor.2005.09.014.
- Perini, T., A. Langville, G. Kramer, J. Shrager, and M. Shapiro (2022). *Weight Set Decomposition for Weighted Rank Aggregation: An interpretable and visual decision support tool*. DOI: 10.48550/ARXIV.2206.00001.
- Pitsoulis, L. S. (2014). *Topics in matroid theory*. Springer. DOI: 10.1007/978-1-4614-8957-3.
- Pramanik, S. and P. Biswas (2012). “Multi-objective Assignment Problem with Generalized Trapezoidal Fuzzy Numbers”. In: *International Journal of Applied Information Systems* 2.6. Published by Foundation of Computer Science, New York, USA, pp. 13–20. DOI: 10.5120/ijais12-450375.
- Przybylski, A. and X. Gandibleux (2017). “Multi-objective branch and bound”. In: *European Journal of Operational Research* 260.3, pp. 856–872. DOI: 10.1016/j.ejor.2017.01.032.
- Przybylski, A., X. Gandibleux, and M. Ehrgott (2008). “Two phase algorithms for the bi-objective assignment problem”. In: *European Journal of Operational Research* 185.2, pp. 509–533. DOI: 10.1016/j.ejor.2006.12.054.
- Przybylski, A., X. Gandibleux, and M. Ehrgott (2010). “A Recursive Algorithm for Finding All Nondominated Extreme Points in the Outcome Set of a Multiobjective Integer Programme”. In: *INFORMS Journal on Computing* 22.3, pp. 371–386. DOI: 10.1287/ijoc.1090.0342.
- Raith, A., C. Houtte, J. Wang, and M. Ehrgott (Sept. 2009). “Applying Bi-Objective Shortest Path Methods to Model Cycle Route Choice”. In: *32nd Australasian Transport Research Forum, ATRF 2009*.
- Raith, A., U. Nataraj, M. Ehrgott, G. Miller, and K. Pauw (Sept. 2011). “Prioritising Cycle Infrastructure Projects”. In: *ATRF 2011 - 34th Australasian Transport Research Forum*.
- Rendl, F. and M. Leclerc (1988-1989). “A multiply constrained matroid optimization problem”. In: *Discrete Mathematics* 73, pp. 207–212. DOI: 10.1016/0012-365X(88)90149-5.
- Roy, R., S. Dehuri, and S. B. Cho (2011). “A Novel Particle Swarm Optimization Algorithm for Multi-Objective Combinatorial Optimization Problem”. In: *International Journal of Applied Metaheuristic Computing* 2.4, pp. 41–57. DOI: 10.4018/jamc.2011100104.

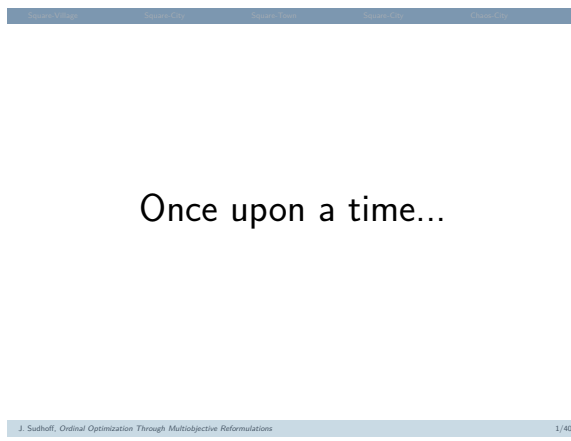
- Ruzika, S. and H. Hamacher (2009). “A survey on multiple objective minimum spanning tree problems”. In: *Algorithmics*. Ed. by J. Lerner, D. Wagner, and K. A. Zweig. Vol. 5515/2009. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, pp. 104–116. DOI: 10.1007/978-3-642-02094-0_6.
- Ruzika, S. (2009). “A linear-time algorithm for the inary bicriteria spanning tree problem”. Presented at 23rd european conference on operational research (EURO XXIII). URL: https://www.euro-online.org/media_site/reports/EUR023_AB.pdf.
- Sauvanet, G. and E. Néron (2010). “Search for the best compromise solution on multiobjective shortest path problem”. In: *Electron. Notes Discrete Math.* 36, pp. 615–622. DOI: 10.1016/j.endm.2010.05.078.
- Sawaragi, Y., H. Nakayama, and T. Tanino (1985). *Theory of Multiobjective Optimization*. Vol. 176. Mathematics in Science and Engineering. Elsevier Science. 322 pp. URL: https://www.ebook.de/de/product/15181299/theory_of_multiobjective_optimization.html.
- Schnepper, T., K. Klamroth, J. Puerto, and M. Stiglmayr (2021). “A local analysis to determine all optimal solutions of p-k-max location problems on networks”. In: *Discrete Applied Mathematics* 296, pp. 217–234. DOI: 10.1016/j.dam.2020.05.013.
- Schrijver, A. (2003). *Combinatorial Optimization*. Springer Berlin Heidelberg.
- Schrijver, A. (2017). *A Course in Combinatorial Optimization*. URL: <https://homepages.cwi.nl/~lex/files/dict.pdf>.
- Schulze, B. (Jan. 2017). “New Perspectives on Multi-Objective Knapsack Problems”. PhD thesis. Bergische Universität Wuppertal.
- Schäfer, L. E., T. Dietz, M. Barbati, J. R. Figueira, S. Greco, and S. Ruzika (2021). “The binary knapsack problem with qualitative levels”. In: *European Journal of Operational Research* 289.2, pp. 508–514. DOI: 10.1016/j.ejor.2020.07.040.
- Schäfer, L. E., T. Dietz, N. Fröhlich, S. Ruzika, and J. R. Figueira (2020). “Shortest paths with ordinal weights”. In: *European Journal of Operational Research* 280.3, pp. 1160–1170. DOI: 10.1016/j.ejor.2019.08.008.
- Seipp, F. (2013). “On Adjacency, Cardinality, and Partial Dominance in Discrete Multiple Objective Optimization”. PhD thesis. TU Kaiserslautern.
- Serafini, P. (1987). “Some considerations about computational complexity for multi objective combinatorial problems”. In: *Recent Advances and Historical Development of Vector Optimization*. Ed. by J. Jahn and W. Krabs. Vol. 294. Lecture Notes in Economics and Mathematical Systems. Springer, pp. 222–232. DOI: 10.1007/978-3-642-46618-2.
- Shioura, A., A. Tamura, and T. Uno (1997). “An Optimal Algorithm for Scanning All Spanning Trees of Undirected Graphs”. In: *SIAM Journal on Computing* 26.3, pp. 678–692. DOI: 10.1137/s0097539794270881.
- Sitarz, S. (2013). “The medal points’ incenter for rankings in sport”. In: *Applied Mathematics Letters* 26.4, pp. 408–412. DOI: 10.1016/j.aml.2012.10.014.
- Srinivas, M. A. (1995). “Matroid optimization with generalized constraints”. In: *Discrete Applied Mathematics* 63, pp. 161–174. DOI: 10.1016/0166-218X(94)00031-8.
- Stanley, R. P. (2013). *Algebraic Combinatorics*. Springer New York. DOI: 10.1007/978-1-4614-6998-8.

-
- Steuer, R. E. and E.-U. Choo (1983). “An interactive weighted Tchebycheff procedure for multiple objective programming”. In: *Mathematical Programming* 26.3, pp. 326–344. DOI: 10.1007/bf02591870.
- Stidsen, T., K. A. Andersen, and B. Dammann (2014). “A Branch and Bound Algorithm for a Class of Biobjective Mixed Integer Programs”. In: *Management Science* 60.4, pp. 1009–1032. DOI: 10.1287/mnsc.2013.1802.
- Tamby, S. and D. Vanderpooten (2021). “Enumeration of the Nondominated Set of Multiobjective Discrete Optimization Problems”. In: *INFORMS Journal on Computing* 33.1, pp. 72–85. DOI: 10.1287/ijoc.2020.0953.
- Tammer, C. and A. Göpfert (2003). “Theory of Vector Optimization”. In: *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*. Springer US, pp. 1–70. DOI: 10.1007/0-306-48107-3_1.
- Verma, S., M. Pant, and V. Snasel (2021). “A Comprehensive Review on NSGA-II for Multi-Objective Combinatorial Optimization Problems”. In: *IEEE Access* 9, pp. 57757–57791. DOI: 10.1109/access.2021.3070634.
- Villarreal, B. and M. H. Karwan (1981). “Multicriteria integer programming: A (hybrid) dynamic programming recursive approach”. In: *Mathematical Programming* 21.1, pp. 204–223. DOI: 10.1007/bf01584241.
- Yuan, J. and Y. Li (2021). “Solving binary multi-objective knapsack problems with novel greedy strategy”. In: *Memetic Computing* 13.4, pp. 447–458. DOI: 10.1007/s12293-021-00344-7.
- Zhang, C. W. and H. L. Ong (2007). “An efficient solution to biobjective generalized assignment problem”. In: *Advances in Engineering Software* 38.1, pp. 50–58. DOI: 10.1016/j.advengsoft.2006.06.003.
- Zhou, G. and M. Gen (1999). “Genetic algorithm approach on multi-criteria minimum spanning tree problem”. In: *European Journal of Operational Research* 114, pp. 141–152. DOI: 10.1016/S0377-2217(98)00016-2.
- Ziegler, G. M. (1995). *Lectures on Polytopes*. Springer New York. DOI: 10.1007/978-1-4613-8431-1.

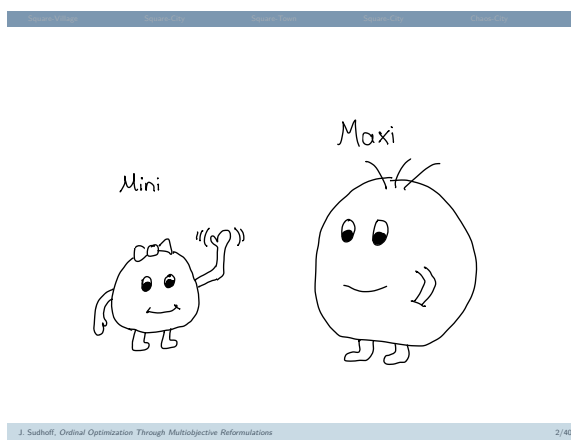
A. A Story About Ordinal Optimization Through Multi-objective Reformulation



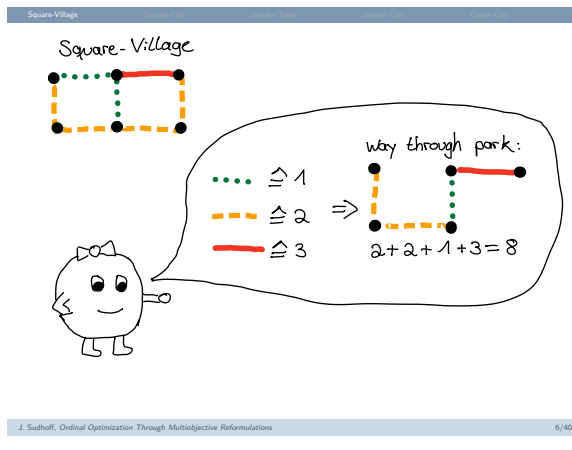
This comic was presented at the international conference Multicriteria Decision Making (MCDM) 2022. The slides are the same as presented on the conference, with some minor corrections. The presentation is based on Klamroth et al., 2022b.



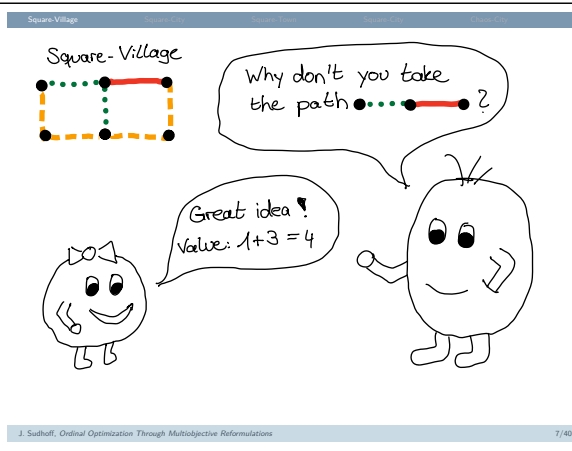
Once upon a time, in a distant country there lived a ...



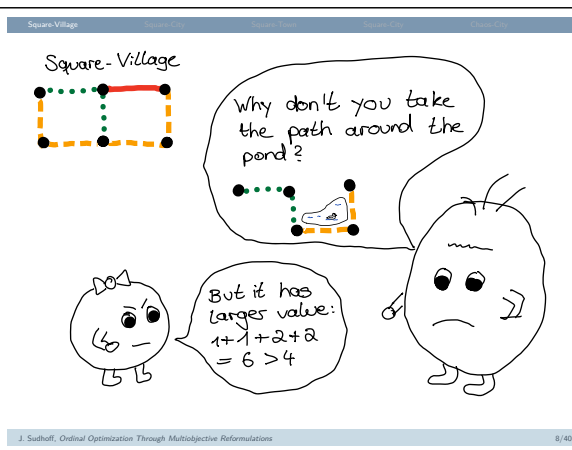
... little girl named Mini together with her father Maxi.



Mini suggested: “Well, lets try to find the best solution by giving the different categories numerical values. Let’s simply give green edges the value one, orange edges the value 2 and red edges the value 3. Hmmm, I could take this way through the park. This would have the numerical value $2 + 2 + 1 + 3 = 8$.”



Maxi wondered: “That’s a long path, why don’t you go through the shopping street?”
 Mini agreed: “Yes, that is a great idea. This path has a numerical value of $1 + 3 = 4$!”



Maxi complained: “I still don’t like this path. It contains a red street! I think you should take the road around the pond.”
 Mini disagreed: “But this path has numerical value $1 + 1 + 2 + 2 = 6$ which is worse than 4!”

Square-Village

Handwritten text: "Square-Village"

Handwritten text: "You choose the wrong numerical values:"

Handwritten text: $\dots \cong 1$, $\dots \cong 11$, $\dots \cong 2 \Rightarrow \dots \cong 6$, $\dots \cong 10$

Handwritten text: "But it is longer and < ?"

Small text at the bottom: "J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 9/40"

Maxi explained: "I think you choose the wrong numerical values. I would assign the value 1 to the green edges, the value 2 to the orange edges and the value 10 to the red edges! Then your path has the value 11 and my path has value 6."

Mini complained: "But it is longer and contains two orange edges – this can be even more dangerous than one red edge!"

Square-Village

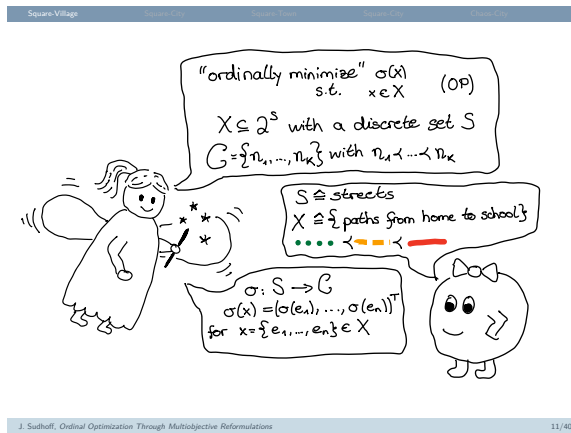
Small text at the bottom: "J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 10/40"

Suddenly a creature appeared and said: "Please don't argue!"

Maxi wondered: "Who are you? What do you want?"

The creature answered: "I am the good math fairy and I hate disputes. I think I can help you."

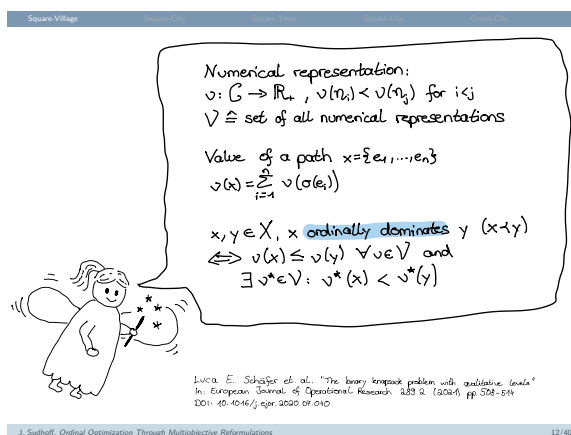
Maxi asked: "What do you suggest?"



The fairy explained: “You need the concept of ordinal optimality. Your problem can be formally described as to ordinally minimize $o(x)$ with $x \in X$. Here, X is a subset of the powerset of a discrete set S . Furthermore, we have a set C of K categories which can be strictly ordered.”

Mini stated: “This means in our case S contains all streets of Square-Village and X contains all paths from my home to the school. A category is green dotted which is strictly preferred over orange dashed which is strictly preferred over the category red solid. Is that correct?”

The fairy answered: “Yes, you are right. Now, we have a function o that assigns a category to every element of S . You have done this already, as you gave every edge a color. The objective function $o(x)$ is a vector which contains one component for every element of the solution, that specifies the category of the element. The order of the components of the vector o is not important, as we will see later.”



The fairy explained further: “To compare the quality of the paths you already used numerical representations, which means that you assigned numerical values in increasing order to the categories. Then, we can calculate the value of a path by addition of the numerical values of its edges, like you have done it already. Now, we can say that a path x ordinally dominates a path y , if for all numerical representations the value of path x is smaller or equal than the value of path y and there exists at least one numerical representation such that the value of path x is strictly better than the value of path y . This concept was first introduced for knapsack problems by Schäfer et al., 2021.”

Square-Village

because
 $v(\dots) + v(-) < 2 \cdot v(\dots) + v(\dots) + v(-) \forall v \in V$

Which path is better?

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 13/40

Mini asked: “I see, so we can say that the path through the park is ordinally dominated by the path through the shopping mall with the red edge, because the path has just two additional orange edges and is hence worse for every numerical representation?”

The fairy rejoiced: “Yes, correct!”
 Mini wondered: “But which path is optimal, the path with the red edge or the one around the pond?”

Square-Village

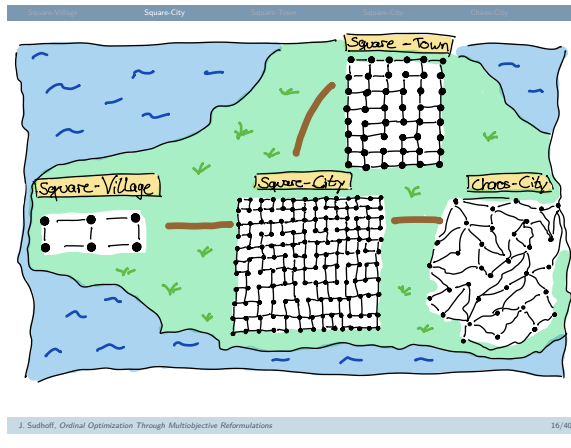
$x \in X$ is an ordinal efficient path
 $\Leftrightarrow \nexists y \in X: y \prec x$

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 14/40

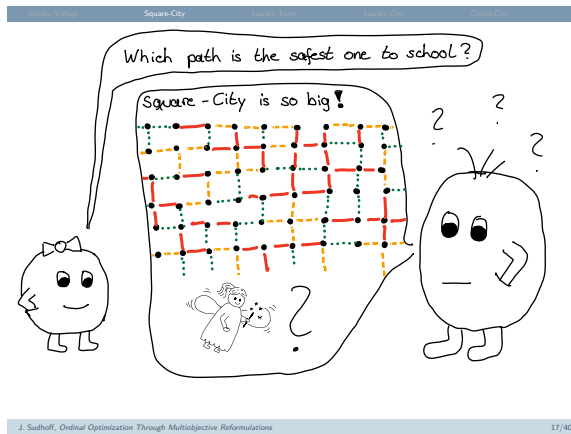
The fairy answered: “Well, in this context we talk about efficiency instead of optimality, because there can be more than one optimal solution. We call a path efficient, if there is no other path that dominates it. Here you can see all possible paths without loops from your home to the school in your village and the dominance relations.”

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 15/40

Maxi wondered: “There is more than one optimal solution?!”
 The fairy explained: “Yes, you found all efficient solutions for your village.”
 Mini cheered: “Brilliant! I can vary between both solutions – this means the way to school does not get boring!”



Mini and Maxi were reconciled as they were both correct. They thanked the fairy and the fairy offered to help them again, whenever they would need it. Mini enjoyed to vary between both paths to school and she came home safely every day. So they lived in harmony for a few years. But then Maxi got a job-offer in Square-City. Mini and Maxi decided to move to Square-City and they were lucky and quickly found a nice apartment.



A day before Mini had to go to the new school, she asked her dad: "Which path is the safest one to school now?" Maxi had no idea and replied: "That's a good question. I don't know. Square-City is so big. I don't want to compare all possible paths by hand to calculate the safest one. Let us ask the good math fairy for help." As soon as he finished the last sentence, the good math fairy appeared and asked: "How may I help you?" Maxi replied: "Fairy, do you know a way to compute ordinally safest paths efficiently?"

Square-Village

$c: X \rightarrow \mathbb{Z}_{\geq}^K, c_i(x) = |\{e \in x : o(e) = \eta_i\}|$
 $c(\rightarrow) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, c(\leftarrow) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, c(\uparrow) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$
 $c(\rightarrow \rightarrow) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, c(\rightarrow \uparrow) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$
 $c(\rightarrow \uparrow \rightarrow) = \begin{pmatrix} 2 \\ 1 \\ 1 \\ 0 \end{pmatrix}, c(\rightarrow \uparrow \rightarrow \rightarrow) = \begin{pmatrix} 3 \\ 1 \\ 1 \\ 0 \end{pmatrix}$
 $c_i(x) \hat{=} \text{number of edges of category } i$

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 18/40

The fairy explained: “Well, we can reformulate the problem. To make the idea clear, we still consider Square-Village. We can identify every edge by a vector which has the dimension of the number of categories. Every component is zero except for the entry for the category of the corresponding edge, which is one. For a path we take the sum over all those vectors and hence get a vector c that counts, in its i -th component, the number of edges in the i -th category. For example, the vector c would be $(1, 0, 1)^T$ for the path through the shopping mall with the red edge, $(1, 2, 1)^T$ for the path through the park and $(2, 2, 0)^T$ for the one around the pond. The path which contains only orange edges has the vector $(0, 4, 0)^T$.”

Square-Village

$x, y \in X, x \text{ tail dominates } y (x \leq_t y)$
 $\Leftrightarrow \sum_{j=i}^K c_j(x) \leq \sum_{j=i}^K c_j(y) \forall i=1, \dots, K$
 and $c(x) \neq c(y)$

$c(\rightarrow \rightarrow) = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, c(\rightarrow \uparrow) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$
 $i=1: 1+0+1 < 1+2+1$
 $i=2: 0+1 < 2+1$
 $i=3: 1 \leq 1$

$x \text{ ordinally dominates } y (x \leq_o y)$
 $\Leftrightarrow x \text{ tail dominates } y (x \leq_t y)$

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 19/40

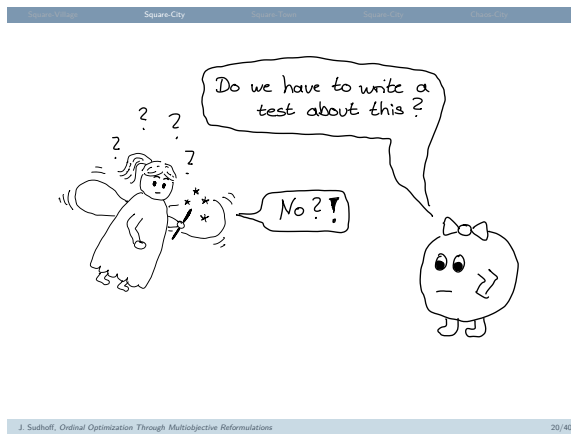
Now we can introduce tail-dominance. We say a path y is tail-dominated by a path x if

$$\sum_{j=i}^K c_j(x) \leq \sum_{j=i}^K c_j(y) \text{ for } i = 1, \dots, K$$

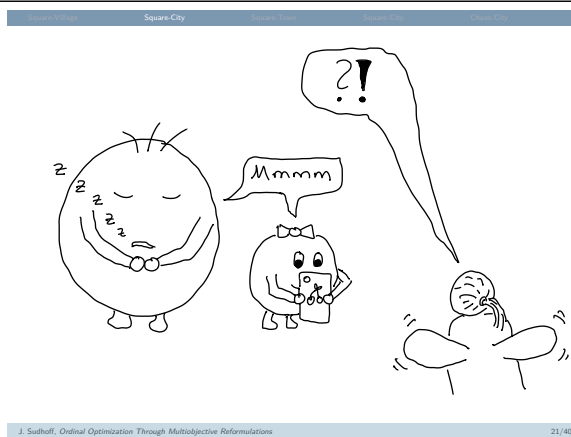
and there exists $i^* \in \{1, \dots, K\}$ such that $\sum_{j=i^*}^K c_j(x) < \sum_{j=i^*}^K c_j(y)$. We call it “tail-dominance” as we take the sum over the last components of the vector c , starting with component i .”

Mini realized: “Even with this dominance concept the path through the shopping mall tail-dominates the path through the park. Because, if $i = 1$, $1 + 0 + 1 < 1 + 2 + 1$, for $i = 2$ we get $0 + 1 < 2 + 1$, and in the last component both vectors have the value 1.”

The fairy answered: “Yes and that holds in general, hence x ordinally dominates y if and only if x tail-dominates y . The important result is that ordinal dominance and tail dominance are equivalent. I will quickly prove this result. Let’s assume ...”




Mini interrupted: "Wait! Do we have to write a test about this?"
The fairy is confused and hesitantly answered: "No... However, we assume that a path x ordinally dominates a path y ."



Then... - Are you listening?"
Mini and Maxi murmured: "Mmmm"

Square-City



Square-Village

$x \prec y \Leftrightarrow \sum_{j=i}^K c_j(x) \leq \sum_{j=i}^K c_j(y)$ for all $i=1, \dots, K$, $c(x) \neq c(y)$
 $\Leftrightarrow d_i(x) \leq d_i(y)$ for all $i=1, \dots, K$, $d(x) \neq d(y)$
 with $d \in \mathbb{Z}_{\geq}^K$, $d_i(x) = \sum_{j=i}^K c_j(x)$

Pareto dominance

$d(\leftarrow) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, $d(\dashrightarrow) = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$, $d(\rightarrow) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
 $d(\leftarrow \dashrightarrow) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$, $d(\leftarrow \rightarrow) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}$
 $d(\dashrightarrow \rightarrow) = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$

number of edges in a path

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 22/40

The fairy said: “I see... Let’s skip this part. We just use the result that ordinal and tail dominance are equivalent. Now we define a new vector d , where $d_i(x) = \sum_{j=i}^K c_j(x)$. Hence, the i -th component of d tells us the number of elements in category η_i or worse. Then obviously, tail dominance on c is equivalent to Pareto dominance on d . Pareto dominance means that $d(x)$ dominates $d(y)$ if $d(x)$ is in each component smaller or equal to $d(y)$ and it has a strictly smaller value in at least one component. We consider Square Village to illustrate this concept. If we identify the green edges with the vector $(1, 0, 0)^\top$ the orange edge with $(1, 1, 0)^\top$ and the red one with $(1, 1, 1)^\top$, we can again compute the outcome vector d of a path by taking the sum over the vectors of the edges. For example for the first path we compute $(1, 0, 0)^\top + (1, 1, 1)^\top = (2, 1, 1)^\top$ which Pareto dominates the vector $(4, 3, 1)^\top$ of the second path, as expected. Furthermore, we notice that the first component of d represents the number of edges in the path.

Square-City

$C \subseteq \mathbb{R}^K$ is a **cone** $\Leftrightarrow \alpha d \in C$ for all $d \in C$, $\alpha \in \mathbb{R}$, $\alpha > 0$
 A cone C is called

- **convex** $\Leftrightarrow C$ is a convex set
- **pointed** $\Leftrightarrow -d \notin C$ for all $d \in C \setminus \{0\}$

$C \subseteq \mathbb{R}^K$ is a **polyhedral cone**
 $\Leftrightarrow \exists A \in \mathbb{R}^{m \times K} \setminus \{0\}: C = \text{cone}(A) = \{y \in \mathbb{R}^K: Ay \geq 0\}$

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 23/40

The tail dominance and Pareto dominance can be visualized by their corresponding ordering cones. We quickly recall the definition and some properties of cones. A cone is a subset C of \mathbb{R}^K , if and only if it holds that $\alpha d \in C$ for all vectors $d \in C$ and all $\alpha > 0$. The cone is called convex, if it is a convex set and it is called pointed if $-d \notin C$ for all vectors $d \in C \setminus \{0\}$. A cone is called a polyhedral cone, if it is induced by an $m \times K$ matrix A . This means that every K -dimensional vector y is in the cone if and only if $Ay \geq 0$.

Square-City

R binary relation on \mathbb{R}^K , compatible with scalar multiplication and addition, then:

- $C_R := \{y^2 - y^1 : (y^1, y^2) \in R\}$ is a cone
- $0 \in C_R \Leftrightarrow R$ is reflexive
- C_R is pointed $\Leftrightarrow R$ is antisymmetric
- C_R is convex $\Leftrightarrow R$ is transitive

\leq_ε is a strict partial order $*$
 $C_{\leq_\varepsilon} := \{y^2 - y^1 : y^1 \leq_\varepsilon y^2\}$ is a pointed and convex cone, that does not contain 0

*Lucia E. Schäfer et al.: "The binary knapsack problem with qualitative 'avoids'" In: European Journal of Operational Research, 2021, 1(2021), pp. 5728-5744. DOI: 10.1016/j.ejor.2020.07.010

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 24/40

There exists a close connection between binary relations and cones. Every binary relation on the real vector space that is compatible with scalar multiplication and addition induces a cone. The tail dominance relation is a strict partial order, see Schäfer et al., 2021, and compatible with addition. Hence, we can conclude that as the relation is antisymmetric, the cone induced by this relation is pointed. Because the relation is transitive, the cone is convex. As the relation is not reflexive, the cone does not contain zero.

Square-City

Pareto cone: $\mathbb{R}_{\geq}^K := \{y \in \mathbb{R}^K : y \geq 0, y \neq 0\}$
 closure of the Pareto cone: $\mathbb{R}_{\geq}^K = \mathbb{R}_{\geq}^K \cup \{0\} = \text{cone}(I)$
cone induced by the identity matrix I

Theorem: $\text{cone}(A_{\leq_t}) \setminus \{0\} = C_{\leq_t}$ with

$$A_{\leq_t} = \begin{pmatrix} 1 & \dots & 1 \\ 0 & \ddots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix} \quad a_{ij} = \begin{cases} 1 & \text{if } i \leq j \\ 0 & \text{otherwise} \end{cases}$$

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 25/40

The closure of the Pareto cone is a polyhedral cone, which is induced by the identity matrix. The Pareto cone can be equivalently seen as the cone induced by the componentwise relation. Similarly, the ordinal cone induced by the relation of tail dominance can be equivalently seen as a polyhedral cone without the zero vector. The corresponding polyhedral cone is induced by a $K \times K$ matrix A_{\leq_t} with ones on the diagonal as well as the upper triangle and zeros otherwise.

Square-City

Tail Dominance Pareto Dominance

2D

3D

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 26/40

Here we see the Pareto cone and the ordinal cone for two and three categories. Every point inside the cones is dominated by the origin. As you see, the ordinal cone includes the Pareto cone. Note that tail dominance is used for the vector c while Pareto dominance is used for the vector d .

Square-City

Cone Dominance:
 $Y \subseteq \mathbb{R}^k, Y \neq \emptyset, C_{\mathcal{R}}$ cone induced by a strict binary relation \mathcal{R}

- non-dominated set $N(Y, C_{\mathcal{R}}) := \{\hat{y} \in Y : (\hat{y} - C_{\mathcal{R}}) \cap Y = \emptyset\}$
- y^1 dominates y^2 if $(y^1, y^2) \in \mathcal{R}$ and $y^1 \neq y^2$

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 27/40

Tail and Pareto dominance can be equivalently formulated with their ordering cones. Hence, I explain the concept of optimality for general cones. Assume a set Y of possible outcome vectors of an optimization problem is given and we consider a ordering cone C , which is induced by a binary relation \mathcal{R} . Then the non-dominated set of Y is defined by all vectors $\hat{y} \in Y$ such that $(\hat{y} - C_{\mathcal{R}}) \cap Y = \emptyset$. And we say a vector y^1 dominates a vector y^2 if $(y^1, y^2) \in \mathcal{R}$. This picture visualizes the definition of cone dominance. The point y^* is non-dominated, while \hat{y} is dominated by every point in the dark green area.

Square-City

Ordinal Cone Optimization Problem
 $\min_{c \in \mathbb{R}^k} c(x) \quad (\text{COP})$
 s.t. $x \in X$

Transformed Pareto Cone Optimization Problem
 $\min_{d \in \mathbb{R}_{\leq t}^k} d := A_{\leq t} c(x) \quad (\text{TOP})$
 s.t. $x \in X$

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 28/40

Now, we are able to formulate the ordinal cone optimization problem (COP), which minimizes the vector $c(x)$ w.r.t. the ordinal cone such that $x \in X$. Furthermore, we get the transformed Pareto cone optimization problem (TOP) which minimizes the vector d such that $x \in X$. The vector d can be computed by multiplying the vector c with the matrix $A_{\leq t}$, which induces the ordinal cone. We show, that those problems are equivalent.

Square-City

Theorem:
 $Y \subseteq \mathbb{R}^k, Y \neq \emptyset, C = \text{cone}(A), \text{ matrix } A \in \mathbb{R}^{m \times k}$
 $\Rightarrow A \cdot N(Y, \text{cone}(A)) \subseteq N(A \cdot Y, \mathbb{R}_{\leq t}^m)$
 If A has maximal rank $\Rightarrow A \cdot N(Y, \text{cone}(A)) = N(A \cdot Y, \mathbb{R}_{\leq t}^m)$

Corollary
 The efficient solutions for problems (OP), (COP) and (TOP) are the same.
 Proof: $A_{\leq t}$ has maximal rank

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 29/40

Towards this end, we refer to a well-known result, which says that

$$A \cdot N(Y, \text{cone}(A)) \subseteq N(A \cdot Y, \mathbb{R}_{\leq t}^m)$$

for a general matrix A . If A has maximal rank, then the sets are equal. Obviously the matrix $A_{\leq t}$, which induces the ordinal cone, has maximal rank and hence the efficient solutions of the problems (COP) and (TOP) are equivalent.

Square-City

Non-dominated outcome vectors
 $\bar{c} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, c^* = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ $\bar{d} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, d^* = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 30/40

For example, we consider again Square-Village. In this picture we see both non-dominated outcome vectors of Square-Village and their dominance cones, respectively. We look from the origin to the positive orthant. We see the non-dominated outcome vectors and the region, which is dominated by them.”

Square-Town

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 31/40

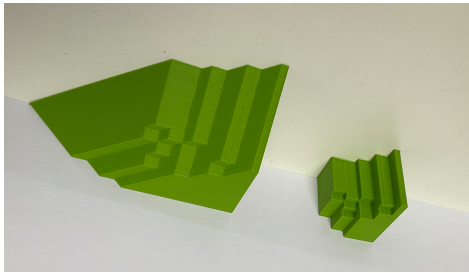
To show you another visualization of dominance cones, we will leave Mini and Maxi for a while and we make a trip to Square-Town.

Square-Town

10 efficient paths

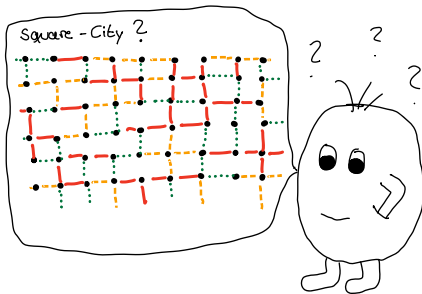
J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 32/40

Here we see a map from Square-Town. Imagine, we would like to find all ordinal-efficient paths from s to t . Square-Town has 10 efficient paths from s to t .



Rabea Freese, „Dreidimensionale Repräsentation von ordinalen und klassischen Ordnungsgesetzen“
Bachelor Thesis

Rabea Freese, a bachelor student from the University of Wuppertal, printed the corresponding dominated areas in 3D for her Bachelor Thesis, see Freese, 2022.



Let's go back to Mini, Maxi and the fairy. Maxi seemed to be confused and complained: "Fairy, I like your visualizations, but I still don't understand how this can help me to find all safe paths from home to school in Square-City."

"ordinally minimize" $\sigma(x)$
 st. $x \in X$

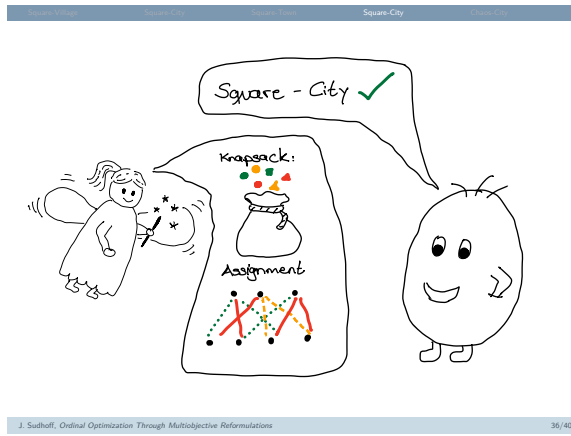
\Leftrightarrow \min_{cst} $c(x)$
 st. $x \in X$

\Leftrightarrow $\min_{st.}$ $d(x)$
 st. $x \in X$

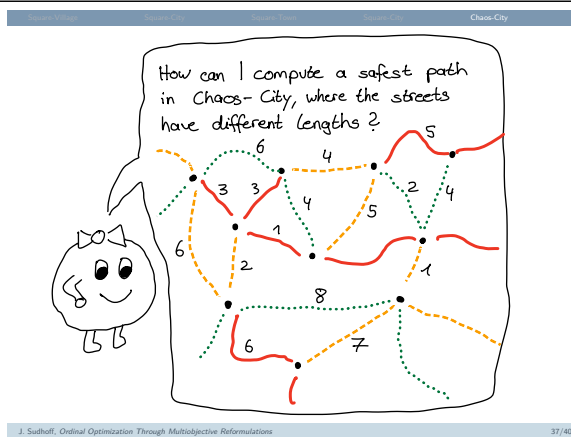
Multiobjective Shortest Path Problem
 Ernesto G.V. Martins "On a multicriteria shortest path problem"
 In: European Journal of Operational Research 45-3 (1988) pp.236-245
 DOI: 10.1016/0377-2217(88)90079-8

P.M. de las Casas et al. "An Improved Multiobjective Shortest Path Algorithm"
 In: Computers & Operations Research 135 (2021)
 DOI: 10.1016/j.cor.2021.105424

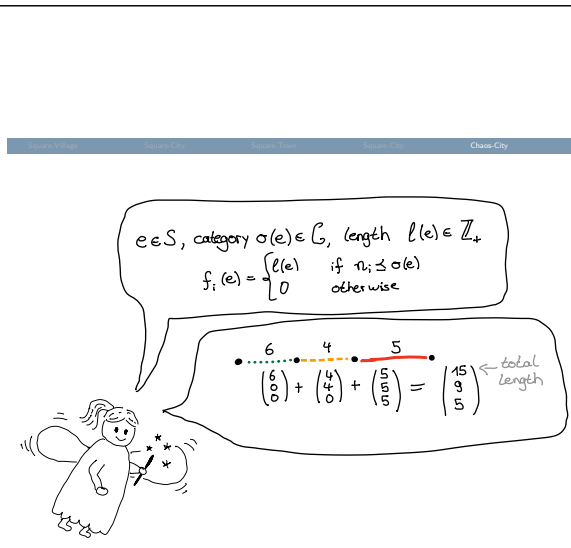
The fairy explained: "Well, we transformed our original problem into a multi-objective shortest path problem with binary coefficients, which can be solved by standard methods. For the shortest path problem I would suggest the algorithm from Martins, 1984 or the improved version of it from Casas et al., 2021."



Maxi realized: “Great, that makes the computation of the solutions possible.”
 The fairy added: “Yes and I have even more good news: You can apply this strategy to all ordinal combinatorial optimization problems and not only to shortest path problems.”



Mini asked: “That sounds good, but my big dream is to move one day to Chaos-City! There the streets have different lengths, which I denoted next to the edges. How can I compute a safest path in this case?”



The fairy replied: “Well in this case, every edge e has a category $o(e)$ and a value for the length $l(e)$. Now, we identify every edge e with a vector of the dimension of the number of categories, which has the value $l(e)$ in the first $o(e)$ entries and zero otherwise. For example, consider a path with a green edge with length 6, an orange edge with length 4 and a red edge with length 5. Hence, we get the vectors $(6, 0, 0)^T$, $(4, 4, 0)^T$ and $(5, 5, 5)^T$. The path has the outcome vector $(15, 9, 5)^T$. This can be computed similarly for every path. Note that the first component of the resulting vector gives us the total length of the path. Again, the resulting problem is a multi-objective optimization problem, which can be solved by standard methods.”

Ordinal shortest path with equal length

Knapsack:

Assignment

Ordinal shortest path with different length

Transformation:

... and they lived happily ever after

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 39/40

Mini smiled and said: “Thank you very much for your help, fairy! Now we are able to cope with many kinds of ordinal problems by transforming them into standard multi-objective optimization problems.”
 ... and they lived happily ever after.

Thank you for your attention!

J. Sudhoff, Ordinal Optimization Through Multiobjective Reformulations 40/40