BERGISCHE
UNIVERSITÄT
WUPPERTAL

Network Flow Problems with Uncertain Input Data in the Context of Supply Chain Management Applications

Zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften - Dr. rer. nat. -
am Fachbereich C - Mathematik und Naturwissenschaften - der
Bergischen Universität Wuppertal vorgelegte

genehmigte Dissertation

von

Simone Gast

2010

Tag der mündlichen Prüfung:	17.12.2010
Vorsitzende der Prüfungskommission:	Prof. Dr. Kathrin Klamroth
Erstgutachter:	Prof. Dr. Kathrin Klamroth
Zweitgutachter:	Prof. Dr. Stefan Bock

Zusammenfassung

In Anbetracht der zunehmenden Komplexität von Supply Chains gewinnt die Optimierung von Supply Chains zunehmend an Bedeutung. In vielen Fällen sind jedoch die Eingangsdaten der Supply Chains nicht exakt bekannt, weshalb eine Reihe an Unsicherheitsfaktoren entlang der gesamten Supply Chain auftreten kann. Im industriellen Bereich sind Datenunsicherheiten große Risikofaktoren, die erkannt und abgedeckt werden müssen. Daher ist es oftmals nicht ratsam, die Realität durch ein einfaches deterministisches Optimierungsmodell abzubilden.

Die Betrachtung von Unsicherheiten in mathematischen Optimierungsproblemen erfordert die Entwicklung von speziellen Methoden. Etablierte Konzepte zur Behandlung von Unsicherheiten sind Methoden aus dem Bereich der Robusten Optimierung und der Stochastischen Programmierung. In dieser Arbeit werden verschiedene existierende Ansätze untersucht und speziell auf Supply Chain Probleme erweitert und angepasst. Zusätzlich werden neue heuristische Optimierungsmethoden entwickelt, welche zu neuen Robustheitskonzepten führen.

Der Schwerpunkt liegt dabei insbesondere auf der Anwendbarkeit der entwickelten Konzepte auf reale Probleme. Um die entwickelten Algorithmen zu untersuchen und zu validieren, wird ein Referenzmodell in der Form eines Supply Chain Netzwerks realer Größe, basierend auf realen Eingangsdaten, entwickelt. Die verschiedenen präsentierten Optimierungskonzepte werden auf das Referenzmodell angewendet und die Ergebnisse verglichen.

Abstract

In view of the increasing complexity of supply chains, the optimization of supply chains becomes more and more important. However, in many cases, the input data of supply chains are not known exactly, consequence of which is that a great variety of uncertainty factors can occur along the supply chain. In the industry, data uncertainties are significant risk factors that must be detected and considered. Therefore it is often not advisable to represent the reality by means of a simple deterministic optimization model.

The consideration of uncertainties in mathematical optimization problems requires the development of particular methods. Established concepts for dealing with uncertainty are methods of Robust Optimization and Stochastic Programming. In this thesis, various existing approaches will be examined and extended to specially suit supply chain problems. In addition, the development of heuristic optimization methods leads to new robustness concepts.

In particular, the main emphasis is on the applicability of the developed concepts to real-world problems. In order to examine and validate the developed algorithms, a reference model in the form of a supply chain network of realistic size, based on real-world input data including uncertainty, is developed. The different presented optimization concepts are applied to the reference model and the results are compared.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, the manufacturing process of a commodity is not only a single process, but a chain of processes that comprises not only one, but several different organizations. In the literature, this chain is called *supply chain*. According to J.T. Mentzer, see [69], a supply chain is a network of organizations that are linked by upstream and downstream flows of products, services, finances and information from a source to a customer.

As stated in [69], the definition of the term "supply chain management" differs across authors. In this thesis, the term supply chain management denotes the planning and management of various tasks concerning supply chains, for example the choice of suppliers, the production and processing and tasks in logistics.

Supply chain management is an important research area in Operations Research. The supply chain optimization is based on input data like volumes of orders, production locations, routes and costs of transport or product portfolio.

For details about supply chain management and logistics, we refer the reader to [69] and [88].

One of the major aims of manufacturers is a cost-saving production process. However, in some cases, costs can not be handled as fixed data. A typical example is the fluctuating oil price which immensely influences all production and transportation costs.

Furthermore, a manufacturer has to optimize the production volume: A safety stock which is too large results in high storage costs, while a too small safety stock involves the risk that customers cannot be delivered in out-of-stock situations. One of the main difficulties in this optimization process is the fact, that the demand can not be predicted precisely.

In this thesis we consider the problem of data uncertainties in supply chains modeled as network flow problems. The subject of this thesis arises from a research project with the German industry partners Axxom Software AG and Henkel KGaA.

Uncertainties can occur along the entire supply chain. A typical risk factor is uncertainty in cost values, for example fluctuating transportation costs due to an instable oil price. Moreover, the demand values in supply chains often cannot be predicted precisely, as they are affected by various factors like customer behavior or seasonal fluctuations. These two kinds of uncertainties will be in the focus of this thesis.

The classical approach of modeling supply chain problems with uncertain input data is to formulate deterministic surrogate problems and solve them using various scenario sets of input data, for example best case, average case and worst case. Then every scenario corresponds to a different configuration of the input data.

This classical approach has several drawbacks: The number of possible scenarios can be very high, consequently a large number of optimization problems must be solved. If only a limited number of scenarios is considered, the actually realized scenario is possibly not covered by the scenario set. Furthermore, the optimal solutions for different scenario sets in general cannot be combined to a global optimal solution.

Hence, new approaches for dealing with uncertainty in supply chain problems must be developed.

This thesis is organized as follows.

In Chapter 2, we give a short introduction to the mathematical fields needed in this thesis.

In Chapter 3, we describe the problem and arising challenges in detail. Furthermore, we give a survey over previous developments in research in the concerned mathematical fields.

In Chapter 4, the Ant Colony Optimization metaheuristic is adapted to solve Supply Chain Management problems that are modeled as Network Flow Problems.

In Chapter 5, we examine the avoidance of small amounts of flow in supply chain networks. We develop an approach based on the Ant Colony Optimization metaheuristic as well as a Branch and Bound method using a special branching scheme based on SOS 2 conditions.

In Chapter 6, we propose methods to deal with data uncertainty in the cost values that are based on Robust Optimization. First, we extend an existing approach of D. Bertsimas and M. Sim from single-commodity to multicommodity flow problems. Additionally, we combine ideas of Robust Optimization and the Ant Colony Optimization metaheuristic and develop an ant algorithm for network flow problems with uncertain cost.

In Chapter 7, we describe the transformation of uncertain demands to uncertain costs. We develop an algorithm using additional information about costs arising due to supply deficiency and surplus delivery. In addition, a general algorithm is developed in case that no additional information is available.

In Chapter 8, algorithms for supply chain problems with uncertain demands are examined. First, we extend the Ant Colony Optimization metaheuristic concept and develop an ant algorithm for network flow problems with uncertain demand values. Moreover, we adapt the concept of recoverable robustness to supply chain problems with uncertainty in the demand values.

In Chapter 9, we propose various methods based on Stochastic Programming in order to solve supply chain problems with uncertain demands. We extend the existing approach of the two-stage model to multicommodity problems and introduce a new recourse scheme. Furthermore, we examine time-expanded networks in order to solve inventory management problems.

In Chapter 10, the supply chain optimization problem with uncertain demands is regarded from a multicriteria perspective. In particular, we elaborate the relation between the recoverable robustness approach and the Stochastic Programming concept.

In Chapter 11, the developed heuristic optimization methods are evaluated by means of several fully layered networks, which are generated at random.

In Chapter 12, the proposed methods are examined in the context of a reference model developed in cooperation with Axxom and Henkel, which represents a typical supply chain network. Different algorithms proposed in previous chapters are applied to this reference model.

Finally, Chapter 13 gives a conclusion and an outlook to future research prospects.

Parts of this thesis were previously published:

- The general transformation algorithm of uncertain demands into uncertain costs, given in Chapter 7, was previously published in [45].
- In parts, the Stochastic Programming methods concerning supply chain

problems with uncertain demands, presented in Chapter 9, were previously published in [28].

- To some extent, the computational results of Chapter 12 were previously published in [29].

Einführung

In der heutigen Zeit ist der Herstellungsprozess eines Gutes nicht einfach ein einzelner Prozess, sondern eine Kette aus Prozessen, die nicht nur eine, sondern mehrere verschiedene Einrichtungen einschließen. In der Literatur wird diese Kette als *Supply Chain* bezeichnet. Gemäß J.T. Mentzer, siehe [69], ist eine Supply Chain ein Netzwerk aus Einrichtungen, die durch auf- und abwärts fließende Flüsse aus Produkten, Dienstleistungen, Finanzen und Informationen von einer Quelle zu einem Kunden verbunden sind.

Wie in [69] dargelegt, unterscheidet sich die Definition des Begriffs "Supply Chain" von Autor zu Autor. In dieser Arbeit bezeichnet der Begriff Supply Chain Management die Planung und das Management von verschiedenen Aufgaben innerhalb einer Supply Chain, beispielsweise die Wahl der Lieferanten, die Produktion sowie die Ablaufsteuerung und Aufgaben im Bereich der Logistik.

Supply Chain Management ist ein bedeutender Forschungsbereich im Bereich Operations Research. Die Supply Chain Optimierung basiert auf Eingangsdaten wie dem Bestellvolumen, Produktionsstandorten, Transportrouten und -kosten oder dem Produktportfolio.

Für Details zu den Themen Supply Chain Management und Logistik sei der Leser auf [69] und [88] verwiesen.

Eines der Hauptziele von Herstellerfirmen ist ein kostensparender Produktionsprozess. In manchen Fällen können Kosten jedoch nicht als unveränderliche Daten behandelt werden. Ein typisches Beispiel hierfür ist der schwankende Ölpreis, der immensen Einfluss auf die Produktions- und Transportkosten hat.

Weiterhin muss eine Herstellerfirma ihren Produktionsumfang optimieren: Ein zu hoher Sicherheitsbestand zieht zu hohe Lagerkosten nach sich, während ein zu niedriger Sicherheitsbestand das Risiko beinhaltet, dass Kunden im Falle leerer Lager nicht mehr beliefert werden können. Eine der Hauptschwierigkeiten in diesem Optimierungsprozess ist die Tatsache, dass der Bedarf nicht präzise vorausgesagt werden kann.

In dieser Arbeit betrachten wir das Problem von Datenunsicherheiten in Supply Chains modelliert als Netzwerkflussproblem. Das Thema dieser Arbeit ist aus einem gemeinsamen Forschungsprojekt mit den deutschen Industriepartnern Axxom Software AG und Henkel KGaA entstanden.

Unsicherheiten können entlang der gesamten Supply Chain entstehen. Ein typischer Risikofaktor ist Unsicherheit in den Kostenwerten, beispielsweise schwankende Transportkosten aufgrund eines instabilen Ölpreises. Darüber hinaus können Bedarfswerte in Supply Chains oftmals nicht genau vorausgesagt werden, da sie durch verschiedene Faktoren wie Kundenverhalten oder saisonale Schwankungen beeinflusst werden. Diese zwei Arten von Unsicherheiten werden im Fokus dieser Arbeit stehen.

Der klassische Ansatz zur Modellierung von Supply Chain Problemen mit unsicheren Eingangsdaten ist die Formulierung von deterministischen Ersatzproblemen, welche für verschiedene Szenarienmengen an Eingangsdaten gelöst werden, beispielsweise Bestfall, Durchschnittlicher Fall und Schlimmstfall. Dann entspricht jedes Szenario einer unterschiedlichen Konfiguration der Eingangsdaten.

Dieser klassische Ansatz hat verschiedene Nachteile: Die Anzahl an möglichen Szenarien kann sehr hoch sein. Folglich muss eine große Anzahl an Optimierungsproblemen gelöst werden. Falls nur eine begrenzte Menge an Szenarien betrachtet wird, ist möglicherweise das tatsächlich realisierte Szenario nicht durch diese Szenarienmenge abgedeckt. Weiterhin können im Allgemeinen die optimalen Lösungen für verschiedene Szenarienmengen nicht zu einer global optimalen Lösung kombiniert werden.

Folglich müssen neue Ansätze zum Umgang mit Unsicherheiten in Supply Chains entwickelt werden.

Diese Arbeit ist wie folgt aufgebaut:

In Kapitel 2 wird eine kurze Einführung in die mathematischen Felder gegeben, die in dieser Arbeit benötigt werden.

In Kapitel 3 wird die Problemstellung und daraus hervorgehende Herausforderungen im Detail beschrieben. Weiterhin wird eine Übersicht über frühere Forschungsergebnisse in den jeweiligen mathematischen Feldern gegeben.

In Kapitel 4 wird die Ant Colony Optimization Metaheuristik auf die Lösung von Supply Chain Problemen, welche als Netzwerkflussprobleme formuliert sind, angepasst.

In Kapitel 5 wird die Vermeidung von kleinen Flussmengen in Supply Chain Netzwerken beschrieben. Dazu werden ein Ansatz basierend auf der Ant Colony

Optimization Metaheuristik sowie ein Branch-and-Bound Verfahren unter Verwendung eines speziellen Branchingschemas, basierend auf SOS 2 Bedingungen, entwickelt.

In Kapitel 6 werden Verfahren zur Behandlung unsicherer Eingangsdaten beschrieben, welche auf Robuster Optimierung basieren. Zunächst wird ein bereits existierender Ansatz von D. Bertsimas und M. Sim von Netzwerkflussproblemen mit einem Gut (single-commodity) auf Netzwerkflussprobleme mit mehreren Gütern (multicommodity) erweitert. Zusätzlich werden Ideen der Robusten Optimierung und der Ant Colony Optimization Metaheuristik kombiniert und ein Ameisenalgorithmus zur Lösung von Netzwerkflussproblemen mit unsicheren Kosten entwickelt.

In Kapitel 7 wird die Transformation von unsicheren Bedarfswerten auf unsichere Kostenwerte beschrieben. Ein Algorithmus, der Zusatzinformationen über Kosten, die durch zu geringe Liefermengen oder überzählig gelieferte Güter entstehen, verwertet, wird entwickelt. Zusätzlich wird ein allgemeiner Algorithmus für den Fall, dass derartige Informationen nicht vorliegen, entwickelt.

In Kapitel 8 werden Algorithmen für Supply Chain Probleme mit unsicherem Bedarf untersucht. Zunächst wird das Konzept der Ant Colony Optimization Metaheuristik erweitert und ein Ameisenalgorithmus für Netzwerkflussprobleme mit unsicheren Bedarfswerten entwickelt. Weiterhin wird das Konzept der Recoverable Robustness an Supply Chain Probleme mit Unsicherheiten in den Bedarfswerten angepasst.

In Kapitel 9 werden verschiedene Methoden zur Lösung von Supply Chain Problemen mit unsicherem Bedarf vorgestellt, welche auf Stochastischer Programmierung basieren. Der bereits existierende Ansatz des Two-Stage Verfahrens wird auf Multicommodity-Probleme erweitert und ein neues Kompensationsschema eingeführt. Darüber hinaus werden zeitexpandierte Netzwerke zur Lösung von Bestandsmanagementproblemen untersucht.

In Kapitel 10 wird das Supply Chain Optimierungsproblem mit unsicherem Bedarf aus einer multikriteriellen Perspektive betrachtet. Insbesondere werden dabei Zusammenhänge zwischen dem Ansatz der Recoverable Robustness und dem Konzept der Stochastischen Programmierung herausgearbeitet.

In Kapitel 11 werden die entwickelten heuristischen Optimierungsverfahren an Hand verschiedener vollständig geschichteter Netzwerke (fully layered networks) evaluiert, welche zufallsgestützt generiert werden.

In Kapitel 12 werden die vorgestellten Verfahren im Kontext eines Referenzmodells untersucht, welches in Zusammenarbeit mit Axxom und Henkel entwickelt wurde und ein typisches Supply Chain Netzwerk darstellt. Verschiedene Optimierungsalgorithmen, welche in den voranstehenden Kapiteln vorgestellt wurden, werden auf dieses Referenzmodell angewendet.

Schließlich gibt Kapitel 13 ein abschließendes Fazit sowie einen Ausblick auf potentielle zukünftige Forschungsbereiche.

Teile dieser Arbeit wurden zuvor veröffentlicht:

- Der allgemeine Transformationsalgorithmus von unsicherem Bedarf in unsichere Kosten, beschrieben in Kapitel 7, wurde zuvor veröffentlicht in [45].
- Zu Teilen wurden die Verfahren der Stochastischen Programmierung betreffend der Supply Chain Probleme mit unsicherem Bedarf, vorgestellt in Kapitel 9, in [28] veröffentlicht.
- Zu Teilen wurden die numerischen Ergebnisse aus Kapitel 12 in [29] veröffentlicht.

Chapter 2

Mathematical Background

This chapter contains a brief overview regarding some mathematical fields which are important for this thesis. First, a brief introduction to graph theory is given. Moreover, we consider the formulation of linear optimization and network flow problems. Furthermore, we state the basic concepts of Multicriteria Optimization. Finally, discrete and continuous probability spaces in probability theory are introduced.

2.1 Graphs

In this section, a short introduction to the basic concepts of graphs is given. The given descriptions are based on the books of M. Aigner [5] and H. Hamacher and K. Klamroth [55].

A *graph* $G = (\mathcal{N}, \mathcal{A})$ consists of a finite set of *nodes* (vertices) $\mathcal{N}$ and a finite set of *arcs* (edges) $\mathcal{A}$. An arc is defined by two endpoints i, $j \in \mathcal{N}$ and is denoted by $[i, j]$.

A *directed graph* $G = (\mathcal{N}, \mathcal{A})$ is a graph, where a direction is associated with each arc. An arc of a directed graph consists of a start node $i \in \mathcal{A}$ and an end node $j \in \mathcal{A}$ and is denoted by (i, j). A directed graph is also called *digraph.*

A *path* P in G is a sequence of nodes $P = (i_1, i_2, \ldots, i_l)$ where $i_p \in \mathcal{N}$, $p = 1, \ldots, l$, and

$$\begin{cases} e_p = [i_p, i_{p+1}] \in \mathcal{A} & \text{for graphs} \\ e_p = (i_p, i_{p+1}) \in \mathcal{A} \text{ or } (i_{p+1}, i_p) \in \mathcal{A} & \text{for digraphs} \end{cases}$$

and $i_p \neq i_{p+1}$ $\forall p = 1, \ldots, l-1$.

A (di-)path $P = (i_1, i_2, \ldots, i_l)$ is called a *(di-)cycle* if $i_l = i_1$.

A (di-)graph G is called *connected*, if every pair of nodes in $\mathcal{A}$ is connected by a path.

A (di-)graph G is called *acyclic* if G does not contain a (di-)cycle.

Let $n := |\mathcal{N}|$ be the number of nodes and $m := |\mathcal{A}|$ be the number of arcs. The *adjacency matrix* $AD = AD(G)$ of a (di-)graph G is an $n \times n$-matrix defined by

$$a_{ij} = \begin{cases} 1 & \text{if } [i,j] \in \mathcal{A} \text{ (or } (i,j) \in \mathcal{A} \text{ respectively, for digraphs)} \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathcal{A} = \{e_1, \dots, e_m\}$. The *node-arc incidence matrix* $A = A(G)$ of a graph G is an $n \times m$-matrix defined by

$$a_{ij} = \begin{cases} 1 & \text{if } i \text{ is endpoint of } e_j \\ 0 & \text{otherwise} \end{cases}$$

The incidence matrix $A = A(G)$ of a digraph G is an $n \times m$-matrix defined by

$$a_{ij} = \begin{cases} 1 & \text{if } i \text{ is start node of } e_j \\ -1 & \text{if } i \text{ is end node of } e_j \\ 0 & \text{otherwise} \end{cases}$$

2.2 Linear Optimization

The given definitions are based on the book of G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd, see [74].

A general linear optimization problem, also called *linear program (LP)*, can be given in the *inequality form*

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t. } & Ax \le b \\ & x \ge 0 \end{aligned} \tag{2.1}$$

and in the *standard form*

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t. } & Ax = b \\ & x \ge 0 \end{aligned} \tag{2.2}$$

where $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. These two forms are completely equivalent.

The set $S = \{x \in \mathbb{R}^n : Ax = b, x \leq 0\}$ is called the *feasible region* and an $x \in S$ is called a *feasible solution* of the LP in standard form. An $x^* \in S$ with $cx^* \leq cx \quad \forall x \in S$ is called an *optimal solution* of the linear optimization problem.

There exist very efficient methods to solve linear optimization problems. One example is the Simplex Method, which was developed by G. Dantzig in 1947. For details, see [55] or [74], for example. Furthermore, a class of algorithms called *interior point methods* can be efficiently applied on linear optimization problems, see [96] for details.

2.3 Network Flows

In this section, we will give a short introduction to network flows.

2.3.1 Network Optimization

The introduction to Network Optimization follows the books of R.K. Ahuja, T.L. Magnanti and J.B. Orlin, see [4], and H. Hamacher and K. Klamroth, see [55].

Let $G = (\mathcal{N}, \mathcal{A})$ be a connected digraph with node set $\mathcal{N}$ and arc set $\mathcal{A}$. For every node $i \in \mathcal{N}$, a value $b_i \in \mathbb{Z}$ gives the information about supply (if $b_i > 0$) and demand (if $b_i < 0$) in the respective node. Node i is called a *transshipment node* if $b_i = 0$. We usually assume that $\sum\limits_{i \in \mathcal{N}} b_i = 0$. The decision variables x_{ij} represent the flow on an arc $(i,j) \in \mathcal{A}$. The cost per flow unit on arc (i,j) is given by c_{ij}, while l_{ij} und u_{ij} with $l_{ij} \leq u_{ij}$ define the lower and upper capacity bounds on arc (i,j), respectively.

The *Network Flow Problem (NFP)* is defined as

$$\begin{aligned}
\min \quad & \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij} & & \\
\text{s.t.} \quad & \sum_{j:(i,j) \in \mathcal{A}} x_{ij} - \sum_{j:(j,i) \in \mathcal{A}} x_{ji} = b_i & \forall i \in \mathcal{N} & \qquad (2.3) \\
& l_{ij} \leq x_{ij} \leq u_{ij} & \forall (i,j) \in \mathcal{A}. &
\end{aligned}$$

The NFP is also called *Minimum Cost Flow Problem (MCFP)*.

In matrix form, the NFP can also be written as

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t. } & Ax = b \\ & l \leq x \leq u \end{aligned} \tag{2.4}$$

where A is the node-arc incidence matrix of G.

The equality constraints in (2.3) and (2.4) are called *mass balance constraints* or *flow conservation constraints*. The inequality constraints are referred to as *flow bound constraints*.

Each network containing arcs with nonzero lower bounds can be transformed to an equivalent network with zero lower bounds only (see [4]). Furthermore, each network can be transformed to a single-source single-sink network (see [55]).

In many cases, all arc and node data are assumed to be integral. In practice, this so-called *integrality assumption* is not very restrictive, because rational data can be converted into integer data by multiplication with a suitably large number. Furthermore, irrational numbers are transformed to rational numbers when represented on a computer, anyway.

For solving minimum cost flow problems, there exist a variety of efficient algorithms (see Subsection 3.3.1), for example the Cost Scaling Algorithm and the Network Simplex Method. An exhaustive survey on various algorithms can be found in [4], for example.

The network flow model is a static model that has no underlying temporal dimensions. In order to obtain a dynamic network flow model, we can generate a copy of the static network for each point in time. Furthermore, the static networks are connected via arcs as temporal linkages. The resulting dynamic network is called *time-expanded network*. See [4] for details.

2.3.2 Multicommodity Flows

In Subsection 2.3.1, we considered network models composed of one commodity which is sent from the source node(s) to the sink node(s). A network flow problem with multiple commodities sharing the same network facilities is called *Multicommodity Flow Problem* (MFP). According to [4], the MFP for p commodities is defined as follows:

$$\begin{aligned} \min \ & \sum_{k=1}^{p} c^k x^k \\ \text{s.t.} \ & \sum_{k=1}^{p} x_{ij}^k \leq \kappa_{ij} & \forall (i,j) \in \mathcal{A} \\ & A x^k = b^k & k = 1, \ldots, p \\ & 0 \leq x_{ij}^k \leq u_{ij}^k & \forall (i,j) \in \mathcal{A},\ k = 1, \ldots, p \end{aligned} \tag{2.5}$$

where A is the node-arc incidence matrix of the network graph $G = (\mathcal{N}, \mathcal{A})$. c^k defines the cost values in matrix form for commodity k, x_{ij}^k and x^k the flow values for arc (i,j) and in matrix form, u_{ij}^k the upper capacity bounds for arc (i,j) and b^k gives the information about supply and demand in matrix form.

The first constraint can be explained as follows: As the individual commodities must share common arc capacities, the so-called *bundle constraints*

$$\sum_{k=1}^{p} x_{ij}^k \leq \kappa_{ij} \ \forall (i,j) \in \mathcal{A}$$

restrict the total flow $\sum\limits_{k=1}^{p} x_{ij}^k$ of all commodities on arc (i,j) to at most κ_{ij}. As without the bundle constraints, the MFP could be decomposed into p single-commodity problems, these constraints are also called *complicating constraints.* Due to the special problem structure, most solution methods are based on decomposition methods (see [38]).

2.4 Multicriteria Optimization

The basic definitions given in this section are based on the book of M. Ehrgott, see [36].

In connection with Multicriteria Optimization, we shall use the following notations concerning orders in $\mathbb{R}^n$ as given in [36]. Let $x,\ y \in \mathbb{R}^n$.

Notation	Definition
$x \leq y$	if $x_i \leq y_i, \quad i = 1, \ldots, n$
$x < y$	if $x_i \leq y_i, \quad i = 1, \ldots, n; \quad x \neq y$

Let X be the feasible set for the optimization problem and $f_i : X \to \mathbb{R}$ scalar objective functions, $i = 1, \dots, Q$. Then the *multicriteria optimization problem* (MOP) can be written as

$$\begin{aligned} &\text{vmin } (f_1(x), \dots, f_Q(x)) \\ &\quad \text{s.t. } x \in X \end{aligned} \tag{2.6}$$

Let $f(x) := (f_1(x), \dots, f_Q(x))$.

A solution x^* of Problem (2.6) is called *Pareto optimal* if there is no $x \in X$ such that $f(x) < f(x^*)$. A solution x^* is called *weakly Pareto optimal* if there is no $x \in X$ such that $f_i(x) < f_i(x^*)$ for all $i = 1, \dots, Q$. A solution x^* is called *strictly Pareto optimal* if there is no $x \in X$, $x \neq x^*$, such that $f(x) \leq f(x^*)$.

The set of all Pareto optimal solutions is called the *Pareto set.*

If x^* is Pareto optimal, $f(x^*)$ is called *efficient.* If x^* is weakly Pareto optimal, $f(x^*)$ is called *weakly efficient.* The set of all efficient points is called the *efficient set.*

If x^1, $x^2 \in X$ and $f(x^1) < f(x^2)$, we say that x^1 dominates x^2 and $f(x^1)$ dominates $f(x^2)$.

A *scalarization* is a single-objective problem that combines different objective functions and constraints into a single objective optimization problem, where the aim is to find all efficient points of the initial multiobjective problem by repeatedly solving the single-objective optimization problem with varying parameters.

2.5 Probability Theory

In this section, we give a short overview over some important terms and definitions in probability theory following the books of U. Krengel, see [63], and A. Papoulis, see [78].

2.5.1 Discrete Probability Spaces

Let Ω be a non-empty, at most countable set. A function $P : \Omega \to [0, 1]$ is called a *probability mass*, if

$$\begin{aligned} P(\Omega) &= 1 \\ P(A) &\geq 0 \ \forall A \subseteq \Omega \\ P(\bigcup_{i=1}^{\infty} A_i) &= \sum_{i=1}^{\infty} P(A_i) \text{ for any series of pairwise disjoint } A_i \in \Omega \end{aligned} \tag{2.7}$$

The pair (Ω, P) is called a *discrete probability space.*

Let (Ω, P) be a discrete probability space and χ an arbitrary set. Then a function $X : \Omega \to \chi$ is called a χ-valued *random variable.*

A probability mass P_X on $\chi_X := \{X(\omega) : \omega \in \Omega\}$ where $P_X(x) = P(\{\omega \in \Omega : X(\omega) = x\})$ is called the *distribution function* of X.

If (Ω, P) is a discrete probability space and X a real-valued random variable, the *expected value* $\mathbb{E}(X)$ of X is defined as $\mathbb{E}(X) = \sum\limits_{\omega \in \Omega} X(\omega)P(\omega)$.

If $E(X^2)$ exists, the *variance* of X is defined as $Var(X) = \mathbb{E}((X - \mathbb{E}(X))^2)$. The *standard deviation* of X is defined as $\sigma_X = \sqrt{Var(X)}$.

2.5.2 Continuous Probability Spaces

Let Ω be a non-empty set and $\mathcal{A}$ a σ-algebra of subsets of Ω. A function P: $\mathcal{A} \to [0, 1]$ is called *probability mass* if

$$\begin{aligned} P(\Omega) &= 1 \\ P(A) &\geq 0 \ \forall A \in \mathcal{A} \\ P(\bigcup_{i=1}^{\infty} A_i) &= \sum_{i=1}^{\infty} P(A_i) \text{ for pairwise disjoint } A_i \in \mathcal{A} \end{aligned} \tag{2.8}$$

The triplet $(\Omega, \mathcal{A}, P)$ is called *probability space.* The pair $(\Omega, \mathcal{A})$ is called *measurable space.*

A function $F : \mathbb{R} \to [0, 1]$ is called *distribution function*, if F is continuous from the right, monotonically increasing, $\lim\limits_{x \to -\infty} F(x) = 0$ and $\lim\limits_{x \to \infty} F(x) = 1$.

A nonnegative function $f : \mathbb{R} \to \mathbb{R}_0^+$ with $\int\limits_{-\infty}^{\infty} f(x)\,dx = 1$ is called *density function.*

If f is a density function, then $F(x) = \int\limits_{-\infty}^{x} f(t)\,dt$ is a distribution function.

Let $(\Omega, \mathcal{A})$ and $(\Omega', \mathcal{A}')$ be measurable spaces. A function $f : \Omega \to \Omega'$ is called *measurable*, if $f^{-1}(A') \in \mathcal{A}$ for all $A' \in \mathcal{A}'$.

Let $(\Omega, \mathcal{A}, P)$ be a probability space and let $(\Omega', \mathcal{A}')$ be a measurable space. A *random variable* X is a measurable function $X : \Omega \to \Omega'$.

The *distribution* of X is a probability mass P_X defined by $P_X(A') = P(X \in A')$. If X is real-valued, the distribution function is $F_X(x) = P(X \leq x) = P_X(]-\infty, x])$. If the distribution of X has a density function f, we shortly say that X has the *density* f.

The *expected value* or *mean* of X is defined as $\mathbb{E}(X) = \int\limits_{\Omega} X \; dP$. If X has a density f then $\mathbb{E}(X) = \int\limits_{-\infty}^{\infty} x f(x) \; dx$.

The *variance* of X is defined by $Var(X) = \mathbb{E}[(X - \mathbb{E}(X))^2]$. $\sigma_X = \sqrt{Var(X)}$ is called the *standard-deviation* of X.

The distribution with density function

$$\varphi_{\mu,\sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

is called the *Gaussian distribution* or *normal distribution* with expected value μ and variance σ^2, short $\mathcal{N}(\mu, \sigma^2)$.

Chapter 3

Problem Formulation and Objectives

The optimization of supply chains depends on diverse input data. Dependent on volumes of orders, routes and costs of transport and diverse commodities, the entire supply chain shall be optimized. When modeling a supply chain as optimization problem, a compromise between a simple and tractable model and a model that sufficiently represents the reality is needed.

3.1 Network Representation of Supply Chain Problems

One possible model of a supply chain is the network representation as a network G with nodes $\mathcal{N}$ and arcs $\mathcal{A}$, cost values, demand data and capacity restrictions. Then the corresponding optimization problem is a network flow problem.

As for the advantages concerning computational tractability, the network representation is maintained as a base model, both in the research project with Axxom Software AG and Henkel KGaA in general and in particular in this thesis.

In the context of the supply chain applications in our research project, a supply chain model consists of various supply chain stages. Examples are transportation of raw materials, production and distribution stages. In practice, these stages must be processed successively.

Therefore, except as noted otherwise, we always assume in the following that the network G is a layered network. The term *layered network* is defined as follows:

Definition 3.1. *A network $G = (\mathcal{N}, \mathcal{A})$ with node set $\mathcal{N}$ and arc set $\mathcal{A}$ is called a layered network with s layers if the node set can be partitioned into s subsets $l_1, \ldots, l_s$, such that the following conditions hold:*

1. *For every node $i \in l_k$, $k = 1, \ldots, s-1$: For every arc $(i,j) \in \mathcal{A}$ it holds that node $j \in l_{k+1}$.*
2. *For every node $i \in l_k$, $k = 2, \ldots, s$: For every arc $(j,i) \in \mathcal{A}$ it holds that node $j \in l_{k-1}$.*
3. *There are no incoming arcs for nodes in l_1. There are no outgoing arcs for nodes in l_s.*

The subsets l_k, $k = 1, \ldots, s$, are called layers.

A layered network G is called fully layered, if for every node $i \in l_k$, $k = 1, \ldots, s-1$ and for every node $j \in l_{k+1}$, the arc (i,j) is contained in the arc set $\mathcal{A}$.

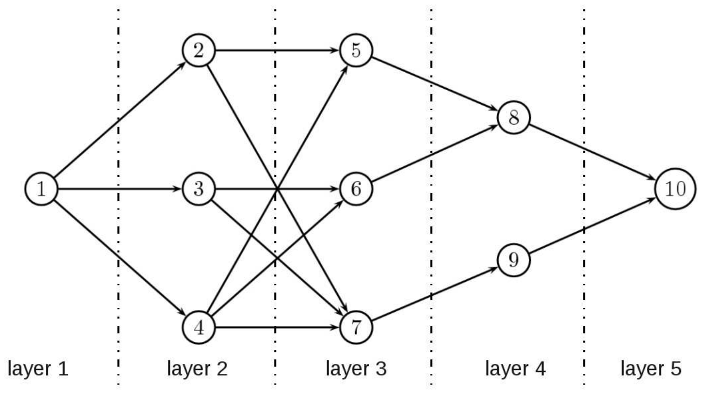


Figure 3.1: Layered network

3.2 Uncertainties in Supply Chains

In the industry, data uncertainties, which can occur along the entire supply chain, are significant risk factors. Target figures like the total cost or the service level can significantly be influenced by data uncertainty.

Typical risk factors are:

- Uncertainties in <u>costs</u>:
  - raw material cost
  - production and processing cost
  - logistics and stock cost
- Uncertainties in <u>demand</u>:
  - customer behavior
  - seasonal fluctuation
  - fluctuations in economic activity
  - competitor behavior

  If the uncertainty in demand is very high, the safety stock must usually be increased in order to avoid an "out-of-stock" situation.

There exist more risk factors like capacity fluctuations, failures of machines or fluctuations in material quality, for example. These risk factors will not be considered in this thesis, because we want to focus on the two major risk factors, uncertainties in costs and uncertainties in demand.

3.3 Literature Review

In this thesis, we focus on supply chain management problems that are formulated as Network Flow Problems. Therefore we give a short literature review concerning algorithmic approaches for minimum cost flow problems and network flow problems in supply chains first.

In order to deal with uncertainty, various mathematical methods exist. Two major fields have been established: Robust Optimization and Stochastic Programming. We give a brief overview over the developments in these two fields, as the research in the following chapters of this thesis is based on concepts from the fields Robust Optimization and Stochastic Programming in large part.

Furthermore, we will develop Robust Optimization methods based on the Ant Colony Optimization metaheuristic in this thesis. We therefore give a short literature survey concerning Ant Colony Optimization.

Finally, we review other heuristic approaches in the context of supply chain management applications.

The literature on the covered mathematical fields is exhaustive, such that we can only give a small sample of references here. We will focus on well-established references as well as papers that are specially of interest in the context of this thesis.

3.3.1 Network Flow Problems

Minimum Cost Flow Problems There exist a variety of algorithms for solving minimum cost flow problems, which can be divided in two main classes: Feasible Flow Algorithms (FFA) and Optimal Infeasible Flow Algorithms (OIFA), see [97]. Feasible Flow Algorithms start with an arbitrary feasible solution and generate a new feasible solution in each iteration, until the optimal solution is found. Optimal Infeasible Flow Algorithms use infeasible flows during the optimization process which comply with optimality conditions and which are adapted until all flow conservation and capacity constraints are satisfied.

Important representatives of the FFA class are the Cycle Canceling Algorithm, the Cost Scaling Algorithm and some implementations of the Network Simplex Algorithm.

The Cycle Canceling Algorithm was developed by M. Klein in 1967, see [60]. The algorithm improves the objective function by searching negative cost cycles in the residual network and augmenting the flow on these cycles. Klein demonstrates the approach by means of assignment and transportation problems.

The Cost Scaling technique was developed independently by H. Röck in 1980, see [82], and R.G. Bland and D.L. Jensen in 1985, see [19] and [20]. The Cost Scaling Algorithm uses a successive approximation technique based on cost scaling. The algorithm of Röck was improved by A.V. Goldberg and R.E. Tarjan in [50]. Further improvements by R.K. Ahuja et al., see [3], and J.B. Orlin, see [76], followed.

The Network Simplex Algorithm is a specialized version of the linear programming simplex method. The operations of the network simplex method can be simplified compared to linear programming due to the spanning tree structure of bases of the minimum cost flow problem. A variety of Network Simplex implementations can be found in the literature, see [59] for example. Some variations of the Network Simplex Algorithm belong to the FFA class.

Important representatives of the OIFA class are the Successive Shortest Path Algorithm, the Capacity Scaling Algorithm and the Parametric and Dual Network Simplex Algorithm.

The Successive Shortest Path Algorithm was proposed by J. Edmonds and R.M. Karp, see [35]. Starting with the zero-flow, the algorithm compensates imbalances between nodes with surplus flow and nodes with flow deficiency, until the flow conservation constraints are satisfied for all nodes. The reduced cost optimality criterion is maintained throughout the optimization process.

The Capacity Scaling Algorithm was presented by J. Edmonds and R.M. Karp, see [35]. The algorithm is a variation of the Successive Shortest Path Algorithm. The running time of the original algorithm is enhanced by the following improvement: In every iteration only those shortest paths are considered on which a minimum amount of flow can be sent.

The Parametric and the Dual Network Simplex Algorithm are variations of the Network Simplex Algorithm which belong to the OIFA class. During the optimization phase of the Parametric Network Simplex Algorithm, the flow conservation constraints are violated, while the Dual Network Simplex Algorithm violates the capacity constraints. In [2], R.K. Ahuja et al. describe a Parametric Network Simplex Algorithm. In [6], A.I. Ali et al. give an implementation of the Dual Network Simplex Algorithm using dual re-optimization procedures.

As we can only give a short cross-section of algorithms for minimum cost flow problems in this context, we refer the reader to [4] for an exhaustive survey.

Supply Chain Problems The modeling of supply chain problems as minimum cost flow problems is a wide-spread approach. As the list of supply chain applications modeled as minimum cost flow problems is exhaustive, we can only give a small sample of references here.

In [57], W.S. Jewell formulates the warehousing and distribution of a seasonal product as minimum cost flow problem.

In [34], R.C. Dorsey et al. model a multifacility, multiproduct production scheduling problem as minimum cost flow problem. A finite planning horizon of discrete production periods is considered. Standard network flow algorithms as well as special purpose algorithms are applied.

In [41], J.R. Evans proposes the modeling of dynamic multistage production and inventory planning problems as single commodity network flow problems. In particular, single period problems are examined in detail.

In recent years, new approaches in theory and practice of minimum cost flow problems were developed, integrating different mathematical fields. We will give two examples for this:

In [26], S.-W. Chioua examines a minimum cost product flow in a supply chain network consisting of manufacturers, retailers and consumers. A combinatorial approximation algorithm based on simplicial decomposition is presented.

In [49], M. Ghateea and S.M. Hashemi consider fuzzy supply and demand in network design problems, modeled as multi-objective fuzzy minimum cost flow problems. In particular, the most optimistic and the most pessimistic solutions are examined.

Time-Expanded Networks The theory of time-expanded networks is an important feature in recent applications of network flow problems. We will give a small sample of references here:

In [54], H.W. Hamacher and S. Tifecki model building evacuation problems as time-expanded network flow problem. The simultaneous handling of various objective functions by means of lexicographical optimization is examined.

In [61], E. Köhler et al. examine time-expanded networks in the context of applications in road traffic control. A special focus is on flow-dependent transit times on the arcs.

In [43], L. Fleischer and M. Skutella examine the storage of flow at intermediate nodes in time-expanded minimum cost flow problems. A capacity scaling method for time-expanded minimum cost flow problems with costs proportional to transit times is proposed.

M. Skutella summarizes the research concerning dynamic flows in [89]. Research results starting from the first approach to time-expanded networks by L.R. Ford and D.R. Fulkerson ([44]) as well as recent applications are covered.

3.3.2 Robust Optimization

Robust Optimization Models Robust Optimization is an approach to address optimization problems with uncertain input data. The goal of Robust Optimization is to find a solution to the optimization problem, which can cope best with all possible realizations of the uncertain data. In general, this solution is not optimal for every realization of the uncertain input data, but performs well even in the worst case. In many cases, Robust Optimization deals with uncertain input parameters which are known only within certain bounds.

The first robust approach for linear optimization problems was presented by A.L. Soyster in 1973 (see [90]). Soyster proposes a linear optimization model which leads to a solution that is feasible for all input data. However, this approach generates rather over-conservative solutions.

In [72], J.M. Mulvey et al. present a scenario based robust approach. The model combines solution robustness and model robustness. A main feature of the approach of Mulvey et al. is the fact that the robust solutions contain common components for all scenarios as well as components which are scenario specific. However, the computational efforts for large problem instances can be very high.

Less conservative robust models were proposed by A. Ben-Tal and A. Nemirovski starting from 1998 (see [11], [12] and [13], for example) as well as L. El-Ghaoui et al. (see [39] and [40]). In these papers, uncertain linear problems with ellipsoidal uncertainties are considered, which allow the approximation of more complex uncertainty sets. The resulting robust optimization problems, called *robust counterparts*, involve conic quadratic problems.

A robust approach which allows to control the level of conservatism in the robust solution was proposed by D. Bertsimas and M. Sim (see [14] and [15]). A robustness parameter specifies the number of coefficients that are allowed to change such that the optimal solution can be guaranteed to be feasible. Moreover, the approach of Bertsimas and Sim is directly applicable to discrete optimization problems. In [16], an application on optimal control of supply chains subject to stochastic demand is presented.

P. Kouvelis and G. Yu present a scenario-based robust framework specially designed for discrete optimization problems (see [62]). A drawback of their approach is the fact, that the robust counterpart of many polynomially solvable discrete optimization problems is NP-hard.

In [64] and [91], the new approach *recoverable robustness* of S. Stiller et al. is presented. In recoverable robustness, the aim is to find solutions to an optimization problem with uncertain input data, which can be made feasible, or *recovered*, within a given budget for all possible situations. The optimization process consists of two phases, a planning phase and a recovery phase.

M. Fischetti and M. Monaci propose a robustness concept called *Light Robustness* in [42]. Light Robustness is based on the approach given by Bertsimas and Sim ([14]). The aim is to balance a possible violation of the constraints and the quality of the solution concerning the objective function value.

In [85], A. Schöbel and A. Kratz analyze the so-called price of robustness, i.e. the trade-off between the best possible solution and a robust solution. A bicriteria problem is formulated by adding the trade-off as an additional objective function to the original problem. The bicriteria approach is developed in the context of aperiodic timetabling problems.

For a detailed survey on the major existing robust approaches in the context of Network Flow Problems, see [46].

Supply Chain Problems In the context of supply chain problems, Robust Optimization has been successfully applied. We give some example references here.

In [98], C.-S. Yu and H.-L. Li propose a Robust Optimization model for stochastic logistic problems, inspired by the approach of Mulvey et al. ([72]). Yu and Li focus on linear logistic models with a scenario-based data set. In comparison to the original approach of Mulvey et al., the number of variables and constraints of the robust problem is decreased, resulting in a lower running time of the method of Yu and Li. The computational performance of the model is demonstrated by means of two logistic management examples. However, even though the resulting robust counterpart of a linear problem remains linear when applying the method of Yu and Li, the robust formulation of a network flow problem can no longer be represented as a network flow problem.

In [16], D. Bertsimas and A. Thiele present a Robust Optimization approach for the problem of optimally controlling a supply chain in discrete time. Uncertain demand values in the supply chain without specific distribution are considered. The optimization method is based on the robust optimization framework of Bertsimas and Sim ([14], [15]). The resulting optimization model is a linear programming problem in case that no fixed costs occur along the supply chain and a mixed integer programming problem otherwise. In contrast to the approach presented in this thesis, the supply chain problem is not modeled as a network flow problem.

In [1], E. Adida and G. Perakis examine demand uncertainty in dynamic pricing and inventory control problems. A robust approach based on a demand-based fluid model is presented. The optimization model includes linear and convex functions. As a dynamic problem setting is considered, the coefficients of the input data are time-dependent, such that the optimal pricing and production policy in the fluid model are determined for a time horizon. In contrast to the time-discrete network flow problem representation in this thesis, the model of Adida and Perakis is a time-continuous one.

3.3.3 Stochastic Programming

Another framework for modeling optimization problems with uncertain input data is Stochastic Programming. In contrast to Robust Optimization, Stochastic Programming works with given or estimated probability distributions of the uncertain input data in general. The optimal policy of a Stochastic Programming model is to find a solution that is feasible for all (or almost all) possible data realizations and that minimizes the expectation of a cost function which is dependent on the random variables.

Two-Stage Linear Programming A comprehensively studied model in Stochastic Programming is the two-stage linear program, see [18] and [22], for example. In the first stage, one decision is made for all possible scenarios. In the second stage, after uncertainty is disclosed, a recourse decision is made for each scenario that can compensate bad effects that might result for the specific scenario from the first stage decision.

There exist various methods for solving two-stage linear programs, the so-called L-shaped algorithm being the best known one, see [18], for example. In [65], J. Linderoth and S. Wright propose serial and asynchronous versions of the L-shaped algorithm as well as a trust-region method. In [86], R. Schultz et al. describe a framework using Gröbner basis methods for solving stochastic problems with integer recourse.

Chance Constraint Model Another important model is the so-called chance constraint model, see [58] and [80] for example. In the chance constraint model, constraints need not hold almost surely, but with some probability level. In addition, A. Prékopa examines the maximization of the probability that the constraints are satisfied in [80].

Supply Chain Problems Stochastic Programming has successfully been applied to supply chain problems. We give a small sample of references here.

In [7], K.-P. Arnold examines the planning of transport and storage in the context of distribution planning on the basis of stochastic planning models, where a main focus lies on the applicability of the developed concepts in practice. Arnold considers both static (single time period) and dynamic (multiple time periods) transportation problems. For single time period problems, various Stochastic Programming policies are demonstrated, including both the two-stage method and the chance constraint model. The multiple time period problems are considered under two perspectives: a formulation as a deterministic linear program as well an application of dynamic programming techniques are examined.

In [79], W.B. Powell and H. Topaloglu examine freight transportation problems with uncertain input data, formulated as linear programming problems. In this context, Powell and Topaloglu consider resource allocation problems using concepts from Stochastic Programming and in particular model the freight transportation problem as two-stage linear program, exploiting the underlying network structure. Various different classes of applications are examined and a case study based on the distribution of freight cars for a railroad is carried through.

J. Böttcher examines transportation and transsshipment problems as well as location/transport problems in [22]. The two-stage method as well as the chance-constraint model are regarded. In the context of the two-stage method, Böttcher distinguishes deterministic and stochastic compensation in the second stage. Furthermore, an optimization method similar to the general L-shaped method, called cutting plane method of van Slyke and Wets, is discussed. The practical applicability of the Stochastic Programming methods is demonstrated by means of different example applications.

In [27], P. Dächert gives a survey over Stochastic Programming in the context of Supply Chain Management applications. She applies the two-stage method as well as the chance constraint model to several single-commodity production and transportation planning problems with uncertain demand. The thesis of Dächert covers the problem class of supply chain problems with network representation.

3.3.4 Ant Colony Optimization

ACO Theory The first Ant Colony Optimization (ACO) approach was developed in 1996 by M. Dorigo et al., see [32], known under the name of Ant System (AS). The main application was the Classical Traveling Salesman Problem, though the applicability to other optimization problems was also shown in [32].

Various variants of the ACO metaheuristic were developed in order to improve the performance of the ant algorithm.

The main idea of the Elitist Strategy for Ant System (EAS), see [33], is the reinforcement of the pheromone trails of the best-so-far solution in each iteration.

In the Rank-based Ant System (AS_{rank}), see [33], the best solutions that are found are ranked according to the quality of the solutions. Only the best solutions obtain new pheromone, where the amount of pheromone is multiplied with the rank number of the solution.

In [92], T. Stützle and H. Hoos present a variant called Max-Min Ant System ($MMAS$), where only the iteration-best or best-so-far ant is allowed to drop pheromone in each iteration and where the pheromone levels are bounded below and above by iteration dependent pheromone bounds. Furthermore, the pheromone trails are reinitialized as soon as a stagnation behavior occurs.

Industrial Applications and Supply Chain Problems In recent years, a large variety of ant algorithms and applications to industrial problems, including supply chain management problems, has appeared. As the list of industrial applications of ACO techniques is exhaustive, we can only give a small sample of references here.

K. Doerner et al. examine the full truckload transportation problem in [31], using a hybrid ACO approach. The objective is to minimize the total costs, consisting of fixed costs occurring per vehicle that is brought into use and variable costs depending on the traveled distance. A problem specific heuristic is developed for the initialization of the pheromone trails and for the solution construction. Supply chain stages different from the transportation stage are not considered in the paper of Doerner et al.

The application of a hybrid algorithm called Beam-ACO to supply chain management problems is presented by A. Caldeira et al. in [23]. Beam-ACO fuses Beam-Search and Ant Colony Optimization and was initially presented in the context of job-shop problems (see [21]). The different operational activities within the supply chain are solved using ACO techniques, where the pheromone information is used for information exchange in the supply chain. The algorithm optimizes the supplying, the distributer and the logistic agents of a supply chain.

In [87], C.A. Silva et al. propose a decentralized supply chain management concept based on a multi-agent system, where the corresponding optimization problem is formulated as a set of distributed supply chain problems. As in [23], the pheromone matrix of the ACO metaheuristic is used to achieve information exchange within the supply chain. The coordination of the colony agents of the different supply chain members is realized in a multi-agent system.

In [24], F.T.S. Chan and N. Kumar propose a design technique for supply chain networks based on a multiple ACO technique with main focus on the allocation of customers to distribution centers. The goal is to design an efficient supply chain network which balances the transit time and customers service, which distinguishes the paper of Chan and Kumar and this thesis, where the focus is on the cost-efficient strategic and operational planning of supply chain networks.

Stochastic Optimization Problems Though there exist many applications of the ACO metaheuristic to supply chain problems, most of these models are developed for deterministic optimization problems. However, the use of the Ant Colony Optimization metaheuristic concept for solving stochastic optimization problems has received more attention in recent years.

In [53], T.J. Gutjahr proposes an Ant Colony Optimization based algorithm for solving stochastic combinatorial optimization problems with deterministic constraints. The objective function, which is the expected value of a random variable, is estimated using Monte-Carlo sampling.

In [17], M. Birattari et al. present an algorithm for combinatorial optimization problems under uncertainty, which is based on Ant Colony Optimization and on F-Race. The algorithm is evaluated on the probabilistic Traveling Salesman Problem.

In [8], P. Balaprakash et al. examine Ant Colony Optimization algorithms for the probabilistic Traveling Salesman Problem, where the cost of the solutions constructed by the ants is evaluated by an empirical estimation approach.

In this thesis, we want to apply the ACO to supply chain problems with uncertain input data by the introduction of concepts of Robust Optimization in the ACO metaheuristic.

3.3.5 Heuristics in Supply Chain Management

Apart from Ant Colony Optimization, various other heuristics have successfully been applied to supply chain management problems. We will give a small sample of references here:

General Heuristics In [52], H. Gunnarsson and M. Rönnqvist propose two heuristic approaches to the planning of production and distribution for a pulp company including transportation of raw materials, production mix and contents, inventory management, distribution and customers' orders. The first presented heuristic approach is based on a rolling planning horizon, while the second heuristic approach is based on Lagrangian decomposition and subgradient optimization.

Genetic Algorithm In [81], P. Radhakrishnan et al. examine the determination of the optimal level of inventory at different stages in a supply chain. An approach based on genetic algorithms is presented.

In [73], D. Naso et al. propose a hybrid genetic algorithm approach combined with constructive heuristics for the delivery of ready-mixed concrete. A case study concerning ready-mixed concrete delivery based on industrial data is provided.

Simulated Annealing In [56], V. Jayaraman and A. Ross examine the design and management of distribution networks. The optimal design of the distribution system as well as utilization strategies are generated using a simulated annealing approach.

In [67], S. Afshin Mansouri considers a bicriteria sequencing problem concerning successive stages of a supply chain. The handling of two competing objective functions by means of a multi-objective simulated annealing approach is examined.

3.4 Challenges

In supply chain management, an additional objective for cost efficiency arises: Usually, it is cheaper (per unit) to produce, process and transport large amounts of the same commodity instead of small amounts. We call this objective the avoidance of small amounts of flow. As this additional claim is not regarded by the existing approaches, we will develop a new approach which respects the claim of avoidance of small amounts in Chapter 5.

A second goal is to adapt existing approaches to multicommodity supply chain problems. We will extend the approaches of Bertsimas and Sim as well as the two-stage linear programming approach to multicommodity problems, cf. Chapters 6 and 9.

Another challenge is the inclusion of inventory management strategies into existing approaches. The challenge consists of the insertion of a time component in a so-far time-independent model. We will consider inventory management strategies in the context of Stochastic Programming in Chapter 9.

A drawback of some existing approaches is the size and complexity of the underlying models and resulting optimization problems. Therefore, a fourth goal is the development of models and methods that can handle large supply chain problems with uncertain input data. In Chapters 6 to 9, diverse optimization approaches for this purpose are developed. These approaches will be examined and evaluated on a reference model based on real-world data, which will be developed in Chapter 12.

Chapter 4

Adaption of the ACO Metaheuristic to Network Flow Problems

In this chapter, we give a short description of the ACO metaheuristic. Subsequently, we describe the adaption of the ACO metaheuristic to network flow problems.

4.1 The Ant Colony Optimization Metaheuristic

The Ant Colony Optimization (ACO) metaheuristic is a population-based metaheuristic which was originally applied to combinatorial optimization problems. The metaheuristic is inspired by the natural ants' foraging behavior: Ants are able to find the shortest path to a food source by exploiting pheromone trails which they deposit on the ground while moving.

In ACO, ants traverse a construction graph in order to construct solutions to the combinatorial optimization problem. The decisions, which arcs of the construction graph are chosen by an ant, are based on the pheromone trails of preceding ants and of heuristic information of the arcs. Once the solution construction is finished, the pheromone trails on the arcs are updated.

The ACO metaheuristic is (cf. [33]):

Algorithm 4.1 ACO metaheuristic

```
1: set parameters, initialize pheromone trails
2: SCHEDULE_ACTIVITIES
3:     ConstructAntSolutions
4:     DaemonActions {optional}
5:     UpdatePheromones
6: END_SCHEDULE_ACTIVITIES
```

Here DaemonActions denote actions which can only be executed by a daemon possessing global knowledge and not by single ants, like laying down pheromone on the trails of the iteration best solution, for example.

For details concerning the ACO metaheuristic, see [33].

The ACO metaheuristic has two main features that justify the application to supply chain problems with uncertain input data:

(i) The ACO metaheuristic is random-based. Therefore the introduction of (random based) uncertainty data into the construction graph seems straightforward - see Chapters 6 and 8.

(ii) The ants' behavior of following other ants by following their pheromone trails is unique for the ACO metaheuristic compared to other heuristics like Genetic Algorithms or Simulated Annealing. This behavior turns out to be the key for the fulfillment of a practical requirement in supply chain practice, the avoidance of small amounts of flow - see Chapter 5.

Nevertheless, we naturally do not want to exclude the possibility of modeling the supply chain problems by other heuristic methods hereby.

4.2 ACO for Single-Commodity Flows

In this section, we consider the adaption of the ACO optimization to network flow problems. Let $G = (\mathcal{N}, \mathcal{A})$ be a given single-source single-sink layered network with source node s and sink node t, integral cost values c_{ij} and integral capacity constraints u_{ij} for all arcs $(i, j) \in \mathcal{A}$.

We consider the following minimum cost flow problem for a layered single-source single-sink network G:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} && \\
\text{s.t.} \quad & \sum_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji} = b_i && \forall i \in \mathcal{N} \qquad (4.1)\\
& 0 \le x_{ij} \le u_{ij} && \forall (i,j) \in \mathcal{A}.
\end{aligned}$$

Let the minimum cost flow problem be feasible.

The basic idea for the application of the ACO metaheuristic to the minimum cost flow problem is to replace flow units by ants. For every flow unit of the supply at the source node, we set an ant at this node. Then all ants choose their way through the network until they reach the sink node. The tracks of the ants represent the routing of the flow units as a solution.

We slightly adapt the ACO metaheuristic to our purpose:

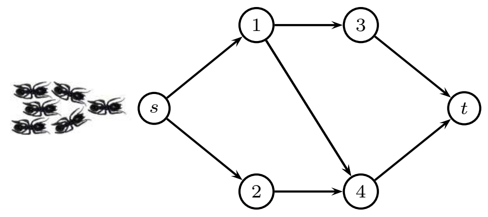
Algorithm 4.2 Adapted ACO metaheuristic

```
1: set parameters
2: initialize pheromone trails
3: while stopping criterion is not fulfilled do
4:     construct ant solution
5:     update pheromone trails
6: end while
```

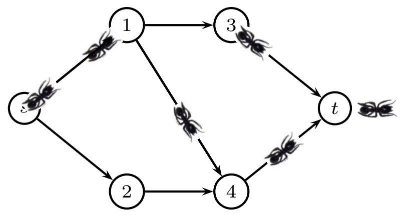
In the following subsections, we describe in detail the adaption of the ACO metaheuristic to the single-commodity minimum cost flow problem. Our definitions are mainly guided by the basic ACO metaheuristic as given in [33]. Additions concerning bounds for the pheromone trails and heuristic information are inspired by the ideas given in [92].

4.2.1 Construction Graph

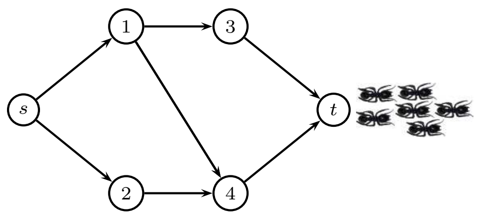
The construction graph is identical to the network flow problem graph G. All ants start their walk at the source node of the given network. At each node they choose one of the outgoing arcs and pass on to the next node. The ants' walk is finished at the sink node.



(a) Ants at source node (start)



(b) The ants' walk



(c) Ants at sink node (finish)

Figure 4.1: Construction graph

4.2.2 Constraints

The walk of the ants is constrained by the arc capacities: For each arc $(i, j) \in \mathcal{A}$ with capacity u_{ij}, no more than u_{ij} ants per iteration are allowed to walk on this arc.

4.2.3 Initialization of the Pheromone Trails

If no additional information is available, a uniform initialization of the pheromone trails is adequate.

A better initialization of the pheromone trails can be obtained by the optimal solution of the original minimum cost flow problem 4.1:

The MCFP can be solved efficiently by a special-purpose network flow algorithm, for example the Cost Scaling Algorithm (CSA). Let x^* be the optimal solution of the original minimum cost flow problem. Then we initialize the pheromone trails τ_{ij} for all arcs $(i, j) \in \mathcal{A}$ as follows:

$$\tau_{ij} := \begin{cases} \tau_{min} & \text{if } \bar{x}^* + \frac{x^*_{ij} - \bar{x}^*}{x^*_{max}} < \tau_{min} \\ \bar{x}^* + \frac{x^*_{ij} - \bar{x}^*}{x^*_{max}} & \text{if } \tau_{min} \leq \bar{x}^* + \frac{x^*_{ij} - \bar{x}^*}{x^*_{max}} \leq \tau_{max} \\ \tau_{max} & \text{if } \tau_{max} < \bar{x}^* + \frac{x^*_{ij} - \bar{x}^*}{x^*_{max}} \end{cases} \tag{4.2}$$

where $\bar{x^*}$ is the average flow over all arcs and x^*_{max} is the maximum flow on one arcs with respect to the optimal solution x^*.

As arcs with pheromone trails equal to zero have a probability of 0 to be chosen by any ant, we impose a lower pheromone bound $\tau_{min} > 0$ for all arcs (i, j), for example $\tau_{min} = 1$. Furthermore, we restrict the pheromone trails by an upper pheromone bound τ_{max}, in order to avoid that particular arcs accumulate too much pheromone and therefore become too attractive.

A second reasonable choice would be the direct initialization of the pheromone trails by the optimal solution of the minimum cost flow problem:

$$\tau_{ij} := \begin{cases} \tau_{min} & \text{if } x^*_{ij} < \tau_{min} \\ x^*_{ij} & \text{if } \tau_{min} \leq x^*_{ij} \leq \tau_{max} \\ \tau_{max} & \text{if } \tau_{max} < x^*_{ij} \end{cases} \tag{4.3}$$

The advantage of the pheromone initialization (4.2) is the fact that the variance of the pheromone levels is smaller than for initialization (4.3), as it is bundled around $\bar{x^*}$. Consequently, the (dis-)favoring of different arcs is not too strong. However, the computation of initialization (4.3) is slightly faster, which can be advantageous in large networks.

4.2.4 Heuristic Information

The heuristic information η_{ij}, also called *visibility*, for arc (i, j) is defined dependent on the cost value c_{ij}.

Let $\bar{c}$ be the arithmetic average cost over all arcs and c_{max} the maximum cost over all arcs. We assume the following condition for each cost value c_{ij}:

$$c_{ij} \leq \bar{c} \cdot c_{max} + 1. \tag{4.4}$$

In practice, this assumption is not restrictive, as for $\bar{c} \geq 1$ the condition is always fulfilled. As c is assumed to be integral, the condition could be violated only for networks where many cost values are less or equal to 0.

Then we define the visibility as

$$\eta_{ij} := \bar{c} - \frac{c_{ij} - \bar{c}}{c_{max}}. \tag{4.5}$$

The visibility is always positive because of the condition (4.4).

If all cost values are positive and the integrality assumption holds, another reasonable choice would be

$$\eta_{ij} := \begin{cases} \dfrac{1}{c_{ij}} & \text{if } c_{ij} > 0 \\ 1 & \text{if } c_{ij} = 0 \end{cases} \tag{4.6}$$

This straightforward choice corresponds to the standard calculation of the heuristic values in the ant algorithm for the Traveling Salesman Problem, see [33].

As for the calculation of the pheromone bounds, it holds that the variance in (4.5) is smaller, while the computation of the heuristic information in (4.6) is slightly faster. Furthermore, for very large cost values, the choice (4.6) would produce values very close to 0, such that in that case (4.5) would be the better choice.

Moreover, a lower visibility bound η_{min} can be imposed in order to avoid that expensive arcs have a too low visibility.

The heuristic information is essential for the generation of cheap solutions. Without the heuristic information, which is based on the arc cost, the ants would generate a solution where as many ants as possible use the same arcs, no matter how expensive these arcs are.

The combination of pheromone information and heuristic information helps the ants to construct a solution which is rather cheap and where many ants use the same arcs.

4.2.5 Solution Construction

Each ant starts its walk at the source node. At each step, the ant chooses one outgoing arc of the current node at random and walks to the end node of the chosen arc. When choosing the next arc, the capacity conditions must be respected: each arc has a limited capacity, i.e. only a certain number of ants may use this arc. If the maximum number was already reached by previously walking ants, the arc cannot be chosen by any ant anymore.

The process of choosing an arc can be implemented as follows: The interval $[0, 1]$ is subdivided into several subintervals according to the number of outgoing arcs and their respective probability. Then a random number in $[0, 1]$ is generated. Dependent on the subinterval, in which the generated random number lies, the respective arc is chosen as next arc.

The capacity constraints on the arcs may lead to a dead end, if for a node the capacity of all outgoing arcs is already fully exhausted. If an ant reaches a dead end, the dead end node is added to a tabu list and the ant starts its walk again at the source node. The ant (and all following ants) are not allowed to choose arcs leading to a node in the tabu list, in order to avoid a repeated choice of dead end nodes.

It might happen that for a specific ant it is not possible to reach the sink node due to the path choice of the preceding ants. This means that for this ant, the source node is finally added to the tabu list and therefore the continuation of the solution construction is no longer possible. In this case, all ants must start over again. As the ants' walk is random based, the ants will finally be able to construct a feasible solution:

Proposition 4.1. *The ants will finally be able to construct a feasible solution.*

Proof. Let $0 < p \leq 1$ be the probability that the ants construct a feasible solution in one try θ. Then the probability for not constructing a feasible solution in try θ is $1 - p$.

We can calculate the probability P for finding at least once a feasible solution in the first θ tries:

$$P(a\, feasible\, solution\, is\, found\, until\, try\, \theta) = 1 - (1-p)^{\theta} \tag{4.7}$$

Therefore, we have in the limit:

$$\begin{aligned} P(a\, feasible\, solution\, is\, finally\, found) &= \lim_{\theta \to \infty} 1 - (1-p)^{\theta} \\ &= 1 - \lim_{\theta \to \infty} (1-p)^{\theta} \\ &= 1 - 0 \\ &= 1 \end{aligned} \tag{4.8}$$

□

The walk of one ant is finished when the ant reaches the sink node.

As the capacity conditions (maximum number of ants per arc) have to be respected, the ants must walk successively.

Let u_{ij}^R be the remaining capacity of arc (i, j), i.e. the maximum capacity u_{ij} minus the number of ants that have already used this arc so far in the current iteration.

The calculation of the arc probability p_{ij} for the selection of an arc (i, j) which has a remaining arc capacity u_{ij}^R greater than 0 is based on the values for the arc visibility η_{ij} and the arc pheromone level τ_{ij}:

$$p_{ij} = \frac{\eta_{ij}^{\alpha} \tau_{ij}^{\beta}}{\sum\limits_{\substack{k:(i,k)\in\mathcal{A} \\ u_{ik}^R > 0}} \eta_{ik}^{\alpha} \tau_{ik}^{\beta}} \tag{4.9}$$

Here $\alpha \geq 0$ and $\beta \geq 0$ are two parameters that determine the relative influence of the pheromone trails and the visibilities. If $\alpha = 0$, only the pheromone trails would be taken into account without any heuristic bias. If $\beta = 0$, the arc probability would be influenced by heuristic values only. Reasonable values for α and β often have to be determined experimentally.

Note that in the sum in the denominator of (4.9), we consider only arcs having a positive remaining capacity.

The pseudo-code for the solution construction is given in the following algorithm. The term "current node" is considered as a variable. The value of the variable "current node" corresponds to the actual location of the ant. Let N be the number of ants.

Algorithm 4.3 construct ant solution

```
1:  set remaining capacity for all arcs to original arc capacity
2:  set tabu list to $\emptyset$
3:  for ant $k = 1$ to $N$ do
4:      set current node to source node.
5:      while current node not equal to sink node do
6:          for all outgoing arcs of the current node do
7:              if end node of the arc is on the tabu list then
8:                  set arc probability to 0
9:              else if remaining arc capacity $= 0$ then
10:                 set arc probability to 0
11:             else
12:                 calculate arc probability considering arc visibility and arc
    pheromone level
13:             end if
14:         end for
```

```
15:         **if** arcs with strict positive probability exist **then**
16:             choose next arc $a$
17:             set current node $\leftarrow$ end node of arc $a$
18:         **else**
19:             **if** current node is the source node **then**
20:                 reset pheromone trails and restart algorithm (goto (1))
21:             **else**
22:                 add current node to tabu list
23:                 restart ant at source node (goto (4))
24:             **end if**
25:         **end if**
26:     **end while**
27:     lower remaining capacity for all visited arcs by 1
28: **end for**
```

4.2.6 Update of the Pheromone Trails

The pheromone trails must be updated after each iteration such that the pheromone values on arcs visited by more ants are increased and pheromone values on arcs visited by fewer ants are decreased. The pheromone update is processed after each iteration, i.e. when all ants have finished their walk.

First, all pheromone values are decreased by evaporating pheromone on all arcs. For all arcs $(i,j) \in \mathcal{A}$, the pheromone level τ_{ij} is set to

$$\tau_{ij} := (1-\rho)\tau_{ij}. \tag{4.10}$$

where $\rho \in [0,1]$. ρ is called *evaporation factor.*

After evaporation, the pheromone values on all arcs must be increased proportional to the number of ants which chose that arc in the current iteration. That means that arcs not being chosen by any ant receive no new pheromone. We increase the pheromone level τ_{ij} on arc (i,j) according to the following rule:

$$\tau_{ij} := \tau_{ij} + \sum_{k=1}^{N} \triangle\tau_{ij}^{k} \tag{4.11}$$

where $\triangle\tau_{ij}^{k}$ is the amount of new pheromone on arc $(i,j) \in \mathcal{A}$ laid down by ant k. One possible choice for $\triangle\tau_{ij}^{k}$ would be

$$\triangle\tau_{ij}^{k} := \begin{cases} 1 & \text{if arc } (i,j) \text{ was chosen by ant } k \\ 0 & \text{else} \end{cases} \tag{4.12}$$

However, this choice does not respect the total cost of the constructed solution in the amount of pheromone laid down. Here the term "constructed solution" denotes the combination of the chosen paths of all ants in the current iteration, not only the current path of one single ant. In order to comprise the total cost of the current solution in the update of the pheromone trails in iteration l, we introduce a factor ω_l which is multiplied to $\triangle\tau_{ij}^{k}$:

$$\omega_l = \begin{cases} 1 & \text{in iteration } l = 1 \\ \frac{\text{total cost of the best-so-far solution}}{\text{total cost of the current solution}} & \text{in iteration } l \geq 2 \end{cases} \tag{4.13}$$

Because of the pheromone evaporation, the pheromone levels for arcs which are not chosen by any ant would converge to zero and therefore would be very unlikely to be chosen in later iterations. To avoid this exclusion of arcs, we respect the lower pheromone bound τ_{min} on all arcs (i,j). Furthermore we respect the upper pheromone bound τ_{max} to prevent all arcs from accumulating too much pheromone.

Let $q_f^l(x)$ be the function of the pheromone update for a solution x in iteration l:

$$q_f^l(x) := (1-\rho)\tau_{ij} + \omega_l \sum_{k=1}^{N} \triangle\tau_{ij}^{k} \tag{4.14}$$

Then in iteration l the pheromone on arc (i,j) is updated as follows:

$$\tau_{ij}^{l} := \begin{cases} \tau_{min} & \text{if } q_f^l(x) < \tau_{min} \\ (1-\rho)\tau_{ij} + \omega_l \sum_{k=1}^{N} \triangle\tau_{ij}^{k} & \text{if } \tau_{min} \leq q_f^l(x) \leq \tau_{max} \\ \tau_{max} & \text{if } \tau_{max} < q_f^l(x) \end{cases} \tag{4.15}$$

Alternatively, the pheromone update can be restricted to the cost minimal "best-so-far" solution only, as this was proposed in the Max-Min Ant System ($MMAS$), see [92]. This means that the costs of the currently constructed solution are compared to the costs of the cheapest ever found solution. If the new solution is cheaper, the pheromone update is performed according to the ants' paths in the new solution. If not, the pheromone update is performed based on the ants' paths of the best-so-far solution.

Note that for this alternative pheromone update, the cost minimal solution "best-so-far" solution must be saved in each iteration.

4.3 ACO for Multicommodity Flows

The ACO metaheuristic for single-commodity flows can be extended to multi-commodity flows. Let p be the number of commodities.

Let b^k denote the supply at the source node for commodity k, $k = 1, \dots, p$. Then the total number of ants at the source node is $\sum\limits_{k=1}^{p} b^k$. Each ant is firmly associated to one commodity, i.e. we have b^k ants for commodity k, $k = 1, \dots, p$.

Construction Graph The construction graph is equal to the construction graph of the single-commodity problem.

Constraints On the arcs of the construction graph, we now have two different capacity constraints for each ant. The capacity constraints as well as the bundle constraints must be respected. Let k be the commodity associated with the currently considered ant a. Then ant a can only use arc (i, j) if no more than $u_{ij}^k - 1$ ants which are associated with commodity k have used arc (i, j) so far and if no more than $\kappa_{ij} - 1$ ants have used arc (i, j) so far, all in all.

Heuristic Information As we have different arc costs for different commodities (and therefore for different ants), the heuristic information has to be defined dependent on the commodities. The visibilities are defined as in the single-commodity case using c_{ij}^k as cost value for arc (i, j) and ants corresponding to commodity k.

Pheromone Trails We have no longer one pheromone trail on every arc, but p different pheromone trails. Each ant can only lay down pheromone on the pheromone trail corresponding to the ant's commodity. The pheromone initialization as well as the pheromone update are performed as described for single-commodity flows.

Solution Construction The solution construction is performed as in the single-commodity case, whereas each ant considers only the heuristic and pheromone values corresponding to its associated commodity.

Alternatively, the arc probability in the solution construction could be influenced by both the pheromone of the respective commodity of the ant as well as by the total amount of pheromone over all commodities, for example in a weighted sum. By this means, the number of used arcs is not only reduced per commodity, but over all commodities. However, if there exist arcs that are cheap for some commodities but expensive for other commodities, this can lead to increased total costs.

Succession of the Ants An important issue is the succession of the ants. If the succession of the ants corresponded to the succession of the commodities, i.e. the ants associated to commodity 1 started first, ants associated to commodity 2 second, and so on, the ants of commodities early in the succession would be treated with favor, as early starting ants have a greater range of arcs left, where the common capacity is not exhausted yet.

As we do not want to favor any commodity, we randomize the succession of the ants in each iteration. At the beginning of the ant algorithm, we sort the ants according to an arbitrary criterion and put them into an ordered list. In each iteration, we shuffle this ant list and let then start the ants successively.

4.4 Convergence Theory

When considering optimization with the ACO metaheuristic, the following question arises: On what terms does the ant algorithm find a global optimal solution? As before, we assume that an optimal solution exists.

In this section, we will give an overview regarding convergence theory in ACO, based on [33]. We maintain the notation given by M. Dorigo and T. Stützle in [33], slightly adapted to our version of the ant algorithm.

G	construction graph
E	arcs in the construction graph
$q_f(x)$	function of the pheromone update
M	maximum number of outgoing arcs over all nodes in G
N	number of ants
l	number of layers in the construction graph

Table 4.1: Ant algorithm notation

For the first part of this section, the following assumption (4.4.1) holds:

- $\exists\, \eta_{min} > 0,\, \exists\, \eta_{max} < \infty : \; \eta_{min} \leq \eta_{ij}(\theta) \leq \eta_{max} \; \forall (i,j) \in E, \forall \text{ iterations } \theta$
- $\exists\, \tau_{min} > 0 : \; \tau_{min} \leq \tau_{ij}(\theta) \; \forall (i,j) \in E, \forall \text{ iterations } \theta$
- x^* is an optimal solution

This means that the heuristic data are bounded by fixed upper and lower bounds and that the pheromone levels are bounded below, where the lower bounds have to be greater than 0. In the most frequently applied variants of the ant algorithm, these assumptions can be easily fulfilled.

The following proposition, given by M. Dorigo and T. Stützle in [33], shows that the maximum possible pheromone level is bounded on all arcs.

Proposition 4.2. *For every arc $(i,j) \in E$ holds:*

$$\lim_{\theta \to \infty} \tau_{ij}(\theta) \leq \tau_{max} := \frac{q_f(x^*)}{1-\rho}$$

Furthermore, M. Dorigo and T. Stützle show in [33] that under assumption (4.4.1), the ant algorithm eventually finds an optimal solution:

Theorem 4.3. *Let $P^*(\theta)$ be the probability for the ant algorithm to find an optimal solution at least once in the first θ iterations. For arbitrary $\epsilon > 0$ and sufficiently large θ, we have $P^*(\theta) \geq 1 - \epsilon$ and $\lim_{\theta \to \infty} P^*(\theta) = 1$.*

Proof. Let i be a fixed node in G. For every outgoing arc (i,j), it holds for every ant that

$$p_{ij} \geq \hat{p}_{min} := \frac{\tau_{min}^{\alpha} \eta_{min}^{\beta}}{(M-1)\tau_{max}^{\alpha} \eta_{max}^{\beta} + \tau_{min}^{\alpha} \eta_{min}^{\beta}} > 0. \tag{4.16}$$

This estimation holds because the ant has at most M possibilities for its decision, where in the worst case arc (i, j) has pheromone level τ_{min} and visibility η_{min} and all other arcs (at most $M - 1$) have pheromone level τ_{max} and visibility η_{max}. It holds that $\hat{p}_{min} > 0$, because τ_{min}, τ_{max}, η_{min} and η_{max} are greater than 0.

Hence, every arc (i, j) is chosen with a probability greater or equal to $\hat{p}_{min}$. Therefore every solution, consisting of l arcs per ant, having a total number of N ants, can be generated in each iteration with a probability $\hat{p} \geq \hat{p}_{min}^{N \cdot l}$. This also holds for the optimal solution. The probability for not finding an optimal solution in the first θ iterations is therefore smaller or equal to $(1 - \hat{p})^{\theta}$. Consequently, the probability $P^*(\theta)$ to find an optimal solution at least once in the first θ iterations is $P^*(\theta) \geq \hat{P}^*(\theta) := 1 - (1 - \hat{p})^{\theta}$.

For a sufficiently large θ and arbitrary $\epsilon > 0$, we have $(1 - \hat{p})^{\theta} \leq \epsilon$ and therefore $1 - (1 - \hat{p})^{\theta} \geq 1 - \epsilon$.

Furthermore, we have $\lim_{\theta \to \infty} (1 - (1 - \hat{p})^{\theta}) = 1$. $\square$

For the remaining part of this section, the following assumptions (4.4.2) hold:

- $\exists\ \eta_{min} > 0, \exists\ \eta_{max} < \infty :\ \eta_{min} \leq \eta_{ij}(\theta) \leq \eta_{max}\ \forall (i, j) \in E, \forall \text{ iterations } \theta$
- $\exists\ \tau_{min}(\theta) > 0 :\ \tau_{min}(\theta) \leq \tau_{ij}(\theta)\ \forall (i, j) \in E, \forall \text{ iterations } \theta$
- $M \geq 2$ (nontrivial problem)

This means that the heuristic data are bounded by fixed upper and lower bounds. The pheromone levels, however, are bounded below by an iteration dependent value.

In addition, we assume that only the best-so-far solution obtains new pheromone, which we proposed as an alternative in Subsection 4.2.6.

The following lemmata, previously given in [99], are needed in order to prove Theorem 4.6 below.

Lemma 4.4. *Let $x \in \mathbb{R}$. Then:*

$$x < 1 \Rightarrow \ln(1 - x) \leq -x$$

Proof. For $x \geq -1$ it holds that

$$\begin{aligned}\exp(x) - (x+1) &= \sum_{\nu=0}^{\infty} \frac{x^\nu}{\nu!} - (\frac{x^1}{1!} + \frac{x^0}{0!}) \\ &= \frac{x^2}{2!}(1+\frac{x}{3}) + \frac{x^4}{4!}(1+\frac{x}{5}) + \dots \qquad (4.17) \\ &\geq 0\end{aligned}$$

With (4.17), it holds that

$$\begin{aligned}1 + x \leq \exp(x)\ \forall x > -1 \ &\Rightarrow\ 1 - x \leq \exp(-x) \qquad \forall x < 1 \\ &\Rightarrow \ln(1-x) \leq -x \qquad \forall x < 1 \qquad (4.18)\end{aligned}$$

□

Lemma 4.5.
$$\sum_{\theta=1}^{\infty} \frac{1}{(\ln(\theta+1))^i} \quad \text{diverges for } i > 0.$$

Proof. Let $i > 0$.

Let $\theta > 3$, $y := \ln\theta$ and $k \in \mathbb{N}_0$ such that $k < i \leq k+1$. Then $y \geq 1$ and it holds:

$$\begin{aligned}\exp(y) &= \sum_{\nu=0}^{\infty} \frac{y^\nu}{\nu!} \\ &= \sum_{\nu=0}^{k} \frac{y^\nu}{\nu!} + \frac{y^{k+1}}{(k+1)!} + \sum_{\nu=k+2}^{\infty} \frac{y^\nu}{\nu!} \\ &\geq \frac{y^{k+1}}{(k+1)!} \qquad (4.19) \\ &\geq \frac{y^i}{(k+1)!}\end{aligned}$$

and therefore

$$\exp(y)(k+1)! \geq y^i \qquad (4.20)$$

and

$$\theta(k+1)! \geq (\ln\theta)^i \tag{4.21}$$

Consequently, we have

$$\frac{1}{(\ln\theta)^i} \geq \frac{1}{\theta(k+1)!} \tag{4.22}$$

As $\sum\limits_{\theta=2}^{\infty} \frac{1}{\theta(k+1)!}$ diverges, we have $\sum\limits_{\theta=2}^{\infty} \frac{1}{(\ln\theta)^i}$ and therefore $\sum\limits_{\theta=1}^{\infty} \frac{1}{(\ln(\theta+1))^i}$ diverges, too. □

M. Dorigo and T. Stützle show in [33] that for suitably chosen $\tau_{min}(\theta)$, the ant algorithm eventually finds an optimal solution:

Theorem 4.6. *$\forall\theta \geq 1$ let $\tau_{min}(\theta) := \frac{d}{\ln(\theta+1)}$, where $d = const.$ and $d^\alpha < \frac{M\cdot\tau_{max}^\alpha\eta_{max}^\beta\cdot(\ln 2)^\alpha}{\eta_{min}^\beta}$. Let $P^*(\theta)$ be the probability, that in the first θ iterations at least one optimal solution is found. Then $\lim\limits_{\theta\to\infty} P^*(\theta) = 1$.*

Proof. Let x^* be an arbitrary optimal solution. Let B_θ be the event, that in iteration θ an optimal solution is found for the first time.

Then $P^*(\theta) = P(\bigcup_{i=1}^{\theta} B_i)$ and $1 - P^*(\theta) = P(\neg\bigcup_{i=1}^{\theta} B_i) = P(\bigcap_{i=1}^{\theta} \neg B_i)$.

Now we consider $P(\bigcap_{i=1}^{\theta} \neg B_i)$, the probability, that an optimal solution is not found within the first θ iterations. We want to show that 0 is an upper bound for this probability.

The event $\bigcap_{i=1}^{\theta} \neg B_i$ implies that x^* is not found within the first θ iterations. Therefore we have:

$$P(\bigcap_{i=1}^{\theta} \neg B_i) \leq P(x^* is\, never\, found). \tag{4.23}$$

As shown in the proof of Theorem 4.3, it can be guaranteed that in iteration θ at node i every feasible decision can be made with a probability p_{min}, where

$$\begin{aligned} p_{min} \geq \hat{p}_{min}(\theta) &:= \frac{\tau_{min}(\theta)^\alpha\eta_{min}^\beta}{\tau_{max}^\alpha\eta_{max}^\beta(M-1) + \tau_{min}(\theta)^\alpha\eta_{min}^\beta} \\ &\geq \frac{\tau_{min}(\theta)^\alpha\eta_{min}^\beta}{M\cdot\tau_{max}^\alpha\eta_{max}^\beta} =: \hat{p}'_{min}(\theta) > 0. \end{aligned} \tag{4.24}$$

Then the probability for the construction of the optimal solution x^* is $\hat{p}_{min}(\theta) \geq \hat{p}'_{min}(\theta)^{N \cdot l}$.

Consequently, the following estimation holds:

$$\begin{aligned} P(x^* is\, never\, found) &\leq \prod_{\theta=1}^{\infty}(1-(\hat{p}'_{min}(\theta))^{N\cdot l}) \\ &= \prod_{\theta=1}^{\infty}\left(1-\left(\frac{\tau_{min}(\theta)^{\alpha}\eta_{min}^{\beta}}{M\cdot\tau_{max}^{\alpha}\eta_{max}^{\beta}}\right)^{N\cdot l}\right) \end{aligned} \tag{4.25}$$

We want to show that the product on the right hand side is equal to 0. We consider the logarithm of the product given above, which is defined because of $0 < \hat{p}'_{min}(\theta) < 1$ for $M \geq 2$.

We have that:

$$\begin{aligned} \ln\left[\prod_{\theta=1}^{\infty}\left(1-\left(\frac{\tau_{min}(\theta)^{\alpha}\eta_{min}^{\beta}}{M\cdot\tau_{max}^{\alpha}\eta_{max}^{\beta}}\right)^{N\cdot l}\right)\right] &= \sum_{\theta=1}^{\infty}\ln\left[1-\left(\frac{\tau_{min}(\theta)^{\alpha}\eta_{min}^{\beta}}{M\cdot\tau_{max}^{\alpha}\eta_{max}^{\beta}}\right)^{N\cdot l}\right] \\ &= \sum_{\theta=1}^{\infty}\ln\left[1-\left(\frac{\left(\frac{d}{\ln(\theta+1)}\right)^{\alpha}\eta_{min}^{\beta}}{M\cdot\tau_{max}^{\alpha}\eta_{max}^{\beta}}\right)^{N\cdot l}\right] \\ &= \sum_{\theta=1}^{\infty}\ln\left[1-\left(\frac{D}{(\ln(\theta+1))^{\alpha}}\right)^{N\cdot l}\right] \\ &\leq \sum_{\theta=1}^{\infty}-\left(\frac{D}{(\ln(\theta+1))^{\alpha}}\right)^{N\cdot l} \\ &= -D^{N\cdot l}\sum_{\theta=1}^{\infty}\frac{1}{(\ln(\theta+1))^{\alpha N l}} \\ &= -\infty \end{aligned} \tag{4.26}$$

with $D = \frac{d^{\alpha}\eta_{min}^{\beta}}{M\cdot\tau_{max}^{\alpha}\eta_{max}^{\beta}} = const.$

It follows that

$$\ln\left[\prod_{\theta=1}^{\infty}\left(1-\left(\frac{\tau_{min}(\theta)^{\alpha}\eta_{min}^{\beta}}{M\cdot\tau_{max}^{\alpha}\eta_{max}^{\beta}}\right)^{N\cdot l}\right)\right] = -\infty \tag{4.27}$$

and

$$\prod_{\theta=1}^{\infty} \left(1 - \left(\frac{\tau_{min}(\theta)^{\alpha} \eta_{min}^{\beta}}{M \cdot \tau_{max}^{\alpha} \eta_{max}^{\beta}} \right)^{N \cdot l} \right) = 0. \tag{4.28}$$

Therefore the probability never to find x^* is equal to 0 and consequently $P(\bigcap_{i=1}^{\infty} \neg B_i) = 0$. Since we have $P^*(\theta) = 1 - P(\bigcap_{i=1}^{\theta} \neg B_i)$, it holds that $\lim_{\theta \to \infty} P^*(\theta) = 1$.

$\square$

Theorem 4.3 shows that the ant algorithm guarantees to find an optimal solution, if the pheromone trails are bounded below by a fixed lower bound. Theorem 4.6 extends this result to a lower bound of the pheromone level which decreases to 0 logarithmically.

As already stated in [33], we have to note that Theorem 4.6 can not be proved for an exponentially fast decrement of the pheromone levels as given by a constant pheromone evaporation rate without additional lower bound. This means that in practice, a fixed or logarithmically decreasing lower pheromone bound must be introduced in order to guarantee that an optimal solution is found. As we introduced a fixed lower bound in the models proposed in Sections 4.2 and 4.3, we ensure that our ant algorithms find an optimal solution.

However, Theorems 4.3 and 4.6 do not make a statement about the necessary time or number of iterations which is required to find an optimal solution. In the worst case, this time can be astronomically large.

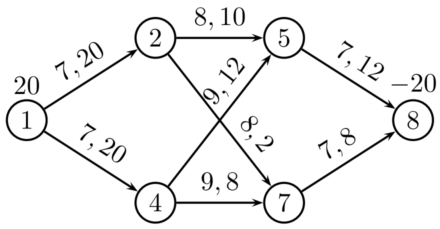
Chapter 5

Avoiding Small Amounts of Flow in Minimum Cost Network Flow Problems

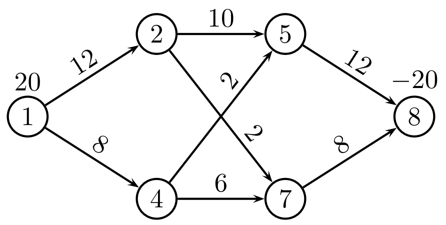
The optimal solutions of minimum cost network flow problems possibly contain many small amounts of flow on diverse arcs. In practice, however, small amounts of flow are often not desired.

Consider for example the production planning process for a given commodity. If the optimal solution contains many different production sites with only a small amount of production output per site, all of these sites must be configured to produce the commodity, but only a small amount of the commodity is produced on each production site. Usually in practice, a solution which contains fewer different production sites would be preferred, even if it is slightly more expensive, because of the savings in configuration expenditure.

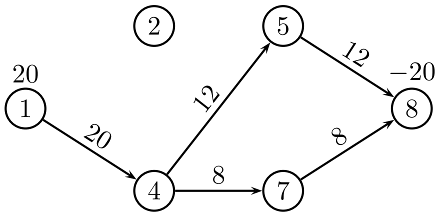
The following figure illustrates the optimal solution to a given minimum cost flow problem and a possibly more desired solution which respects the additional goal of avoiding small amounts of flow.



(a) MCFP network with costs, capacities and supply/demand



(b) Optimal solution: eight arcs used, total cost 448



(c) Desired solution: five arcs used, total cost 460

Figure 5.1: Optimal and desired solution of a minimum cost flow problem

Let $G = (\mathcal{N}, \mathcal{A})$ be a given network with node set $\mathcal{N}$ and arc set $\mathcal{A}$. Let $n = |\mathcal{N}|$ be the number of nodes and $m = |\mathcal{A}|$ the number of arcs. Let $c \in (\mathbb{R}_0^+)^m$ be the corresponding cost vector, $u \in (\mathbb{R}^+)^m$ the capacity vector, $A \in \mathbb{R}^{n \times m}$ the node-arc incidence matrix and $b \in \mathbb{R}^n$ the vector of supply and demand.

As the notation "small flow" is rather vague, we specify a threshold value θ_i for every arc $i \in \mathcal{A}$. Then a "small flow" on arc i is a flow greater than zero and smaller than θ_i. Hence we want to avoid positive flows which do not exceed the threshold values, if this is possible.

In the following, we call the minimum cost flow problem with the additional goal of surpassing the threshold values (if possible) the *threshold minimum cost flow problem* (TMCFP).

There are different approaches to solve this problem. The introduction of lower bounds for the flows on the arcs is not leading to the desired model as the possibility of sending no flow on these arcs is excluded. Another possibility to model this problem is through the definition of nonlinear cost functions. The drawback of this approach clearly is the fact that the LP structure of the problem is destroyed.

In this chapter, we consider both a heuristic approach and an exact approach to deal with this problem. In the heuristic approach, we apply the Ant Colony Optimization metaheuristic to solve the given problem. In the exact approach, we define piecewise linear cost functions on the arcs. In order to exploit the linear structure of the problem, we formulate the optimization problem using so-called SOS 2 conditions, which will be introduced in Section 5.2. This approach requires the introduction of integer variables. Finally, we combine the heuristic and the exact approach in a special-purpose Branch and Bound algorithm for this problem.

5.1 Application of the Ant Colony Optimization Metaheuristic to the TMCFP

In order to apply the ACO metaheuristic for minimum cost flow problems, which we proposed in Chapter 4, to the threshold minimum cost flow problem, we must add some specifications and extensions.

For simplification, we initially assume that the threshold value θ_i is equal for all arcs $e_i \in \mathcal{A}$. Let θ be this threshold value.

5.1.1 Parameters

For solving the minimum cost flow problem with the additional goal of avoiding small amounts of flow, the number s of ants must be equal to the supply at the source node.

A suitable setting for the parameters α and β, which appear in the calculation of the arc probabilities, see Subsection 4.2.5, is chosen experimentally. In our test runs, the choice $\alpha = 1$ and $\beta = 3$ turned out to be reasonable.

The pheromone evaporation factor ρ was also determined experimentally. $\rho = 0.25$ seemed to be a reasonable choice.

5.1.2 Penalty Costs

One possibility to enforce the influence of the threshold value θ even more is the introduction of penalty costs.

When calculating the total cost of the current solution, we penalize arcs where the amount of flow, i.e. the amount of ants, are greater than zero but less than the threshold value. Therefore we introduce a penalization cost value c^{max}, which is greater or equal to the maximum possible cost that can arise on any arc (i, j) in the network (arc cost times flow value). If the amount of ants are greater than zero and less than the threshold value, we set the cost for using this arc to c^{max}.

As the costs of a penalized arc are constant within the interval $]0, \theta[$, the total cost will not rise if more ants use this arc and will even be reduced if the amount of ants surpasses the threshold value.

If different threshold values θ_i for arcs $e_i \in \mathcal{A}$ are given, the penalty costs can be modeled analogously.

5.1.3 Stopping Criterion

Possible stopping criteria are a maximum number of iterations or stagnation in the solution construction, i.e. when finally the same solution is constructed by the ants in each iteration.

5.1.4 Variation

The main idea of the variation is the following: When ant a has chosen its path p_a through the network, we want to exploit the maximal capacity for this path.

First, we determine the maximal remaining capacity u for the chosen path. Here we consider the remaining capacity u before ant a has walked over the path p_a. Let u be the minimum of all remaining arc capacity values for all arcs included in path p_a. After ant a has finished its walk, we manipulate the next $u - 1$ ants such that they choose the same path p_a as ant a.

In the test runs, this variation turned out to produce similar results as the original algorithm, but was significantly faster in finding good solutions.

5.2 An Exact Approach using SOS 2 Conditions

In this section, we present an exact approach to the threshold minimum cost flow problem using so called SOS 2 conditions.

5.2.1 Approximation of Nonlinear Functions using SOS 2 Conditions

In this subsection, we introduce the so called Special Ordered Set of Type 2 condition in the context of the approximation of nonlinear functions by piecewise linear ones as proposed in [71]. We consider a nonlinear function of the form $f : [a, b] \to \mathbb{R}$.

In order to obtain grid points $x^1, \dots, x^k$ with corresponding function values $f(x^1), \dots, f(x^k)$, the interval $[a, b]$ is divided such that $a = x^1 \leq x^2 \leq \cdots \leq x^k = b$. Using the grid points, the function can be approximated by a piecewise linear function.

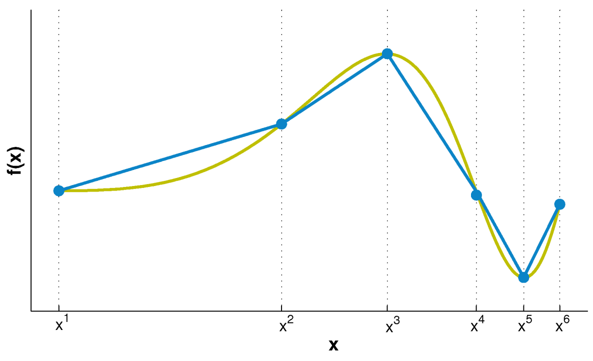


Figure 5.2: Piecewise linear approximation of a nonlinear function

In order to approximate the function f by a piecewise linear function, we introduce new variables λ^j for every grid point $(x^j, f(x^j))$. The goal is to approximate the function value $f(x)$ for arbitrary $x \in [a, b]$ by the following convex combination:

$$x = \sum_{j=1}^{k} x^j \lambda^j \tag{5.1}$$

$$f(x) \approx \sum_{j=1}^{k} f(x^j) \lambda^j \tag{5.2}$$

$$\sum_{j=1}^{k} \lambda^j = 1 \tag{5.3}$$

$$\lambda^j \geq 0 \qquad j = 1, \dots, k \tag{5.4}$$

$$\lambda \in \text{SOS } 2 \tag{5.5}$$

The last condition $\lambda \in$ SOS 2 has the following meaning: At most two entries of $\lambda = (\lambda^1, \dots, \lambda^k)$ are positive, and if two are positive they must be adjacent. This condition is called the *Special Ordered Set of Type 2 condition*, briefly *SOS 2 condition*.

This condition can be modeled implicitly by incorporating it into the branching phase of a Branch and Bound algorithm, see [9] and [10], for example.

Alternatively, the SOS 2 condition can be modeled explicitly via binary variables. This method is called the *lambda method* or the *convex combination method*, see [71].

In addition, there exists an alternative binary approach called the *delta method* or the *incremental method*, see [71].

In recent years, new approaches concerning SOS2 and MIP formulations were developed:

In [94], J.P. Vielma et al. analyse several MIP formulations for mixed integer programs with piecewise linear continuous cost functions with special focus on non-convex piecewise fcuntions. Theoretical properties and relative computational performance are compared. Moreover, an extension to piecewise linear lower semicontinuous functions is presented.

In [95], J.P. Vielma et al. present an SOS2 based formulation for lower semicontinuous functions. Their approach extends a branch-and-cut algorithm for linear programs with piecewise linear continuous cost functions. A computational analysis using the CPLEX solver is carried through.

The Lambda Method

The lambda method models the SOS 2 condition explicitly by introducing new binary variables $y^j,\ j = 1, \ldots, k-1$ corresponding to the $k-1$ segments of $[a, b]$, given by the grid points. The additional constraint

$$\sum_{j=1}^{k-1} y^j = 1$$

ensures that exactly one segment of $[a, b]$ is selected. By adding the constraints

$$\begin{aligned}
&\lambda^1 \leq y^1 \\
&\lambda^j \leq y^{j-1} + y^j \quad j = 2, \ldots, k-1 \\
&\lambda^k \leq y^{k-1}
\end{aligned}$$

we ensure that only the λ-variables which are adjacent to the chosen segment can be positive.

Hence, the piecewise linear approximation of $f(x)$ can be described by the following mixed integer linear formulation:

$$x = \sum_{j=1}^{k} x^j \lambda^j \tag{5.6}$$

$$f(x) \approx \sum_{j=1}^{k} f(x^j)\lambda^j \tag{5.7}$$

$$\sum_{j=1}^{k} \lambda^j = 1 \tag{5.8}$$

$$\sum_{j=1}^{k-1} y^j = 1 \tag{5.9}$$

$$\lambda^1 \leq y^1 \tag{5.10}$$

$$\lambda^j \leq y^{j-1} + y^j \quad j = 2, \ldots, k-1 \tag{5.11}$$

$$\lambda^k \leq y^{k-1} \tag{5.12}$$

$$\lambda^j \geq 0 \qquad j = 1, \ldots, k \tag{5.13}$$

$$y^j \in \{0, 1\} \qquad j = 1, \ldots, k-1 \tag{5.14}$$

The Delta Method

The so-called *delta method* describes the linear interpolation of the grid points by using continuous variables $\delta^j,\ j = 1, \ldots, k-1$, corresponding to the subintervals $[x^j, x^{j+1}]$ of $[a, b]$ and binary variables $w^j,\ j = 1, \ldots, k-2$. Then the piecewise linear approximation of $f(x)$ is given by

$$x = a + \sum_{j=1}^{k-1} (x^{j+1} - x^j) \cdot \delta^j \tag{5.15}$$

$$f(x) \approx f(a) + \sum_{j=1}^{k-1} (f(x^{j+1}) - f(x^j)) \cdot \delta^j \tag{5.16}$$

$$\delta \text{ fulfills the filling condition} \tag{5.17}$$

The filling condition for the δ-variables is defined as follows:

$$\text{If } \delta^j > 0 \text{ with } 2 \leq j \leq k-1 \text{ then } \delta^l = 1 \text{ for } 1 \leq l < j. \tag{5.18}$$

The filling condition guarantees that if interval $[x^j, x^{j+1}]$, $2 \leq j \leq k-1$, is chosen, i.e. $\delta^j > 0$, then all intervals $[x^l, x^{l+1}]$ to its left must also be used completely, i.e. $\delta^l = 1,\ 1 \leq l < j$.

The filling condition can be modeled explicitly by the introduction of binary variables $w^j,\ j = 1, \ldots, k-2$ and additional constraints:

$$\begin{aligned}
\delta^{j+1} &\leq w^j && j = 1, \ldots, k-2 && (5.19)\\
w^j &\leq \delta^j && j = 1, \ldots, k-2 && (5.20)\\
0 \leq \delta^j &\leq 1 && j = 1, \ldots, k-1 && (5.21)\\
w^j &\in \{0, 1\} && j = 1, \ldots, k-2 && (5.22)
\end{aligned}$$

Then the conditions (5.19) and (5.20) ensure the filling condition for the δ-variables.

All in all, the piecewise linear approximation of $f(x)$ can be described by the following mixed integer linear formulation:

$$\begin{aligned}
x &= a + \sum_{j=1}^{k-1} (x^{j+1} - x^j) \cdot \delta^j\\
f(x) &\approx f(a) + \sum_{j=1}^{k-1} (f(x^{j+1}) - f(x^j)) \cdot \delta^j\\
\delta^{j+1} &\leq w^j && j = 1, \ldots, k-2 && (5.23)\\
w^j &\leq \delta^j && j = 1, \ldots, k-2\\
0 \leq \delta^j &\leq 1 && j = 1, \ldots, k-1\\
w^j &\in \{0, 1\} && j = 1, \ldots, k-2
\end{aligned}$$

In [77], it is shown that the bounds produced by the linear programming relaxation of the lambda method are always worse than the bounds for the relaxation of the delta method. Therefore, in practice, the delta method is computationally superior to the lambda method.

5.2.2 TMCFP with SOS 2 Conditions

Now we consider the threshold minimum cost flow problem

$$\begin{array}{ll} \min & c^\top x \\ \text{s.t.} & Ax = b \\ & 0 \leq x \leq u \end{array} \tag{5.24}$$

where $x \in \mathbb{R}^m$, $c \in (\mathbb{R}_0^+)^m$, $u \in (\mathbb{R}^+)^m$, $A \in \mathbb{R}^{n\times m}$ and $b \in \mathbb{R}^n$.

Let c_i be the cost given for $i \in \mathcal{A}$, $u^{max} = \max_i u_i$ and $c^{max} = u^{max} \cdot \max_i c_i$. Let θ_i be the threshold value for arc i, i.e. the minimum amount of flow that is desired on arc e_i if the flow is not zero.

Now we consider each arc e_i, $i = 1, \dots, m$, separately.

In order to respect the threshold value for arc e_i, we would choose a discontinuous cost function. The following figure visualizes the desired cost function:

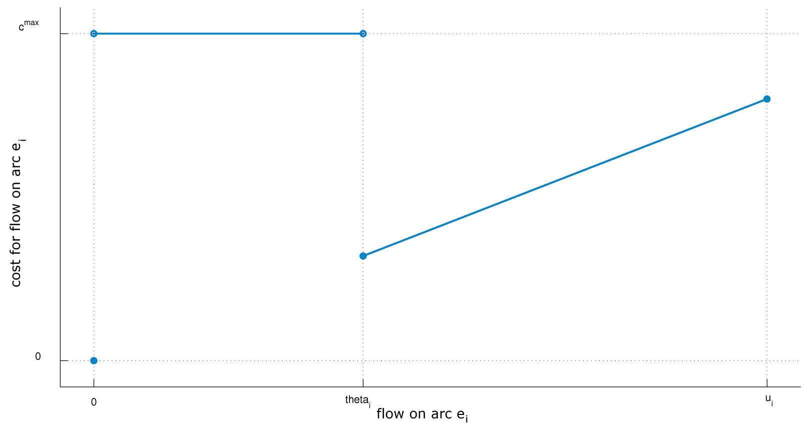


Figure 5.3: Desired cost function for arc e_i

For $x_i = 0$ or $\theta_i \leq x_i \leq u_i$ the cost is identical to the cost of the original minimum cost flow problem.

For $0 \leq x_i \leq \theta_i$ the cost is constant with a cost value of c^{max}, where c^{max} is greater or equal to the maximum possible cost that can arise on any arc $e_i \in \mathcal{A}$ in the network. This choice is due to the following consideration: Any feasible solution with $x_i = 0$ or $\theta_i \leq x_i \leq u_i$ for all arcs $e_i \in \mathcal{A}$, if one exists, is then less expensive than a solution with $0 < x_i < \theta_i$. If no such solution exists, i.e. there has to be at least one arc e_i with $0 < x_i < \theta_i$, the number of those arcs will be as small as possible, as the cost value c^{max} arises for all used arcs e_i with $0 < x_i < \theta_i$.

Now we want to approximate the discontinuous cost function by a piecewise linear cost function.

First we define grid points as described in Subsection 5.2.1. For every $i = 1, \dots, m$ we choose the x-coordinates $x_i^1, ..., x_i^5$ of the grid points such that

$$\begin{aligned} x_i^1 &= x_i^2 = 0 \\ x_i^3 &= x_i^4 = \theta_i \\ x_i^5 &= u_i. \end{aligned}$$

If $u_i < \theta_i$, we choose the x-coordinates $x_i^1, ..., x_i^5$ of the grid points such that

$$\begin{aligned} x_i^1 &= x_i^2 = 0 \\ x_i^3 &= x_i^4 = x_i^5 = u_i. \end{aligned}$$

The case $u_i < \theta_i$ is possible for example if the same threshold value shall be chosen for every arc in the network. Note that for $u_i < \theta_i$, the grid points x_i^4 and x_i^5 are not necessary and could be dropped in order to reduce the number of optimization variables and conditions. For uniformity of notation, however, we will keep five grid points per arc in the following considerations.

The y-coordinates of the five grid points are given by the following assignment:

$$\begin{aligned} c(x_i^1) &= 0 \\ c(x_i^2) &= c(x_i^3) = c^{max} \\ c(x_i^4) &= c_i x_i^4 \\ c(x_i^5) &= c_i x_i^5 \end{aligned}$$

if $\theta_i \leq u_i$, and

$$\begin{aligned} c(x_i^1) &= 0 \\ c(x_i^2) &= c(x_i^3) = c(x_i^4) = c(x_i^5) = c^{max} \end{aligned}$$

if $u_i < \theta_i$. Here c_i denotes the i-th component of the cost vector c of the minimum cost flow problem.

The following figure visualizes the cost assignment for an arc e_i where $\theta_i < u_i$.

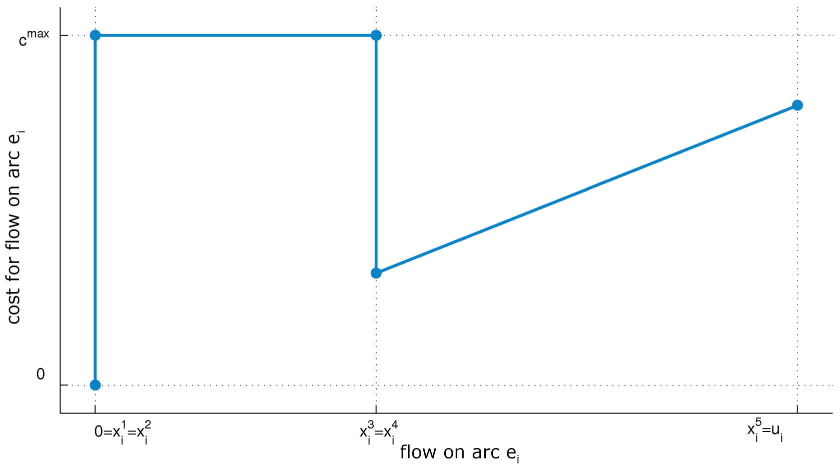


Figure 5.4: Cost assignment for arc e_i

Note that if the range of the costs varies considerably for different arcs $e_i \in \mathcal{A}$, it is possible to choose the value c^{max} differently than described above or even to choose different values c_i^{max} for different arcs $e_i \in \mathcal{A}$.

Furthermore it is possible to introduce threshold values θ_i only for some arcs $e_i \in \mathcal{A}$. For example, in production and transportation networks, the threshold values could only be introduced for the nodes representing the production process. Then the increase of the size of the resulting optimization problem is reduced.

For the rest of this chapter, we assume that we have threshold values on all arcs, that the cost values on all arcs lie within a small range and that $\theta_i < u_i$ for all arcs.

Application of the Lambda Method

For every arc e_i, $i = 1, \ldots, m$, we introduce the λ-variables λ_i^j, $j = 1, \ldots, 5$, $0 \leq \lambda_i^j \leq 1$, corresponding to the grid points x_i^j. Assuming that for every i, $i = 1, \ldots, m$, the SOS 2 condition holds for the corresponding λ-variables $\lambda_i^j, j = 1, \ldots, 5$, $x_i \in [0, u_i]$ can be represented as

$$x_i = \sum_{j=1}^{5} \lambda_i^j x_i^j.$$

Note that for $x_i = x_i^1 = x_i^2$ or $x_i = x_i^3 = x_i^4$ this representation is not unique. For $x_i \neq x_i^j$, $j = 1, \ldots, 4$, the representation is unique because of the SOS 2 condition.

Then the new overall cost function is

$$\begin{aligned} c(x) &:= \sum_{i=1}^{m} c(x_i) \\ &= \sum_{i=1}^{m} c(\sum_{j=1}^{5} \lambda_i^j x_i^j) \\ &= \sum_{i=1}^{m} \sum_{j=1}^{5} c(x_i^j) \lambda_i^j \\ &=: \hat{c}^\top \lambda =: \hat{c}(\lambda) \end{aligned}$$

where

$$\begin{aligned}
\lambda &:= (\lambda_1^1, \dots, \lambda_m^1, \dots, \lambda_1^5, \dots, \lambda_m^5)^\top \\
\hat{c} &:= (c(x_1^1), \dots, c(x_m^1), \dots, c(x_1^5), \dots, c(x_m^5))^\top .
\end{aligned}$$

Obviously $\hat{c}(\lambda)$ is a well defined linear function depending on λ.

Additionally, the constraints $Ax = b$ of the minimum cost flow problem must be transformed to constraints dependent on λ instead of x.

We have

$$\begin{aligned}
& & Ax &= b \\
&\Leftrightarrow & A_k x &= b_k \quad k = 1, \dots, n \\
&\Leftrightarrow & \sum_{i=1}^{m} A_{ki} x_i &= b_k \quad k = 1, \dots, n \\
&\Leftrightarrow & \sum_{i=1}^{m} A_{ki} \sum_{j=1}^{5} x_i^j \lambda_i^j &= b_k \quad k = 1, \dots, n \\
&\Leftrightarrow: & \hat{A}\lambda &= b
\end{aligned}$$

where

$$\hat{A} = (A_{k1}x_1^1, \dots, A_{km}x_m^1, \dots, A_{k1}x_1^5, \dots, A_{km}x_m^5)_{k=1,\dots,n}$$

and λ as given above.

Now we replace the optimization variables x by the new continuous variables λ. In order to satisfy the SOS 2 conditions, we insert the binary variables y and the corresponding constraints.

The resulting mixed-integer problem is of the following structure:

$$\begin{array}{lll} \min & \sum_{i=1}^{m}\sum_{j=1}^{5} c(x_i^j)\lambda_i^j & \\ \text{s.t.} & \sum_{i=1}^{m} A_{ki}\sum_{j=1}^{5} x_i^j\lambda_i^j = b_k & k=1,\dots,n \\ & 0 \le \sum_{j=1}^{5} x_i^j\lambda_i^j \le u_i & i=1,\dots,m \qquad\qquad (**) \\ & \sum_{j=1}^{5}\lambda_i^j = 1 & i=1,\dots,m \\ & \sum_{j=1}^{4} y_i^j = 1 & i=1,\dots,m \\ & \lambda_i^1 \le y_i^1 & i=1,\dots,m \\ & \lambda_i^j \le y_i^{j-1}+y_i^j & i=1,\dots,m,\ j=2,3,4 \\ & \lambda_i^5 \le y_i^4 & i=1,\dots,m \\ & \lambda_i^j \ge 0 & i=1,\dots,m,\ j=1,\dots,5 \\ & y_i^j \in \{0,1\} & i=1,\dots,m,\ j=1,\dots,4 \end{array} \tag{5.25}$$

Because of

$$0 \le \lambda_i^j \le 1,\ i=1,\dots,m,\ j=1,\dots,5,$$

$$x_i^j \ge 0,\ i=1,\dots,m,\ j=1,\dots,5,$$

$$x_i^j \le x_i^5,\ i=1,\dots,m,\ j=1,\dots,5$$

and

$$\sum_{j=1}^{5}\lambda_i^j = 1,\ i=1,\dots,5$$

we have

$$0 \le \sum_{j=1}^{5} x_i^j\lambda_i^j \le x_i^5\sum_{j=1}^{5}\lambda_i^j = x_i^5 = u_i,\ i=1,\dots,m$$

such that the constraint $(**)$ is redundant.

In order to write the optimization problem in an extended standard form containing equality and inequality constraints, we rearrange the optimization problem such that the inequality constraints take the form $A_{\le}(\lambda,y) \le b_{\le}$ and the equality constraints take the form $A_{=}(\lambda,y) = b_{=}$, where $\lambda := (\lambda_1^1,\dots,\lambda_m^1,\dots,\lambda_1^5,\dots,\lambda_m^5)$ and $y := (y_1^1,\dots,y_m^1,\dots,y_1^4,\dots,y_m^4)$.

Then the matrix $A_{\le}$ is of the following form:

I_m					$-I_m$			
	I_m				$-I_m$	$-I_m$		
		I_m				$-I_m$	$-I_m$	
			I_m				$-I_m$	$-I_m$
				I_m				$-I_m$

Figure 5.5: Structure of matrix $A_{\leq}$ of the inequality constraints (λ-method)

Here I_m denotes the identity matrix of dimension m. Empty rectangles denote zero-filled $m \times m$-matrices.

The vector $b_{\leq}$ is a zero vector.

The matrix $A_{=}$ is of the following form:

		$*$	$*$	$*$				
I_m	I_m	I_m	I_m	I_m				
					I_m	I_m	I_m	I_m

Figure 5.6: Structure of matrix $A_{=}$ of the equality constraints (λ-method)

Rectangles filled with a $*$ denote matrices of size $n \times m$ which have non-zero entries on the positions corresponding to the 1s and -1s in the incidence matrix A of the graph of the given minimum cost flow problem. I_m denotes the identity matrix of dimension m. Empty rectangles denote zero-filled $n \times m$-matrices and zero-filled $m \times m$-matrices, respectively.

The vector $b_{=}$ is $(b, 1, \ldots, 1, 1, \ldots, 1)^{\top}$.

The optimization problem has $5m + 4m = 9m$ optimization variables and $5m + 3m = 8m$ constraints.

Application of the Delta Method

For every arc e_i, $i = 1, \ldots, m$, we introduce the δ-variables δ_i^j, $j = 1, \ldots, 4$, $0 \leq \delta_i^j \leq 1$, corresponding to the subintervals $[x_i^j, x_i^{j+1}]$ of $[0, u_i]$. Assuming that the filling condition holds, $x_i \in [0, u_i]$ can be represented as

$$x_i = \sum_{j=1}^{4} (x_i^{j+1} - x_i^j) \cdot \delta_i^j .$$

Note that for $x_i = x_i^1 = x_i^2$ or $x_i = x_i^3 = x_i^4$ this representation is not unique. For $x_i \neq x_i^j$, $j = 1, \ldots, 4$, the representation is unique because of the filling condition.

Now we consider the new overall cost function

$$\begin{aligned}
c(x) &:= \sum_{i=1}^{m} c(x_i) \\
&= \sum_{i=1}^{m} c(\sum_{j=1}^{4}(x_i^{j+1} - x_i^j) \cdot \delta_i^j) \\
&= \sum_{i=1}^{m}\sum_{j=1}^{4}(c(x_i^{j+1}) - c(x_i^j)) \cdot \delta_i^j \\
&=: \hat{c}^\top \delta =: \hat{c}(\delta)
\end{aligned}$$

where

$$\begin{aligned}
&\delta := (\delta_1^1, \dots, \delta_m^1, \dots, \delta_1^4, \dots, \delta_m^4)^\top \\
&\hat{c} := (c(x_1^2) - c(x_1^1), \dots, c(x_m^2) - c(x_m^1), \dots, c(x_1^5) - c(x_1^4), \dots, c(x_m^5) - c(x_m^4))^\top.
\end{aligned}$$

We can see that $\hat{c}(\delta)$ is a well defined linear function depending on δ.

In addition to the transformation of the cost function, we must also transform the linear constraints $Ax = b$ of the minimum cost flow problem to constraints dependent on δ instead of x.

We have

$$\begin{aligned}
& & Ax = b & \\
&\Leftrightarrow & A_k x = b_k & \quad k = 1, \dots, n \\
&\Leftrightarrow & \sum_{i=1}^{m} A_{ki} x_i = b_k & \quad k = 1, \dots, n \\
&\Leftrightarrow & \sum_{i=1}^{m} A_{ki} \sum_{j=1}^{4}(x_i^{j+1} - x_i^j) \cdot \delta_i^j = b_k & \quad k = 1, \dots, n \\
&\Leftrightarrow: & \hat{A}\delta = b &
\end{aligned}$$

where

$$\hat{A} = (A_{k1}(x_1^2 - x_1^1), \dots, A_{km}(x_m^2 - x_m^1), \dots, A_{k1}(x_1^5 - x_1^4), \dots, A_{km}(x_m^5 - x_m^4))_{k=1,\dots,n}$$

and δ as given above.

Consequently, we can replace the optimization variables x by the new continuous variables δ. Additionally we insert the binary variables w and the corresponding constraints in order to satisfy the filling condition.

The resulting mixed-integer problem is of the following structure:

$$\begin{array}{lll l}
\min & \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{4}(c(x_i^{j+1})-c(x_i^j))\cdot\delta_i^j & & \\
\text{s.t.} & \sum\limits_{i=1}^{m}A_{ki}\sum\limits_{j=1}^{4}(x_i^{j+1}-x_i^j)\cdot\delta_i^j = b_k & k=1,\dots,n & \\
 & 0\le\sum\limits_{j=1}^{4}(x_i^{j+1}-x_i^j)\cdot\delta_i^j\le u_i & i=1,\dots,m & (**) \\
 & \delta_i^{j+1}\le w_i^j & i=1,\dots,m,\ j=1,2,3 & \\
 & w_i^j\le\delta_i^j & i=1,\dots,m,\ j=1,2,3 & \\
 & 0\le\delta_i^j\le 1 & i=1,\dots,m,\ j=1,\dots,4 & \\
 & w_i^j\in\{0,1\} & i=1,\dots,m,\ j=1,2,3 &
\end{array} \tag{5.26}$$

Because of

$$0\le\delta_i^j\le 1,\ i=1,\dots,m,\ j=1,\dots,4$$

and

$$x_i^j\ge 0,\ i=1,\dots,m,\ j=1,\dots,5$$

we have

$$0\le\sum_{j=1}^{4}(x_i^{j+1}-x_i^j)\cdot\delta_i^j\le\sum_{j=1}^{4}(x_i^{j+1}-x_i^j)=x_i^5-x_i^1=u_i-0=u_i,\ i=1,\dots,m$$

such that the constraint $(**)$ is redundant.

Now we rearrange the optimization problem such that the inequality constraints take the form $A_{\le}(\delta,\omega)\le b_{\le}$ and the equality constraints take the form $A_{=}(\delta,\omega)=b_{=}$, where $\delta:=(\delta_1^1,\dots,\delta_m^1,\dots,\delta_1^4,\dots,\delta_m^4)$ and $\omega:=(\omega_1^1,\dots,\omega_m^1,\dots,\omega_1^3,\dots,\omega_m^3)$.

Then the matrix $A_{\le}$ is of the following form:

	I_m			$-I_m$		
		I_m			$-I_m$	
			I_m			$-I_m$
$-I_m$				I_m		
	$-I_m$				I_m	
		$-I_m$				I_m

Figure 5.7: Structure of matrix $A_{\le}$ of the inequality constraints (δ-method)

Here I_m denotes the identity matrix of dimension m. Empty rectangles denote zero-filled $m \times m$-matrices. Moreover, we can state the following proposition:

Proposition 5.1. *The matrix $A_{\leq}$ is totally unimodular.*

Proof. $A_{\leq}$ is a $(0, 1, -1)$-matrix with no more than two nonzero entries in each column. Furthermore we have $\sum_i (a_{\leq})_{ij} = 0$ if column j contains two nonzero coefficients. Then $A_{\leq}$ is totally unimodular (cf. [75]). $\square$

$b_{\leq}$ is the zero vector.

The matrix $A_{=}$ is of the following form:

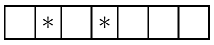


Figure 5.8: Structure of matrix $A_{=}$ of the equality constraints (δ-method)

Rectangles filled with a $*$ denote matrices of size $n \times m$ which have non-zero entries on the positions corresponding to the 1s and -1s in the incidence matrix A of the graph of the given minimum cost flow problem. Empty rectangles denote zero-filled $n \times m$-matrices.

$b_{=}$ is the supply/demand-vector b.

The optimization problem has $4m + 3m = 7m$ optimization variables and $6m + m = 7m$ constraints.

5.3 Branching for SOS 2 Conditions

Instead of introducing binary variables in order to guarantee the SOS 2 condition for the λ-variables in the lambda method, it is possible to incorporate these conditions directly into the branching phase of a Branch and Bound (B&B) algorithm.

First we consider a set of consecutive variables $\{\lambda^1, \dots, \lambda^s\}$. This set of λ-variables is said to be SOS 2, if at most two of these variables are nonzero and if exactly two of them are nonzero, they must be adjacent.

Instead of branching on individual variables, the classical branching idea for SOS 2 conditions, as presented in [9] and [10], for example, is to branch on sets of variables. So, if there are two nonnegative λ-variables λ^k and λ^l, which are not adjacent, we choose one of them and generate two new problems. If we choose λ^k as branching variable, we add the equation $\sum\limits_{j=1}^{k} \lambda^j = 1$ to subproblem

1 and the equation $\sum\limits_{j=k}^{s} \lambda^j = 1$ to subproblem 2. Strategies for the selection of the branching variable are presented in the literature cited above.

We will now consider a Branch and Bound algorithm which is adapted to the TMCFP as presented in Subsection 5.2.2. Remember that we have several sets of λ-variables $\lambda_i = \{\lambda_i^1, \dots, \lambda_i^5\}$, one for each arc $i = 1, \dots, m$, and we require each set λ_i to be SOS 2.

The optimization problem is

$$(P) = \begin{cases} \min\limits_{\lambda} & \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{5} c(x_i^j)\lambda_i^j & \\ \text{s.t.} & \sum\limits_{i=1}^{m} A_{ki} \sum\limits_{j=1}^{5} x_i^j \lambda_i^j = b_k & k = 1, \dots, n \\ & \sum\limits_{j=1}^{5} \lambda_i^j = 1 & i = 1, \dots, m \\ & \lambda_i^j \geq 0 & i = 1, \dots, m,\ j = 1, \dots, 5 \\ & \lambda_i \in \text{SOS } 2 & i = 1, \dots, m \end{cases} \tag{5.27}$$

where $x_i^j, i = 1, \dots, m, j = 1, \dots, 5$, c, A and b as defined in Subsection 5.2.2.

We propose the following Branch and Bound algorithm:

Algorithm 5.1 Branch and Bound algorithm for the threshold minimum cost flow problem (SOS 2 B&B method)

```
1: let $L$ be a list of unsolved problems; set $L := \{(P)\}$
2: apply the ant algorithm to $(P)$
3: if the ant algorithm doesn't find a feasible solution then
4:     let $\bar{z} := \infty$ be an upper bound for $(P)$
5: else
6:     let $\tilde{\lambda}$ be the ant solution with cost $\tilde{z}$
7:     let $\bar{z} := \tilde{z}$ be an upper bound for $(P)$
8: end if
```

```
 9: while $L$ is not empty do
10:     choose a problem $P$ from $L$
11:     solve the SOS 2 relaxation of $P$
12:     let $\lambda^*$ be an optimal solution
13:     let $\underline{z}_P := \sum_{i=1}^{m} \sum_{j=1}^{5} c(x_i^j)\lambda^{*j}_i$ be a lower bound for $P$
14:     if $\lambda_i^* \in$ SOS 2 for $i = 1, \dots, m$ then
15:         if $\bar{z} > \underline{z}_P$ then
16:             $\bar{z} := \underline{z}_P$
17:             $\tilde{\lambda} := \lambda^*$
18:             delete all $P'$ from $L$ where $\underline{z}_{P'} \geq \bar{z}$
19:         end if
20:         goto 9
21:     end if
22:     divide $P$ into subproblems and add them to $L$
23: end while
24: if $\tilde{\lambda}$ exists then
25:     $\tilde{\lambda}$ is an optimal solution to $(P)$
26: else
27:     $(P)$ is infeasible
28: end if
```

Note that (P) denotes problem (5.27), while P is any problem in L.

In order to obtain a tight upper bound, we first apply the ant algorithm which we proposed in Section 5.1 to the threshold minimum cost flow problem in step 2. It is possible that the ant algorithm doesn't find a feasible solution in a reasonable time (though there might exist one!). In this case, we set the upper bound to ∞.

The SOS 2 relaxation of problem P in step 11 is the problem P without the SOS 2 condition for $\lambda_i, i = 1, \dots, n$.

For the branching phase, step 22, we suggest the following procedure. First choose $i^* \in \{1, \dots, n\}$ such that $\lambda_{i^*} \notin$ SOS 2. Then create three subproblems: Add the equation $\sum_{j=1}^{1} \lambda_{i^*}^j = 1$ to subproblem 1, the equation $\sum_{j=2}^{3} \lambda_{i^*}^j = 1$ to subproblem 2 and the equation $\sum_{j=4}^{5} \lambda_{i^*}^j = 1$ to subproblem 3.

In subproblem 1, the flow on arc i^* is restricted to be zero. In subproblem 2, the flow on arc i^* is restricted to be greater or equal to zero and smaller or equal to the threshold value θ_{i^*}. In subproblem 3, the flow on arc i^* is restricted to be greater or equal to the threshold value θ_{i^*}.

This procedure excludes the possibility that $\lambda^1_{i^*} > 0$ and $\lambda^2_{i^*} > 0$ at the same time (case 1) or $\lambda^3_{i^*} > 0$ and $\lambda^4_{i^*} > 0$ at the same time (case 2). Regarding the fact that $x^1_{i^*} = x^2_{i^*}$ and $x^3_{i^*} = x^4_{i^*}$ and the definition of the new cost function c in Subsection 5.2.2, it becomes obvious that in a minimization problem the optimal solution would never comprise the cases mentioned above. Instead, in an optimal solution we would always have $\lambda^1_{i^*} = 1$ and $\lambda^2_{i^*} = 0$ in case 1 or $\lambda^3_{i^*} = 0$ and $\lambda^4_{i^*} = 1$ in case 2.

In step 10, we suggest choosing subproblems of types 1 and 3 prior to subproblems of type 2 as next problems from L.

In order to accelerate the Branch and Bound algorithm, diverse strategies for the choice of the next subproblem, for example best-first or best-projection, should be tried.

Summing up the arguments, we can state the following theorem:

Theorem 5.2. *Algorithm 5.1 terminates with an optimal solution of the threshold minimum cost problem, if one exists, or proves that the TMCFP is infeasible.*

5.4 Example-Problems

Now we examine the approach described the previous subsections for three example threshold minimum cost flow problems.

In the first problem, the construction of a solution that respects the goal of surpassing the threshold value is possible. We compare the optimal solution of the MCFP and the TMCFP. Furthermore, we illustrate the cost assignments for two arcs.

In the second problem, where it is not possible to find a solution where the threshold value is exceeded on every arc, we show that with the threshold model a reasonable solution is constructed nevertheless.

In the third problem, we compare the computational performance of the different presented models for the TMCFP.

5.4.1 Avoiding Small Flows Example 1

We consider the following MCFP

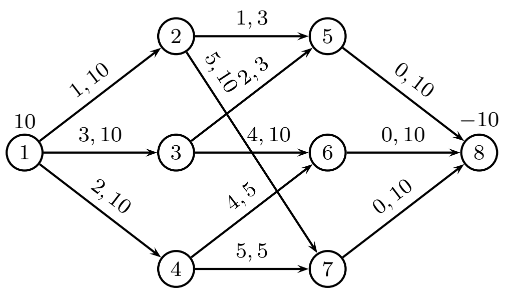


Figure 5.9: Avoiding small flows example 1: Minimum cost flow problem

where c, u on the arcs denotes the cost and the capacity, respectively.

The numbering of the arcs (i, j) is lexicographical.

The problem can formally be written as

$$\begin{array}{ll} \min & c^\top x \\ \text{s.t.} & Ax = b \\ & 0 \le x \le u \end{array}$$

where

$$c = (1,3,2,1,5,2,4,4,5,0,0,0)^\top$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 \end{pmatrix}$$

$$b = (10,0,0,0,0,0,0,-10)^\top$$

$$u = (10,10,10,3,10,3,10,5,5,10,10,10)^\top$$

The optimal solution to this problem is $x = (3,3,4,3,0,3,0,4,0,6,4,0)^\top$ with a total cost of 45.

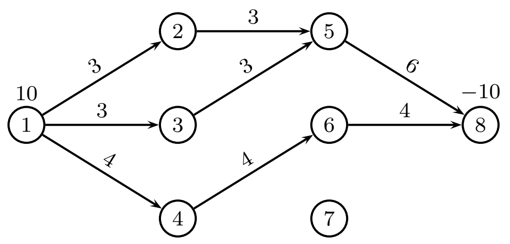


Figure 5.10: Avoiding small flows example 1: Optimal solution of the original minimum cost flow problem

Now we consider the corresponding TMCFP with threshold value 4. That means we introduce the additional goal that for all arcs having a positive flow, the amount of flow shall be greater or equal to 4, so we have $\theta_i = 4, i = 1, \ldots, m$. Considering the given values for costs and capacities, we have

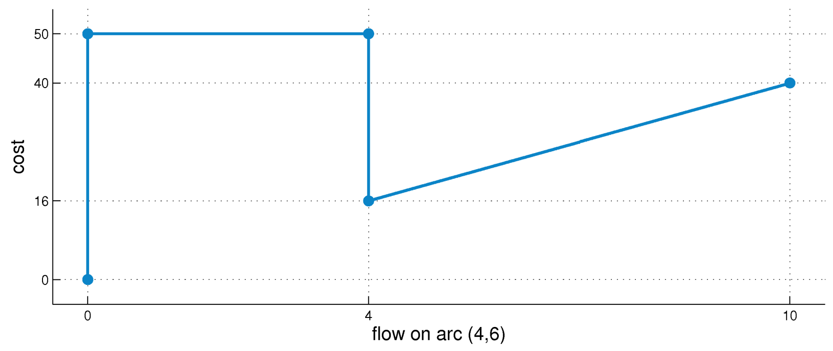
$$c^{max} = u^{max} \cdot \max_i c_i = u_5 \cdot c_5 = 10 \cdot 5 = 50.$$

For all arcs except arc $(2,5)$ and arc $(3,5)$ we have $\theta_i < u_i$. We define therefore the grid points x_i^j in the following way:

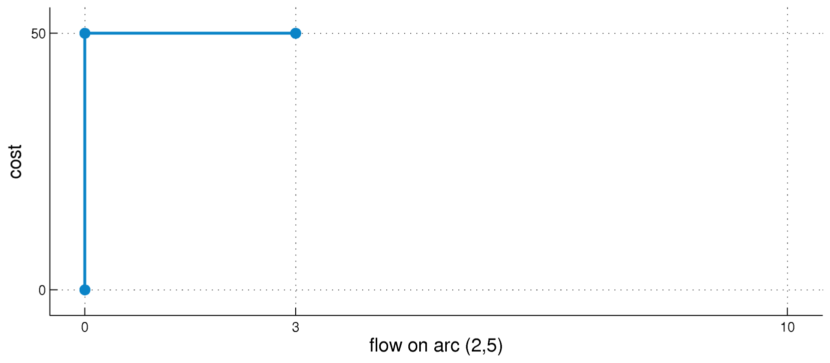
i	1	2	3	4	5	6	7	8	9	10	11	12
x_i^1	0	0	0	0	0	0	0	0	0	0	0	0
x_i^2	0	0	0	0	0	0	0	0	0	0	0	0
x_i^3	4	4	4	3	4	3	4	4	4	4	4	4
x_i^4	4	4	4	3	4	3	4	4	4	4	4	4
x_i^5	10	10	10	3	10	3	10	5	5	10	10	10

The corresponding cost functions are defined as given in Subsection 5.2.2.

The following figure illustrates the cost assignment for arcs (4,6) and (2,5).



(a) Cost assignment for arc (4,6)



(b) Cost assignment for arc (2,5)

Figure 5.11: Example cost assignments

We can see that small amounts of flow, i.e. amounts smaller than θ_i, are more expensive than large amounts of flow exploiting the full capacity of an arc with capacity greater than θ_i. Nevertheless it is still possible to send small amounts of flow if this can not be avoided.

As we defined the cost function to be constant on the interval $]0, \theta_i[$, we increase the possibility of getting only some "larger" small amounts of flow, but not many very small amounts of flow.

Solving the threshold minimum cost flow problem with one of the methods described above leads to an optimal solution $x = (10, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 10)^\top$ with a total cost of 60.

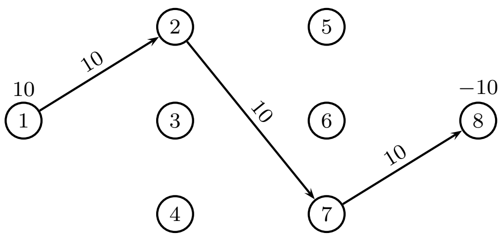


Figure 5.12: Avoiding small flows example 1: Optimal solution of the threshold minimum cost flow problem

The total cost for this solution in the original network is also 60.

In the solution of the original MCFP, we have four arcs carrying an amount of flow smaller than 4 and greater than 0. In the new solution of the TMCFP, all arcs have an amount of flow greater or equal to 4 or equal to 0.

5.4.2 Avoiding Small Flows Example 2

Now we consider the following minimum cost flow problem

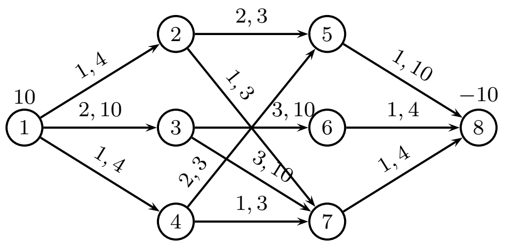


Figure 5.13: Avoiding small flows example 2: Minimum cost flow problem

The optimal solution to this problem is $x = (3, 3, 4, 3, 0, 3, 0, 4, 0, 6, 4, 0)^\top$ with a total cost of 40.

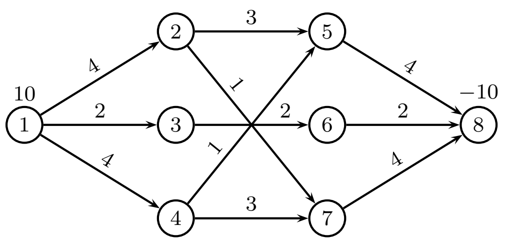


Figure 5.14: Avoiding small flows example 2: Optimal solution of the original minimum cost flow problem

The threshold value is set to $\theta_i = 4, i = 1, \dots, m$. In the original optimal solution, we have seven arcs that carry an amount of flow smaller than four. All in all, 11 out of 12 arcs are used in the optimal solution.

Considering the given values for costs and capacity, we have

$$c^{max} = u^{max} \cdot \max_i c_i = 10 \cdot 3 = 30.$$

The optimal solution of the threshold minimum cost flow problem is then

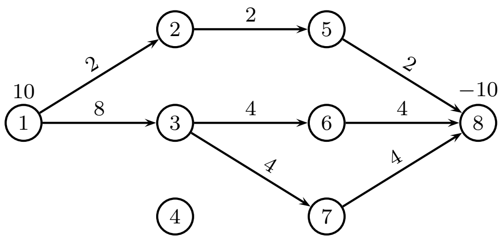


Figure 5.15: Avoiding small flows example 2: Optimal solution of the threshold minimum cost flow problem

The total cost for this solution in the original network is 56.

We can see that on three arcs the amount of flow is not respecting the threshold value. This is due to the structure of the network: If we removed all arcs having a capacity smaller than four, the network flow problem would be infeasible. However, our approach reduces the number of arcs having a flow smaller than the threshold value.

5.4.3 Avoiding Small Flows Example 3

We consider the following minimum cost flow problem, which was previously discussed in [45]:

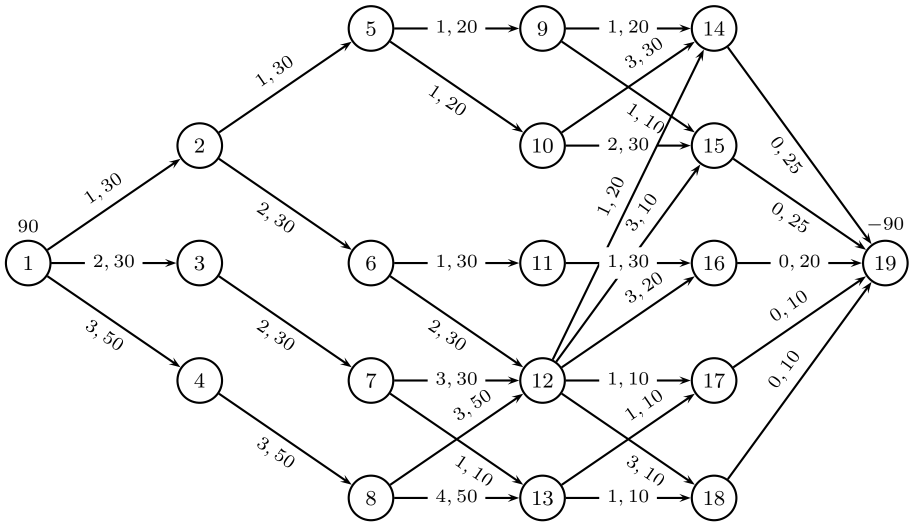


Figure 5.16: Avoiding small flows example 3: Minimum cost flow problem

The optimal solution to this problem is shown below. The optimal solution has a total cost of 675 and contains 27 arcs with positive flow, where 3 arcs have a flow value greater than 0 and smaller than 10.

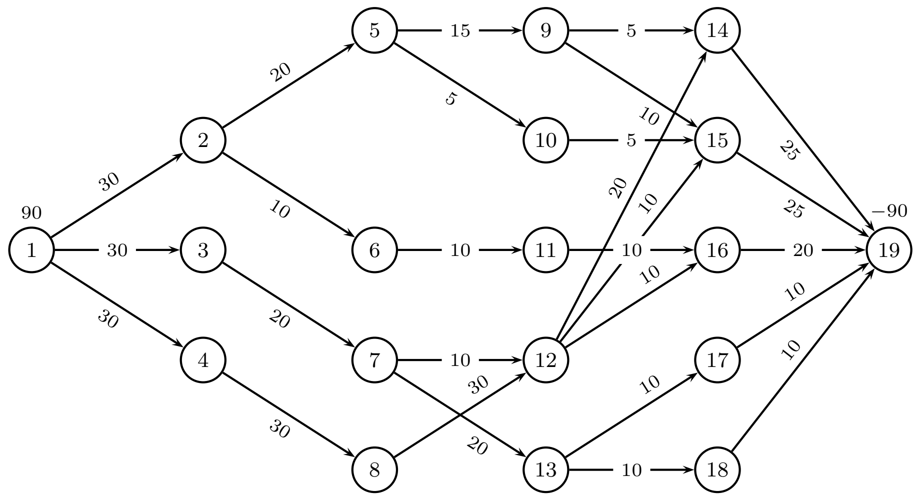


Figure 5.17: Avoiding small flows example 3: Optimal solution of the original minimum cost flow problem

We set the threshold values $\theta_i = 10$, $i = 1, \dots, m$. The optimal solution of the original problem contains three arcs with an amount of flow smaller than 10. Considering the given values for costs and capacity, we have $c^{max} = 200$.

The ant algorithm produces the following solution for parameter settings $\alpha = 3$, $\beta = 3$:

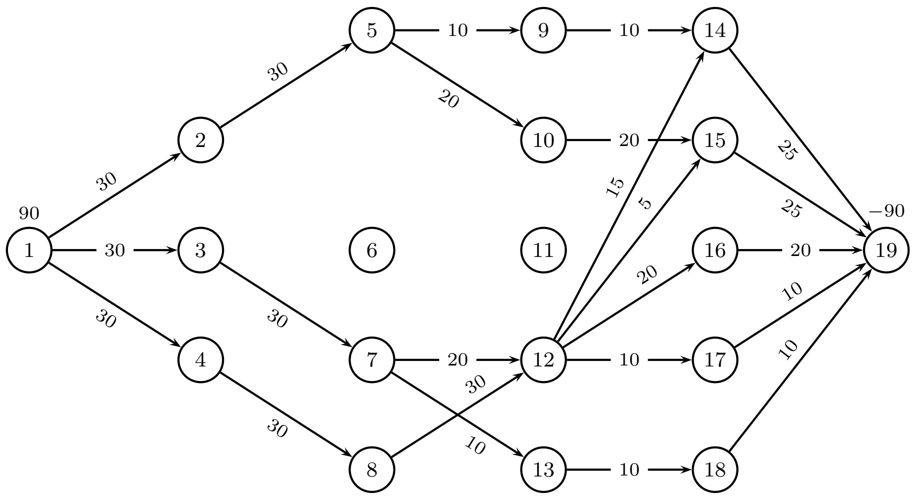


Figure 5.18: Avoiding small flows example 3: Ant solution of the threshold minimum cost flow problem

The ant solution has a total cost of 710 (regarding the cost values in the original network) and contains 23 arcs with positive flow, where 1 arc has a flow value greater than 0 and smaller than 10.

The optimal solution of the threshold minimum cost flow problem is then

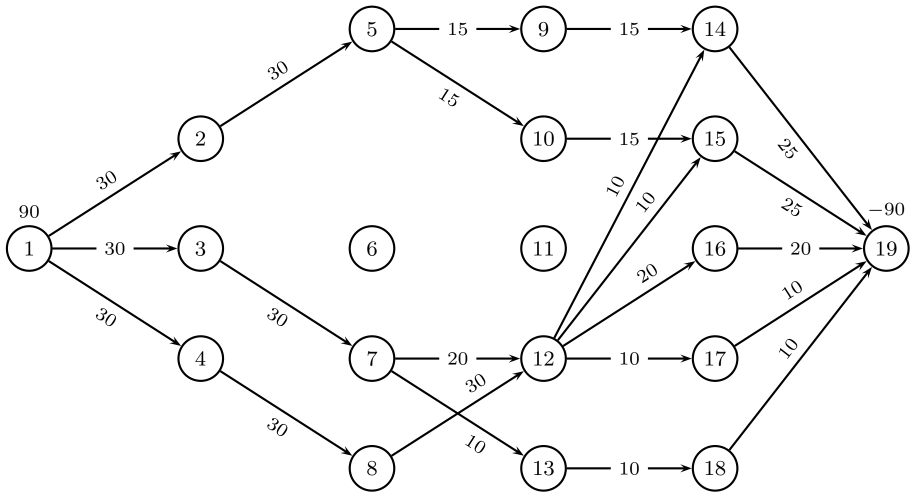


Figure 5.19: Avoiding small flows example 3: Optimal solution of the threshold minimum cost flow problem

In the optimal solution of the threshold minimum cost flow problem, 23 arcs have a positive flow. No arc with positive flow carries an amount of flow smaller than 10. The total cost of this solution is 715 (regarding the cost values in the original network).

The following table lists a running time comparison for the λ and δ-method and the SOS 2 B&B method for the avoiding small flows example 3. The comparisons are all based on a standard Branch and Bound implementation for mixed-integer programs in Matlab. For the SOS2 B&B method, only the branching scheme was modified as explained in Section 5.3. The ant algorithm was implemented in Java. All test runs were performed on a standard PC with Pentium Dual-Core 1.60GHz.

algorithm	running time	remark
δ-method	13.2 seconds	
λ-method	179.2 seconds	
SOS2 B&B method	10.4 seconds	upper bound = ∞ (no ant preprocessing)
SOS2 B&B method	4.2 seconds	upper bound from ant preprocessing
ant algorithm	0.21 seconds	(*)

Table 5.1: Avoiding small amounts of flow: Numerical comparison

Note that the ant algorithm (*) did not find the optimal solution of the TMCFP.

As already shown in [77], the running time comparison confirms that the δ-method is computationally superior to the λ-method. However, the SOS2 B&B method leads to an even shorter computation time, even when running the algorithm without the ant algorithm preprocessing (i.e. with upper bound set to ∞ at the initialization of the algorithm). This is due to the fact that the branching scheme is specially adapted to the problem structure, while for the λ- and δ-method, a general branching scheme is used. Utilizing the ant algorithm preprocessing and therefore a tighter upper bound leads to a reduction of the running time of almost 60%.

The running time of the standalone version of the ant algorithm is considerably less than for the exact methods. However, the ant algorithm does not find the optimal solution of the TMCFP: The ant solution contains as many arcs with positive flow as the exact solution (23 arcs) and has a total cost of 710 which is slightly less than the total cost of the exact solution (715). However, the threshold goal is violated for one arc. In comparison to the optimal solution of the original MCFP, the number of arcs not exceeding the threshold value is reduced from 3 to 1. Therefore, for large networks, where the running time of the algorithms might be critical, the ant algorithm may be a reasonable compromise between solution quality and computational effort.

5.5 Summary and Conclusions

The optimal solutions of MCFPs often contain many small amounts of flow on diverse arcs. In practice, however, small amounts of flow are not desired in many cases. Solutions with each arc having either a flow of zero or a flow exceeding a certain threshold value would be preferred, even if they are more expensive. Two solution methods for this kind of problem, called the threshold minimum cost flow problem, were developed:

In Section 5.1, a heuristic method based on the ACO metaheuristic was presented. The ants' behavior of following other ants by following their pheromone trails leads to solutions where the number of ants on each arc, and therefore the corresponding flow, is either zero or relatively large. The introduction of penalty costs for constructed solutions that violate the threshold claim enforces this effect.

In Section 5.2, an exact approach using a linearization of non-continuous cost functions with the help of SOS 2 conditions, was developed. Two binary approaches for modeling the SOS 2 conditions as binary variables, the *lambda method* and the *delta method*, were presented.

In Section 5.3, a Branch and Bound method for solving the TMCFP was developed. The method is based on the SOS 2 formulation and relaxation of the TMCFP and uses a special branching scheme based on SOS conditions. Using the heuristic solution of the ant algorithm of Section 5.1 as first upper bound, the computation time of the B&B method can be reduced significantly.

The presented approaches were applied to example problems in Section 5.4. A running time comparison shows that concerning the computational time, the ant algorithm is superior to the SOS 2 based approaches. Furthermore, the SOS 2 Branch and Bound method, which was presented in Section 5.3, has a shorter computation time than a general Branch and Bound method, which was applied to the problem formulations of the lambda and delta method. However, it has to be kept in mind that the ant algorithm as heuristic method does not guarantee to find an optimal solution.

Chapter 6

Robust Optimization in Network Flows: Uncertain Costs

In this chapter, we consider network flow problems with uncertain costs. We first give a short introduction into the problem formulation and objectives. Then we give a short description of the existing approach of Bertsimas and Sim [14]. Subsequently, we extend their approach to multicommodity flows. Furthermore, we introduce uncertain costs into the ant algorithm described in Chapter 4. Finally, we compare the results of the approaches for different distributions of the uncertain cost values.

6.1 Problem Formulation and Objectives

In network flow problems, the cost values on the arcs can be subject to uncertainty, where not necessarily all arcs must be concerned. The aim is to find a solution which can cope best with all possible realizations of the uncertain data in some sense. In case of uncertain costs, this means that the calculated solution will not become extremely expensive, no matter which cost values are actually realized.

In the considered approaches, the level of robustness can be controlled by a robustness parameter. The robustness level can be chosen anywhere between maximum risk, i.e. only the minimum cost values are considered for all arcs, and maximum robustness, i.e. the maximum cost values on every arc are considered.

In Robust Optimization, the distribution of the uncertainty values is usually not considered - the uncertainty data are given in interval form. This differs from the Stochastic Programming approach, see Chapter 9, where the distribution of the uncertainty values is taken into account. Though in Robust Optimization the uncertainty values are a priori considered as equally distributed, a robustness parameter can regulate to which region of the uncertainty interval more importance is assigned.

6.2 Approach of Bertsimas and Sim

In [14], D. Bertsimas and M. Sim propose a general approach to Robust Optimization which is extended to a robust approach for solving network flow problems with uncertain cost. Based on this approach, Bertsimas and Sim develop an efficient algorithm for solving single-commodity minimum cost network flow problems with uncertainty in the cost vector.

6.2.1 Model Formulation

In the following, the fundamentals of the robust approach of Bertsimas and Sim are explained. For details, see [14].

A given network $G = (\mathcal{N}, \mathcal{A})$ and the corresponding minimum cost network flow problem

$$\begin{array}{lll} \min & \sum\limits_{(i,j)\in\mathcal{A}} \tilde{c}_{ij} x_{ij} & \\ \text{s.t.} & \sum\limits_{\{j:(i,j)\in\mathcal{A}\}} x_{ij} - \sum\limits_{\{j:(j,i)\in\mathcal{A}\}} x_{ji} = b_i & \forall i \in \mathcal{N} \\ & 0 \leq x_{ij} \leq u_{ij} & \forall (i,j) \in \mathcal{A} \end{array}$$

are considered, where $\tilde{c}_{ij} \in [c_{ij}, c_{ij} + d_{ij}], c_{ij}, d_{ij} \geq 0,\ (i,j) \in \mathcal{A}$. c_{ij} is the nominal cost on arc (i,j), while d_{ij} describes the possible uncertainty in the cost value on arc (i,j). d_{ij} will also be called the (maximum) possible extra cost.

Let Γ be a given number with $\Gamma \in \{0, \ldots, |J|\}$, where $J = \{(i,j) : d_{ij} > 0\}$. The parameter Γ regulates the robustness in the objective function: It determines the number of arcs where cost fluctuations are expected. The goal is to find an optimal solution that minimizes the objective function for all scenarios where at most Γ cost coefficients are assumed to change within the given uncertainty interval. This means that we want to be protected against all cases, where up to Γ cost coefficients are allowed to take at most their worst case value $c_{ij} + d_{ij}$, while the other cost coefficients take their nominal cost value c_{ij}.

The robust minimum cost network flow problem is then

$$\begin{array}{lll}
\min & \sum\limits_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} + \max\limits_{\{S:S\subseteq J,|S|\leq\Gamma\}} \sum\limits_{(i,j)\in S} d_{ij}x_{ij} & \\
\text{s.t.} & \sum\limits_{\{j:(i,j)\in\mathcal{A}\}} x_{ij} - \sum\limits_{\{j:(j,i)\in\mathcal{A}\}} x_{ji} = b_i & \forall i\in\mathcal{N} \\
 & 0\leq x_{ij}\leq u_{ij} & \forall (i,j)\in\mathcal{A}.
\end{array}$$

By formulating the dual of the inner maximization problem and applying strong duality, Bertsimas and Sim show in [14] that this problem is equivalent to

$$\min_{\theta\geq 0} \quad Z(\theta)$$

where

$$Z(\theta) = \left\{\begin{array}{lll}
\Gamma\theta + \min & \sum\limits_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} + \sum\limits_{(i,j)\in J} d_{ij}\max\{x_{ij} - \frac{\theta}{d_{ij}}, 0\} & \\
\text{s.t.} & \sum\limits_{\{j:(i,j)\in\mathcal{A}\}} x_{ij} - \sum\limits_{\{j:(j,i)\in\mathcal{A}\}} x_{ji} = b_i & \forall i\in\mathcal{N} \\
 & 0\leq x_{ij}\leq u_{ij} & \forall (i,j)\in\mathcal{A}
\end{array}\right.$$

It is shown in [14] that $Z(\theta)$ can be evaluated by solving a minimum cost flow problem and that $Z(\theta)$ is a convex function of θ.

Bertsimas and Sim propose a bisection method in order to solve the robust minimum cost flow problem. The major advantage of their approach is the fact, that the solution of the robust problem can be obtained by solving a series of network flow problems. A disadvantage is that in practice, the number of network flow problems that need to be solved can be very high.

6.2.2 Extension to Multicommodity Flows

The approach of Bertsimas and Sim, as proposed in [14], can efficiently be applied to network flow problems, such that a robust minimum cost flow problem can be solved by solving a collection of modified nominal minimum cost flow problems.

While Bertsimas and Sim consider single-commodity minimum cost flow problems only, we extend their approach to multicommodity minimum cost flow problems.

Let p be the number of commodities and $K := \{1,\ldots,p\}$. Let $G = (\mathcal{N},\mathcal{A})$ be a directed graph with cost c_{ij}^k and capacity u_{ij}^k for every arc $(i,j)\in\mathcal{A}$ and commodity $k\in K$ and bundle constraints κ_{ij} for every arc $(i,j)\in\mathcal{A}$. The bundle constraints κ_{ij} restrict the total flow of all commodities on arc (i,j), cf. Section 2.3.2. Let b_i^k be the supply/demand values for node $i\in\mathcal{N}$ and commodity $k\in K$.

Then the multicommodity minimum cost flow problem is given as:

$$\begin{array}{lll} \min & \sum_{k=1}^{p} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k & \\ \text{s.t.} & \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^k = b_i^k & i \in \mathcal{N}, k \in K \\ & \sum_{k=1}^{p} x_{ij}^k \le \kappa_{ij} & (i,j) \in \mathcal{A} \\ & 0 \le x_{ij}^k \le u_{ij}^k & (i,j) \in \mathcal{A}, k \in K \end{array} \tag{6.1}$$

This formulation corresponds exactly to the formulation of problem (2.5), which was partially written in matrix form.

Let X be the set of feasible solutions of problem (6.1).

Now we consider uncertainty in the cost vector. Let $d_{ij} := (d_{ij}^1, \ldots, d_{ij}^p), d_{ij}^k \ge 0, (i,j) \in \mathcal{A}, k = 1, \ldots, p$ such that the actual costs for commodity k on arc (i,j) take a value in $[c_{ij}^k, c_{ij}^k + d_{ij}^k]$. Let $J_K := \{(k,i,j) : d_{ij}^k > 0\}$.

The robust multicommodity minimum cost flow problem is:

$$\begin{array}{ll} \min & \sum_{k=1}^{p} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k + \max_{\{S: S\subseteq J_K, |S|\le\Gamma\}} \sum_{(k,i,j)\in S} d_{ij}^k x_{ij}^k \\ \text{s.t.} & x \in X. \end{array} \tag{6.2}$$

where $\Gamma \in [0, |\mathcal{A}| \cdot p]$. Problem (6.2) can be rewritten as follows:

$$\begin{array}{lllll} \min_{x\in X} & \Big(\sum_{k=1}^{p} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k + & \max_{v} & \sum_{(k,i,j)\in J_K} d_{ij}^k x_{ij}^k v_{ij}^k \Big) & \\ & & \text{s.t.} & 0 \le v_{ij}^k \le 1 & (k,i,j) \in J_K \\ & & & \sum_{(k,i,j)\in J_K} v_{ij}^k \le \Gamma. & \end{array} \tag{6.3}$$

For a fixed $x \in X$, the dual of the inner maximization problem of problem (6.3) is:

$$\begin{array}{lll} \min_{\theta,y} & \Gamma\theta + \sum_{(k,i,j)\in J_K} y_{ij}^k & \\ \text{s.t.} & y_{ij}^k + \theta \ge d_{ij}^k x_{ij}^k & (k,i,j) \in J_K \\ & y_{ij}^k \ge 0 & (k,i,j) \in J_K \\ & \theta \ge 0. & \end{array} \tag{6.4}$$

Applying strong duality, we can rewrite problem (6.3) as follows:

$$\begin{array}{lllll} \min_{x\in X} & \sum_{k=1}^{p} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k + & \min_{\theta,y} & \Big(\Gamma\theta + \sum_{(k,i,j)\in J_K} y_{ij}^k \Big) & \\ & & \text{s.t.} & y_{ij}^k + \theta \ge d_{ij}^k x_{ij}^k & (k,i,j) \in J_K \\ & & & y_{ij}^k \ge 0 & (k,i,j) \in J_K \\ & & & \theta \ge 0. & \end{array} \tag{6.5}$$

As the optimization variables in problem (6.5) are coupled and therefore the outer and inner minimization problem have to be solved simultaneously, problem

(6.5) can be rewritten as:

$$\begin{array}{lll}
\min\limits_{x,\theta,y} & \sum\limits_{k=1}^{p} \sum\limits_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k + \Gamma\theta + \sum\limits_{(k,i,j)\in J_K} y_{ij}^k & \\
\text{s.t.} & y_{ij}^k + \theta \geq d_{ij}^k x_{ij}^k & (k,i,j) \in J_K \\
& y_{ij}^k \geq 0 & (k,i,j) \in J_K \\
& \theta \geq 0 & \\
& x \in X. &
\end{array} \tag{6.6}$$

Problem (6.6) is equivalent to the following problem:

$$\min_{\theta\geq 0} \quad Z(\theta) \tag{6.7}$$

where

$$\begin{array}{llll}
Z(\theta) = \Gamma\theta + & \min\limits_{x,y} & \left(\sum\limits_{k=1}^{p} \sum\limits_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k + \sum\limits_{(k,i,j)\in J_K} y_{ij}^k \right) & \\
& \text{s.t.} & y_{ij}^k \geq d_{ij}^k x_{ij}^k - \theta & (k,i,j) \in J_K \\
& & y_{ij}^k \geq 0 & (k,i,j) \in J_K \\
& & \theta \geq 0 & \\
& & x \in X. &
\end{array} \tag{6.8}$$

Eliminating the variables y_{ij}^k in equation (6.8), we obtain

$$\begin{array}{lll}
Z(\theta) = \Gamma\theta + & \min\limits_{x} & \left(\sum\limits_{k=1}^{p} \sum\limits_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k + \sum\limits_{(k,i,j)\in J_K} d_{ij}^k \max\{x_{ij}^k - \frac{\theta}{d_{ij}^k}, 0\} \right) \\
& \text{s.t.} & x \in X.
\end{array} \tag{6.9}$$

Now we consider the minimization problem given in equation (6.9) for fixed $\theta \geq 0$:

$$\begin{array}{ll}
\min\limits_{x} & \sum\limits_{k=1}^{p} \sum\limits_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k + \sum\limits_{(k,i,j)\in J_K} d_{ij}^k \max\left(x_{ij}^k - \frac{\theta}{d_{ij}^k}, 0\right) \\
\text{s.t.} & x \in X.
\end{array} \tag{6.10}$$

Our goal is to solve problem (6.10) as a multicommodity minimum cost flow problem. We have to modify the underlying graph $G = (\mathcal{N}, \mathcal{A})$. For every arc (i,j) we introduce two new nodes i' and j'. Furthermore, we insert four new arcs (i,i'), (i',j'), (j',j) and (i',j) and remove the original arc (i,j). The costs, capacities and bundle constraints are set as follows:

$$\begin{array}{lll}
c^k_{ii'} = c^k_{ij} & u^k_{ii'} = u^k_{ij} & \kappa_{ii'} = \kappa_{ij} \\
c^k_{i'j'} = d^k_{ij} & u^k_{i'j'} = \infty & \kappa_{i'j'} = \infty \\
c^k_{j'j} = 0 & u^k_{j'j} = \infty & \kappa_{j'j} = \infty \\
c^k_{i'j} = 0 & u^k_{i'j} = \dfrac{\theta}{d^k_{ij}} & \kappa_{i'j} = \infty
\end{array}$$

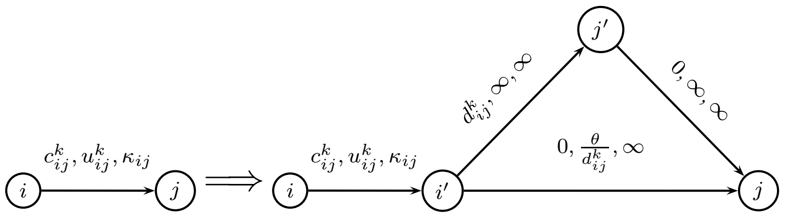

$c^k_{ij}, u^k_{ij}, \kappa_{ij}$
i
j
$\Longrightarrow$
$c^k_{ij}, u^k_{ij}, \kappa_{ij}$
i
i'
d^k_{ij}, ∞, ∞
j'
$0, \infty, \infty$
$0, \frac{\theta}{d^k_{ij}}, \infty$
j

Figure 6.1: Insertion of new arcs

Let the modified graph be $G' = (\mathcal{N}', \mathcal{A}')$.

Theorem 6.1. *For fixed θ, problem (6.10) can be solved by solving the multi-commodity minimum cost flow problem corresponding to $G' = (\mathcal{N}', \mathcal{A}')$.*

Proof. Consider an optimal solution x of problem (6.10) transfered to the context of the graph G'. If $x^k_{ij} \le \frac{\theta}{d^k_{ij}}$ for a given commodity k and a given arc $(i,j) \in \mathcal{A}$ with $(k,i,j) \in J_K$, then in G' the flow would be routed along the arcs (i,i') and (i',j). The arising cost is

$$c^k_{ii'}x^k_{ij} + c^k_{i'j}x^k_{ij} = c^k_{ij}x^k_{ij}.$$

If $x^k_{ij} \ge \frac{\theta}{d^k_{ij}}$ for a given commodity k and a given arc $(i,j) \in \mathcal{A}$ with $(k,i,j) \in J_K$, then in G' the flow would be first routed along the arc (i,i'). Then an amount of $\frac{\theta}{d^k_{ij}}$ would be routed along arc (i',j) and the excess amount $x^k_{ij} - \frac{\theta}{d^k_{ij}}$ would be routed along (i',j') and (j',j). The arising cost is

$$c^k_{ii'}x^k_{ij} + c^k_{i'j}\frac{\theta}{d^k_{ij}} + c^k_{i'j'}\left(x^k_{ij} - \frac{\theta}{d^k_{ij}}\right) + c_{j'j}\left(x^k_{ij} - \frac{\theta}{d^k_{ij}}\right) = c^k_{ij}x^k_{ij} + d^k_{ij}\left(x^k_{ij} - \frac{\theta}{d^k_{ij}}\right).$$

The bundle constraint $\kappa_{i'j}$ on arc (i',j) is irrelevant as the actual bundle capacity is determined by the bundle constraint on arc (i,i') and therefore equal to the bundle constraint on the original arc (i,j). The contribution to cost matches the objective function of problem (6.10).

Furthermore, an optimal solution x^* in G' corresponds to a solution in G, as the capacity constraints on arc (i, i') in G' and likewise on arc (i, j) in G are u_{ij}^k for commodity k and the bundle capacity is κ_{ij}. Moreover, the sum of flows on arcs (i', j) and (i', j') as well as on arcs (i', j) and (j', j') in G' is indirectly limited by the same capacity bounds, as the arc (i, i') is the only incoming arc in node i' in G'. □

Theorem 6.2. *$Z(\theta)$ is a convex function of θ.*

Proof. Let $\theta_1,\ \theta_2 \geq 0$. Let (x_1, y_1) and (x_2, y_2) be the optimal solutions of problem (6.8) corresponding to θ_1 and θ_2, respectively. The feasible region of problem (6.8) is convex, so for all $\lambda \in [0, 1]$ it holds that $(\lambda x_1 + (1-\lambda)x_2, \lambda y_1 + (1-\lambda)y_2)$ is feasible to the problem with $\theta = \lambda\theta_1 + (1-\lambda)\theta_2$. Consequently,

$$\begin{aligned}
\lambda Z(\theta_1) + (1-\lambda)Z(\theta_2) &= \lambda\left(\Gamma\theta_1 + \sum_{k=1}^{p}\sum_{(i,j)\in\mathcal{A}} c_{ij}^k (x_1)_{ij}^k + \sum_{(k,i,j)\in J_K} (y_1)_{ij}^k\right) + \\
&\quad (1-\lambda)\left(\Gamma\theta_2 + \sum_{k=1}^{p}\sum_{(i,j)\in\mathcal{A}} c_{ij}^k (x_2)_{ij}^k + \sum_{(k,i,j)\in J_K} (y_2)_{ij}^k\right) \\
&= \Gamma(\lambda\theta_1 + (1-\lambda)\theta_2) + \\
&\quad \sum_{k=1}^{p}\sum_{(i,j)\in\mathcal{A}} \left(c_{ij}^k(\lambda(x_1)_{ij}^k + (1-\lambda)(x_2)_{ij}^k\right) + \\
&\quad \sum_{(k,i,j)\in J_K} \left(\lambda(y_1)_{ij}^k + (1-\lambda)(y_2)_{ij}^k\right) \\
&\geq Z(\lambda\theta_1 + (1-\lambda)\theta_2)
\end{aligned}$$

□

Algorithm

Using the results of Theorems 6.1 and 6.2, we propose the following algorithm to find the optimal solution θ^* of Problem (6.7). The algorithm bisects an interval $[p, q]$ in which θ^* is to be found. The stopping criterion is the length of interval $[p, q]$, i.e. the algorithm terminates as soon as an interval of maximum length l_{STOP} is found, in which θ^* is sure to lie.

Instead of searching for the minimum of $Z(\theta)$ by a bisection method, as proposed in [14] and Algorithm 6.1, other search techniques like the more efficient golden section method could be applied.

Algorithm 6.1 Multicommodity Bertsimas and Sim Algorithm

```
1: Let the robust multicommodity minimum cost flow problem be given by the graph $G = (\mathcal{N}, \mathcal{A})$.
2: Let $\Gamma \in [0, \bar{\Gamma}]$ be the given robustness parameter, $\bar{\theta} := \max_{k \in K, (i,j) \in \mathcal{A}} u_{ij}^k d_{ij}^k$,
   $\bar{\Gamma} := |\{(k,i,j) : d_{ij}^k > 0\}| = |J_K|$ and $l_{STOP}$ the desired interval length.
3: Evaluate $Z(\theta)$ for $\theta = 0, \frac{1}{4}\bar{\theta}, \frac{1}{2}\bar{\theta}, \frac{3}{4}\bar{\theta}, \bar{\theta}$. Respecting the convexity of $Z$, determine $p$ and $q$ such that $\theta^* \in [p,q]$: $[p,q] = [0, \frac{1}{2}\bar{\theta}]$ or $[p,q] = [\frac{1}{4}\bar{\theta}, \frac{3}{4}\bar{\theta}]$ or $[p,q] = [\frac{1}{2}\bar{\theta}, \bar{\theta}]$.
4: Until $|q - p| \leq l_{STOP}$:
    1. Evaluate $Z(\frac{3}{4}p + \frac{1}{4}q)$ and $Z(\frac{1}{4}p + \frac{3}{4}q)$.
    2. Update the interval $[p,q]$: $[p,q] = [p, \frac{1}{2}p + \frac{1}{2}q]$ or $[p,q] = [\frac{3}{4}p + \frac{1}{4}q, \frac{1}{4}p + \frac{3}{4}q]$ or $[p,q] = [\frac{1}{2}p + \frac{1}{2}q, q]$ such that $\theta^* \in [p,q]$.
5: $\theta^*$ belongs to interval $[p,q]$. Set $\theta^* = \frac{1}{2}(p+q)$. STOP.
```

The network flow corresponding to θ^* is the solution to problem (6.8), which is determined by the evaluation of Z. Z can be evaluated by solving a multicommodity MCFP.

Multicommodity Uncertain Cost Example

In the following, we consider a 2-commodity minimum cost flow problem with uncertain cost consisting of 10 nodes and 17 arcs as illustrated below.

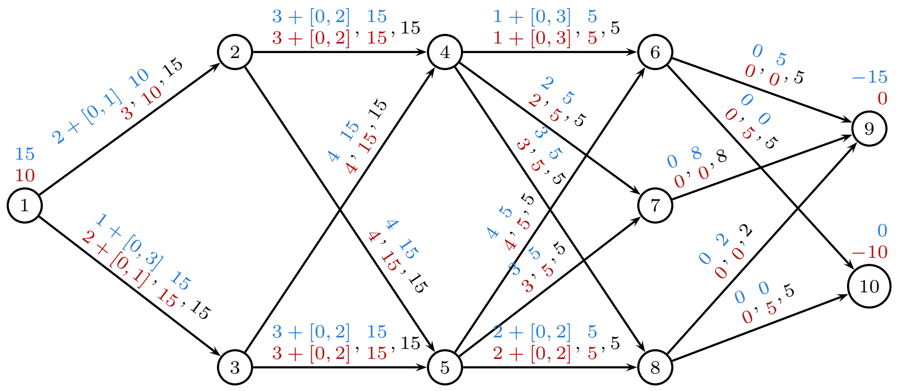
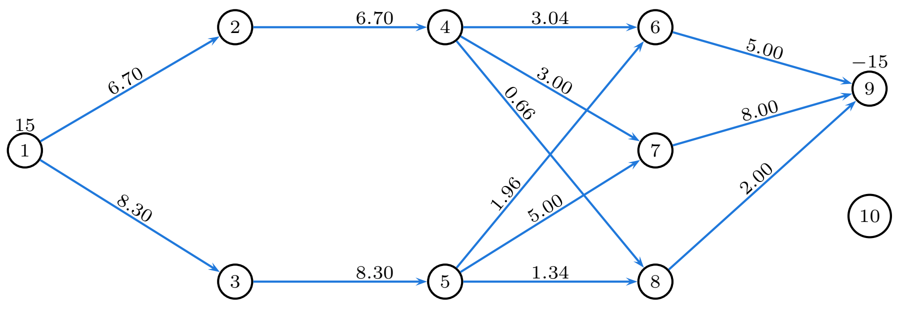


Figure 6.2: Multicommodity uncertain cost example: Multicommodity minimum cost flow problem

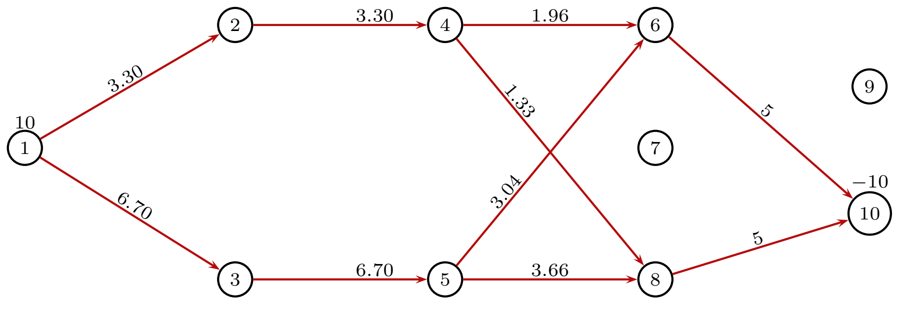
The numbers on the arcs indicate the cost, the capacity and the bundle constraint. The upper line denotes commodity 1, the lower line commodity 2. The cost values are written in the form $c + [0, d]$, which means that the nominal cost is c and the possible extra cost is at most d. Note that some arcs, e.g. arcs $(2, 4)$ and $(4, 6)$ are relatively cheap in the nominal cost c_{ij}^k but have a high potential extra cost d_{ij}^k.

In our example we have $\bar{\Gamma} = |\{(k, i, j) : d_{ij}^k > 0\}| = 11$.

The robust problem with $\Gamma = 0$ is equivalent to the original 2-commodity minimum cost flow problem, as $\Gamma = 0$ means that no cost variation is considered. For $\Gamma = 0$, the following solution is obtained:



(a) Commodity 1

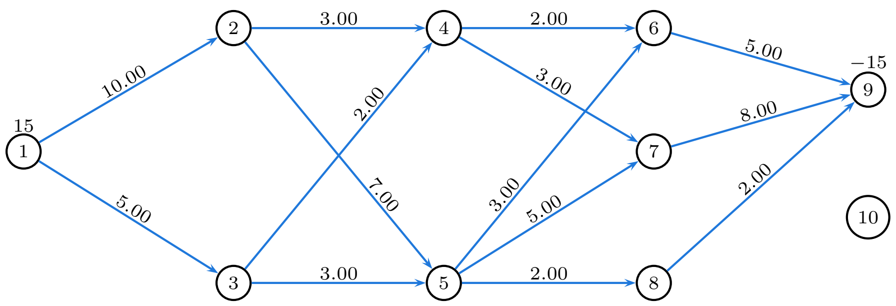


(b) Commodity 2

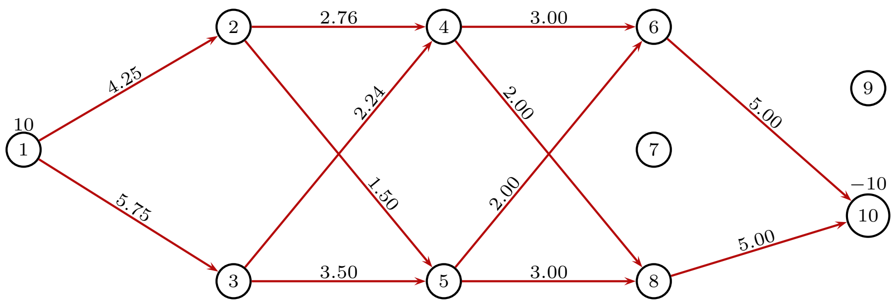
Figure 6.3: Multicommodity uncertain cost example: Solution for $\Gamma = 0$

For $\Gamma = 0$, we obtain a best case cost of 167 (i.e. for all arcs $(i, j) \in \mathcal{A}$ and all commodities k, the minimum cost c_{ij}^k is considered as actual cost) and a worst case cost of 280 (i.e. for all arcs $(i, j) \in \mathcal{A}$ and all commodities k, the maximum cost $c_{ij}^k + d_{ij}^k$ is considered as actual cost).

Now we consider a medium robust value for Γ, i.e. $\Gamma = 6$. This means that we consider the scenarios where at most 6 cost values c_{ij}^k vary in $[c_{ij}^k, c_{ij}^k + d_{ij}^k]$. We solve the robust multicommodity minimum cost flow problem by applying the algorithm given above and we obtain the following solution.



(a) Commodity 1

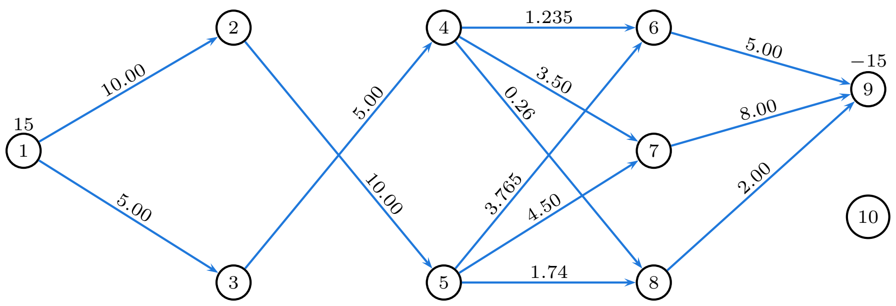


(b) Commodity 2

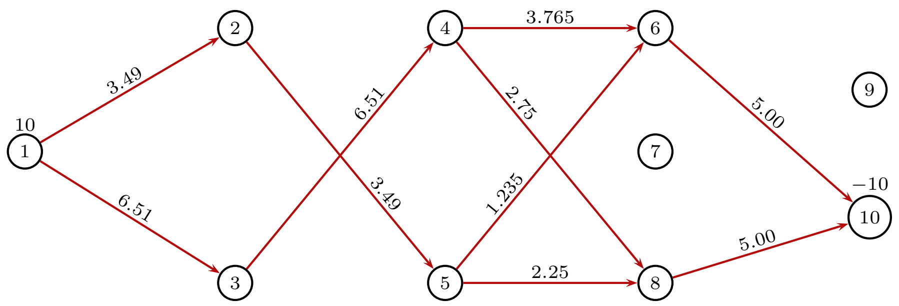
Figure 6.4: Multicommodity uncertain cost example: Robust solution for $\Gamma = 6$

For $\Gamma = 6$, we obtain a best case cost of 184 and a worst case cost of 264. We can see that potentially expensive arcs like arcs $(2, 4)$ and $(4, 6)$ are used less than for $\Gamma = 0$, whereas arcs without potential extra cost like arcs $(2, 5)$ and $(3, 4)$, which hadn't been used for $\Gamma = 0$, now have a real positive flow in the optimal solution.

Now we consider the maximum value $\Gamma = \bar{\Gamma} = 11$. That means we consider the robust problem with cost variation for all arc/commodity combinations with uncertain costs.



(a) Commodity 1



(b) Commodity 2

Figure 6.5: Multicommodity uncertain cost example: Robust solution for $\Gamma = 11$

For $\Gamma = 11$, we obtain a solution with a best case cost of 197.5 and a worst case cost of 252. Considering the arcs $(2,4)$,$(2,5)$,$(3,4)$ and $(3,5)$, we can clearly see that for the most conservative choice of Γ, arcs with a potential high costs are avoided (arcs $(2,4)$ and $(3,5)$) whereas arcs with a higher nominal cost but no potential extra cost are preferred (arcs $(2,5)$ and $(3,4)$).

The following table shows the best case and worst case cost for the solutions for different values of Γ.

Γ	best case	worst case
0	167.00	280.30
1	167.00	278.00
2	174.76	272.29
3	177.00	270.36
4	177.60	269.78
5	180.57	267.27
6	183.99	264.26
7	186.18	262.91
8	187.84	261.23
9	195.84	253.41
10	197.51	252.00
11	197.51	252.00

Table 6.1: Multicommodity uncertain cost example: Best case and worst case costs

As we would expect, for rising Γ the best case cost increases while the worst case cost decreases, as higher values of Γ enforce more conservative solutions.

Coupled Uncertainty

In practice, there are scenarios where data uncertainty is not stochastically independent. Consider for example a transportation network, where some arcs represent transportation on the road and the remaining arcs transportation on the rail. Let the data uncertainty for the road arcs be given by the fluctuations of the oil price. Then it is likely that for rising oil prices, the actual price for transportation on the road will be higher for all arcs representing transportation on the road.

A possibility for coupled uncertainty in multicommodity flow problems could be the following: If the cost on one arc is subject to uncertainty, then all commodities are effected simultaneously. Consider again a transportation network. Let arc (i, j) represent transportation on the road by a certain transportation company from i to j. If the transportation company increases their prices for transportation, the price for transportation will be higher for all commodities simultaneously.

One solution to this problem for a problem with coupled uncertainty of the type mentioned secondly is to choose only values of Γ which are a multiple of p, where p is the number of commodities. Remember that according to Bertsimas and Sim, Γ represents the maximum number of arcs where uncertainty is actually considered, where each arc is multiply counted, once for each commodity having an uncertain cost value on this arc. If Γ is a multiple of p, it cannot happen that for one specific arc (i, j) the uncertainty is considered for less than all p commodities: In Problem (6.2), the maximum over <u>all</u> contemplable 3-tuples (k, i, j) is considered, which means that the 3-tuples $(1, i, j)$, $(2, i, j)$, ..., (p, i, j) are comprised in every case, if Γ is a multiple of p.

6.2.3 Non-Integer Values for the Robustness Parameter

In Subsections 6.2.1 and 6.2.2, we required the robustness parameter Γ to be integral.

In [14], Bertsimas and Sim propose a general robust approach for arbitrary mixed integer programming problems, which is the basis for the robust approach for network flow problems, which we presented in Subsection 6.2.1. In the general robust approach, Bertsimas and Sim allow Γ to take non-integer values. We will shortly summarize the explanations given in [14], adapted to the minimum cost flow problem:

As defined in Subsection 6.2.1, let $J := \{(i, j) : d_{ij} > 0\}$. Then $|J|$ is the maximum number of arcs with possible cost fluctuations. Now let the robustness parameter Γ be an arbitrary, not necessarily integral, value in $[0, |J|]$. Then a non-integral value Γ has the following meaning: Up to $\lfloor \Gamma \rfloor$ of the possibly fluctuating cost coefficients are allowed to take at most their worst case value

$c_{ij} + d_{ij}$. In addition, one cost coefficient is allowed to take at most the value $c_{ij} + (\Gamma - \lfloor \Gamma \rfloor) d_{ij}$, i.e. this value may fluctuate by $(\Gamma - \lfloor \Gamma \rfloor) d_{ij}$.

As the robust approach for network flow problems, as presented in [14], is a special case of the general robust approach for arbitrary mixed integer programming problems, the non-integer value concept for the robustness parameter is directly applicable for the robust approach for network flow problems in Subsection 6.2.1.

The extension of the non-integer value concept for the robustness parameter to multicommodity flow problems, as presented in Subsection 6.2.2, can be realized analogously.

6.3 Ant Algorithm

The algorithm of Bertsimas and Sim for single-commodity flows and the extension to multicommodity flows is a suitable approach for minimum cost flow problems with uncertain costs. However, for large networks, the computational effort is considerably large, most notably in the multicommodity case, as a series of deterministic minimum cost flow problems must be solved.

In this section, we present an extension of the ant algorithm, which we proposed in Chapter 4, to uncertain costs. As the ant algorithm is random-based and therefore contains stochastic components, the introduction of uncertain cost data into the algorithm is straightforward. We will especially focus on uniformly distributed costs and Gaussian distributed costs, but we will also consider arbitrary distributions.

The extension is applicable to both the single-commodity and the multicommodity ant algorithm.

6.3.1 Uniformly Distributed Costs

Let c_{ij}^k be the nominal cost on arc (i,j) for commodity k and $d_{ij}^k \geq 0$ the possible extra cost. Now we assume, that all possible cost values in $[c_{ij}^k, c_{ij}^k + d_{ij}^k]$ are equally likely, i.e. we have a uniform distribution of the costs in $[c_{ij}^k, c_{ij}^k + d_{ij}^k]$.

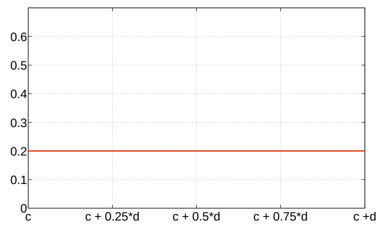


Figure 6.6: Density function for uniformly distributed costs in $[c, c+d]$ for nominal cost $c = 3$ and extra cost $d = 5$

The basic idea is the following: For every arc and every commodity, we have an interval $[c^k_{ij}, c^k_{ij} + d^k_{ij}]$ of possible cost values. At the beginning of each iteration, we generate a random cost value $\hat{c}^k_{ij}$ out of this interval and set the current arc cost for arc (i, j) and commodity k for this iteration to $\hat{c}^k_{ij}$.

First, we consider an arc (i, j) with low nominal cost c^k_{ij} and high extra cost d^k_{ij}. With growing number of iterations, the ants are confronted with different arc costs on the same arc. The visibility information makes the same arc rather attractive in some iterations, if a low cost value is generated, or less attractive, if a high cost value is generated. Consequently, an arc with high extra costs d^k_{ij} will be chosen by more ants in some iterations and less ants in other iterations and therefore will receive more or less pheromone.

By contrast, an arc (i', j') with a nominal cost $c^k_{i'j'}$ that is slightly higher than c^k_{ij} and with extra cost $d^k_{i'j'} = 0$ has a constant visibility over all iterations, which is only slightly lower than the best case visibility of arc (i, j) but much higher than the worst case visibility of arc (i, j). This arc is more attractive than arc (i, j) in average and will therefore receive more pheromone at long sight.

With an increasing number of iterations, the ants are confronted with a variety of different cost values on the uncertain arcs and therefore the pheromone levels on the arcs will level off in the sense that arcs that are a good choice for the majority of uncertain cost values will have a higher pheromone level, arcs that are a rather bad choice will have a lower pheromone level. An appropriate choice of the pheromone evaporation rate is important: If the pheromone evaporation is too low, arcs that were a good choice in early iterations might settle in all constructed solutions even though they aren't a good choice in general. If the evaporation rate is too high, valuable collected information might be lost too soon.

6.3.2 Gaussian Distributed Costs

Again, we consider an interval $[c_{ij}^k, c_{ij}^k + d_{ij}^k]$ of possible arc costs for arc (i, j) and commodity k.

We introduce a robustness parameter $\Gamma \in [0, 1]$ in order to control the level of robustness of the ant solution. We define that for $\Gamma = 0$ the solution shall be minimally robust and for $\Gamma = 1$ the solution shall be maximally robust. When increasing Γ from 0 to 1, the solution shall become more and more robust.

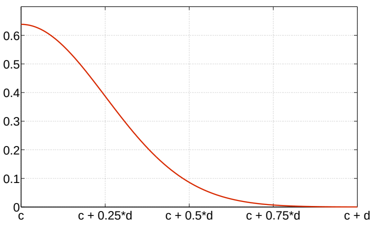
As proposed in Subsection 6.3.1, we generate a random cost value in $[c_{ij}^k, c_{ij}^k + d_{ij}^k]$ for arc (i, j) and commodity k at the beginning of each iteration.

However, as we want to respect the robustness parameter Γ, we can not use a uniform distribution as before. Instead, we generate cost values that have a Gaussian distribution.

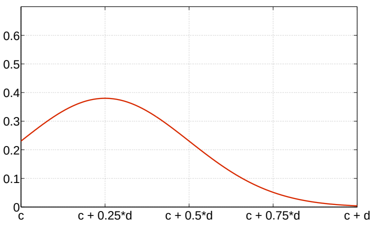
The expected value of the Gaussian distribution is set to $\mu_{ij}^k = c_{ij}^k + \Gamma d_{ij}^k$. The standard deviation is set to $\sigma_{ij}^k = \frac{1}{4} d_{ij}^k$. Furthermore, the distribution is truncated at the interval boundaries c_{ij}^k and $c_{ij}^k + d_{ij}^k$.

The expected value μ has a direct impact on the generated cost values: For a low value of Γ, the expected generated cost values are near the lower interval boundary c_{ij}^k, for a high value of Γ they are more likely to be near the upper interval boundary $c_{ij}^k + d_{ij}^k$.

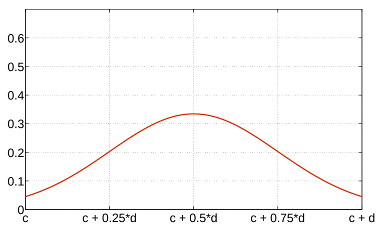
The choice of σ is based on the fact that approximately 95.45% of all values have a deviation less or equal to 2σ to the mean value.



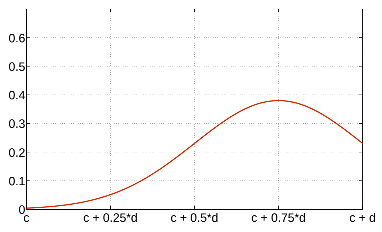
(a) $\Gamma = 0$



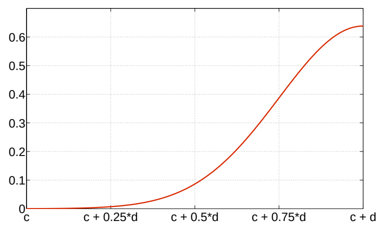
(b) $\Gamma = 0.25$



(c) $\Gamma = 0.5$



(d) $\Gamma = 0.75$



(e) $\Gamma = 1$

Figure 6.7: Density functions for Gaussian distributed costs in $[c, c + d]$ for nominal cost $c = 3$ and extra cost $d = 5$ for $\Gamma = 0,\ 0.25,\ 0.5,\ 0.75$ and 1

In practice, we generate a cost value with the Gaussian distribution $N(\mu_{ij}^k, {\sigma_{ij}^k}^2)$ without truncation. If the generated value lies outside the interval $[c_{ij}^k, c_{ij}^k + d_{ij}^k]$, we generate a new cost value and continue this procedure until we generate a cost value that lies inside the interval.

6.3.3 Arbitrary Distribution

The ant algorithm can easily be extended from the Gaussian distribution to an arbitrary distribution. If the distribution of the uncertain input data is known, we can generate the cost values according to this distribution in each iteration. The approach of Bertsimas and Sim, on the contrary, can not use the additional information about the distribution of the uncertain input data even if it is known.

Note that a fixed given distribution for the ant algorithm means that we have no longer a robustness parameter, as the robustness parameter was introduced in order to shift the expected value of the Gaussian distribution for the generation of less or more expensive cost values in case that no information about the distribution is given.

6.4 Problems

In this section, we examine two example problems. We apply the presented methods and compare the results.

6.4.1 Single-Commodity Uncertain Cost Example

In this section, we apply the proposed robust ant algorithms to an example NFP and compare the results to the robust approach of Bertsimas and Sim.

We consider the minimum cost flow problem defined by the following digraph:

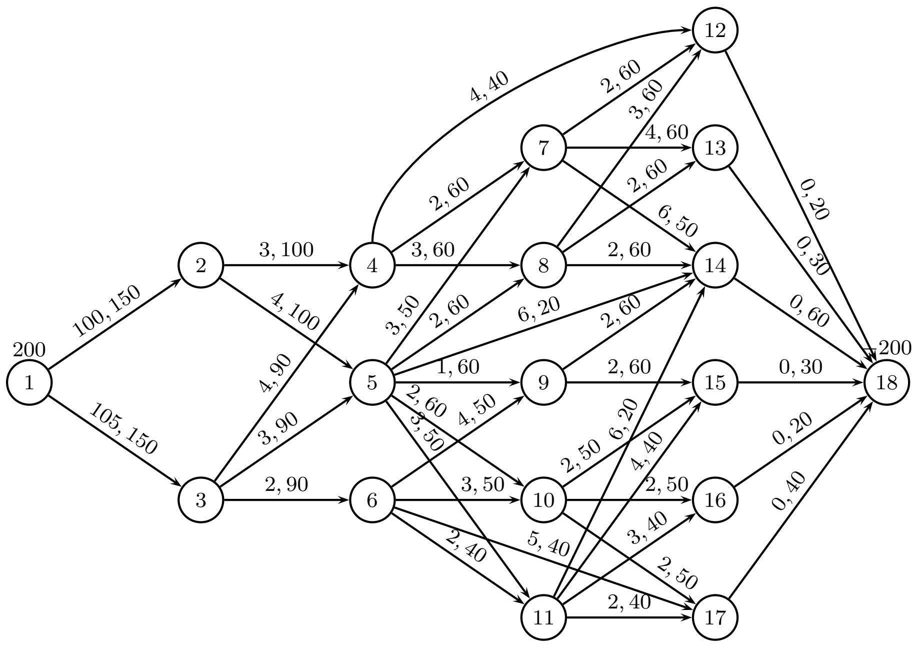


Figure 6.8: Single-commodity uncertain cost example: Minimum cost flow problem

Note that this network does not exactly correspond to the definition of a layered network (see Definition 3.1), as we have three arcs going from layer 3 directly to layer 5. In practice, this effect can occur for example if in two successive distribution stages intermediate storage can be omitted. Note that by the insertion of new dummy nodes in layer 4 and splitting each of the concerned arcs into two arcs from layer 3 to layer 4 and from layer 4 to layer 5, the network can easily be transformed into a layered network.

The numbering of the arcs is lexicographically ordered.

The minimum cost flow problem has the optimal solution $x = (150, 50, 80, 70, 0, 0, 50, 20, 60, 0, 0, 0, 60, 10, 0, 0, 0, 10, 40, 0, 20, 0, 0, 0, 20, 30, 30, 30, 0, 20, 0, 20, 30, 60, 30, 20, 40)^\top$.

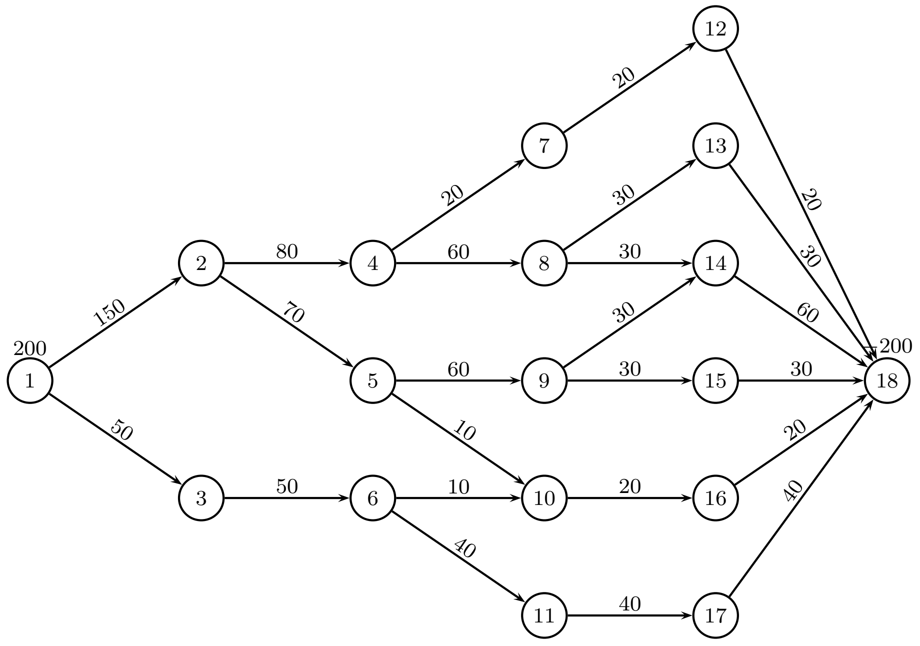


Figure 6.9: Single-commodity uncertain cost example: Optimal solution of the minimum cost flow problem

Now we introduce uncertain costs on arcs $(2, 4)$, $(2, 5)$ and $(3, 6)$. We add a possible extra cost of 10 per arc.

Note that the robustness parameter Γ in the algorithm of Bertsimas and Sim is not fully consistent with the robustness parameter Γ in the ant algorithm with Gaussian distributed costs. Both algorithms have a minimal robustness parameter of 0. The maximum value of the robustness parameter of the algorithm of Bertsimas and Sim equates the number of arcs with uncertain data and is therefore equal to 3. The maximum value of the robustness parameter of the ant algorithm is fixed to 1. With fixed minimum and maximum values for the robustness parameter, we can obtain some corresponding values for Γ for the example network with three arcs with uncertain input data by dividing the robustness parameter interval into four equal parts. The relations are given in the following table:

Γ in ant algorithm	Γ in Bertsimas and Sim
0	0
0.25	0.75
0.5	1.5
0.75	2.25
1.0	3

Table 6.2: Corresponding values for Γ in Bertsimas and Sim and the ant algorithm

Note that we no longer require Γ to be integral for the algorithm of Bertsimas and Sim, see Subsection 6.2.3 for details.

The algorithm of Bertsimas and Sim gives the following results for different values of Γ.

Γ	best case	worst case
0	21680	23638
0.75	21680	23651
1.5	22142	22938
2.25	22165	22915
3	22240	22740

Table 6.3: Single-commodity uncertain cost example: Robust solutions (Bertsimas and Sim)

Now we examine the single-commodity uncertain cost example using the ant algorithm. As the ant algorithm is a random-based metaheuristic, we cannot simply state the result of one run of the ant algorithm, but have to consider multiple runs. As a "result" of the ant algorithm, we can consider a "typical" run, if most runs result in the same solution, or the average of all runs and the maximum deviation, if the runs differ notably.

First, we consider uniformly distributed costs as proposed in Subsection 6.3.1. The results given below are the results of an average run.

best case	worst case
21921.5	23256.5

Table 6.4: Single-commodity uncertain cost example: Robust solution (ant algorithm - uniformly distributed costs, average run)

In our test series, we had 20 runs of the ant algorithm. The maximum deviations were -0.64% and $+0.86\%$ for the best case cost and -1.01% and $+1.21\%$ for the worst case cost.

When comparing the results of the ant algorithm to the results of Bertsimas and Sim, we can see that the solution of the ant algorithm is a little less robust than the solution of Bertsimas and Sim for $\Gamma = 1.5$ but is more robust than the solution for $\Gamma = 0.75$. We use $\Gamma = 1.5$ as reference value as this is the average value of all possible values for $0 \leq \Gamma \leq 3$ and therefore corresponds best to the uniformly distributed costs in the ant algorithm.

Furthermore, we examine the application of the ant algorithm with Gaussian distributed costs, as presented in Subsection 6.3.2, to the multicommodity uncertain cost example. Again, we consider the results of an average run per value of Γ.

Γ	best case	worst case	deviation (bc)		deviation (wc)	
0	21721	23681	-0.05%	$+0.18\%$	-0.68%	$+0.33\%$
0.25	21818	23148	-0.31%	$+0.28\%$	-0.73%	$+0.61\%$
0.5	21875	22995	-0.16%	$+0.16\%$	-0.07%	$+0.20\%$
0.75	22300	22810	-0.40%	$+0.04\%$	-0.00%	$+0.00\%$
1	22310	22810	-0.00%	$+0.00\%$	-0.00%	$+0.00\%$

Table 6.5: Single-commodity uncertain cost example: Robust solutions (ant algorithm - Gaussian distributed costs, average runs)

In our test series, we had 20 runs of the ant algorithm per value of Γ.

The results can be directly compared to the results of the algorithm of Bertsimas and Sim considering Table 6.2. Furthermore, we compare the expected cost values for the optimal solutions for different values of Γ, both for the algorithm of Bertsimas and Sim and the ant algorithm. We consider the Γ values of the ant algorithm, where $\Gamma \in [0, 1]$. Then for a given Γ, we expect the actual cost value on each arc $(i, j) \in \mathcal{A}$ to be $(1 - \Gamma)c_{ij} + \Gamma d_{ij}$. Consequently, for a given value of $\Gamma \in [0, 1]$, the expected cost value of a solution x^* can be calculated as

$$\begin{aligned} ((1-\Gamma)c + \Gamma d)^\top x^* &= (1-\Gamma)c^\top x^* + \Gamma d^\top x^* \qquad (6.11) \\ &= (1-\Gamma) \cdot (\text{best case cost}) + \Gamma \cdot (\text{worst case cost}). \end{aligned}$$

We obtain the following expected cost values:

Γ	Bertsimas & Sim	ant algorithm
0	21680	21721
0.25	22173	22151
0.5	22540	22435
0.75	22728	22683
1	22740	22810

Table 6.6: Single-commodity uncertain cost example: Expected cost values for the optimal solutions for different values of Γ: Bertsimas/Sim and ant algorithm (Gaussian distributed costs)

The following figure illustrates the best case and worst case costs of both algorithms and compares the expected values for the optimal solutions for different values of Γ.

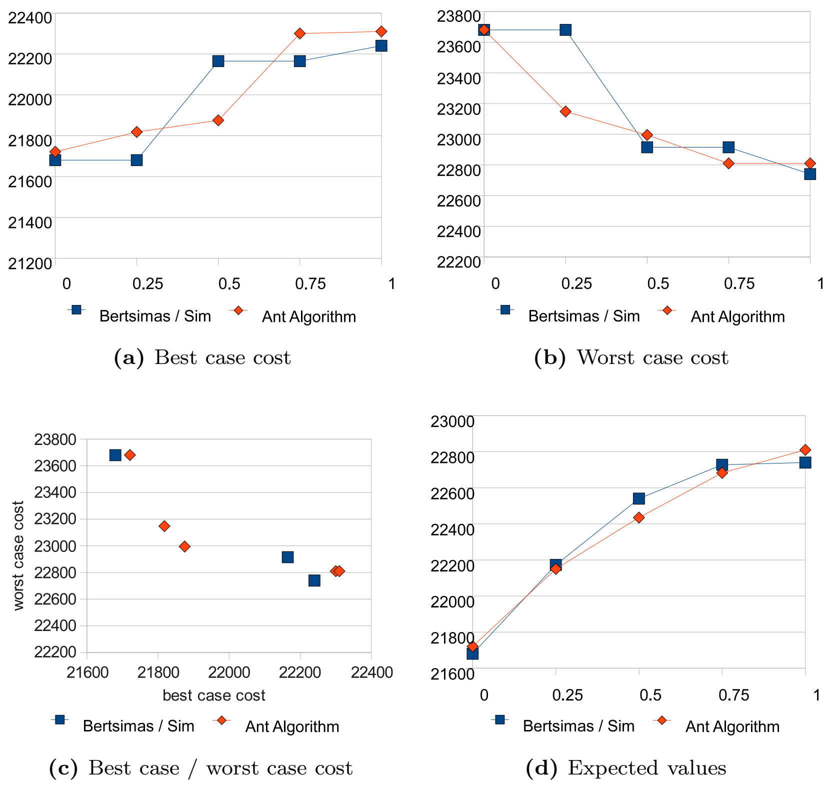


Figure 6.10: Single-commodity uncertain cost example: Comparison of best case and worst case costs and expected cost values for different values of Γ: Bertsimas/Sim and ant algorithm (Gaussian distributed costs)

Note that in our figures, we use the Γ values which correspond to the ant algorithm.

6.4.2 Single-Commodity Triangular Distributed Cost Example

We consider again the network of the single-commodity uncertain cost example, where the cost values of arcs $(2,4)$, $(2,5)$ and $(3,6)$ shall be subject to uncertainty with a possible extra cost of 10 per arc. This time, we consider a triangular distribution of the cost values on these arcs. The following table lists the minimum, maximum and most probable cost values on these arcs.

arc	minimum cost value	most probable cost value	maximum cost value
$(2,4)$	3	5	13
$(2,5)$	4	12	14
$(3,6)$	2	4	12

Table 6.7: Single-commodity triangular distributed cost example: Triangular distribution of the cost values

The algorithm of Bertsimas and Sim only considers cost data given in the form of an interval $[c_{ij}, c_{ij} + d_{ij}]$. Hence, given distributions of the cost values within the interval bounds are not used by the algorithm. However, even though the algorithm of Bertsimas and Sim a priori is not perfectly suitable for solving problems with given distributions, we want to test the algorithm of Bertsimas and Sim on the triangular distribution and compare the results to the results of the ant algorithm.

In our example, arcs $(2,4)$, $(2,5)$ and $(3,6)$ all have a possible extra cost of 10. For arcs $(2,4)$ and $(3,6)$, however, the most probable cost value is much lower than for arc $(2,5)$. The following figure illustrates the flow between node layers 2 and 3 (subgraph given by nodes 2 to 6) of the example network for the optimal solution of the algorithm of Bertsimas and Sim for different values of Γ.

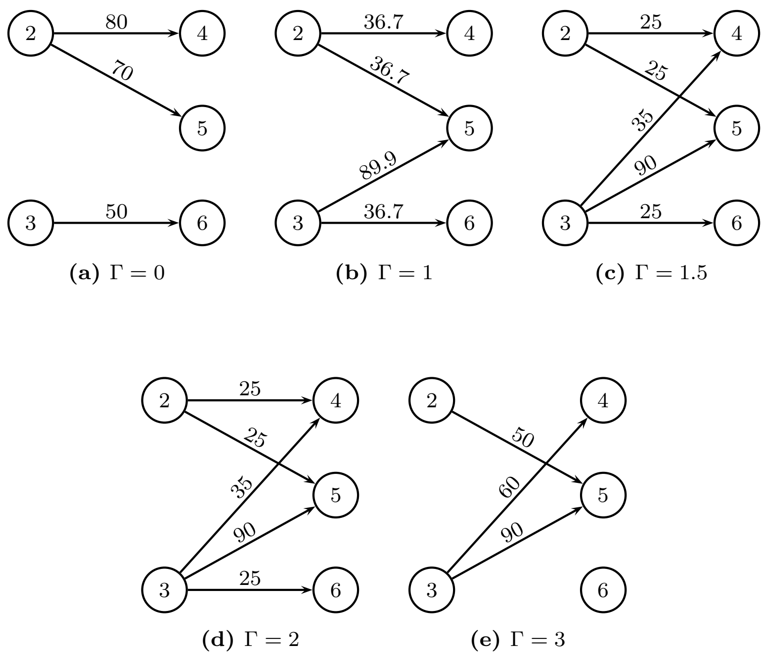


Figure 6.11: Single-commodity triangular distributed cost example: Flows between node layers 2 and 3 for Bertsimas and Sim

As we would expect, the algorithm of Bertsimas and Sim does not respect the fact that the most probable cost value for arc $(2, 5)$ is much higher than the one for arcs $(2, 4)$ and $(3, 6)$. Therefore, even in the most robust case ($\Gamma = 3$), 50 flow units are sent on arc $(2, 5)$ and none on arcs $(2, 4)$ and $(3, 6)$.

Now we modify the ant algorithm such that it generates cost data that correspond to the triangular distribution for arcs $(2, 4)$, $(2, 5)$ and $(3, 6)$. This results in the following flow between node layers 2 and 3 of the example network:

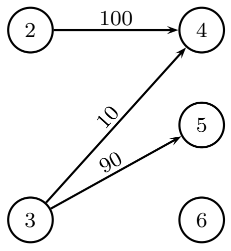


Figure 6.12: Single-commodity triangular distributed cost example: Flows between node layers 2 and 3 for the ant algorithm

We can see that no flow units are sent on the possibly most expensive arc $(2, 5)$.

The following tables list the best case cost, most probable case cost, expected case cost and worst case cost for the algorithms of Bertsimas and Sim and the ant algorithm. Note that the expected cost for arc $(2,4)$ is 7, for arc $(2,5)$ is 10 and for arc $(3,6)$ is 6.

Γ	best case	most probable case	expected case	worst case
0	21680	22500	22620	23680
1	21990	22430	22679	23090
1.5	22165	22290	22515	22915
2	22165	22290	22515	22915
3	22240	22640	22540	22740

(a) Bertsimas and Sim

best case	most probable case	expected case	worst case
22000	22200	22400	23000

(b) Ant algorithm

Table 6.8: Single-commodity triangular distributed cost example: Comparison of the results of Bertsimas and Sim and the ant algorithm

For all values of Γ, the algorithm of Bertsimas and Sim produces a solution that has a most probable cost which is higher than for the ant algorithm. If we compare the solution of Bertsimas and Sim for $\Gamma = 1.5$, which has the lowest most probable cost, to the solution of the ant algorithm, we can see that both best case and most probable case cost are higher for Bertsimas and Sim, while the worst case cost is slightly lower. For $\Gamma = 3$, the most robust case of Bertsimas and Sim, the most probable case cost is significantly higher than for the ant algorithm.

6.4.3 Summary

It can be clearly seen that the ant algorithm is more flexible and adaptable than the algorithm of Bertsimas and Sim, if more information about the distribution of the uncertain cost values is known. This is due to the fact that the algorithm of Bertsimas and Sim only considers the uncertainty interval data, but ignores information about distributions, if given.

An important advantage of the ant algorithm in comparison to the algorithm of Bertsimas and Sim is the computational time for network flow problems with a large number of arcs having uncertain input data. Considering a supply chain network with a fixed number of arcs, we first examine the problem for a small number of arcs actually having uncertain input data and then the same network with a large number of arcs with uncertain input data. While the running time of the algorithm of Bertsimas and Sim increases immensely, it remains almost constant for the ant algorithm: The number of arcs having uncertain input data influences the ant algorithm only in the way that at the beginning of each iteration, a specific cost value has to be generated for every uncertain arc, which is not computationally intensive.

However, the fact that the ant algorithm does not guarantee to find a global optimal solution has to be kept in mind.

6.5 Summary and Conclusions

In this chapter, network flow problems with uncertain cost values on the arcs were examined. For every arc subject to uncertainty, a nominal cost value and a maximum extra cost value were considered. Both an exact and a heuristic solution to this kind of problem were developed:

In Section 6.2, the Robust Optimization based method for network flow problems with uncertain costs, which was proposed by D. Bertsimas and M. Sim in [14], was extended from single-commodity to multicommodity problems.

In Section 6.3, an ACO based heuristic method for network flow problems with uncertain costs is presented. The introduction of uncertain cost data is in line with the random-based stochastic components of the ant algorithm in the solution construction process. The presented method is suitable for both single-commodity and multicommodity problems.

For both approaches, the level of robustness can be controlled via a robustness parameter Γ.

The presented approaches were applied to example problems in 6.4. A main difference of the two approaches is the fact that the algorithm of Bertsimas and Sim does not consider the distribution of the uncertainty values, while the ant algorithm can easily be adapted to diverse probability distributions. Concerning the computational time, the ant algorithm was superior to the algorithm of Bertsimas and Sim in our test runs. However, it has to be kept in mind that the ant algorithm does not guarantee to find a global optimal solution.

Chapter 7

Transformation of Uncertain Demands to Uncertain Costs

In the previous chapter, we presented algorithms for supply chain problems with uncertain cost values. However, there is another important type of uncertainty in supply chains: Uncertainty in demand.

In the context of supply chain management, the demand is situated at the end of the supply chain in the form of a warehouse or a customer, for example. We assume wlog that the demand arises at a warehouse and will therefore use the term “warehouse” as a synonym for a demand node in the following.

If the demand in a supply chain is subject to uncertainty, we want to find a production and transportation strategy which can cope best with the uncertain demand values - that means that we have to find a compromise between a full safety stock production and the risk of deficiency at the warehouses. As deficiency can be “worse” - i.e. more expensive - at some warehouses, we have to decide which warehouses should be delivered with extra product units.

In this chapter, we describe the transformation of uncertain demands to uncertain costs in a given network $G = (\mathcal{N}, \mathcal{A})$. Our goal is to apply the algorithms for supply chain problems with uncertain costs, which we presented in Chapter 6, to supply chain problems with uncertain demands.

We assume in the following that G is a layered network with positive costs only, where all demand nodes (warehouses) are situated in the last layer. For the definition of the term *layered network*, see Section 4.2.

7.1 Transformation using Additional Information: Penalty and Storage Costs

This section describes the transformation of uncertain demands to uncertain costs using additional information. The additional information is given in the form of penalty costs in case of deficiency and storage costs for surplus delivery.

Let $G = (\mathcal{N}, \mathcal{A})$ be a given single-source supply chain network with source node s. Let i be a demand node with nominal demand b_i. Now we consider uncertain demand in node i. Let z_i be the possible extra demand in node i.

$$\begin{array}{ccccc} & \nearrow & \cdots & \searrow & \\ \Rightarrow \textcircled{s} & \longrightarrow & \cdots & \longrightarrow & \textcircled{i} \Rightarrow [|b_i|, |b_i| + |z_i|] \\ & \searrow & \cdots & \nearrow & \end{array}$$

Figure 7.1: Network with uncertain demand at node i

In a fixed time-period, a warehouse is supplied with a specific amount of goods. If not all units of this delivery can be sold, they have to be stored at the warehouse. The warehouse charges a storage cost of $c^{storage}$ per unit to be stored. However, if the current demand at the warehouse is higher than the amount of goods which were supplied, the warehouse charges a penalty cost of $c^{penalty}$ per unavailable unit.

7.1.1 The Transformation Process

The additional information given by penalty and storage costs can be used when transforming uncertain demands to uncertain costs.

Let node i be a demand node. First we replace the original node i by new nodes i,, i' and j and three new arcs:

$$\begin{tikzcd} i \arrow[rr] \arrow[dr] & & j \\ & i' \arrow[ur] & \end{tikzcd}$$

Figure 7.2: Network extension near node i

The demand at node i is transferred to node j and set to the full demand $b_i + z_i$. The demand at node i is consequently set to 0.

The arc (i, j) represents the path of the flow of the basic demand b_i. As we assume that the basic demand is always requested, the cost c_{ij} on arc (i, j) is set to 0, that means that no additional costs occur.

The arcs (i, i') and (i', j) represent the path of the flow of the additional demand z_i. The cost of the path (i, i', j) is set to $c^{storage}$ in order to represent the storage of additional units.

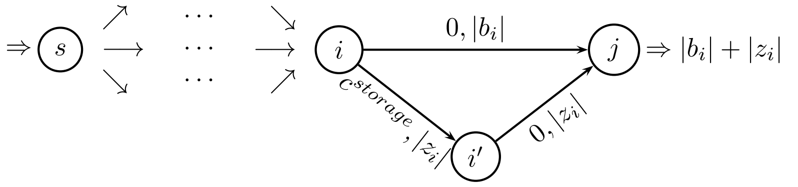


Figure 7.3: Network extension near node i with arc data

Now we introduce an additional arc from the source node s to node j with capacity $|z_i|$. This arc is introduced with the intent to represent the path of flow units which do not pass the production and delivery cycle and are therefore not delivered to the demand nodes and consequently evoke penalty costs, if the demand at the demand node is not satisfied.

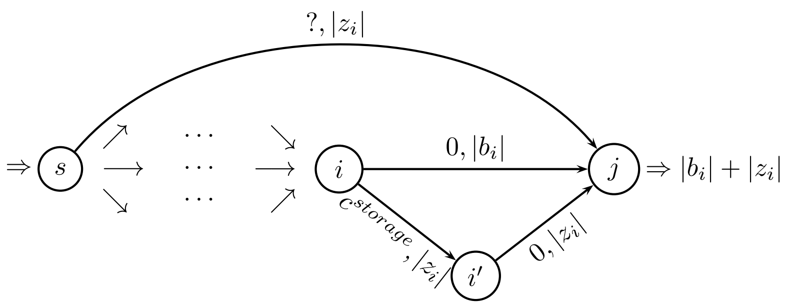


Figure 7.4: Transformed network with additional arc from s to j

We set the nominal costs of the additional arc (s, j) to $-c^{storage}$. This effects that the higher the storage cost, the more attractive is the additional arc.

Moreover, we want to embed both penalty costs and uncertainty in the demand in the cost value of the additional arc (s, j).

According to the construction of our extended network, in an optimal solution the basic demand b_i is always sent through the network via node i and arc (i, j). The costs for this flow are equal to the costs in the original network, as the cost on arc (i, j) is 0. In the extended network, it is possible to send the extra demand z_i either "through" the network via the nodes i and i' or "around" the network via arc (s, j).

We want to treat the uncertain demand in the following way: In the most "optimistic" case, we assume that the current demand at the demand node i is the minimum (basic) demand b_i. Therefore in this case, we want all extra units of flow to use the additional arc (s, j). In the most "pessimistic" case, we assume that the current demand at the demand node i is the maximum (basic plus extra) demand $b_i + z_i$. In this case, each unit of flow which is sent via the additional arc (s, j) evokes penalty costs in the amount of $c^{penalty}$.

In order to model this situation, we introduce uncertain costs on arc (s, j). As stated above, the nominal cost on arc (s, j) is $-c^{storage}$. Now we add extra costs lying in the interval $[0, c^{storage} + c^{penalty}]$. Hence we have negative costs in the amount of $-c^{storage}$ in the most optimistic case and positive costs in the amount of $c^{penalty}$ in the most pessimistic case.

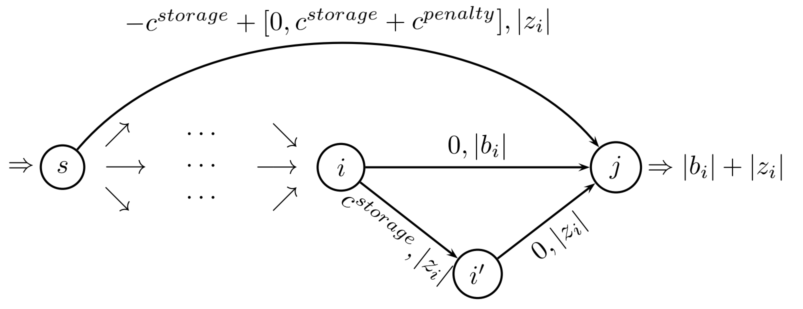

$-c^{storage} + [0, c^{storage} + c^{penalty}], |z_i|$
s
i
j
i'
$0, |b_i|$
$c^{storage}, |z_i|$
$0, |z_i|$
$\Rightarrow |b_i| + |z_i|$

Figure 7.5: Transformed network with uncertain costs

Finally, the whole transformation process has transformed a network with uncertain demand data into a network with uncertain cost data.

Note that an alternative model for the cost values on the additional arcs is imaginable: The nominal costs on the additional arcs are set to 0, the extra costs to $c^{penalty}$. For this model, it likewise holds that the higher the transportation cost through the network and the higher the storage cost, the more attractive is the additional arc. In this alternative cost model, the storage costs only effect the cost values on arc (i, i'), while in the cost value setting proposed in Figure 7.5, the storage costs concern both arc (s, j) and arc (i, i'). Therefore, we propose to select the alternative cost model for problems where the influence of the storage costs is considered less essential and where the main focus is on the penalty costs.

7.1.2 Uncertain Demands Transformation Example 1

We consider a minimum cost flow problem with uncertain demands. The given network corresponds to a supply chain planning problem, where uncertain demands arise at the nodes corresponding to warehouses.

The supply chain network consists of five node layers: The first node layer (node 1) is the source layer, the second node layer (nodes 2 to 4) is the production layer, the third node layer (nodes 5 to 8) is the layer of supra regional warehouses, the fourth node layer (nodes 9 to 13) is the layer of regional warehouses and the fifth node layer (nodes 14 to 18) is the customer node layer, i.e. the demand nodes. The demand intervals are given at the right hand side of the demand nodes.

In our example, the warehouse nodes are nodes 14 to 18. The parameters at the demand nodes specify the uncertainty intervals of the demands.

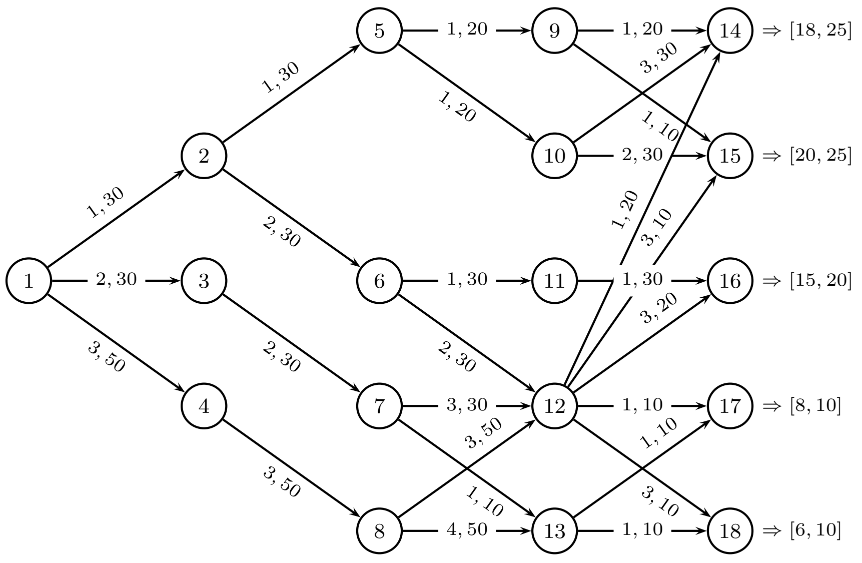


Figure 7.6: Uncertain demands transformation example 1: Minimum cost flow problem with uncertain demands

The following table lists the penalty and storage costs at the warehouse nodes.

node	14	15	16	17	18
penalty cost	40	30	30	30	30
storage cost	4	4	4	12	12

Table 7.1: Uncertain demands transformation example 1: Penalty and storage costs

We modify the network by applying the rules to transform uncertain demand into uncertain costs which we presented above. The transformed network is shown in the following figure.

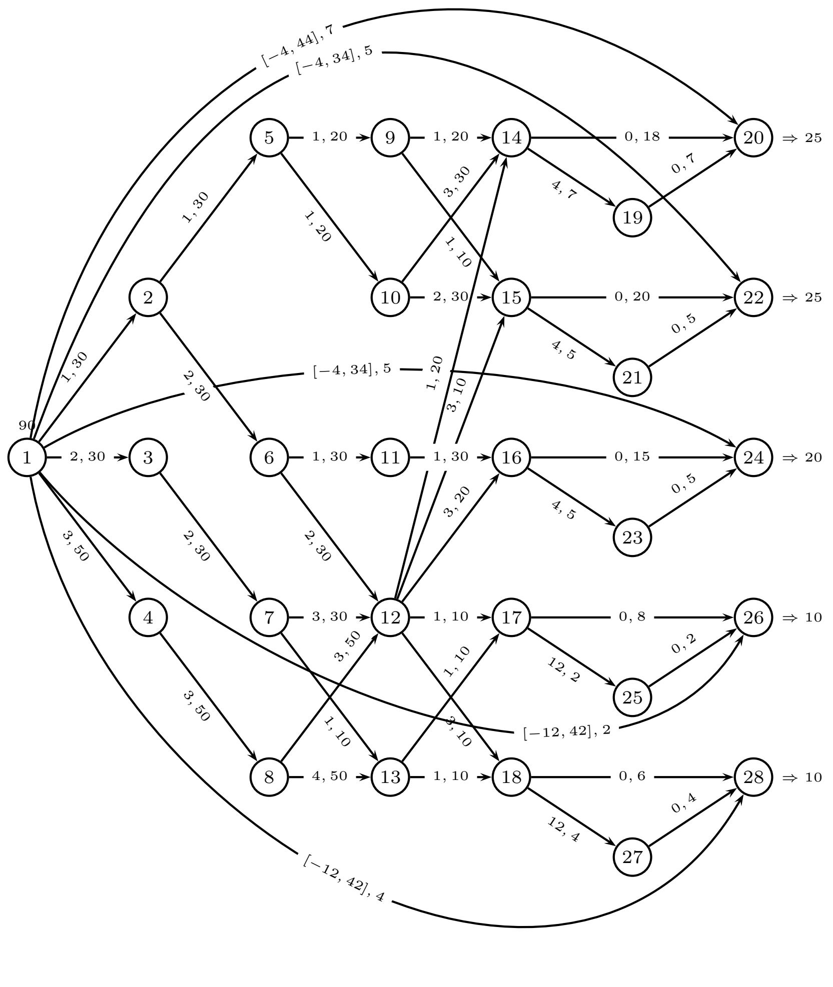


Figure 7.7: Uncertain demands transformation example 1: Transformed network

Now we have a minimum cost flow problem with uncertain costs. This problem can be solved using the algorithm of Bertsimas and Sim or the ant algorithm with Gaussian distributed costs. We apply both algorithms to the minimum cost flow problem and calculate the optimal solution for several values of Γ.

The following tables list the number of flow units of extra demand which are not delivered to the demand nodes but along the additional arcs around the original network. We consider several values of Γ.

node \ Γ	0	1.25	2.5	3.75	5
14	7	3.96	2.25	0	0
15	5	5	2.84	0	0
16	5	5	2.84	0	0
17	2	2	2	0	0
18	4	3.52	2	0	0

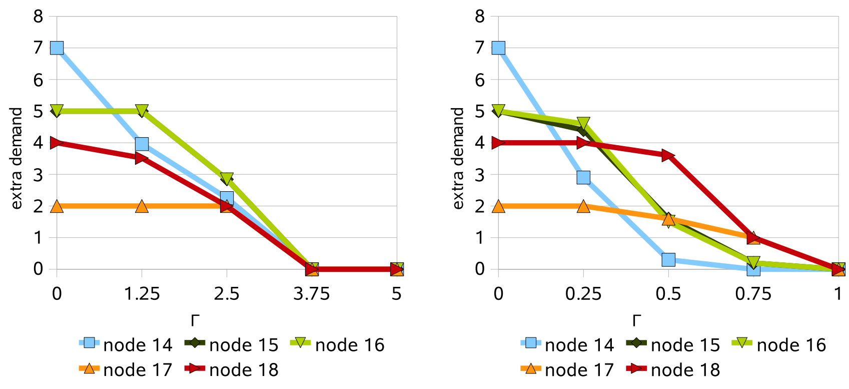
(a) Bertsimas and Sim

node \ Γ	0	0.25	0.5	0.75	1
14	7	2.9	0.3	0	0
15	5	4.4	1.6	0.2	0
16	5	4.6	1.5	0.2	0
17	2	2	1.6	1	0
18	4	4	3.6	1	0

(b) Ant algorithm

Table 7.2: Flow units of extra demand which are not delivered to the demand nodes

The following figure illustrates the table data.



(a) Bertsimas and Sim **(b)** Ant algorithm

Figure 7.8: Flow units of extra demand which are not delivered to the demand nodes

Note that for the demand nodes 15 and 16, the number of flow units which are not delivered is (almost) identical.

We can see that for both algorithms, in the minimally robust case (Γ = 0) all extra demand units are not delivered to the demand nodes, hence sent to the sink node via the additional arcs. In the maximally robust case (Γ = 5 and Γ = 1, respectively), no extra demand unit is sent via the additional arcs. In the intermediate robust cases, the number of not delivered demand units decreases for decreasing values of Γ.

The following tables list the total number of flow units which are delivered to the demand nodes along the arcs of the original network.

node \ Γ	0	1.25	2.5	3.75	5
14	18	21.04	22.75	25	25
15	20	20	22.16	25	25
16	15	15	17.16	20	20
17	8	8	8	10	10
18	6	6.48	8	10	10

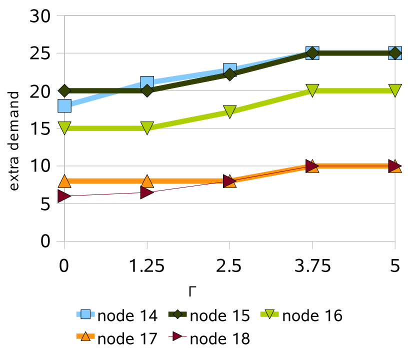
(a) Bertsimas and Sim

node \ Γ	0	0.25	0.5	0.75	1
14	18	22.1	24.7	25	25
15	20	20.6	23.4	24.8	25
16	15	15.4	18.5	19.8	20
17	8	8	8.4	9	10
18	6	6	6.4	9	10

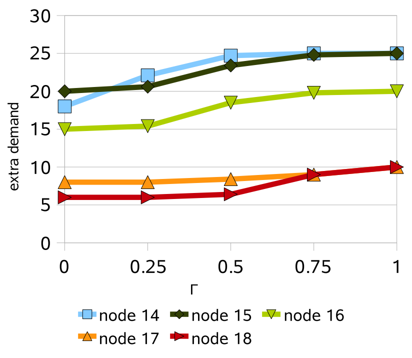
(b) Ant algorithm

Table 7.3: Total number of flow units which are delivered to the demand nodes

The following figure illustrates the table data.



(a) Bertsimas and Sim



(b) Ant algorithm

Figure 7.9: Total flow units which are delivered to the demand nodes

For both algorithms, we can see that for demand node 14, extra demand units are delivered already for lower values of Γ. This coincides with our expectation, as for demand node 14, the penalty cost is higher than for demand nodes 15 to 18.

When considering the demand nodes 17 and 18, which have the highest storage costs, only the ant algorithm behaves like we expected: For higher values of Γ, the amount of not delivered extra demand units for demand nodes 17 and 18 is higher than for demand nodes 14 to 16, because of the higher storage cost.

The algorithm of Bertsimas and Sim, however, reduces the amount of not delivered demand more uniformly. First the demand nodes with greater amount of uncertain demand units are delivered until all demand nodes have the same remaining amount of not delivered uncertain demand. Then the remaining amount of not delivered uncertain demand is reduced uniformly.

All in all, the general behavior of both algorithms corresponds to our expectations. When considering the solutions in detail, the ant algorithm meets our expectations more precisely. This is presumably due to the fact that in the transformation process, the fluctuations in the demand values and the resulting cost intervals, respectively, are not modeled exactly.

7.2 General Approach

In this section, we describe a general approach for the transformation of uncertain demands to uncertain costs, if no additional information about penalty and storage costs is available.

Let $\bar{\Gamma} = |\{i : b_i < 0, i \in \mathcal{N}\}|$ denote the number of demand nodes.

We introduce a robustness parameter $\Gamma \in [0, \bar{\Gamma}]$, which controls the level of robustness of the robust solution. Note the we do not require Γ to be integral.

Our approach has to satisfy the following criteria:

- For $\Gamma = 0$, only the nominal demand is considered.
- For $\Gamma = \bar{\Gamma}$, the full demand (basic plus extra demand) is considered.
- Nodes with shorter distance to the source are prioritized to nodes with greater distance, i.e. if only an insufficient amount of extra units are available for all demand nodes, nodes with a short distance to the source node shall be served first, because the transportation costs are lower.

In this context, a shorter distance of node i to the source means that there exists a cheaper path from the source to node i.

7.2.1 The Transformation Process

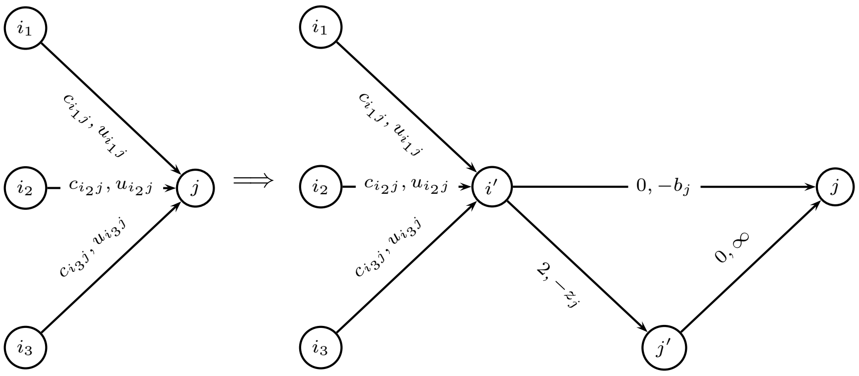
In order to transform the given network G with uncertain demands to a network $\tilde{G}$ with uncertain costs, the following transformation rules are applied:

1. Solve the minimum cost network flow problem for the original network with nominal demand. For every node i with $b_i < 0$, determine the longest used path from the source node to node i and save the value as $lnominal_i$.

2. Solve the minimum cost network flow problem for the original network with full (nominal plus extra) demand. For every node i with $b_i < 0$, determine the longest used path from the source node to node i and save the value as $lfull_i$. Determine the maximum value $lfull_{max} = \max_{i \in \mathcal{N}} lfull_i$.

3. For every node j with $b_j < 0$ and $z_j < 0$, where $|z_j|$ is the possible extra demand, insert two new nodes i' and j' and modify the corresponding arcs:

    - For every arc (i, j), insert a new arc (i, i') with cost $c_{ii'} = c_{ij}$ and capacity $u_{ii'} = u_{ij}$ and delete the arc (i, j).
    - Insert a new arc (i', j) with cost $c_{i'j} = 0$ and capacity $u_{i'j} = -b_j$.
    - Insert a new arc (i', j') with cost $c_{i'j'} = 2$ and capacity $u_{i'j'} = -z_j$.
    - Insert a new arc (j', j) with cost $c_{j'j} = 0$ and capacity $u_{j'j} = \infty$.



Consider an arbitrary path p from the source node to node i' with integer cost c. We set the cost value of arc (i', j') to $c_{i'j'} = 2$. By this means, we maintain the possibility to insert a new path from the source node to node j which is more expensive than path p (cost c), less expensive than path p plus arc (i', j') (cost $c+2$) and still has integer costs, i.e. cost $c+1$.

4. For every node j with $b_j < 0$ and $z_j < 0$, insert a new arc from the source node s to node j. These arcs will be called "additional arcs" in the following.

    - Set the cost $c_{sj} = lnominal_j + 1$, where $lnominal_j$ is the value calculated in step 1.

- Set the capacity to $u_{sj} = -z_j$.
- Set the uncertainty for the cost on arc (s, j) to $d_{sj} = lfull_{max}+3-c_{sj}$, where $lfull_{max}$ is the value calculated in step 2.

The cost values c_{sj} are chosen higher than the costs of the longest used path in the nominal demand minimum cost flow problem (cf. step 1). The purpose of this choice is that if we consider the extended network with nominal demand only, the optimal solution uses the same arcs as in the original network. The capacity values u_{sj} are set to the values of the possible extra demand in node j. The uncertain cost values d_{sj} are chosen such that every additional arc has the same maximum cost of $lfull_{max}+3$. The addition of 3 to $lfull_{max}$ ensures that in the worst case (i.e. the actual cost is $c_{ij} + d_{ij}$ for arc $(i, j) \in \mathcal{A}$), it is more expensive to send flow via (s, j) than via nodes i' and j' as defined in step 3.

5. Adapt the supply and demand:
   - For every node i with $b_i < 0$, set the demand to the full demand $b_i + z_i$.
   - Set the supply for the source node $b_s = -\sum\limits_{i:b_i<0} (b_i + z_i)$.

There exist alternative models for the cost values on the additional arcs, for example a demand node specific choice of the cost values: $d_{sj} = lfull_j + 3 - c_{sj}$. Note that alternative cost models lead to a slightly different interpretation of the definition of robustness in this context.

7.2.2 Interpretation

The flow in the extended network can be interpreted as follows: The flow on the additional arcs, inserted in step 4, represents flow, which is not actually sent “through” the network but is sent “around” it. In reality, this amount of flow would not be produced, would not be sent through the network - would not even exist. It represents the possible extra demand which is not supplied. The flow on the arcs added in step 3 represents the flow which is sent “through” the network. In fact the flow goes through the original network and strains the capacities. In reality, this amount of flow would be produced and sent through the network to the points of demand. It represents the possible extra demand which is supplied.

The transformed network can now be solved with either the ant algorithm or the algorithm of Bertsimas and Sim for uncertain cost. For interpretation, we apply different values of the robustness parameter Γ. Recall that for the ant algorithm, $\Gamma \in [0,1]$, while for the algorithm of Bertsimas and Sim, we have $\Gamma \in [0,\bar{\Gamma}]$. $\Gamma = 1$ in the ant algorithms corresponds to $\Gamma = \bar{\Gamma}$ in the algorithm of Bertsimas and Sim.

If the interpretation given above is respected, we can see that the transformation rules fulfill the given requirements:

- For $\Gamma = 0$, only the nominal cost is considered for every arc. The possible extra costs (uncertainty) are not considered. Since the nominal costs on the additional arcs are chosen greater than the costs for the longest used path in the minimum cost flow problem with nominal demand and smaller than the costs on the path which must be chosen for extra demand, for $\Gamma = 0$ the flow corresponding to the extra demand would choose the way on the additional arcs.
- For $\Gamma = 1$ (ant algorithm) and $\Gamma = \bar{\Gamma}$ (Bertsimas and Sim), the maximum possible cost is considered for every arc. Since the nominal plus the extra costs on the additional arcs are chosen such that these costs are greater than the costs for the longest used path in the minimum cost flow problem with full demand, for a maximum Γ the flow corresponding to the extra demand would choose the way on the arcs corresponding to the arcs in the original network and not on the additional arcs.
- The maximum cost values $c_{sj}+d_{sj}$ are chosen equally for all additional arcs (s,j). For nodes j with short paths from the source node to node j (i.e. low costs c_{sj}), the range from nominal costs to maximum costs is greater than for nodes with longer paths from the source node. This implies that the uncertainty d_{sj} in the costs is higher for these additional arcs and therefore for greater values of Γ, these arcs will be (partly) avoided. Therefore nodes with shorter distance to the source will be supplied with extra flow units prior to nodes with longer distance to the source node.

We sum up the arguments in the following proposition:

Proposition 7.1. *The transformation process described in Subsection 7.2.1 satisfies the following requirements:*

1. *For $\Gamma = 0$, no additional demand is delivered to the demand nodes.*
2. *For $\Gamma = 1$ (ant algorithm) and $\Gamma = \bar{\Gamma}$ (Bertsimas & Sim), the complete additional demand is delivered to the demand nodes.*
3. *Demand nodes with shorter distance to the source node are supplied prior to demand nodes with longer distance.*

7.2.3 Uncertain Demands Transformation Example 2

In the following, the general transformation algorithm is applied on an example network. Note that the network corresponds to the Uncertain Demands Transformation Example 1, see Figure 7.6.

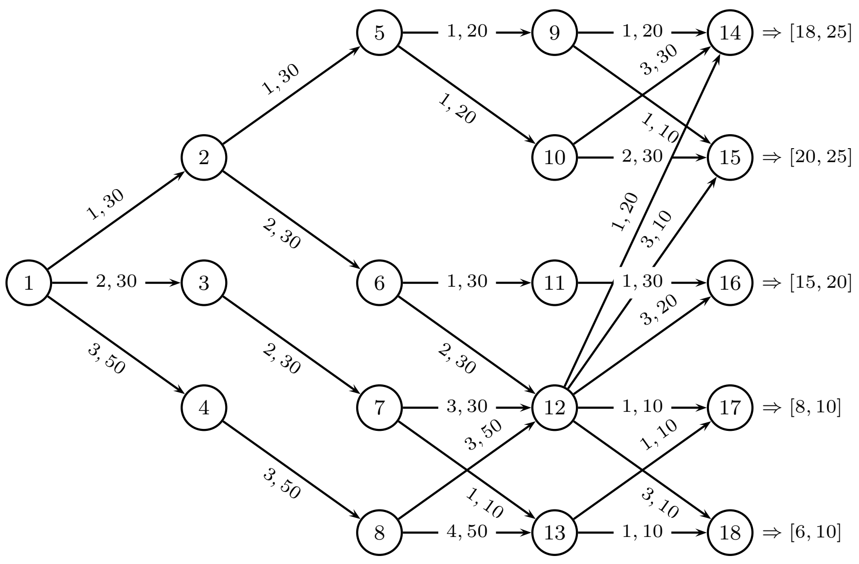


Figure 7.10: Uncertain demands transformation example 2: Minimum cost flow problem with uncertain demands

Considering the minimum cost network flow problem for the given network with nominal demand, the following solution with nominal cost 432 is obtained:

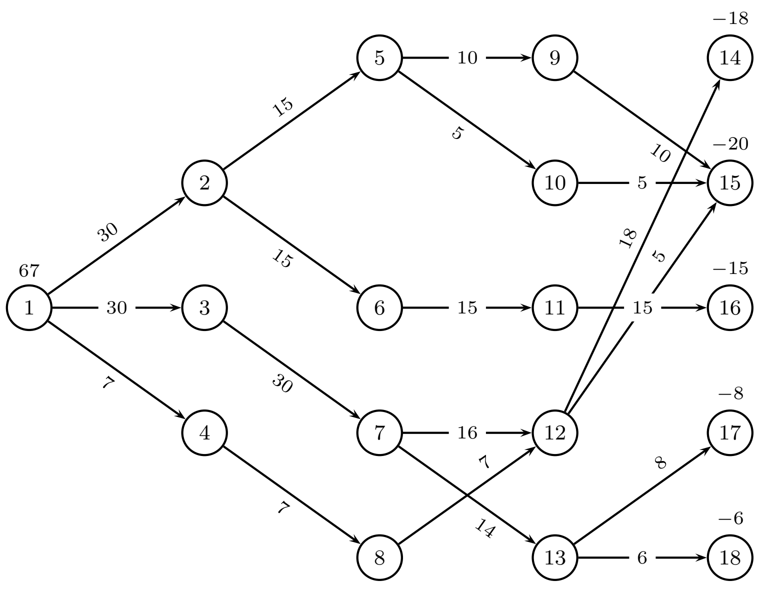


Figure 7.11: Uncertain demands transformation example 2: Optimal solution (nominal demand)

Considering the minimum cost network flow problem for the given network with full demand, the following solution is obtained:

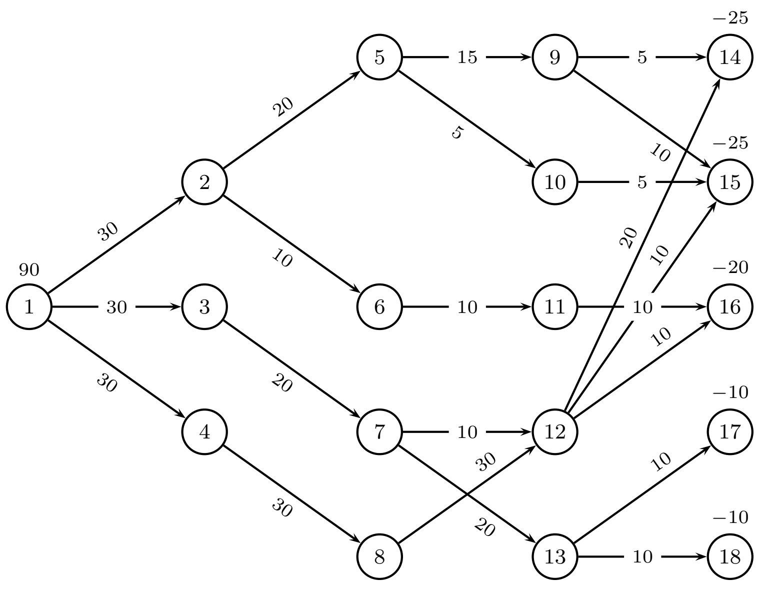


Figure 7.12: Uncertain demands transformation example 2: Optimal solution (full demand)

The nominal cost is 675.

The following table lists the longest paths for nominal demand and full demand:

node j	longest path (nominal demand) $lnominal_j$	longest path (full demand) $lfull_j$
14	8	10
15	12	12
16	5	12
17	6	6
18	6	6

Table 7.4: Uncertain demands transformation example 2: Longest path for nominal and full demand

Consequently, we have $lfull_{max} = 12$.

According to the transformation rules, the following arcs with nominal cost, extra cost and capacities have to be added:

arc (s, j)	nominal cost c_{sj}	extra cost d_{sj}	capacity u_{sj}
$(1, 14)$	9	6	7
$(1, 15)$	13	2	5
$(1, 16)$	6	9	5
$(1, 17)$	7	8	2
$(1, 18)$	7	8	4

Table 7.5: Uncertain demands transformation example 2: Added arcs

The application of the transformation rules result in the following extended network:

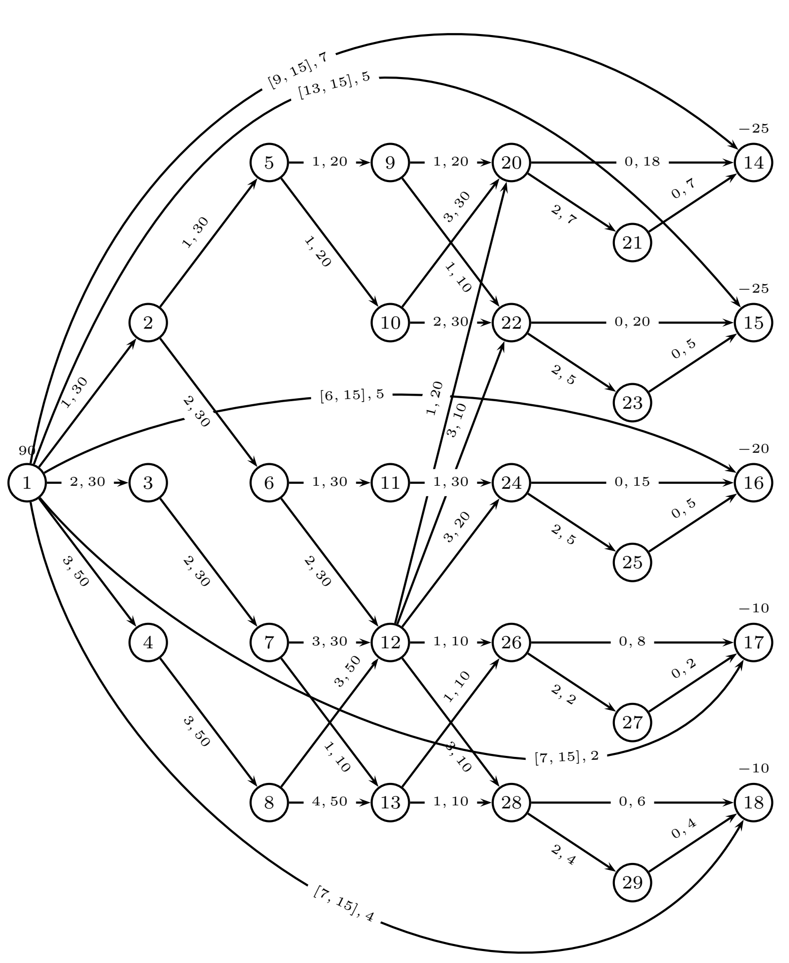


Figure 7.13: Uncertain demands transformation example 2: Extended network

Now the Robust Optimization problem is solved using the algorithm for network flow problems with uncertain costs of Bertsimas and Sim as proposed in [14] for different values of Γ. Note that the choice of the algorithm of Bertsimas and Sim is exemplarily and that we might as well evaluate the problem using the ant algorithm for uncertain costs as proposed in Section 6.3. For this example problem, we want to make a complete analysis of possible values of Γ the focal point.

We calculated the optimal solution for values from $\Gamma = 0.0$ to $\Gamma = 5.0$ in steps of 0.1.

For $0 \leq \Gamma \leq 0.8$, we obtain the following optimal supply at the demand nodes:

node	total supply	extra supply
14	18	0
15	20	0
16	15	0
17	8	0
18	6	0

Table 7.6: Uncertain demands transformation example 2: Robust optimal supply for $0 \leq \Gamma \leq 0.8$

Here the total supply is the nominal supply plus the extra supply, where extra supply denotes the number of extra units that are sent to the demand nodes via the original network.

Obviously, the claim, that for $\Gamma = 0$ the supply may not exceed the nominal demand, is fulfilled.

For $0.9 \leq \Gamma \leq 1.3$ we obtain the following solution:

node	total supply	extra supply
14	18	0
15	20	0
16	$15\frac{1}{3}$	$\frac{1}{3}$
17	8	0
18	6	0

Table 7.7: Uncertain demands transformation example 2: Robust optimal supply for $0.9 \leq \Gamma \leq 1.3$

Obviously, the first node, for which there is extra supply, is node 16. This is due to the fact that the uncertainty in the costs on the additional arc $(1, 16)$ is greater than the uncertainty in the costs for the other additional arcs. Moreover, the extra demand of node 16 is relatively high.

For $1.4 \leq \Gamma \leq 1.7$, the following solution is obtained:

node	total supply	extra supply
14	19.7	1.7
15	20	0
16	16.4	1.4
17	8	0
18	6	0

Table 7.8: Uncertain demands transformation example 2: Robust optimal supply for $1.4 \leq \Gamma \leq 1.7$

Extra supply is now given for nodes 14 and 16. Both nodes have a relatively high extra demand as well as greater uncertainty in the costs on the additional arcs $(1, 14)$ and $(1, 16)$.

For $1.8 \leq \Gamma \leq 1.9$, the following solution is obtained:

node	total supply	extra supply
14	20	2
15	20	0
16	16.7	1.7
17	8	0
18	6.25	0.25

Table 7.9: Uncertain demands transformation example 2: Robust optimal supply for $1.8 \leq \Gamma \leq 1.9$

The extra supply for nodes 14 and 16 is further increased. Node 18 is also delivered with extra supply because of the uncertainty in the cost vector on arc $(1, 18)$. Though the arc $(1, 17)$ has equal values in cost and uncertainty, there is no extra supply for node 17, because the extra demand is less than the extra demand of node 18.

For $2.0 \leq \Gamma \leq 2.3$, the following solution is obtained:

node	total supply	extra supply
14	22.3	4.3
15	20	0
16	18.2	3.2
17	8	0
18	8	2

Table 7.10: Uncertain demands transformation example 2: Robust optimal supply for $2.0 \leq \Gamma \leq 2.3$

The extra supply for nodes 14, 16 and 18 is further increased.

For $2.4 \leq \Gamma \leq 2.8$, the following solution is obtained:

node	total supply	extra supply
14	23.3	5.3
15	20	0
16	18.9	3.9
17	8.75	0.75
18	8.75	2.75

Table 7.11: Uncertain demands transformation example 2: Robust optimal supply for $2.4 \leq \Gamma \leq 2.8$

While node 15 still is not delivered with extra supply because of the relatively low uncertainty on arc $(1, 15)$, node 17 is now delivered with extra supply. Note that for the demand nodes 17 and 18, the amount of demand units that are not delivered to these nodes are of equal value.

For $2.9 \leq \Gamma \leq 5.0$, the following solution is obtained:

node	total supply	extra supply
14	25	7
15	25	5
16	20	5
17	10	2
18	10	4

Table 7.12: Uncertain demands transformation example 2: Robust optimal supply for $2.9 \leq \Gamma \leq 5.0$

As it was claimed, for $\Gamma = 5$, a solution is obtained, where the extra supply is delivered completely for all nodes. There is no flow on the additional arcs $(1, 14)$, ..., $(1, 18)$.

7.3 Summary and Conclusions

In this chapter, we examined network flow problems with uncertain demands. Due to the fact that for NFPs with uncertain costs, efficient algorithms exist, the aim was to develop a transformation model in order to transform NFPs with uncertain demands into NFPs with uncertain costs.

Two different transformation processes were proposed, dependent on how much information about the network input data is known: The first transformation is based on additional information about the cost values, which is given in the form of penalty costs in case of supply deficiency and storage costs for surplus delivery. In case that no additional information is available, we developed a general transformation scheme which is applicable without additional input data.

The major advantage of the proposed algorithms is the fact that the efficient algorithms for NFPs with uncertain costs, cf. Chapter 6, can be applied to the transformed problems.

In the case that additional information, i.e. penalty and storage costs, is available, the transformation process can be carried through very quickly by the insertion of new nodes and arcs. If penalty and storage cost values are not available, additional information is generated by determining longest paths in the network and by solving several minimum cost flow problems in the original network. Using this information, additional nodes and arcs are inserted. For both transformation processes, the increase in network size is moderate: Two new nodes and four new arcs per demand node are inserted.

However, it has to be taken into account that the transformation models are not exact models. In Section 8.3, an exact approach for network flow problems with uncertain demands, based on the concept of recoverable robustness, will be presented. In Chapter 9, a Stochastic Programming approach for multicommodity NFPs will be developed. Furthermore, we will examine an ACO based algorithm for single-commodity and multicommodity NFPs in Section 8.2.

Chapter 8

Robust Optimization in Network Flows: Uncertain Demands

In the previous chapters, we considered supply chain problems with uncertain costs as well as possible transformations of uncertain demands to uncertain costs such that we could solve the uncertain demands problems using the uncertain costs algorithms.

In this section, we will develop special purpose robust algorithms for dealing with uncertain demands.

8.1 Problem Formulation and Objectives

Given a directed network $G = (\mathcal{N}, \mathcal{A})$ with finite node set $\mathcal{N}$ and finite arc set $\mathcal{A}$, the minimum cost network flow problem

$$\begin{array}{lll} \text{minimize} & \sum\limits_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} & \\ \text{s.t.} & \sum\limits_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum\limits_{j:(j,i)\in\mathcal{A}} x_{ji} = b_i & \forall i \in \mathcal{N} \\ & 0 \leq x_{ij} \leq u_{ij} & \forall (i,j) \in \mathcal{A} \end{array}$$

is considered, where c_{ij} denotes the cost values on arc (i,j), x_{ij} denotes the flow on arc (i,j), b_i denotes the nominal supply/demand for node i and u_{ij} denotes the capacity on arc (i,j).

In the following, network flow problems are considered where the underlying network is a single-source network without dicycles. Moreover, it is assumed that only positive costs arise, that means $c_{ij} \geq 0, (i,j) \in \mathcal{A}$.

We consider a problem with uncertain demands in the sense that the demand can only be predicted to be within a certain interval. In particular, let $b_i < 0,\ i \in \mathcal{A}$, be the upper bound of this interval, referred to as "nominal demand", and let $b_i + z_i,\ i \in \mathcal{A}$ with $z_i \leq 0$ be the respective lower bound. $|z_i|,\ i \in \mathcal{A}$, is the maximum possible "extra demand".

For every demand node, we introduce two types of costs: If the actual demand of a demand node i is higher than the number of flow units arriving at the demand node, a penalty cost $c_i^{penalty}$ per unavailable unit is charged. If the actual demand of a demand node is lower than the number of flow units arriving at the demand node, a storage cost $c_i^{storage}$ per redundant unit is charged, cf. Section 7.1.

We introduce a robustness parameter $\Gamma \in [0,1]$, which controls the level of robustness of the robust solution. We have the following aims:

1. For $\Gamma = 0$, only the nominal demand shall be delivered to the demand nodes.
2. For $\Gamma = 1$, the full demand (nominal plus extra demand) shall be delivered to the demand nodes.
3. For increasing Γ, more units shall be delivered to the demand nodes. The choice, which demand nodes are to be prioritized, shall be dependent on the total resulting cost. Here, arc costs, penalty costs and storage costs shall be regarded.

The involvement of penalty and storage costs corresponds exactly to the ideas given in Section 7.1: If one demand node has higher penalty costs than other demand nodes, this demand node shall be delivered with extra units prior to demand nodes with lower penalty cost. If one demand node has higher storage costs than other demand nodes, extra demand units shall be directed to demand nodes with lower storage costs.

8.2 Ant Algorithm

In this section, we want to adapt the ACO metaheuristic for minimum cost flow problems, which we presented in Chapter 4, directly to minimum cost flow problems with uncertain demands, i.e. without using a transformation to a network flow problem with uncertain costs as in Chapter 7.

Let $G = (\mathcal{N}, \mathcal{A})$ be a single-source network with uncertain demand at the demand nodes, i.e. nodes i where $b_i < 0$. Let $z_i \leq 0$ be the possible extra demand at node i.

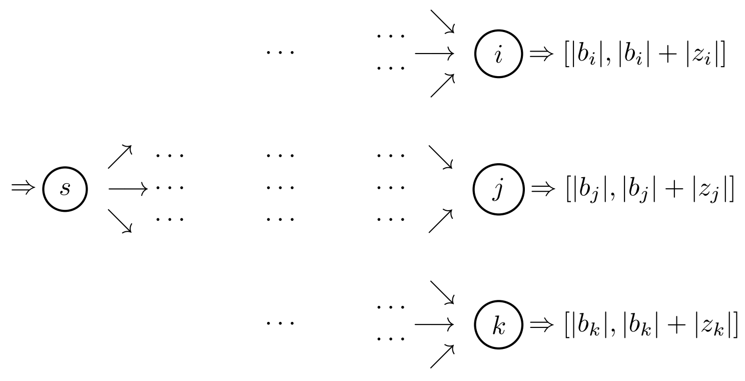

s
i
j
k
$\Rightarrow [|b_i|, |b_i| + |z_i|]$
$\Rightarrow [|b_j|, |b_j| + |z_j|]$
$\Rightarrow [|b_k|, |b_k| + |z_k|]$

Figure 8.1: Network with uncertain demands at nodes i, j and k

8.2.1 Modification of the Network

First, we extend G to a single-sink network by adding a new sink node t and arcs from the original demand nodes to t.

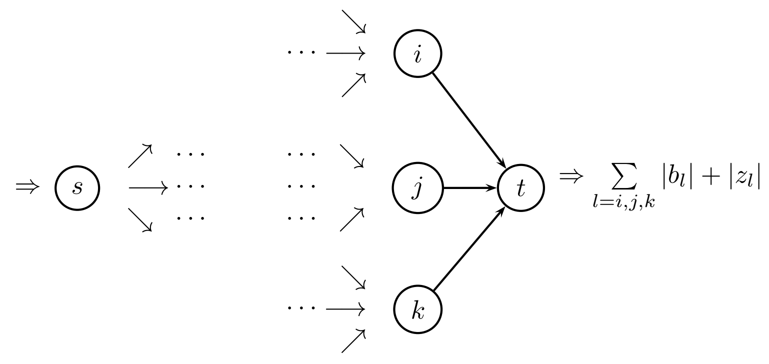


Figure 8.2: Single-sink network

The basic idea of our adapted ant algorithm is the following: We want the first $\sum_{i:b_i<0} |b_i|$ ants for the nominal demand, in the following referred to as *basic ants*, to find their path through the original network as specified in Chapter 4. In addition, we introduce a further set of ants for the extra demand, in the following referred to as *extra ants*. The extra ants shall have the choice whether to walk "through" the network or "around" the network: We introduce a new source node for the extra ants as well as two additional arcs, where one arc is a direct connection from the new source node to the sink node and the other arc is a connection from the new sink node to the old sink node.

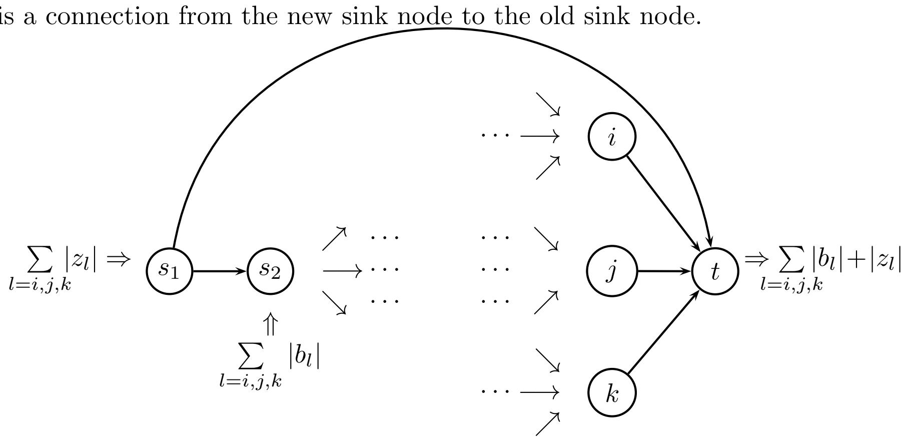
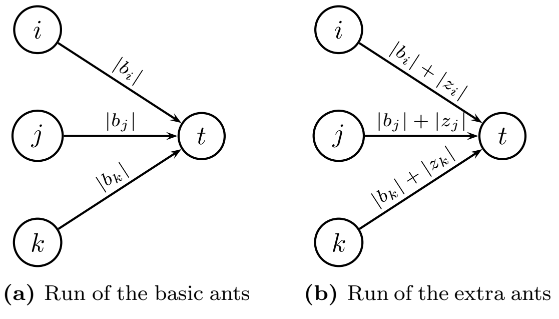

$\sum_{l=i,j,k} |z_l| \Rightarrow$
s_1
s_2
$\Uparrow$
$\sum_{l=i,j,k} |b_l|$
i
j
k
t
$\Rightarrow \sum_{l=i,j,k} |b_l|+|z_l|$

Figure 8.3: Extended network for uncertain demands at nodes i, j and k

In our ant algorithm, the basic ants start their walk at node s_2, the original source node. The extra ants start their walk at node s_1, the new source node.

As we want to ensure that the basic ants choose their path such that the nominal demand at the original demand nodes is fulfilled correctly, we have to adapt the capacities at the arcs from the original demand nodes to the sink node t. For the run of the basic ants, we set the capacity of an arc from an original demand node i to the sink node t to $|b_i|$. As the number of basic ants is $\sum_{i:b_i<0} |b_i|$, it is guaranteed that exactly $|b_i|$ ants walk past node i and therefore the nominal demand at node i is fulfilled. At the beginning of the run of the extra ants, we change the capacity of the arcs mentioned above to $|b_i| + |z_i|$, such that up to $|z_i|$ extra ants can walk past node i.



(a) Run of the basic ants **(b)** Run of the extra ants

Figure 8.4: Extended network: Capacity change

The capacity of the additional arcs (s_1, s_2) and (s_1, t) is set to $\sum_{i:b_i<0} |z_i|$. Hence all extra ants can choose either one additional arc or the other without capacity restriction.

The visibility of the additional arcs is set in dependence of the robustness parameter Γ: The visibility of the additional arc which leads from the new source node to the sink node is set to $1-\Gamma$, the visibility of the additional arc which leads from the new source node to the old source node is set to Γ. Note that we actually need not define cost values for the additional arcs, as for the calculation of the arc probability, pheromone trails and visibilities are sufficient.

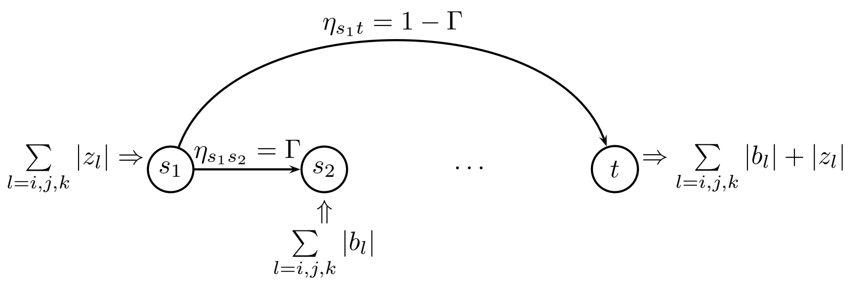


Figure 8.5: Extended network: Visibility on additional arcs

The choice of the visibilities for the arcs (s_1, t) and (s_1, s_2) can be explained as follows: If the pheromone values are not considered, the percentage of ants that use the additional arc will be near to $1-\Gamma$, while the percentage of ants that actually walk through the original network will be near to Γ. This corresponds to the intended definition of the robustness parameter Γ: For small Γ, more ants use the additional arc, which corresponds to the fact that more flow units are not delivered to the demand nodes. For higher values of Γ, more ants walk through the original network, which corresponds to the fact that more flow units are delivered to the demand nodes.

Now we examine different values of the robustness parameter Γ:

For $\Gamma = 0$, the additional arc from node s_1 to node t has a visibility of 1, while the additional arc from node s_1 to node s_2 has a visibility of 0. This means that all extra ants at node s_1 will choose arc (s_1, t) and therefore all extra flow units will be sent around the network. As we have a visibility of 0 on one arc, the pheromone levels can not influence the decision of the ants.

For $\Gamma = 1$, the additional arc from s_1 to t has a visibility of 0, while the additional arc from s_1 to s_2 has a visibility of 1. This means that all extra ants at node s_1 will choose arc (s_1, s_2) and therefore all extra flow units will be sent through the network. As we have a visibility of 0 on one arc, the pheromone levels can not influence the decision of the ants.

For $0 \leq \Gamma \leq 1$, the visibility of the additional arc from s_1 to t corresponds to the current value of Γ, the visibility of arc (s_1, s_2) is $1-\Gamma$. In the first iteration, the pheromone level on both arcs is set to 1, which means that the ants' decision is only based on the current value of Γ. With increasing number of iterations, pheromone can accumulate more or less on both additional arcs and therefore the ants are able to tend towards one arc or the other.

Note that if a positive lower bound is imposed on the visibility values, both arcs (s_1, t) and (s_1, s_2) have a strict positive probability to be chosen for all values of Γ.

It is very important to impose upper bounds for the pheromone trails on the additional arcs. If these upper bounds are not imposed, the extra ants are very likely to choose the same additional arc, that means that either all extra ants walk on additional arc (s_1, t) or all extra ants walk on additional arc (s_1, s_2).

As we already mentioned in Section 4.2.5, it is possible that an ant reaches a dead end during its walk. If this happens, we add the dead end node to the tabu list of the ant and let the ant start over again.

In this case, we have to pay special attention to extra ants. An extra ant can only reach a dead end if it is walking through the original network via arc (s_1, s_2). It is not possible to reach a dead end when using arc (s_1, t). Therefore, a restarting extra ant at node s_1 must remember its first decision at this node, either to walk on arc (s_1, s_2) or to walk on arc (s_1, t), and repeat this choice at a restart. If this determination was not respected, the probability for choosing arc (s_1, t) would actually be increased.

8.2.2 Realization of Uncertain Demands

As proposed in Section 6.3 for uncertain cost values, we address the uncertain input values of the minimum cost flow problem by generating random input values in each iteration.

In the context of uncertain demands, for any demand node i in the original network, we generate a random demand value in $[b_i + z_i, b_i]$ at the beginning of each iteration. As we want to respect the robustness parameter Γ, we generate Gaussian distributed demand values where the distribution function is dependent on Γ.

The expected value μ_i of the Gaussian distribution $N(\mu_i, \sigma_i^2)$ is set to $\mu_i = b_i + \Gamma z_i$. The standard deviation σ_i is set to $\sigma_i = \frac{1}{4}|z_i|$. Furthermore, the distribution is truncated at the interval boundaries $b_i + z_i$ and b_i.

The influence of the robustness parameter Γ on the expected value μ_i makes sure that for high values of Γ, the generated demand value is likely to be near the lower interval bound $b_i + z_i$, while for low values of Γ, the generated demand value is likely to be near the upper interval bound b_i (note that b_i and z_i are negative values).

The choice of the standard deviation guarantees that approximately 95.45% of the generated demand values lie within the uncertainty interval $[b_i + z_i, b_i]$. However, in practice, this choice can lead to widespread demand data for large intervals $[b_i + z_i, b_i]$, such that in this case, a smaller standard deviation may be appropriate.

The generated random demand value is referred to as *current demand*.

After all ants, i.e. basic and extra ants, have finished their walk in an iteration, the constructed solution is evaluated using the generated demand value, the arc cost, the storage cost and the penalty cost. First, the nominal flow cost (arc cost times flow) is evaluated. In addition, for every demand node i in the original network, the current demand is compared to the flow on arc (i, t).

If the flow on arc (i, t) is greater than the current demand at node i, we add a storage cost of $c_i^{storage}$ per redundant unit of flow to the nominal flow cost.

If the flow on arc (i, t) is smaller than the current demand at node i, we add a penalty cost of $c_i^{penalty}$ per unavailable unit of flow to the nominal flow cost.

As in our original ant algorithm in Chapter 4, one unit of pheromone is laid down per ant and per arc at the end of each iteration l. This amount of pheromone is multiplied with the pheromone factor ω_l, where ω_l is defined as follows:

$$\omega_l := \begin{cases} 1 & \text{in iteration } l = 1 \\ \frac{\text{total cost of the best-so-far solution}}{\text{total cost of the current solution}} & \text{in iteration } l \geq 2. \end{cases}$$

Then we have $0 \leq \omega_l \leq 1$. If ω_l takes a too small or too wide variety of values, it can be raised to the power of a value $\varsigma^{pher} \geq 0$ in order to avoid early stagnation behavior in the ant algorithm.

8.2.3 Variations

Separation A possible variation for a minimum cost flow problem with uncertain demands is to separate it into two subproblems and then solve them independently:

1. First solve the minimum cost flow problem for the nominal demands only.

As this problem is deterministic, efficient exact algorithms as the CSA can be applied.

2. Then set up a new minimum cost flow problem for the extra demands. The network structure and the arc costs remain unaffected. The uncertain demand is set to $[0, \text{extra demand}]$ at the demand nodes. The original arc capacity is reduced by the amount of flow of the optimal solution of the minimum cost flow problem for the nominal demand. Solve this minimum cost flow problem with uncertain demands with the ant algorithm.

3. Combine the two solutions.

A disadvantage of this procedure is the fact, that the two subproblems are solved independently and therefore no information is shared. However, the advantage of the variation is that subproblem 1 is deterministic and can be solved exactly and only subproblem 2, which is usually smaller than the original problem because of a smaller number of involved ants, is subject to uncertainty and is solved by a heuristic.

Multicommodity The ACO metaheuristic for minimum cost flow problems with uncertain demands can be adapted to multicommodity minimum cost flow problems with uncertain demands by the means proposed in Section 4.3. Note that initially all ants corresponding to the nominal demand of any commodity have to finish their walk before the extra ants of any commodity start.

Arbitrary Distributions In case that the distribution of the uncertain demand values is known or can be approximated, the ant algorithm can be easily adapted. Then the demand values can be generated according to the given distribution instead of the proposed Gaussian distribution $N(\mu_i, \sigma_i^2)$.

8.2.4 Supplements for the Multicommodity Version

The multicommodity version of the ant algorithm for uncertain demands provides the possibility to enforce the influence of storage and penalty costs.

We modify the visibilities of the additional arcs (s_1, s_2) and (s_1, t) in the following way:

Let

$$\omega_k^{storage} := \frac{\sum\limits_{i:b_i^k<0} (c^{storage})_i^k}{\frac{1}{p}\sum\limits_{l=1}^{p} \sum\limits_{i:b_i^l<0} (c^{storage})_i^l} \tag{8.1}$$

and

$$\omega_k^{penalty} := \frac{\sum\limits_{i:b_i^k<0} (c^{penalty})_i^k}{\frac{1}{p}\sum\limits_{l=1}^{p} \sum\limits_{i:b_i^l<0} (c^{penalty})_i^l} \tag{8.2}$$

for commodity k.

Here $(c^{storage})_i^k$ denotes the storage cost for demand node i and commodity k. $(c^{penalty})_i^k$ denotes the penalty cost for demand node i and commodity k.

Then $\omega_k^{storage}$ determines the proportion of the total possible storage costs for commodity k in relation to the average total possible storage costs over all commodities. Analogously, $\omega_k^{penalty}$ determines the ratio of the total possible penalty costs for commodity k and the average total possible penalty costs over all commodities.

Then we set the visibilities of the additional arcs to

$$\eta_{s_1 s_2}^k := \Gamma \frac{\omega_k^{penalty}}{\omega_k^{storage}} \tag{8.3}$$

$$\eta_{s_1 t}^k := (1-\Gamma) \frac{\omega_k^{storage}}{\omega_k^{penalty}} \tag{8.4}$$

This choice ensures that for a commodity k with higher relative penalty costs and/or lower relative storage costs, the visibility of the additional arc (s_1, t) and therefore the probability of choosing this arc is decreased. Simultaneously, the visibility of the arc (s_1, s_2) is increased. This, in turn, means that for commodity k, more ants will walk through the network than for commodities with lower penalty costs and/or higher storage costs.

If $\omega_k^{storage}$ and $\omega_k^{penalty}$ take values which are too "far" away from 1.0, we propose to introduce two parameters $\varsigma_{arc}^{storage}$ and $\varsigma_{arc}^{penalty}$ and raise $\omega_k^{storage}$ to the power of $\varsigma_{arc}^{storage}$ and $\omega_k^{penalty}$ to the power of $\varsigma_{arc}^{penalty}$, respectively. This avoids early stagnation behavior in the ant algorithm.

Note that these supplements can also be applied to the single-commodity problem in order to enforce the influence of penalty and storage costs on the decision of the ants whether to walk through the original network via arc (s_1, s_2) or to walk around it via the additional arc (s_1, t).

8.2.5 Multicommodity Uncertain Demands Example

In this subsection, we apply the proposed ant algorithm to a multicommodity minimum cost flow problem with uncertain demands. We consider two different commodities. The following figure illustrates the structure of the underlying network. The arc labels refer to the cost and capacity values for every commodity and to the bundle capacity.

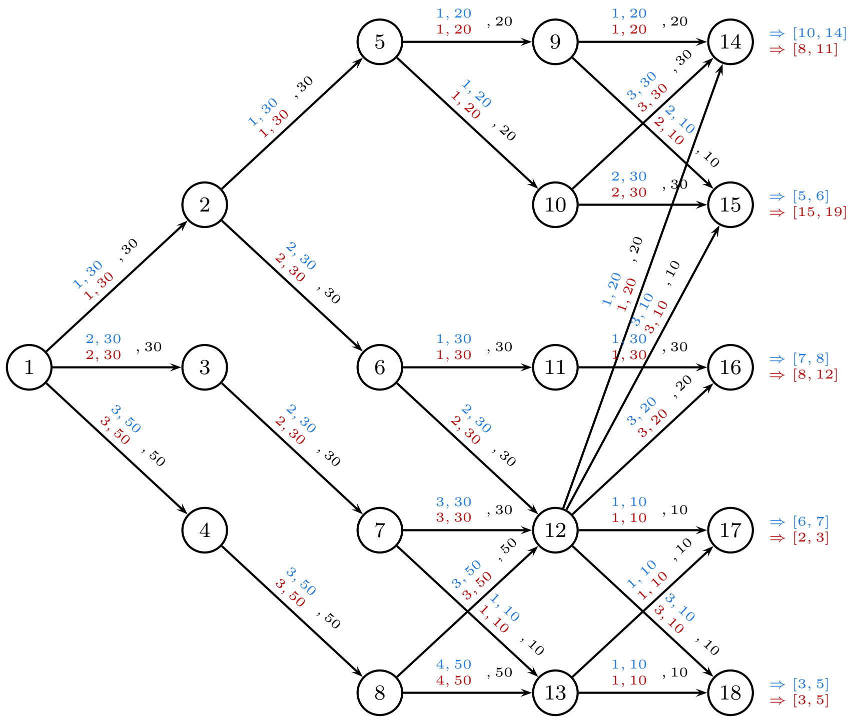


Figure 8.6: Multicommodity uncertain demands example: Minimum cost flow problem with uncertain demands

The underlying network structure corresponds to the networks given in Figures 7.6 and 7.10.

We impose penalty costs for not delivered flow units, which would actually have been requested at the demand nodes, as well as storage costs for supplementary delivered flow units which need to be stored at the demand nodes.

node	14	15	16	17	18
penalty cost	18	18	18	24	24
storage cost	1	1	1	1	1

(a) Commodity 1

node	14	15	16	17	18
penalty cost	12	12	12	12	12
storage cost	2	2	2	2	2

(b) Commodity 2

Table 8.1: Multicommodity uncertain demands example: Penalty and storage costs

We apply the ant algorithm to the multicommodity uncertain demands example problem and examine the results.

As expected, for $\Gamma = 0$, which is the least robust case, no extra units are sent to the demand nodes. In the maximally robust case, i.e. $\Gamma = 1$, the full extra demand is sent to the demand nodes.

We examine in detail the intermediate robust case $\Gamma = 0.5$. The following table lists the delivered flow units at the demand nodes 14 to 18 for commodities 1 and 2.

run	14	15	16	17	18
1	12	5	7	7	5
2	12	5	7	7	5
3	13	6	7	7	4
4	12	5	7	7	4
5	14	5	7	7	4
6	14	5	7	7	4
7	14	5	8	7	4
8	14	5	8	7	4
9	12	5	7	7	5
10	13	5	7	7	5
11	13	5	8	7	5
12	13	6	7	7	5
13	13	5	7	7	5
14	13	5	7	7	5
15	11	5	7	7	5
16	13	5	7	7	5
17	11	5	8	7	5
18	14	5	7	7	4
19	13	5	8	7	4
20	12	5	7	7	4
average	12.80	5.10	7.25	7.00	4.55

(a) Commodity 1

run	14	15	16	17	18
1	10	15	8	3	5
2	11	15	10	3	5
3	10	16	8	2	3
4	10	16	8	2	4
5	11	16	9	2	5
6	9	16	8	2	3
7	11	16	10	2	3
8	11	16	8	3	5
9	9	15	8	3	4
10	11	15	8	3	4
11	9	15	8	3	5
12	9	16	8	3	3
13	10	15	10	3	5
14	11	15	8	3	4
15	10	15	8	3	4
16	11	15	8	3	3
17	9	15	8	3	4
18	11	16	8	3	3
19	10	15	8	3	3
20	10	15	9	3	3
average	10.15	15.40	8.40	2.75	3.90

(b) Commodity 2

Table 8.2: Multicommodity uncertain demands example: Delivered demand at demand nodes for different runs of the ant algorithm

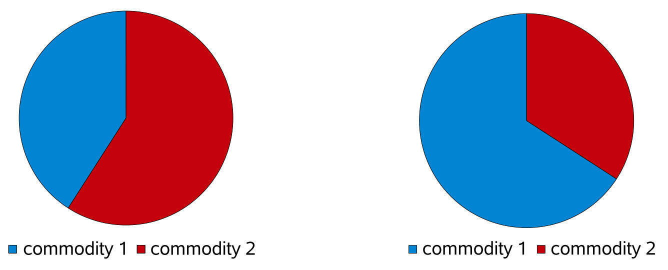
We can see that for different runs of the ant algorithm, we obtain different solutions. This is due to the fact, that the ant algorithm is random based and moreover, the random generation of the demands differs from run to run.

Therefore it is not recommended to consider one single ant run only.

In the following, we consider the results of an average run.

For commodity 1, we have a possible extra demand of 9 units, for commodity 2 of 13 units, i.e. 41% of the overall additional demand is associated with commodity 1, 59% with commodity 2.

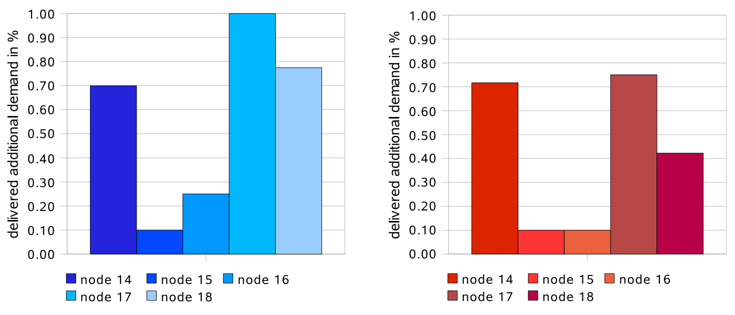
Examining the penalty and storage costs, we can see that for commodity 1, the penalty costs are higher and the storage costs are lower than for commodity 2. We can see that the ant solution reflects this fact: For commodity 1, 63% of the possible extra demand is actually delivered, while only 33% of the possible extra demand of commodity 2 is delivered.



(a) Possible extra demand **(b)** Delivered extra demand

Figure 8.7: Multicommodity uncertain demands example: Distribution of possible and delivered uncertain demand

The following figure illustrates the delivered uncertain demand (in percent), on the average, at the demand nodes for commodities 1 and 2.



(a) Commodity 1 **(b)** Commodity 2

Figure 8.8: Multicommodity uncertain demands example: Delivered uncertain demand for commodities 1 and 2

For commodity 1, we can see that for nodes 17 and 18 the amount of delivered extra units is higher than for nodes 14, 15 and 16. This is due to the higher penalty costs for nodes 17 and 18.

For both commodities, the amount of delivered extra units to node 14 is relatively high. This can be explained by the fact that there exists a very cheap path $(1, 2, 5, 9, 14)$ from the source node to node 14 which is cheaper than any other path from the source node to a demand node. Therefore it is relatively cheap to send extra units to node 14.

This example shows that the heuristic ant algorithm gives a good idea or trend. However, different runs can differ notably, such that it is recommended to consider more than one run for a specific optimization problem.

Both the ant algorithm for uncertain demand and the transformation based algorithms, which were proposed in Section 7.1, exploit the information given by penalty and storage costs. The control of the level of robustness by a robustness parameter Γ is directly given for the ant algorithm. The optimization problem with uncertain cost values, which is generated by the transformation algorithm in Section 7.1, can also be solved using a robustness parameter with the ant algorithm for uncertain costs or the algorithm of Bertsimas and Sim. It is very hard to recommend one approach or the other in a special situation. We will compare the results of the two approaches by means of a reference model in Chapter 12.

8.3 Recoverable Robustness

In this section, we will discuss an alternative approach for modeling robustness, called *recoverable robustness*. The recoverable robustness model is specially suited for supply chain problems with uncertain demands, as it considers uncertainties in the matrix and the right-hand side of the constraints of an optimization problem.

Recoverable robustness is a recent approach of S. Stiller et al., see [64] and [91], in Robust Optimization. The aim is to find solutions to optimization problems, which can be made feasible - or *recovered* - within a given budget. The recovery possibilities are formalized via a class $\mathcal{A}$ of admissible recovery algorithms.

According to [91], a solution x of the original optimization problem is *recovery robust* against the imperfection of information (scenarios) and for the limited recovery possibilities (recovery budget), if for all scenarios a feasible solution can be recovered from x by means of an admissible recovery algorithm.

The optimization process consists of two phases, a planning phase and a recovery phase. In the planning phase, a solution x is computed, which can possibly become infeasible in the realized scenario. Furthermore, an admissible recovery algorithm is chosen. In the recovery phase, the chosen recovery algorithm is used to make x feasible in the realized scenario.

For details concerning the framework of recoverable robustness, see [64] and [91].

8.3.1 Linear Programming Recovery

In [91], S. Stiller presents basic results for recovery robust optimization problems (RROP), where the original optimization problem as well as the recovery problem are linear programs. In particular, linear programs with inequality constraints only are considered.

We modify the considerations of S. Stiller such that they fit exactly our problem structure. As before, we consider supply chain problems formulated as MCFPs. In this chapter, we focus on uncertainty in the demand values. Let S be a closed scenario set for the uncertain demand values h_s.

We consider the following formulation of a linear supply chain problem:

$$\begin{aligned} \min_x \; & c^\top x \\ \text{s.t. } & Ax \le b \\ & A_{eq}x = b_{eq} \\ & Tx = h \end{aligned} \tag{8.5}$$

where $c,\ x \in \mathbb{R}^m$, $A \in \mathbb{R}^{n_1 \times m}$, $b \in \mathbb{R}^{n_1}$, $A_{eq} \in \mathbb{R}^{n_2 \times m}$ and $b_{eq} \in \mathbb{R}^{n_2}$. $Ax \le b$ contains the capacity constraints and $A_{eq}x = b_{eq}$ contains the flow conservation constraints for transshipment nodes (i.e. nodes that are neither source node nor sink node). Furthermore, we have the equality constraints $Tx = h$, which ensure that the demand h is delivered to the demand nodes, where $T \in \mathbb{R}^{l \times m}$, $h \in \mathbb{R}^l$ and l is the number of demand nodes.

When considering uncertain demand values, we have to determine a recovery algorithm for solutions x that have to be turned to a feasible solution for the realized scenario. Inspired by simple recourse in two-stage stochastic programming, see [18] and Chapter 9, we choose $W = [I_l, -I_l]$ as recovery matrix, where I_l is the l-dimensional unit matrix. The matrix W specifies how deficiencies in demand can be compensated. In case of deficiency, for every demand node, the number of delivered demand units can be smaller or greater than the actual demand. In order to satisfy the actual demand, we have to add or subtract missing or surplus units, which leads to a compensation matrix W as given above.

Let y_s be the compensation vector for scenario $s \in S$. Let $d \in \mathbb{R}^{2l}$ be the cost vector for penalty and storage costs for missing or surplus demand units, respectively. Let D be the total recovery budget.

Note that according to our problem formulation, the possibility that for a specific demand node both penalty and storage costs accrue is not excluded explicitly. This could be the case if the recovery budget is extremely high. In practice, however, the recovery budget is generally small enough to avoid solutions.

Let $h^{min} \in \mathbb{R}^l$ be the vector of the minimum demand values at the demand nodes. Then we add the inequality constraints $Tx \geq h^{min}$ in order to guarantee that the minimum demand h^{min} is always distributed to the demand nodes.

Then the RROP can be formulated as follows:

$$\begin{aligned} \min_x \ & c^\top x \\ \text{s.t. } & Ax \leq b \\ & A_{eq}x = b_{eq} \\ & Tx \geq h^{min} \\ & x \geq 0 \\ & \forall h_s \in S \ \exists y_s \in \mathbb{R}^{2l} : \\ & \qquad\qquad\qquad Tx + Wy_s = h_s \\ & \qquad\qquad\qquad\quad d^\top y_s \leq D \qquad y_s \geq 0 \end{aligned} \tag{8.6}$$

For linear programs, we call the RROP *linear recovery robust program*, short LRP.

For every demand node i, let $[h_i^{min}, h_i^{max}]$ be the uncertainty interval for the demand values. We assume for this moment that all uncertainty intervals have the same length $\triangle$.

Let $S := S^1 := \{h_s : \ \|h_s - h^{min}\|_1 \leq \triangle, \ h_s - h^{min} \geq 0\}$ be the considered scenario set, as proposed in [91]. Here $\|\cdot\|_1$ denotes the L^1-norm, where $\|x\|_1 = \sum_{i=1}^{m} |x_i|$ for $x = (x_1, \ldots, x_m)^\top \in \mathbb{R}^m$. Note that S^1 should be a subset of $[h_1^{min}, h_1^{max}] \times \ldots \times [h_l^{min}, h_l^{max}]$, as if it was not, the worst case h^{max} would always have to be recovered which leads to over-conservative solutions.

We consider the following LRP:

$$\begin{aligned}
\min_x\ & c^\top x \\
\text{s.t. } & Ax \le b \\
& A_{eq}x = b_{eq} \\
& Tx \ge h^{min} \\
& x \ge 0 \\
& \forall i \in \{0, \dots, l\}: \\
& \qquad\qquad Tx + Wy_i = h^{min} + \triangle e_i \\
& \qquad\qquad\qquad d^\top y_i \le D \qquad y_i \ge 0
\end{aligned} \tag{8.7}$$

Here e_i denotes the i-th l-dimensional unit vector for $i = 1, \dots, l$ and the l-dimensional zero vector for $i = 0$.

Theorem 8.1. *If $S = S^1$, an optimal solution of problem (8.7) is also an optimal solution to problem (8.6).*

Proof. Let $(x^*, y_0^*, \dots, y_l^*)$ be an optimal solution of problem (8.7). Let $h_i = h^{min} + \triangle e_i,\ i = 0, \dots, l$ and $S^{LRP} = \{h_i :\ i = 0, \dots, l\}$. Obviously, problem (8.7) is equivalent to the RROP for the scenario set S^{LRP}.

Furthermore, we have that S^1 is the set of convex combinations of vectors $h_i,\ i = 0, \dots, l$, i.e. $S^1 = \{h :\ h = \sum\limits_{i=0}^{l} \lambda_i h_i, \sum\limits_{i=0}^{l} \lambda_i = 1, \lambda_i \ge 0, i = 0, \dots, l\}$. Therefore, we have $S^{LRP} \subseteq S^1$ and consequently, it holds that $c^\top x \ge c^\top x^*$ for all feasible solutions x of problem (8.6).

Finally, we must show that x^* is feasible for problem (8.6). It is sufficient to show that for every $h \in S^1$, x^* can be recovered within the given recovery budget D.

Let $h := (h_1, \dots, h_l) \in S^1$ be arbitrarily chosen. Let y_i be the recovery vector corresponding to h_i. Then there exists $\lambda_i \ge 0$, $i = 0, \dots, l$ such that $h = \sum\limits_{i=0}^{l} \lambda_i h_i$ and $\sum\limits_{i=0}^{l} \lambda_i = 1$.

Let $y = \sum\limits_{i=0}^{l} \lambda_i y_i$. Then

$$\begin{aligned} Tx^* + Wy &= (\sum_{i=0}^{l} \lambda_i) Tx^* + W(\sum_{i=0}^{l} \lambda_i y_i) \\ &= \sum_{i=0}^{l} \lambda_i (Tx^* + Wy_i) \\ &= \sum_{i=0}^{l} \lambda_i h_i \\ &= h. \end{aligned} \tag{8.8}$$

and

$$\begin{aligned} d^\top y &= d^\top (\sum_{i=0}^{l} \lambda_i y_i) \\ &= \sum_{i=0}^{l} \lambda_i (d^\top y_i) \\ &\leq \sum_{i=0}^{l} \lambda_i D \\ &= D. \end{aligned} \tag{8.9}$$

With (8.8) and (8.9), we can show that for arbitrary $h \in S^1$, x^* can be recovered within the given budget by a recovery vector y and therefore x^* is feasible for S^1. Since $c^\top x \geq c^\top x^*$ for all feasible solutions x of problem (8.6), as we showed above, x^* is an optimal solution of problem (8.6). □

Theorem 8.1 can analogously be extended to LRPs with a scenario set S which is a convex polytope. Then it is sufficient to solve the corresponding LRP where only the extreme points of S are considered as scenarios.

In practice, the uncertainty intervals will vary in length from node to node. Let $\triangle_i$ be the length of the uncertainty interval of node i. By scaling row i of T, W and h by $\frac{\triangle}{\triangle_i}$, we can transform the optimization problem into an equivalent problem and have equal uncertainty interval lengths for all demand nodes.

8.3.2 Coincidental Covering

The scenario set S^1 corresponds to the ball around h^{min} in the 1-norm with radius $\triangle$, truncated to the positive orthant relative to h^{min}. As we could see in Subsection 8.3.1, the LRP corresponding to the scenario set S^1 can easily be solved. In practice, however, the uncertainty set S^1 can be too small. Given a minimum demand value and a possible fluctuation $\triangle$ for every demand node, the largest (scaled) scenario set of interest would be $S^\infty := \{h_s : \ \|h_s - h^{min}\|_\infty \leq \triangle, \ h_s - h^{min} \geq 0\}$. For example, the case where the demand is maximal at two or more demand nodes is contained in S^∞, but not in S^1: If in S^1 at one demand node i, the maximum demand h_i^{max} is considered, the demand at all other demand nodes can only be minimal.

However, in many cases, we experience the so-called *coincidental covering* effect. According to S. Stiller, see [91], a robust solution for a certain scenario set possibly also covers other scenarios outside the scenario set, that originally were not intended to be covered.

We will discuss the effect of coincidental covering by means of a supply chain network with two demand nodes.

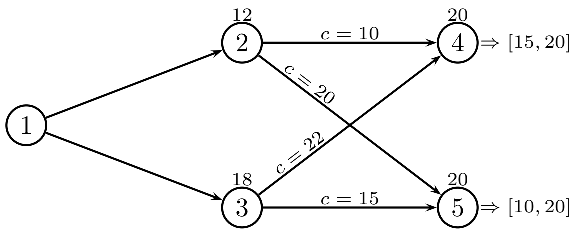


Figure 8.9: Recoverable robustness - 2-dimensional example: Network

The penalty cost for delivery deficiency is 24 for both demand nodes, the storage cost for surplus delivery is 4. The recovery budget is 180. In this model, the recovery budget is the amount which is available for covering the sum of penalty and storage costs in this network.

As we have a possible demand uncertainty of 5 units for demand node 4 and a possible demand uncertainty of 10 units for demand node 5, we have to scale the second row of the matrices T and W as well as h by the factor $\frac{1}{2}$ in order to obtain a common $\triangle$-value for both demand nodes, here $\triangle = 5$.

As scenario set, we consider again the scenario set S^1 as illustrated in the figure below.

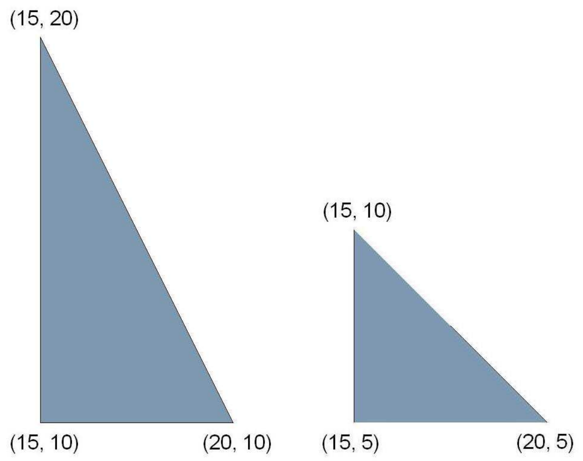


Figure 8.10: Recoverable robustness - 2-dimensional example: Unscaled and scaled scenario set

The optimal recovery robust solution of the 2-dimensional example is $x = (12, 0, 3, 12.5)^\top$:

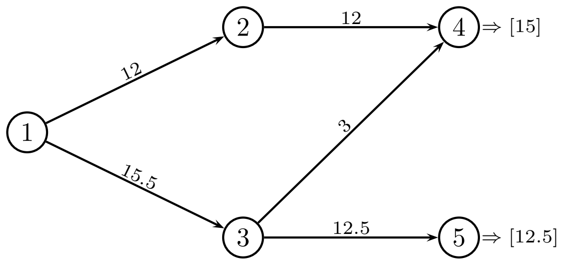


Figure 8.11: Recoverable robustness - 2-dimensional example: Optimal solution

With d and D given, we can determine all feasible recovery vectors y via the inequality $d^\top y \leq D$. With the scaled matrix W, we obtain that the images of all admissible recovery vectors, i.e. Wy, lie in the convex hull of $(7.5,\ 0)^\top$, $(0,\ 3.75)^\top$, $(-45,\ 0)^\top$ and $(0,\ -22.5)^\top$.

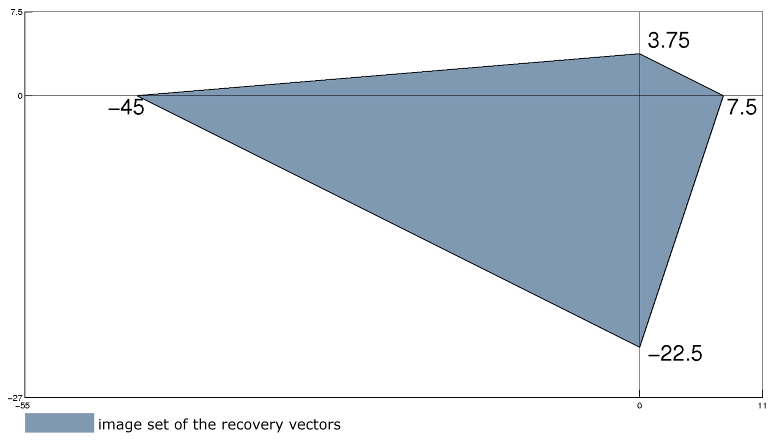


Figure 8.12: Recoverable robustness - 2-dimensional example: Image set of the recovery vectors

The recovery condition $Tx + Wy = h$ of the RROP can be rewritten as follows:

$$Wy = -Tx + h \tag{8.10}$$

This means that for all recoverable solutions x, the negative image of x, translated by h, must lie in the image set of the recovery vectors. As h is subject to uncertainty, we have that $-Tx + h_s$ must lie within the image set of the recovery vectors for all $h_s \in S$. In our example, this means that $-Tx + h^{min}$ plus the triangle corresponding to the uncertain demand scenarios S^1, see Figure 8.10, must lie within the image set of the recovery vectors.

In Figure 8.13, we illustrate again the image set of the recovery vectors Wy. Furthermore, we determine the translated image set of the recoverable solutions $-Tx + h$. This image set consists of all points where the complete uncertainty region (red triangle) can be appended such that it still lies completely inside the image set Wy.

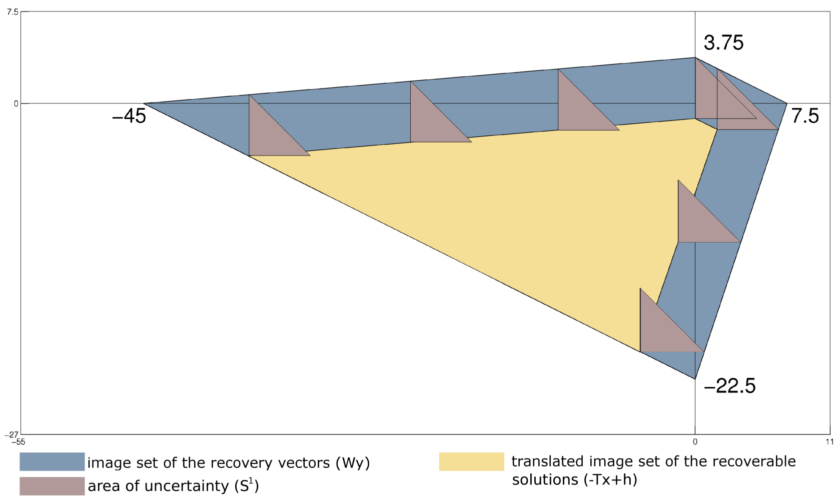

7.5
0
-27
-55
0
11
3.75
-45
7.5
-22.5
image set of the recovery vectors (Wy)
area of uncertainty (S¹)
translated image set of the recoverable solutions (-Tx+h)

Figure 8.13: Recoverable robustness - 2-dimensional example: Translated images of the recoverable solutions S^1

Let $S^\infty := \{h_s : \|h_s - h^{min}\|_\infty \leq \triangle,\ h_s - h^{min} \geq 0\}$ be the maximum (scaled) scenario set for the demand values. Let us suppose for the moment, that the uncertainty scenario set is S^∞. If we choose S^∞ as scenario set, the translated image set of the recoverable solutions is smaller than in case of S^1, having the following shape:

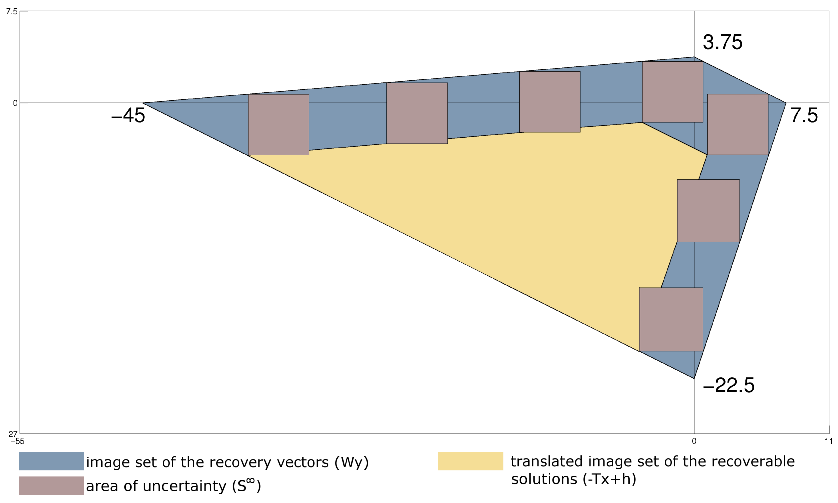

7.5
0
-27
-55
0
11
3.75
-45
7.5
-22.5
image set of the recovery vectors (Wy)
area of uncertainty (S^∞)
translated image set of the recoverable solutions (-Tx+h)

Figure 8.14: Recoverable robustness - 2-dimensional example: Translated images of the recoverable solutions S^∞

Now we return to the scenario set S^1. The following figure illustrates the (not translated) image set of the recovery vectors as well as the images of the recoverable solutions of the 2-dimensional example.

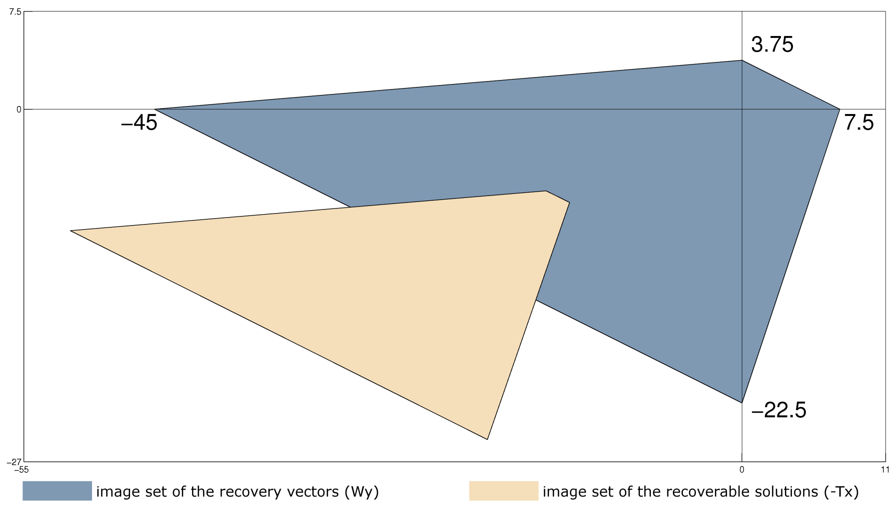


Figure 8.15: Recoverable robustness - 2-dimensional example: Images of the recoverable solutions

As stated above, a solution x is recoverable, if the negative image of x, translated by h^{min} plus the demand triangle, lies within the image set of the recovery vectors. In practice, for a recoverable solution x, often "more" than the demand triangle lies within the image set of the recovery vectors. This corresponds to demand values that are *coincidentally covered*.

Let again $S^\infty := \{h_s : \ \|h_s - h^{min}\|_\infty \leq \triangle, \ h_s - h^{min} \geq 0\}$ be the maximum (scaled) scenario set for the demand values. When calculating optimal solutions for the RROP with scenario set S^1, parts of $S^\infty \setminus S^1$ are coincidentally covered.

The following figure illustrates the optimal solution and the (coincidentally) covered uncertainty set for the 2-dimensional example.

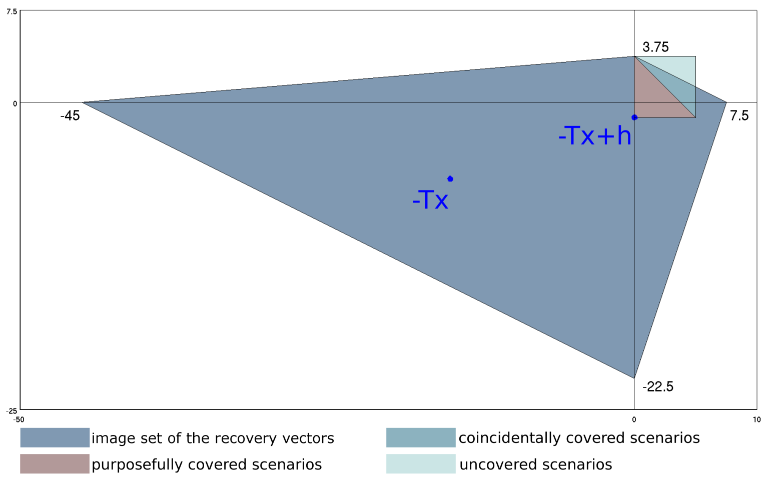


Figure 8.16: Recoverable robustness: 2-dimensional example: Optimal solution and covered uncertainty set

In the two-dimensional example, the scenario set S^1 contains 50% of the demand scenarios in S^∞. However, the optimal solution x covers 75% of the scenarios in S^∞, that means that 25% of S^∞ are covered coincidentally and only 25% of S^∞ are not covered.

In practice, the coincidental covering range reaches from no coincidental covering to a total covering of S^∞.

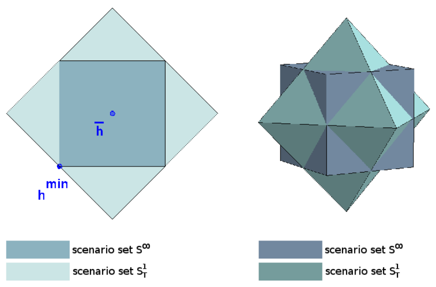
As it can be seen in Figure 8.16, the region which is coincidentally covered is not restricted to S^∞: In our example, the (scaled) scenario point $(7.5, 0)$, for example, which is not contained in S^∞, is also coincidentally covered by the solution x.

8.3.3 Translation of the Demand Scenarios

As stated in the section before, the scenario set S^∞ possibly contains scenarios that are not (coincidentally) covered by the optimal solution for scenario set S^1. In this section, we propose a translation of the demand scenarios such that the covered scenario set is more conservative, i.e. that it contains "more" scenarios of S^∞ than S^1 does. Additionally, we want to keep the structure of S^1 such that the calculation of the optimal solution of the LRP does not become more complicated.

Instead of using h^{min} as base demand vector, we set $\bar{h} := \frac{1}{l} \sum_{i=1}^{l} h_i$ as base demand, where $(h_1, \ldots, h_l)$ is the demand vector of the given scenario. We maintain $\triangle$ as radius for the new scenario set $S^1_T := \{h_s : \ \|h_s - \bar{h}\|_1 \leq \triangle\}$. Note that contrary to the definition of S^1, h_s is not necessarily greater or equal to $\bar{h}$, as otherwise scenarios with low demand values like the scenario h^{min} would not be covered.

The following figure illustrates the relation between the original scenario set S^∞ and S^1_T. The (scaled) scenario set S^∞ corresponds to a square in dimension 2 and a cube in dimension 3, with origin h^{min} and edge length $\triangle$. The scenario set S^1_T is a square rhombus in dimension 2 and a regular octahedron in dimension 3 with center $\bar{h}$ and radius $\triangle$.



(a) Dimension 2 **(b)** Dimension 3

Figure 8.17: Recoverable robustness: Scenario sets S^∞ and S^1_T

For dimensions higher than 2, it holds that $S^\infty \not\subseteq S^1_T$. In dimension 2, the intersection of S^∞ and S^1_T is the complete square S^∞ , while in dimension 3, the intersection is a cuboctahedron.

Note that the new scenario set S^1_T contains new extreme points that are included in neither S^1 nor S^∞. However, as all extreme points of the uncertainty set are considered as scenarios in the recovery stage, these extreme points have an influence on the recovery stage. The extreme point of S^1_T in the positive orthant corresponds to higher demand values at all demand nodes than intended in S^1 or S^∞. In practice, of course, we do not want to obtain solutions that actually deliver demand corresponding to demand values of $S^1_T \setminus S^\infty$. Therefore we add an additional constraint to the LRP that guarantees that not more than h^{max} units are delivered to the demand nodes. Note that nevertheless the extreme points of S^1_T are considered in the recovery stage.

Consequently, as new extreme demand scenarios have to be recovered for scenario set S^1_T, the recovery budget D possibly has to be increased when using scenario set S^1_T instead of S^1. For the 2-dimensional example, the recovery budget D must be increased to 300 in order to ensure the recovery of all possible scenarios of S^1_T.

As we consider a 2-dimensional uncertainty set, 100% of the scenarios in S^∞ are covered by S^1_T.

We have to note that when using the scenario set S^T_1 for a problem with more than two demand nodes, it is not ensured that lower demand values, including h^{min}, can be recovered within the given recovery budget (see Figure 8.17).

The question arises, which scenario set is the better choice in practice. We recommend the use of S^1 for supply chain problems where only small deviations in demand are expected, because these deviations are likely to be covered by the scenario set S^1. In case that greater deviations in demand are to be expected, we recommend the use of S^1_T, as the solutions generated using this scenario set are more conservative in the sense that more units are delivered to the demand nodes.

8.3.4 Extension to Multicommodity Flows

The extension of the general recoverable robustness concept to multicommodity flows is straightforward. Let p be the number of commodities and κ the bundle capacity. Then the multicommodity RROP can be formulated as follows:

$$
\begin{aligned}
\min_x \ & \sum_{k=1}^{p} {c^k}^\top x^k && \\
\text{s.t. } & A^k x^k \leq b^k && k = 1, \dots, p \\
& \sum_{k=1}^{p} A^k x^k \leq \kappa && \\
& A^k_{eq} x^k = b^k_{eq} && k = 1, \dots, p \qquad (8.11) \\
& T^k x \geq {h^k}^{min} && k = 1, \dots, p \\
& \forall h_s :== (h^1_s, \dots, h^p_s) \in S \ \exists y_s \in \mathbb{R}^{2l \times p} : && \\
& \qquad T^k x^k + W^k y^k_s = h^k_s, \ k = 1, \dots, p && \\
& \qquad \sum_{k=1}^{p} {d^k}^\top y^k_s \leq D &&
\end{aligned}
$$

The choice of the scenario set is critical. The standard scenario set $S^1 := \{h_s = (h^1_s, \dots, h^p_s) : \|h_s - h^{min}\|_1 \leq \triangle, h_s - h^{min} \geq 0, h^{min} := ({h^1}^{min}, \dots, {h^p}^{min})\}$

covers the cases with maximum demand for one commodity at one demand node at a time. However, a case with maximum demand for two commodities at one demand node would not be covered by S^1. For a large number of commodities it seems not likely that there is only one deviation for one product at a time, such that S^1 possibly is no suitable scenario set.

We propose a new scenario set S^{multi}. Let h_k^{max} denote the demand vector with maximum demand at all demand nodes for commodity k and minimum demand at all demand nodes for commodities v where $v = 1, \dots, p$ and $v \neq k$. Furthermore, let $h_0^{max} := h^{min}$. Then let $S^{multi} := \{h_s = \sum_{k=0}^{p} \lambda_k h_k^{max},\ 0 \leq \lambda_k \leq 1,\ k = 0, \dots, p,\ \sum_{k=0}^{p} \lambda_k = 1\}$ be the multicommodity scenario set.

This scenario set S^{multi} covers all scenarios where the demand is maximal for one commodity and minimal for all other commodities as well as mixed cases. This scenario set is not very conservative, as for example scenarios where the demand is maximal for two commodities are not covered. However, in practice, the coincidental covering effect (see Subsection 8.3.2) often helps to cover more scenarios than actually intended.

As in the single-commodity case - see Subsection 8.3.3 - the scenario set S^{multi} can be translated in order to obtain a more conservative scenario set. Besides, the definition of other scenario sets is possible, e.g. a scenario set with maximum demand for all commodities at one demand node and minimum demand for all commodities in the other demand nodes.

8.3.5 Recoverable Robustness Example

In this subsection, we apply the recoverable robustness approach to a supply chain management example. We consider a production and transportation network with 18 nodes and 27 arcs where five nodes are demand nodes.

Again, we consider the same network structure as in Figures 7.6, 7.10 and 8.6.

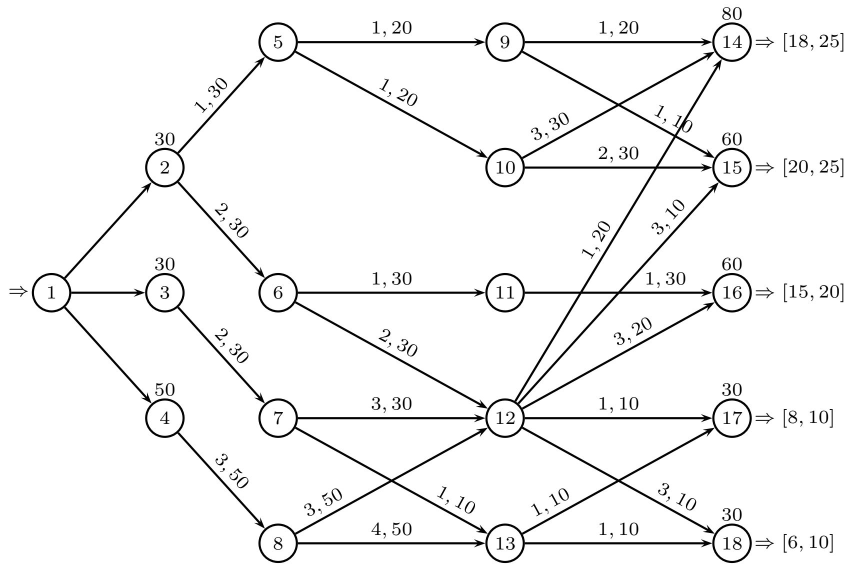


Figure 8.18: Recoverable robustness example: Network

The numbers on the arcs denote costs and capacity restrictions. The numbers on the nodes denote capacity restrictions on the nodes: If node i has a capacity of $v_i \geq 0$, then the sum of the incoming (outgoing) flow units over all incoming (outgoing) arcs must be less or equal to v_i.

The given capacity restrictions on the arcs correspond to capacity restrictions in the production and transportation process. The capacity restrictions on the nodes correspond to storing capacity restrictions at the respective location.

The costs on the arcs correspond to the costs for the respective supply chain stage, i.e. production and transportation costs.

At the demand nodes, storage costs of 4 per surplus unit and penalty costs of 20 per unavailable unit are imposed.

As scenario set, we consider S^1. We scale the demand scenarios such that $\triangle = 7$.

For the calculation of the (coincidental) covering percentage, we use the software polymake, see [47], in order to calculate the intersections and volumes of the involved polyhedra.

Minimum recovery budget First, we solve the LRP with the minimum possible recovery budget. The minimum possible recovery budget can be obtained by solving a problem of the form

$$
\begin{aligned}
\min_{x,y_i,D}\ & D \\
\text{s.t. } & Ax \leq b \\
& A_{eq}x = b_{eq} \\
& Tx \geq h^{min} \\
& \forall i \in \{0,\dots,l\}: \\
& \qquad Tx + Wy_i = h^{min} + \triangle e_i \\
& \qquad d^\top y_i \leq D
\end{aligned}
\tag{8.12}
$$

In our example problem, we have a minimum recovery budget of $D = 75.333$. The following table lists the number of demand units that are transported to the demand nodes in the optimal solution.

node 14	node 15	node 16	node 17	node 18
24.0000	24.3333	19.3333	9.8333	9.5000

Table 8.3: Recoverable robustness example: Minimum recovery budget - transport to demand nodes

As we can see, the numbers of delivered units are close to the upper demand bounds for all demand nodes. This is due to the facts that penalty costs are much higher than storage costs and that the recovery budget is strongly restricted.

Due to the high number of delivered units, the production and transportation cost, for simplicity referred to as production cost in the following, is rather high, such that we have a high "expected" total cost (production cost + recovery budget).

production cost	490.33
recovery budget	75.33
total cost	565.67

Table 8.4: Recoverable robustness example: Minimum recovery budget - production, recovery and total cost

We examine the (coincidental) covering of the demand scenarios in S^∞ by this solution. As we noticed before, the optimal solution with a minimum recovery budget is rather conservative, i.e. many demand units are delivered. Hence it is not surprising that this solution covers 100% of the scenarios in S^∞.

Increased recovery budget Now we increase the recovery budget to $D = 90$ in order to obtain a less conservative solution. The delivered demand units are listed in the following table:

node 14	node 15	node 16	node 17	node 18
20.2222	20.5556	15.5556	8.0000	6.0000

Table 8.5: Recoverable robustness example: Increased recovery budget - transport to demand nodes

For all demand nodes, the number of delivered units is smaller than before. This is due to the facts that less delivered units mean less production costs and that the recovery budget is high enough to compensate potential deficiencies. The production cost is less than for the minimum recovery budget, where less deficiency can be compensated.

production cost	389.50
recovery budget	90.00
total cost	479.50

Table 8.6: Recoverable robustness example: Increased recovery budget - production, recovery and total cost

As the optimal solution for the increased recovery budget is less conservative, the (coincidental) covering of the scenarios in S^∞ is smaller than before: 19.92%.

Recovery budget as optimization variable In practice, the recovery budget sometimes is not given as a fixed value. As we could see before, it is difficult to determine a reasonable recovery budget. One possible approach to this difficulty is to switch over from a fixed recovery budget to a variable recovery budget. This means that we consider D as an optimization variable and minimize over the total cost $c^\top x + D$ instead of the original network cost $c^\top x$ only.

The optimization terminates with $D = 106.67$ as an optimal recovery budget. The following table lists the delivered units at the demand nodes for the optimal solution:

node 14	node 15	node 16	node 17	node 18
19.6667	20.0000	15.0000	8.0000	6.0000

Table 8.7: Recoverable robustness example: Recovery budget as optimization variable - transport to demand nodes

The number of delivered units is close to the lower bound for node 14 and is the lower bound for nodes 15 to 18. The relatively high recovery budget allows the acceptance of high penalty costs while the production cost is decreased.

production cost	340.67
recovery budget	106.67
total cost	447.33

Table 8.8: Recoverable robustness example: Recovery budget as optimization variable - production, recovery and total cost

The (coincidental) covering of the scenarios in S^{∞} is only 5.43% in this case.

Translated demand scenarios In order to obtain a conservative solution, we translate the demand scenarios according to the translation given in Subsection (8.3.3). Again, we consider the recovery budget D as an optimization variable, as we do not want an overly conservative solution with minimum recovery budget.

The optimal solution has a recovery budget of $D = 106.67$ and delivers the following number of units to the demand nodes:

node 14	node 15	node 16	node 17	node 18
21.0667	21.0000	16.0000	8.0000	6.0000

Table 8.9: Recoverable robustness example: Translated demand scenarios - transport to demand nodes

We observe that more demand units are delivered than for the non-translated demand scenarios. Consequently, we have higher production and total costs:

production cost	377.93
recovery budget	106.67
total cost	484.60

Table 8.10: Recoverable robustness example: Translated demand scenarios - production, recovery and total cost

For the translated demand scenarios, we have a (coincidental) covering of 35.30% of the demand scenarios in S^{∞}, which is significantly more than for the non-translated demand scenarios.

Summary The following tables summarizes the covering and cost values of the four variations.

	minimum recovery budget	increased recovery budget	variable recovery budget	translated demand scenarios
(coincidental) covering	100%	19.92%	5.43%	35.30%
production cost	490.33	389.50	340.67	377.93
recovery budget	75.33	90.00	106.67	106.67
total cost	565.67	479.50	447.33	484.60

Table 8.11: Recoverable robustness example: Summary

Obviously, we always have to find a trade-off between low production cost and low recovery budget. For a low recovery budget, on the one hand the production and transportation cost is significantly higher than for a higher recovery budget, but on the other hand the percentage of (coincidental) covering is also significantly higher.

In practice, we have to decide between the different possibilities as the case arises. In cases where large deviations from the nominal demand values are to be expected, a more conservative solution would be advisable. In cases where only small deviations from the nominal demand values are to be expected, a less conservative solution can be expected to cause less expenses.

8.4 Summary and Conclusions

In this chapter, we examined network flow problems with uncertain demands, where for every demand node, a nominal demand value as well as a maximum possible extra demand value is given. Furthermore, we assumed that additional information in the form of penalty costs for supply deficiency and storage costs for surplus delivery is given.

In Section 8.2, an ACO based heuristic model for network flow problems with uncertain demands was presented. The underlying network is transformed by including new nodes and arcs. In each ant iteration, current demand values are randomly generated within the given bounds. Two sets of ants, one corresponding to the nominal demand units and one corresponding to the extra demand units, successively construct solutions in the extended network, leading to a robust solution.

In Section 8.3, the recoverable robustness model proposed by S. Stiller et al. ([64], [91]) was adapted to network flow problems with uncertain demands. The optimization is processed in two phases, a planning phase and a recovery phase. The compensation costs, i.e. the total sum of penalty and storage costs, are limited by a recovery budget. We showed that in practice, often more demand scenarios than originally comprised in the scenario set are covered by the optimal solution of recoverable robustness problem, which is called *coincidental covering* effect.

Both approaches are applicable to single-commodity and multicommdity network flow problems with uncertain demands. In contrast to the ant algorithm proposed in Section 8.2, the level of robustness can not be controlled via a robustness parameter in the recoverable robustness approach.

Chapter 9

Stochastic Programming: Uncertain Demand

In order to model optimization problems with uncertain input data, Stochastic Programming is a widely used alternative to Robust Optimization. In this chapter, we examine network flow problems with uncertain demands. First, we shortly recapitulate the concept of two-stage linear recourse problems following the book of J. Birge and F. Louveaux ([18]) and then extend the model to multicommodity problems. Subsequently, we discuss a special recourse scheme for supply chain problems.

9.1 Two-Stage Linear Recourse Problems

The two-stage model is a scenario-based approach which separates an optimization problem with uncertain input data in two stages. In the first stage, a scenario independent decision is made before information about the actual data realization is known. In the second stage, after uncertainty is disclosed, a second decision - the so-called *recourse decision* - is made in order to compensate violations of the constraints. In the following, we will consider fixed recourse, i.e. problems where the compensation matrix is not subject to uncertainty.

The special feature of the two-stage method is the compensation of constraint violations in the second stage. In general, the two-stage method is applicable to both uncertain cost values and uncertain demand values. However, in case of uncertain cost values, the first-stage solutions are always feasible for all scenarios. In contrast to that, constraint violations in the first stage are prevalent in problems with uncertain demand values. Therefore, we concentrate on uncertain demand values, as for this kind of problem, the full potential of the two-stage method can be demonstrated.

9.1.1 Problem Formulation

According to [18], a two-stage linear recourse problem with fixed recourse is given as

$$\begin{aligned}\min\ & c^\top x + \mathbb{E}_\xi[\min q(\omega)^\top y(\omega)]\\ \text{s.t. } & Ax = b\\ & T(\omega)x + Wy(\omega) = h(\omega)\\ & x \geq 0\\ & y(\omega) \geq 0\end{aligned} \tag{9.1}$$

Here $c \in \mathbb{R}^{m_1}$, $b \in \mathbb{R}^{n_1}$ and $A \in \mathbb{R}^{n_1 \times m_1}$ correspond to the first stage decision x. These vectors and matrices are not subject to uncertainty. $W \in \mathbb{R}^{n_2 \times m_2}$ is called the *recourse matrix* and is assumed to be fixed here. In the second stage, random events $\omega \in \Omega$ may realize. For a given ω, the second stage decision is represented by $y(\omega)$. The second stage data $q(\omega) \in \mathbb{R}^{m_2}$, $h(\omega) \in \mathbb{R}^{n_2}$ and $T(\omega) \in \mathbb{R}^{n_2 \times m_2}$ are known vectors and matrices, respectively, for a given realization ω. T is called the *technology matrix.* $\mathbb{E}_\xi$ denotes the expected values dependent on the random vector $\xi := (T, h, q)$.

Let

$$Q(x, \xi(\omega)) := \min_y \{q(\omega)^\top y | Wy = h(\omega) - T(\omega)x,\ y \geq 0\} \tag{9.2}$$

be the *recourse function* and

$$\mathcal{Q}(x) := \mathbb{E}_\xi Q(x, \xi(\omega)). \tag{9.3}$$

the expected value of the recourse function.

Then problem (9.1) is equivalent to

$$\begin{aligned}\min\ & c^\top x + \mathcal{Q}(x)\\ \text{s.t. } & Ax = b\\ & x \geq 0\end{aligned} \tag{9.4}$$

Problem (9.4) is called the *deterministic equivalent problem.*

If the random vector ξ has K possible realizations $\xi_k = (T_k, h_k, q_k)$, where realization k has a probability of $p_k \geq 0$, $k = 1, \ldots, K$ with $\sum_{k=1}^{K} p_k = 1$, the two-stage problem can be represented in the so-called *extensive form*:

$$\begin{aligned}
\min\ & c^\top x + \sum_{k=1}^{K} p_k q_k^\top y_k \\
\text{s.t. } & Ax = b \\
& T_k x + W y_k = h_k && k = 1, \ldots, K \\
& x \geq 0 \\
& y_k \geq 0 && k = 1, \ldots, K.
\end{aligned} \tag{9.5}$$

There exists a variety of possibilities to model the recourse functions, see [18] for details.

9.1.2 The L-Shaped Method

In [18] and [22], various solution methods for two-stage linear recourse problems are presented. One of the most established solution methods for two-stage linear recourse problems where ξ has finite support, is the L-shaped method, which is a cutting plane technique. In the L-shaped algorithm, the stochastic optimization problem is separated into a master problem in x and several subproblems which are needed to exactly evaluate the recourse function.

The L-shaped algorithm consists of several steps. In step 1, the master problem is solved. Step 2 checks the feasibility of x with the help of the subproblems. If necessary, a feasibility cut is added to the master problem. For a feasible x, optimality is checked in step 3. If required, optimality cuts are added to the master problem. The procedure is repeated until x is feasible and optimal for problem (9.4).

For details concerning the L-shaped method, see [18].

9.1.3 Application to Supply Chain Problems

As the two-stage model and the L-shaped algorithm are generally applicable models, they can be applied to single-commodity as well as multicommodity problems. If the (unmodified) L-shaped algorithm is applied to a multicommodity problem, however, the feasibility cuts and optimality cuts that are added to the master problem in steps 2 and 3, respectively, concern all commodities, no matter if each individual commodity violates feasibility or optimality. This means that redundant constraints are added and let the optimization problem grow unnecessarily. However, if the recourse functions for different commodities are independent, we can add feasibility and optimality cuts only for those products that violate feasibility and optimality. Hence the resulting optimization problems in the L-shaped algorithm grow less quickly.

In the following chapter, we will examine multicommodity two-stage problems and formulate the L-shaped method for multicommodity problems with independent recourse.

9.2 Extension of the L-Shaped Algorithm to Multicommodity Flows

We consider a multicommodity production and transportation planning supply chain problem with uncertain demand. Let N be the number of commodities and K the number of scenarios. Let p_k^n denote the probability of realization k for commodity n where $\sum\limits_{k=1}^{K} p_k^n = 1,\ n = 1, \dots, N$. We consider independent recourse functions for different products. Then we can formulate the supply chain problem as a two-stage linear problem in extensive form:

$$
\begin{aligned}
\min\ & \sum_{n=1}^{N} c^{n\top} x^n + \sum_{n=1}^{N}\sum_{k=1}^{K} p_k^n q_k^{n\top} y_k^n \\
\text{s.t. } & A^n x^n \leq b^n && n = 1, \dots, N \\
& \sum_{n=1}^{N} A^n x^n \leq \kappa && \\
& T_k^n x^n + W^n y_k^n = h_k^n && k = 1, \dots, K,\ n = 1, \dots, N \\
& x^n \geq 0 && n = 1, \dots, N \\
& y_k^n \geq 0 && k = 1, \dots, K,\ n = 1, \dots, N.
\end{aligned}
\tag{9.6}
$$

In Problem (9.6), the bundle constraints of the multicommodity problem are given in $\sum\limits_{n=1}^{N} A^n x^n \leq \kappa$, where $\kappa \in \mathbb{R}^{n_1}$.

Note that in the context of multicommodity NFPs, the matrix A^n describes possible transportation paths in the given network and b^n denotes the capacity restrictions, where every line of A^n corresponds to a capacity bound (the corresponding entry of b^n). Therefore, the structure of the matrix A^n is not the structure of a node-arc incidence matrix.

Note that in most practical cases, A^n is identical for $n = 1, \dots, N$.

In the standard formulation of two-stage linear recourse problem with fixed recourse as given in [18], see (9.1), the constraints $Ax = b$ are given in equality form. In our special case, all corresponding constraints are inequality constraints, which can be transformed to equality constraints by the introduction of slack variables. On the other hand, equality constraints $Ax = b$ can be formulated as inequality constraints as $Ax \leq b$ and $-Ax \leq -b$. As we want to apply the theory presented in this Section to multicommodity NFPs later, we will formulate the constraints as inequality constraints in the following.

9.2.1 The Multicommodity L-Shaped Method

The L-shaped method, as given in [18], can be extended for the multicommodity two-stage problem (9.6). The separation of master problem and subproblems as well as the basic steps 1 (solve the master problem), 2 (check feasibility, add feasibility cuts) and 3 (check optimality, add optimality cuts) can be retained.

The multicommodity L-shaped algorithm proceeds as follows:

Algorithm 9.1 Multicommodity L-shaped Algorithm

```
1: step 0: set $r = s = \nu = 0$ and $\Omega^D = \emptyset$, $\Omega^E = \emptyset$
2: step 1: set $\nu = \nu + 1$
3: solve the LP
```

$$\min \quad z = \sum_{n=1}^{N} c^{n\top} x^n + \sum_{n=1}^{N} \theta^n \tag{9.7}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} A^n x^n \leq \kappa,$$

$$A^n x^n \leq b^n, \qquad n = 1, \dots, N$$

$$D_l^n x^n \geq d_l^n, \qquad l = 1, \dots, r, \quad n = 1, \dots, N : (l, n) \in \Omega^D \tag{9.8}$$

$$E_l^n x^n + \theta^n \geq e_l^n, \qquad l = 1, \dots, s, \quad n = 1, \dots, N : (l, n) \in \Omega^E \tag{9.9}$$

$$x^n \geq 0, \quad \theta^n \in \mathbb{R} \qquad n = 1, \dots, N.$$

```
4:  let $(x^\nu, \theta^\nu)$ be an optimal solution where $x^\nu := (x^{1^\nu}, \dots, x^{N^\nu})^T$ and $\theta^\nu := (\theta^{1^\nu}, \dots, \theta^{N^\nu})^T$.
    if the condition (9.9) does not exist, set $\theta^{n^\nu} = -\infty$ for $n = 1, \dots, N$ and do not consider $\theta^{n^\nu}$ in the calculation of $x^\nu$.
5:  step 2:
6:  for $k = 1, \dots, K$, $n = 1, \dots, N$ do
7:      solve the LP
```

$$\begin{aligned} \min \quad & w_k^{n\prime} = e^\top v_+^n + e^\top v_-^n \\ \text{s.t.} \quad & W^n y^n + I v_+^n - I v_-^n = h_k^n - T_k^n x^{n\nu} \\ & y^n \geq 0, \quad v_+^n \geq 0, \quad v_-^n \geq 0, \end{aligned} \tag{9.10}$$

```
        where $e^\top = (1, \dots, 1)$ and $I$ is the identity matrix
8:      if $w_k^{n\prime} > 0$ then
9:          set $k' := k$
10:         break: goto line 13
11:     end if
12: end for
13: if $w_k^{n\prime} > 0$ for a $k \in \{1, \dots, K\}$ and an $n \in \{1, \dots, N\}$ then
14:     for all $n \in 1, \dots, N$ with $w_{k'}^{n}{}' > 0$ do
15:         let $(\sigma_{k'}^n)^\nu$ be the corresponding dual solution
16:         define $D_{r+1}^n := (\sigma_{k'}^n)^\nu T_{k'}^n$
17:         define $d_{r+1}^n := (\sigma_{k'}^n)^\nu h_{k'}^n$
18:         set $\Omega^D := \Omega^D \cup \{(r+1, n)\}$
19:     end for
20:     set $r := r + 1$
21:     goto step 1 (line 2)
22: else
23:     goto step 3 (line 25)
24: end if
25: step 3:
26: for $k = 1, \dots, K$, $n = 1, \dots, N$ do
27:     solve the LP
```

$$\begin{aligned} \min \quad & q_k^n y^n \\ \text{s.t.} \quad & W^n y^n = h_k^n - T_k^n x^{n\nu}, \\ & y^n \geq 0, \end{aligned} \tag{9.11}$$

```
28: end for
```

```
29: let $\pi_k^{n\nu}$ be the corresponding dual solution to the optimal solution of problem
    $k$ of type (9.11) for commodity $n$
30: define $E_{s+1}^n = \sum_{k=1}^{K} p_k \cdot (\pi_k^{n\nu})^\top T_k^n, \quad n = 1, \dots, N$
31: define $e_{s+1}^n = \sum_{k=1}^{K} p_k \cdot (\pi_k^{n\nu})^\top h_k^n, \quad n = 1, \dots, N$
32: let $w^{n\nu} = e_{s+1}^n - E_{s+1}^n x^{n\nu}, \quad n = 1, \dots, N$
33: **if** for all $n = 1, \dots, N$: $\theta^{n\nu} \geq w^{n\nu}$ **then**
34:     STOP.
35:     $x^\nu$ is an optimal solution
36: **else**
37:     **for all** $n = 1, \dots, N$ with $\theta^{n\nu} < w^{n\nu}$ **do**
38:         set $\Omega^E = \Omega^E \cup \{(s+1, n)\}$, $s = s + 1$
39:     **end for**
40:     goto **step 1** (line 2)
41: **end if**
```

As indicated before, the feasibility of first-stage decisions x^n for commodities $n = 1, \dots, N$ is examined in step 2. If x^n is not second-stage feasible for one or several n, feasibility cuts are added. In step 3, the optimality of x^n is examined and one or several optimality cuts are added.

The major difference of the multicommodity L-shaped algorithm when compared to the single-commodity version is the fact, that we do not only add one feasibility or optimality cut in steps 2 and 3, but one cut for each commodity where feasibility and optimality, respectively, is violated.

9.2.2 Proof of Correctness and Convergence

Now we give a constructive proof of correctness and convergence of the multicommodity L-shaped algorithm, which follows the ideas given by Birge and Louveaux in [18] for the single-commodity version.

Let

$$Q^n(x^n, \xi(\omega)) = \min_{y^n} \{q^n(\omega)^\top y^n | W^n y^n = h^n(\omega) - T^n(\omega) x^n, y^n \geq 0\} \tag{9.12}$$

and

$$\mathcal{Q}^n(x^n) = \mathbb{E}_\xi Q^n(x^n, \xi(\omega)) \tag{9.13}$$

Moreover, let

$$X_1 = \{x | A^n x^n \leq b^n, n = 1, \dots, N, \sum_{n=1}^{N} A^n x^n \leq \kappa, x \geq 0\} \tag{9.14}$$

$$X_2 = \{x | \mathcal{Q}^n(x^n) < \infty, n = 1, \dots, N\} \tag{9.15}$$

where $x = (x^1, \dots, x^N)$.

Then the stochastic problem is

$$\begin{aligned} \min_{x} \quad & \sum_{n=1}^{N} c^{n\top} x^n + \sum_{n=1}^{N} \mathcal{Q}^n(x^n) \\ \text{s.t.} \quad & x \in X_1 \cap X_2 \end{aligned} \tag{9.16}$$

and can be rewritten as

$$\begin{aligned} \min_{x,\theta} \quad & \sum_{n=1}^{N} c^{n\top} x^n + \sum_{n=1}^{N} \theta^n \\ \text{s.t.} \quad & \mathcal{Q}^n(x^n) \leq \theta^n \qquad n = 1, \dots, N \\ & x \in X_1 \cap X_2 \end{aligned} \tag{9.17}$$

with $\theta = (\theta^1, \dots, \theta^N)$ and $\theta^n \in \mathbb{R}$, $n = 1, \dots, N$, as additional optimization variables.

First, we show that $Q^n(x^n, \xi(\omega))$ is a convex function in x^n, $n = 1, \dots, N$.

The concatenation of a convex and a linear function is a convex function, see [83]:

Lemma 9.1. *Let $f(y) : \mathbb{R}^{l_2} \to \mathbb{R}$ be a convex function in y and $g(x) : \mathbb{R}^{l_1} \to \mathbb{R}^{l_2}$ a linear function in x. Then the concatenation $(f \circ g)(x) : \mathbb{R}^{l_1} \to \mathbb{R}$ is a convex function in x.*

Proof. Consider arbitrary vectors $x_1 \in \mathbb{R}^{l_1}$ and $x_2 \in \mathbb{R}^{l_1}$ and an arbitrary $\lambda \in [0, 1]$. Then we have:

$$\begin{aligned} f(g(\lambda x_1 + (1 - \lambda) x_2) &= f(\lambda g(x_1) + (1 - \lambda) g(x_2)) \\ &\leq \lambda f(g(x_1)) + (1 - \lambda) f(g(x_2)) \end{aligned} \tag{9.18}$$

□

Furthermore, Birge and Louveaux show in [18] that in the case that ξ is a finite random variable, X_2 is a convex set. As X_1 clearly is a convex set because it is a polytope, it holds that $X_1 \cap X_2$ is a convex set if ξ is a finite random variable, because the intersection of two convex sets is convex.

With the help of Lemma 9.1, we can prove the following lemma:

Lemma 9.2. *If ξ is a finite random variable, $Q^n(x^n, \xi(\omega)) = \min_{y^n}\{q^n(\omega)^\top y^n | W^n y^n = h^n(\omega) - T^n(\omega)x^n, y^n \geq 0\}$ is a convex function in x^n, $n = 1, \dots, N$, for $x = (x^1, \dots, x^N) \in X_1 \cap X_2$.*

Proof. For this moment, we consider $Q^n(x^n, \xi(\omega))$ as a function of x^n for an arbitrary fixed value $\xi(\omega)$. Then $q^n(\omega)$, $h^n(\omega)$ and $T^n(\omega)$ are fixed values, independent of x^n. Then $h^n(\omega) - T^n(\omega)x^n$ is a linear function in x^n.

It is sufficient to consider $f(b) = \min_a\{{q^n}^\top a | W^n a = b\}$ and to show that f is a convex function in b, as then the convexity of $Q^n(x^n, \xi(\omega))$ in x^n follows with Lemma 9.1. Here q^n represents $q^n(\omega)$ for any fixed ω, a substitutes y^n and b substitutes $h^n(\omega) - T^n(\omega)x^n$ for simplicity in notation.

Consider arbitrary vectors b_1 and b_2 as well as a convex combination $b_\lambda := \lambda b_1 + (1-\lambda)b_2$ where $\lambda \in [0,1]$. Let a_1 and a_2 be an optimal solution of $\min_a\{{q^n}^\top a | W^n a = b\}$ for $b = b_1$ and $b = b_2$, respectively.
Because of

$$W^n(\lambda a_1 + (1-\lambda)a_2) = \lambda W^n a_1 + (1-\lambda)W^n a_2 = \lambda b_1 + (1-\lambda)b_2 = b_\lambda,$$

we have $\lambda a_1 + (1-\lambda)a_2$ is a feasible solution of $\min_a\{{q^n}^\top a | W^n a = b_\lambda\}$. Let a_λ be an optimal solution of $\min_a\{{q^n}^\top a | W^n a = b_\lambda\}$. Then we have

$$\begin{aligned} f(b_\lambda) &= {q^n}^\top a_\lambda \\ &\leq {q^n}^\top(\lambda a_1 + (1-\lambda)a_2) \\ &= \lambda {q^n}^\top a_1 + (1-\lambda){q^n}^\top a_2 \\ &= \lambda f(b_1) + (1-\lambda)f(b_2). \end{aligned}$$

□

We can now show the correctness and termination of the multicommodity L-shaped algorithm.

Theorem 9.3. *If ξ is a finite random variable, the multicommodity L-shaped algorithm converges to an optimal solution, if there exists one, or proves that problem (9.16) is infeasible.*

Proof. First we show that the multicommodity L-shaped algorithm terminates when applied on problem (9.17), which is equivalent to problem (9.16).

In step 3 in iteration ν of the algorithm, problem (9.11) is solved for every $k = 1, \dots, K$ and every $n = 1, \dots, N$ in order to get optimal dual solutions $\pi_k^{n\nu}$,

$k = 1, \ldots, K$, $n = 1, \ldots, N$. By strong duality, for every $k = 1, \ldots, K$ and every $n = 1, \ldots, N$, we have

$$Q^n(x^{n\nu}, \xi_k) = (\pi_k^{n\nu})^\top (h_k^n - T_k^n x^{n\nu}), \tag{9.19}$$

as for the dual, a finite optimal solution could be found and therefore the primal problem also has a finite solution, where the two objective function values are equal.

According to [48], the subgradient inequality $f(x) - f(y) \geq \bigtriangledown f(y)^\top (x - y)$ holds for convex functions $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$, where $x,\ y \in X$.

As $Q^n(x^n, \xi_k)$ is convex in x, which we proved in Lemma 9.2, the inequality given above holds for $Q^n(x^n, \xi_k)$ and we have

$$\begin{aligned} Q^n(x^n, \xi_k) &\geq Q^n(0, \xi_k) + \bigtriangledown Q^n(0, \xi_k)(x^n - 0) \\ &= (\pi_k^{n\nu})^\top h_k^n - (\pi_k^{n\nu})^\top T_k^n 0 + (-(\pi_k^{n\nu})^\top T_k^n) x^n \\ &= (\pi_k^{n\nu})^\top h_k^n - (\pi_k^{n\nu})^\top T_k^n x^n. \end{aligned} \tag{9.20}$$

When calculating the expected value of relations (9.19) and (9.20), we get

$$\mathcal{Q}^n(x^{n\nu}) = \mathbb{E}(\pi^{n\nu})^\top (h^n - T^n x^{n\nu}) = \sum_{k=1}^{K} p_k^n (\pi_k^{n\nu})^\top (h_k^n - T_k^n x^{n\nu}) \tag{9.21}$$

and

$$\mathcal{Q}^n(x^n) \geq \mathbb{E}(\pi^{n\nu})^\top (h^n - T^n x^n) = \sum_{k=1}^{K} p_k^n (\pi_k^{n\nu})^\top h_k^n - \left(\sum_{k=1}^{K} p_k^n (\pi_k^{n\nu})^\top T_k^n \right) x^n, \tag{9.22}$$

respectively.

Because of the constraint $\theta^n \geq \mathcal{Q}^n(x^n)$ in (9.17), a pair (x^n, θ^n) are only feasible for this problem, if $\theta^n \geq \mathbb{E}((\pi^{n\nu})^\top (h^n - T^n x^n))$, which corresponds exactly to inequality (9.9).

On the other hand, if (x^ν, θ^ν) is optimal for (9.17), it follows that $\theta^{n\nu} = \mathcal{Q}^n(x^{n\nu})$, as θ^n is bounded below in (9.17) by $\theta^n \geq \mathcal{Q}^n(x^n)$. This holds if $\theta^{n\nu} = \mathbb{E}((\pi^{n\nu})^\top (h^n - T^n x^{n\nu}))$, therefore if $\theta^{n\nu} = w^{n\nu}$.

Should $\theta^{n\nu} < \mathcal{Q}^n(x^{n\nu})$ from step 3 of the algorithm, then none of the previously defined cuts forces $\theta^n \geq \mathcal{Q}^n(x^n)$ and a new cut must be defined.

The claim that the algorithm converges in a finite number of iterations to an optimal solution follows by the fact that for every (k, n), $k = 1, \ldots, K$, $n = 1, \ldots, N$, only a finite number of possibilities for the dual solutions $\pi_k^{n\nu}$ exists, as every dual solution is defined by one of the finite number of different bases of (9.11) which are independent of x^ν.

Now we have to prove that only a finite number of constraints (9.8) is necessary in order to guarantee that $x = (x^1, \ldots, x^N) \in X_2$. These constraints are generated in step 2 of the algorithm.

The claim that $x = (x^1, \ldots, x^N) \in X_2$ is equivalent to

$$\begin{aligned} x \in \{x| \; \forall k = 1, \ldots, K \quad \exists y = (y^1, \ldots, y^N) \geq 0 \\ \text{s.t. } W^n y^n = h_k^n - T_k^n x^n \quad \forall n = 1, \ldots, N\}. \end{aligned} \tag{9.23}$$

This is equivalent to

$$h_k - T_k x \in posW, \quad k = 1, \ldots, K, \tag{9.24}$$

where $h_k = (h_k^1, \ldots, h_k^N)$, $T_k = (T_k^1, \ldots, T_k^N)$ and $posW$, the *positive hull of* W^n, is defined as

$$posW = \{t = (t^1, \ldots, t^N)| W^n y^n = t^n, y^n \geq 0, \quad n = 1, \ldots, N\}. \tag{9.25}$$

In step 2, we check whether this condition is satisfied for the current x^ν. If this condition is not satisfied, that means that $h_k^n - T_k^n x^{n\nu} \notin posW^n$ for a $k \in \{1, \ldots, K\}$ and an $n \in \{1, \ldots, N\}$, where $posW^n = \{t^n | W^n y^n = t^n, y^n \geq 0\}$, a hyperplane must exist which separates $posW^n$ and $h_k^n - T_k^n x^{n\nu}$. Hence, this hyperplane must satisfy ${\sigma_k^n}^\top t^n \leq 0$ for all $t^n \in posW^n$ and ${\sigma_k^n}^\top (h_k^n - T_k^n x^{n\nu}) > 0$. In order to obtain such a hyperplane, we replace σ_k^n by the dual solution $\sigma_k^{n\nu}$ which we calculated in step 2.

The fact that ${w_k^n}'$ is always strictly positive for some (n, k) is equivalent to $(\sigma_k^{n\nu})^\top (h_k^n - T_k^n x^{n\nu}) > 0$ due to duality. Furthermore we have $(\sigma_k^{n\nu})^\top W^n \leq 0$, as $\sigma_k^{n\nu}$ is an optimal dual solution and therefore the reduced costs of y^n must be nonnegative.

A necessary condition for $x \in X_2$ is that ${\sigma_k^n}^\top (h_k^n - T_k^n x^{n\nu}) \leq 0$ for all $n = 1, \ldots, N$, that means ${w_k^n}' = 0$ for all $k = 1, \ldots, K$ and all $n = 1, \ldots, N$. Consequently, only a finite number of constraints (9.8) may exist. This is true because there exists only a finite number of possible bases for Problem (9.10), which is solved in step 2. This is obvious since X_2 is a convex polyhedron for finite random variables ξ. □

9.2.3 Multicommodity Two-Stage Example

In this example, we apply a special type of recourse, the so-called *simple recourse.* This type of recourse offers several advantages in the application of the multicommodity L-shaped algorithm. As simple recourse is a general type of complete recourse, the first-stage decision x is always feasible and step 2 is redundant. Furthermore, the calculation of the dual variables in step 3 is simplified. See [18] for details.

In the case of production and transportation problems, simple recourse corresponds to a penalization of deficiency or surplus at the demand nodes. For a simple recourse problem, we have $W = [I, -I]$, where I is the identity matrix.

The following figure illustrates the network structure of the given supply chain problem for two commodities.

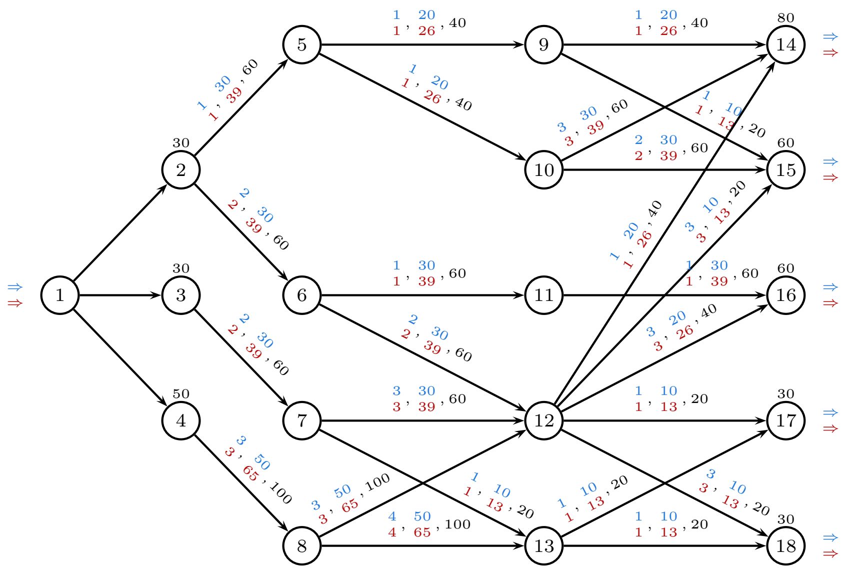


Figure 9.1: Multicommodity two-stage example: Two-stage problem with uncertain demands

Again,the underlying network has the same structure as in Figures 7.6, 7.10, 8.6 and 8.18.

The arc data in Figure 9.1 have the following meaning: Upper line: cost and capacity for commodity 1. Lower line: cost and capacity for commodity 2. Middle line: bundle capacity.

The values above the nodes give the capacity restrictions for the nodes (maximum flow that is allowed to pass the node).

Additionally, we have the following input data for the two-stage formulation:

scenario	product	node 14	node 15	node 16	node 17	node 18
scenario 1	product 1	25.0	25.0	20.0	10.0	10.0
	product 2	30.0	30.0	24.0	12.0	12.0
scenario 2	product 1	27.5	27.5	22.0	11.0	11.0
	product 2	33.0	33.0	26.4	13.2	13.2

Table 9.1: Multicommodity two-stage example: Demand scenarios

	node 14	node 15	node 16	node 17	node 18
penalty cost	40	30	30	30	30
storage cost	4	4	4	12	12

Table 9.2: Multicommodity two-stage example: Penalty and storage costs for commodities 1 and 2

Each of the two scenarios has a probability of 0.5.

We apply the multicommodity L-shaped algorithm to the given problem. After 20 iterations, we obtain the following solution:

product	node 14	node 15	node 16	node 17	node 18
product 1	30.0000	25.3336	20.6657	12.0000	10.9132
product 2	33.0000	28.2689	22.7318	13.2000	12.1902

Table 9.3: Multicommodity two-stage example: Delivered units in optimal solution

This solution causes the following storage and penalty costs per commodity and scenario:

scenario	product	node 14	node 15	node 16	node 17	node 18
scenario 1	product 1	0.0000	0.0000	0.0000	0.0000	0.0000
	product 2	0.0000	0.0000	0.0000	0.0000	0.0000
scenario 2	product 1	0.0000	4.6664	3.3343	0.0000	1.0868
	product 2	0.0000	4.7311	3.6682	0.0000	1.0098

(a) Penalty costs

scenario	product	node 14	node 15	node 16	node 17	node 18
scenario 1	product 1	5.0000	0.3336	0.6657	2.0000	0.9132
	product 2	5.5000	0.7689	0.7318	2.2000	1.1902
scenario 2	product 1	0.0000	0.0000	0.0000	0.0000	0.0000
	product 2	0.0000	0.0000	0.0000	0.0000	0.0000

(b) Storage costs

Table 9.4: Multicommodity two-stage example: Penalty and storage costs in an optimal solution

The solution meets our expectations: For demand node 14 the maximum possible amount of units is delivered for both products 1 and 2, as for this node the penalty costs are higher than for other demand nodes. For demand nodes 17 and 18, the storage cost is higher than for the other demand nodes. For demand node 18, less than the maximum possible amount is sent to this node. For demand node 17, however, a very cheap production and transportation path exists (path $(1, 3, 7, 13, 17)$), such that the maximum possible amount of units is delivered to this demand node, though storage costs are rather high.

Note that the multicommodity two-stage example partially corresponds to the recoverable robustness example which we examined in Subsection 8.3.5, except for different scenario sets. In Chapter 10, we will examine the relations between the Stochastic Programming approach and the recoverable robustness model in detail.

9.3 Recourse by Interexchange

In the multicommodity two-stage example, we considered the recourse of deficiency by a penalization of over- and underproduction (simple recourse). For supply chain problems, however, we propose a new recourse concept - the interexchange recourse scheme. The interexchange recourse scheme is a generalization of a transport and transshipment example, which was given by J. Böttcher in [22].

9.3.1 Introduction of the Interexchange Recourse Scheme

When considering a production and transportation network with uncertain demands, the following situation is likely to happen in case that the demand predictions are not reliable: At one demand node, we have a surplus of goods, while at another demand node, we have a deficiency of goods. In such a situation, it would make sense to interexchange goods between the two demand nodes. This possibility of interexchange will now be integrated in our two-stage model by adding it to the simple recourse scheme, which we will maintain in parallel.

We have to add new arcs between the demand nodes in the underlying network. Every pair of demand nodes must be connected by two arcs, one in each direction.

Furthermore, the compensation matrix W must be modified in order to allow the possibility of interexchange in the recourse. Therefore we add to W one column per new interexchange arc with -1 at the position corresponding to the arc's start node and 1 at the arc's end node. The original columns of W remain unchanged, as these columns represent the compensation of deficiency using penalty and storage costs.

Additionally, we have to add capacity restrictions in order to guarantee that the amount of units that can be interexchanged from one node is limited by the number of units that are actually delivered to this node in the first stage. We introduce a new matrix CR for these capacity restrictions and add the constraints

$$CRy - Tx \leq 0 \tag{9.26}$$

to the two-stage problem. CR is a matrix containing the values 0, 1 and -1, where every interexchange arc is identified by a pair of 1 and -1 in one column. Furthermore, we can add capacity restrictions for the interexchange arcs as well as bundle capacities in the multicommodity case.

Moreover, we have to fix the cost values for the interexchange arcs.

In practice, the cost values on the transportation arcs and the interexchange arcs correspond to the real transportation costs on these paths. Due to the triangle inequality, the transport via another location is more expensive than the direct transport. Therefore, in general the direct transport path will be preferred.

Since the actual demand is not known in the first stage, when the transport routing is fixed, it may occur that, after the realization of the random variable, at one demand node more units than delivered are required, while at another demand node there is a surplus of delivered units. Instead of paying penalty and storage costs at both demand nodes, it may be less expensive to send the surplus units from one node to another via the interexchange arcs. Hence the interexchange arcs will be used despite of the valid triangle inequality for the transportation costs between the concerned nodes.

If no cost values for the interexchange arcs are available, we recommend the following procedure. We respect the following constraints:

- Costs for arcs between the same two nodes must be equal, no matter of the direction.

- Sending units via interexchange arcs must not be cheaper than sending units via the direct path without using interexchange arcs.

In order to fulfill the second constraint for a pair of demand nodes, all possible paths from a production node to these demand nodes must be considered. The minimum cost value for the corresponding interexchange arc is then bounded below by the maximum over all absolute difference values of the path costs. The cost value for the interexchange must be set higher than this lower bound in order to guarantee that interexchange is strictly more expensive than direct sending.

9.3.2 Multicommodity Interexchange Two-Stage Example

The interexchange recourse scheme is demonstrated by means of an example.

We consider again the network given in the multicommodity two-stage example in Subsection 9.2.3. New arcs have to be added between the demand nodes for the interexchange recourse. All in all, 20 new arcs have to be added to the network.

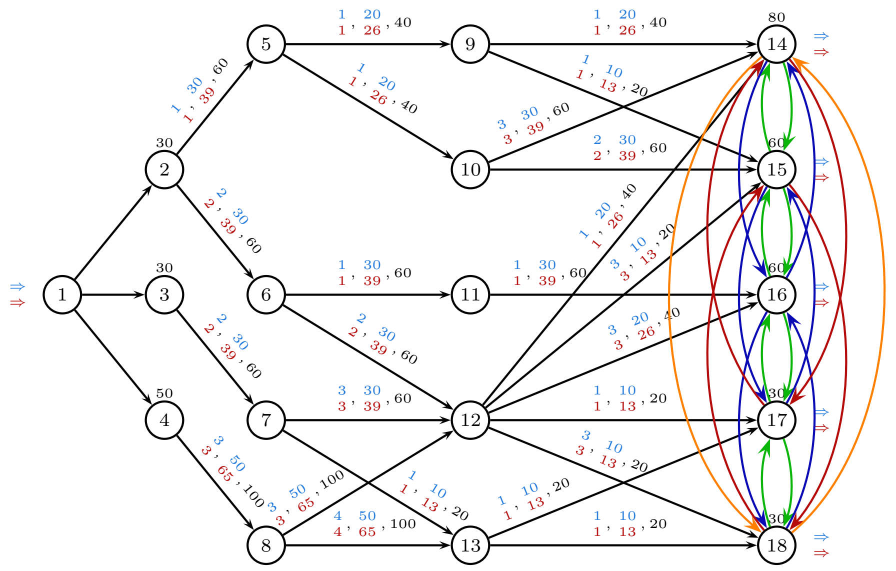


Figure 9.2: Multicommodity interexchange two-stage example: Two-stage problem with uncertain demand and interexchange recourse

The network structure corresponds to the structure used in previous examples, cf. Figures 7.6, 7.10, 8.6 and 9.1.

Again, we consider two scenarios with demand data as well as penalty and storage cost data as given in the multicommodity two-stage example.

Furthermore, we have the following cost values on the interexchange arcs:

node	14	15	16	17	18
14		3	3	1	3
15	3		1	3	1
16	3	1		3	1
17	1	3	3		3
18	3	1	1	3	

Table 9.5: Multicommodity interexchange two-stage example: Arc cost for the interexchange arcs for commodities 1 and 2

For every interexchange arc, the maximum capacity per product is set to 20 and the bundle capacity is set to 50.

We apply the multicommodity L-shaped algorithm to the two-stage problem. The algorithm terminates after 14 iterations with the following optimal solution:

product	node 14	node 15	node 16	node 17	node 18
product 1	26.7076	25.0448	23.2031	16.7013	13.4268
product 2	31.1660	29.5409	24.3376	16.3874	13.4845

Table 9.6: Multicommodity interexchange two-stage example: Delivered units in the first stage of the optimal solution

scenario	product	node 14	node 15	node 16	node 17	node 18
scenario 1	product 1	33.4089	26.9363	24.7385	10.0000	10.0000
	product 2	36.5534	30.6700	25.6930	11.0000	11.0000
scenario 2	product 1	30.0000	27.4333	23.6504	12.0000	12.0000
	product 2	33.0000	30.4012	25.1151	13.2000	13.2000

Table 9.7: Multicommodity interexchange two-stage example: Delivered units including interexchange (second stage)

scenario	product	node 14	node 15	node 16	node 17	node 18
scenario 1	product 1	0.0000	0.0000	0.0000	0.0000	0.0000
	product 2	0.0000	0.0000	0.0000	0.0000	0.0000
scenario 2	product 1	0.0000	2.5667	0.3496	0.0000	0.0000
	product 2	0.0000	2.5988	1.2849	0.0000	0.0000

(a) Penalty costs

scenario	product	node 14	node 15	node 16	node 17	node 18
scenario 1	product 1	8.4089	1.9363	4.7385	0.0000	0.0000
	product 2	9.0534	3.1700	3.6930	0.0000	0.0000
scenario 2	product 1	0.0000	0.0000	0.0000	0.0000	0.0000
	product 2	0.0000	0.0000	0.0000	0.0000	0.0000

(b) Storage costs

Table 9.8: Multicommodity interexchange two-stage example: Penalty and storage costs in optimal solution

The following figure illustrates the interexchange for the two products in the two scenarios.

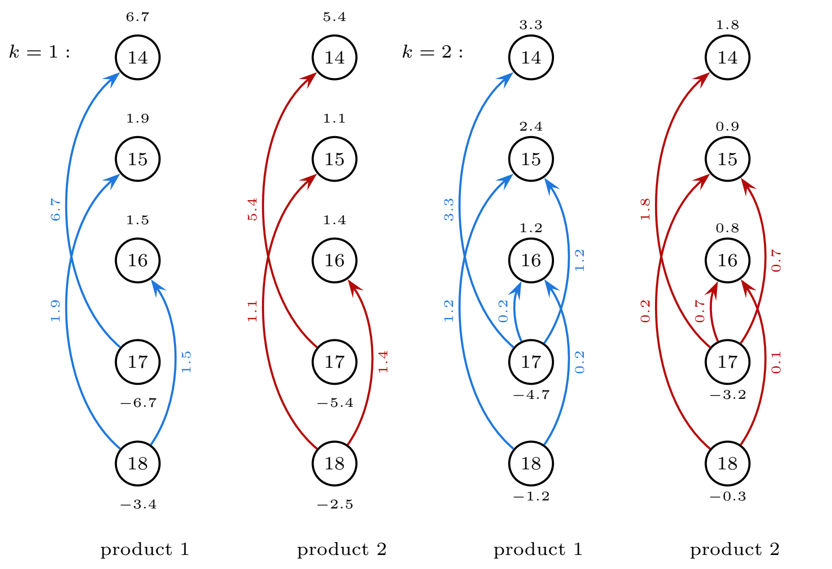


Figure 9.3: Multicommodity interexchange two-stage example: Interexchange for products 1 and 2

Again, the solution meets our expectations. When considering the interexchange, we can see that from demand nodes 17 and 18, the delivered surplus is distributed to other nodes, as for nodes 17 and 18 the storage cost is higher than for the other demand nodes. Most of the surplus is delivered to demand node 14 in order to guarantee that the demand at this node is fulfilled in any case, as the penalty cost is higher at node 14 than at node 15 and 16 and the storage costs are equal for nodes 14, 15 and 16.

Concluding, we want to compare the solution of the multicommodity interexchange example to the solution of the nominal multicommodity two-stage example (see Section 9.2.3). The objective value of the multicommodity interexchange problem is 1549, which is less than the objective value of 1610 of the multicommodity two-stage example. This means that, as we would have expected, the additional compensation possibility of interexchange allows cheaper solutions than simple recourse only.

9.4 Inventory Management

So far, we only considered supply chain management problems for a single time period. Especially in the case of supply chain problems with uncertain demand, the consideration of multiple time periods is an important challenge, as there exists the possibility to store goods from one time period to another. The preproduction and storage of goods is important for time periods with very high demand, where it is not possible to produce enough goods just in time. A maximum storage capacity as well as storage costs are limiting factors for the product storage.

The task of a possible storage of goods between different time periods is called *inventory management* in the following. Note that we still consider uncertainty in the demand values.

9.4.1 Network Extension for Inventory Management

Let TP be the number of considered time periods. Then we have to duplicate the single-period network such that we have TP identical copies of it. In order to connect the single-period networks, we insert the arcs $(i(t), i(t+1))$, $t = 1, \ldots, TP-1$ for all demand nodes i, where $i(t)$, $t = 1, \ldots, TP$ denotes the node i of the single-period network in time period t. These arcs are called *inventory management arcs* in the following.

In our model, we assume that the transportation of goods that is planned for one time period can always be processed within this time period and we therefore neglect the time which is necessary for the transport. In different models, especially in continuous supply chain models like, for example the model of Herty et al. ([51]), the transportation time is integrated in the model.

The following figure illustrates the network extension for two time periods. The dashed lines symbolize the inventory management arcs.

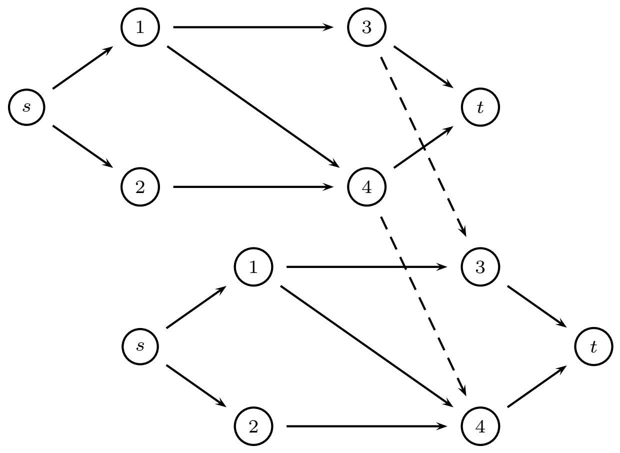


Figure 9.4: Extended network for two time periods

On the inventory management arcs, we have to impose storage costs, as these arcs model the storage of units from one time period to another. One possible choice for the storage costs is the assignment of the same storage costs which we impose in the recourse function.

The single-period constraints must be modified. Let A be the single-period inequality constraint matrix corresponding to the network structure and A_{eq} the single-period equality constraint matrix corresponding to additional equality constraints like flow conservation constraints.

Then we have to duplicate the matrices for every time period:

$$\begin{array}{|c|c|c|c|}\hline A & 0 & \dots & 0 \\ \hline 0 & A & & 0 \\ \hline \vdots & & \ddots & \vdots \\ \hline 0 & 0 & \dots & A \\ \hline A_{eq} & 0 & \dots & 0 \\ \hline 0 & A_{eq} & & 0 \\ \hline \vdots & & \ddots & \vdots \\ \hline 0 & 0 & \dots & A_{eq} \\ \hline \end{array}$$

Figure 9.5: Matrix duplication for the inventory management problem

In addition, we have to add the inventory management arcs to the inequality matrix as shown for $TP = 3$ and l demand nodes:

$$\begin{array}{|c|c|c|c|c|}\hline A & 0 & 0 & \begin{array}{c} 0 \\ \hline -I_l \end{array} & 0 \\ \hline 0 & A & 0 & \begin{array}{c} 0 \\ \hline I_l \end{array} & \begin{array}{c} 0 \\ \hline -I_l \end{array} \\ \hline 0 & 0 & A & 0 & \begin{array}{c} 0 \\ \hline I_l \end{array} \\ \hline \end{array}$$

Figure 9.6: Additional constraints for the inventory management problem (I)

Here I_l is the l-dimensional identity matrix.

Moreover, we have to ensure that the number of units that is stored by inventory management at one demand node in a specific time period is less or equal to the number of units that actually arrives at the demand node. Let T be the part of A which specifies how many units arrive at the demand nodes. Then the additional constraints can be represented by:

$-T$	0	0	I_l	0
0	$-T$	0	$-I_l$	I_l
0	0	$-T$	0	$-I_l$

Figure 9.7: Additional constraints for the inventory management problem (II) for the case $TP = 3$

for $TP = 3$ and l demand nodes.

All in all, the constraints for an inventory management problem have the following structure, illustrated for $TP = 3$ and l demand nodes:

A	0	0	0	0		b
			-I_l	0		
0	A	0	0	0	$\leq$	b
			I_l	-I_l		
0	0	A	0	0		b
			0	I_l		
$-T$	0	0	I_l	0		0
0	$-T$	0	-I_l	I_l	$\leq$	0
0	0	$-T$	0	-I_l		0
A_{eq}	0	0	0	0		b_{eq}
0	A_{eq}	0	0	0	$=$	b_{eq}
0	0	A_{eq}	0	0		b_{eq}

Figure 9.8: Constraints for the inventory management problem

In Figure 9.8, we can see how the size of the constraint matrix and therefore the complexity of the optimization problem increases with the number of considered time periods. As in the single-period problem, the application of a special purpose algorithm like the (multicommodity) L-shaped algorithm instead of a standard linear programming approach is therefore advisable.

If the number of time periods is very large, the computational time increases rapidly. However, in practice, typical periods under consideration are 3 months partitioned in time periods of 13 weeks or 1 year partitioned in time periods of 12 months.

9.4.2 Multicommodity Inventory Management Two-Stage Example

The inventory management approach is not only applicable on single-commodity problems. We can combine the multicommodity L-shaped algorithm with the inventory management approach and then apply it on a multicommodity inventory management problem.

In the following, we consider an inventory management problem for two commodities and two time-periods. The following figure illustrates the problem structure.

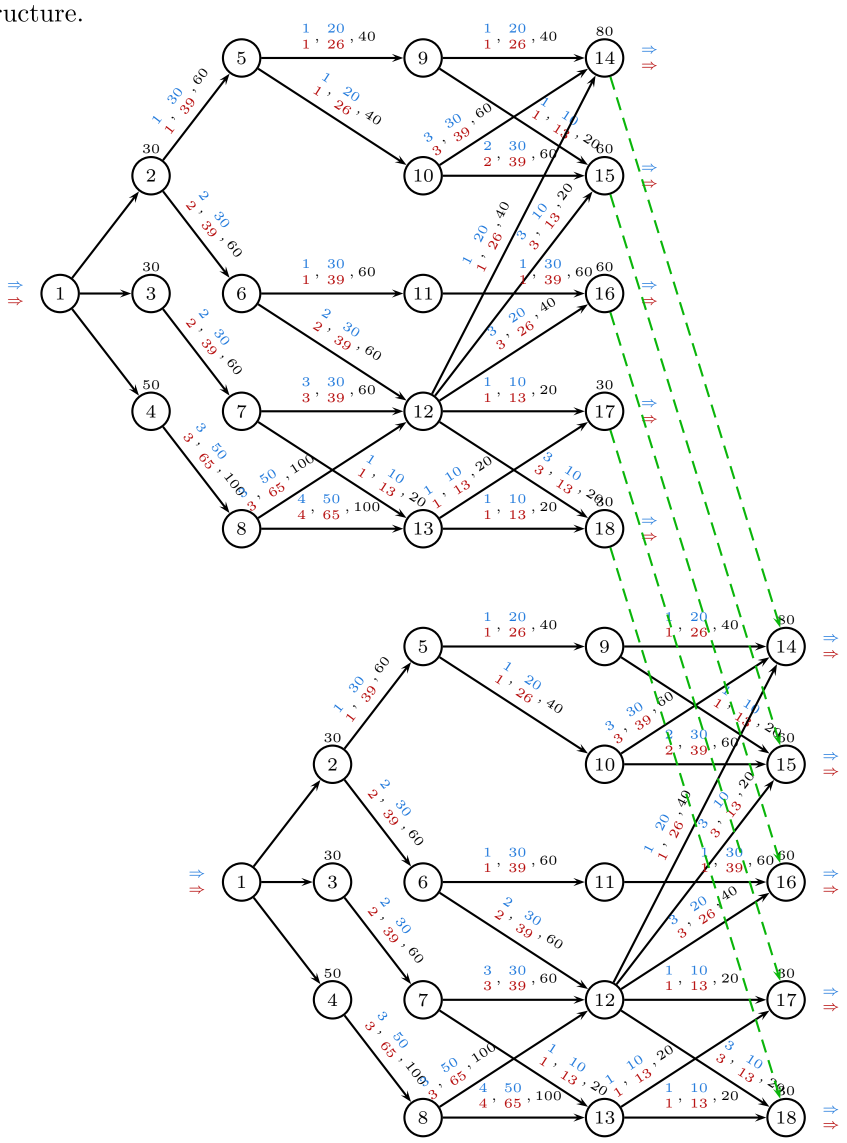


Figure 9.9: Multicommodity inventory management two-stage example: Two-stage problem with uncertain demands and inventory management

Again, the network structure corresponds to the structure used in previous examples, cf. Figures 7.6, 7.10, 8.6, 9.1 and 9.2.

For the demand nodes 14 to 18, we have the following demand scenarios:

time period	scenario	product	node 14	node 15	node 16	node 17	node 18
period 1	scenario 1	product 1	25.00	25.00	20.00	10.00	10.00
		product 2	30.00	30.00	24.00	12.00	12.00
	scenario 2	product 1	27.50	27.50	22.00	11.00	11.00
		product 2	33.00	33.00	26.40	13.20	13.20
period 2	scenario 1	product 1	32.50	32.50	26.00	13.00	13.00
		product 2	39.00	39.00	31.20	15.60	15.60
	scenario 2	product 1	35.75	35.75	28.60	14.30	14.30
		product 2	42.90	42.90	34.32	17.16	17.16

Table 9.9: Multicommodity inventory management two-stage example: Demand scenarios

Again, we consider simple recourse with the following penalty and storage costs:

	node 14	node 15	node 16	node 17	node 18
penalty cost	40	30	30	30	30
storage cost	4	4	4	12	12

Table 9.10: Multicommodity inventory management two-stage example: Penalty and storage costs for commodities 1 and 2

We apply the multicommodity L-shaped algorithm to the two-stage problem. After 45 iterations, the algorithm terminates with the following optimal solution.

	time period 1					time period 2				
node	13	14	15	16	17	13	14	15	16	17
demand sc 1	25.00	25.00	20.00	10.00	10.00	32.50	32.50	26.00	13.00	13.00
demand sc 2	27.50	27.50	22.00	11.00	11.00	35.75	35.75	28.60	14.30	14.30
transport	28.38	25.77	21.26	12	10.58	29.12	22.54	21.29	15.56	13
storage	-3.38	-0.77	-1.26	0	0	3.38	0.77	1.26	0	0
total	25	25	20	12	10.58	32.5	23.3	22.55	15.56	13

(a) Commodity 1

	time period 1					time period 2				
node	13	14	15	16	17	13	14	15	16	17
demand sc 1	30.00	30.00	24.00	12.00	12.00	39.00	39.00	31.20	15.60	15.60
demand sc 2	33.00	33.00	26.40	13.20	13.20	42.90	42.90	34.32	17.16	17.16
transport	31.95	28.64	24	13.2	12.42	31.3	30.2	25.56	17.14	14.3
storage	-4.45	-1.14	-2	0	0	4.45	1.14	2	0	0
total	27.5	27.5	22	13.2	12.42	35.75	31.34	27.56	17.14	14.3

(b) Commodity 2

Table 9.11: Multicommodity inventory management two-stage example: Optimal solution

In our example, the demand values in time period 2 are higher than the demand values in time period 1. Due to tight capacity restrictions, it is not possible to produce the desired amount of goods in the second time period. Hence, we have a surplus production in time period 1, where we have free capacities, and some units are stored until time period 2 in order to avoid too high penalty costs.

The total cost (production, transportation, penalty and storage costs) for the two-period problem is 4551.4. If we optimize the two single-period problems separately without a possibility of inventory management, the total cost is 4667.9.

9.5 Summary and Conclusions

In this chapter, we proposed a Stochastic Programming approach to network flow problems with uncertain demands, based on the concept of two-stage linear recourse problems.

In Section 9.2, the two-stage model was extended from single-commodity to multicommodity network flow problems with uncertain demands. In order to solve the resulting optimization problems, the *multicommodity L-shaped algorithm* was proposed. For this algorithm, correctness and convergence were proven.

In Section 9.3, a new recourse scheme was introduced. The interexchange recourse scheme balances surplus and deficiency of goods by interexchange between demand nodes.

In Section 9.4, we examined supply chain problems for multiple time periods. We introduced the possibility of preproduction and storage of goods in case that it is not possible to produce enough goods just in time in time periods with very high demand.

Both the interexchange recourse scheme and the inventory management approach can be applied to single-commodity and multicommodity problems.

Furthermore, the interexchange recourse scheme can be combined with the inventory management approach for inventory management problems, i.e. the balancing of surplus and deficiency of goods by interexchange between demand nodes can also be introduced in inventory management problems where several time periods are considered.

The Stochastic Programming approach is an exact method, such that the multicommodity L-shaped algorithm as well as its interexchange und inventory management extensions terminate with an exact solution, if one exists. However, this involves large size optimization models for complex problem instances. Especially the inventory management approach, where several time periods are considered, results in large optimization problems.

Chapter 10

A Multicriteria Perspective

By taking a multicriteria perspective, we want to examine the relation between the LRP from recoverable robustness, as presented in Section 8.3, and the Two-Stage Problem from Stochastic Programming, as presented in Chapter 9.

We consider a linear supply chain problem with uncertain demand. Let $S = \{h_0 := h^{min}, h_1 := h^{min} + \triangle e_1, \ldots, h_l := h^{min} + \triangle e_l\}$ be the scenario set of the demand scenarios. Recall that $h^{min} \in \mathbb{R}^l$ denotes the minimum demand, $\triangle$ the length of the uncertainty interval and e_i the i-th l-dimensional unit vector for $i = 1, \ldots, l$ and the l-dimensional zero vector for $i = 0$. In this context, l is the number of demand nodes.

As before, let $x \in \mathbb{R}^m$ be the vector of optimization variables of the original optimization problem ("first stage") and $y_i \in \mathbb{R}^{2l}$ the vector of recovery or compensation variables ("second stage") for scenario i where $i = 0, \ldots, l$.

The LRP for the given scenario set S can be formulated as follows:

$$\begin{aligned}
\min_{x,y_i}\ & c^\top x \\
\text{s.t. } & Ax \le b \\
& A_{eq}x = b_{eq} \\
& Tx \ge h^{min} \\
& x \ge 0 \\
& \forall i \in \{0,\dots,l\}: \\
& \qquad\qquad Tx + Wy_i = h_i \\
& \qquad\qquad d^\top y_i \le D \\
& \qquad\qquad y_i \ge 0
\end{aligned} \tag{10.1}$$

Recall that in the context of supply chain problems, $c \in \mathbb{R}^m$ denotes the production and transportation cost. $Ax \le b$ with $A \in \mathbb{R}^{n_1 \times m}$ and $b \in \mathbb{R}^{n_1}$ contains the capacity constraints. $A_{eq}x = b_{eq}$ with $A_{eq} \in \mathbb{R}^{n_2 \times m}$ and $b_{eq} \in \mathbb{R}^{n_2}$ contains the flow conservation constraints for transshipment nodes. The constraint $Tx \ge h^{min}$ with $T \in \mathbb{R}^{l \times m}$ and $h^{min} \in \mathbb{R}^l$ ensures that at least the minimum demand h^{min} is delivered to the demand nodes. $h_i \in \mathbb{R}^l$, $i \in \{0,\dots,l\}$ are the demand vectors. The matrix $W \in \mathbb{R}^{l \times 2l}$ specifies how deficiencies in demand can be compensated. $d \in \mathbb{R}^{2l}$ denotes the cost vector for penalty and storage costs for missing or surplus demand units, respectively. D denotes the total recovery budget.

Keeping this nomenclature of the input data of the LRP, the two-stage problem corresponding to the scenario set S is:

$$\begin{aligned}
\min\ & c^\top x + \sum_{i=0}^{l} p_i d^\top y_i \\
\text{s.t. } & Ax \le b \\
& A_{eq}x = b_{eq} \\
& Tx \ge h^{min} \\
& Tx + Wy_i = h_i && i = 0,\dots,l \\
& x \ge 0 \\
& y_i \ge 0 && i = 0,\dots,l.
\end{aligned} \tag{10.2}$$

Here p_i denotes the probability for scenario i.

We consider the supply chain problem from a multicriteria point of view. As before, we want to minimize the production and transportation costs $c^\top x$. Furthermore, for every scenario i, $i = 0, \dots, l$, we want to minimize the recovery or compensation cost $d^\top y_i$. The additional equality and inequality constraints concerning flow conservation and capacity constraints remain unchanged. Then the multicriteria optimization problem (MOP) can be formulated as follows:

$$\begin{aligned} \text{vmin } & (c^\top x,\ d^\top y_0,\ \dots,\ d^\top y_l) && \\ \text{s.t. } & Ax \le b && \\ & A_{eq}x = b_{eq} && \\ & Tx + Wy_i = h_i && i = 0, \dots, l \\ & x \ge 0 && \\ & y_i \ge 0 && i = 0, \dots, l \end{aligned} \tag{10.3}$$

10.1 Scalarizations of the Multicriteria Optimization Problem

In this section, we examine two types of scalarizations of the MOP (10.3). The general introduction to scalarizations follows the book of M. Ehrgott, see [36].

We consider the general multicriteria optimization problem of the form

$$\begin{aligned} \text{vmin } & (f_1(x), \dots, f_Q(x)) \\ \text{s.t. } & x \in X \end{aligned} \tag{10.4}$$

10.1.1 e-Constraint Scalarization

Let $k \in \{1, \dots, Q\}$.

A scalarized problem of the type

$$\begin{aligned} \min_{x \in X} \ & f_k(x) && \\ \text{s.t. } & f_i(x) \le \epsilon_i && i = 1, \dots, Q, \quad i \ne k \end{aligned} \tag{10.5}$$

where $\epsilon_i \in \mathbb{R}$, $i = 1, \dots, Q$, $i \neq k$, is called *ϵ-constraint scalarization* of the MOP, short $(P_k(\epsilon))$.

The ϵ-constraint scalarization has the following properties (see [36]):

Theorem 10.1.

1. *Let x^* be an optimal solution of the ϵ-constraint scalarization (10.5) for some k. Then x^* is a weakly Pareto-optimal solution of the MOP (10.4).*
2. *Let x^* be a unique optimal solution of the ϵ-constraint scalarization (10.5) for some k. Then x^* is a Pareto-optimal solution of the MOP (10.4).*
3. *The solution $x^* \in X$ is Pareto optimal if and only if there exists an ϵ^* such that x^* is an optimal solution of $(P_k(\epsilon^*))$ for all $k = 1, \dots, Q$.*

We have the following relation between the MOP (10.3) and the LRP (10.1):

Theorem 10.2. *The LRP (10.1) is an ϵ-constraint scalarization of the multicriteria problem (10.3).*

Proof. Set $\epsilon_i = D$, $i = 0, \dots, l$, where D is the recovery budget of the LRP. □

10.1.2 Weighted Sum Scalarization

A scalarized problem of the MOP of the type

$$\min_{x \in X} \sum_{i=1}^{Q} \lambda_i f_i(x), \tag{10.6}$$

where $\lambda_i \geq 0$, $i = 1, \dots, Q$, $\lambda \neq (0, \dots, 0)$ is called *weighted sum scalarization* of the MOP. In the literature, it is frequently required that $\sum\limits_{i=1}^{Q} \lambda_i = 1$. Note that if $\lambda_{sum} := \sum\limits_{i=1}^{Q} \lambda_i \neq 1$ with $\lambda_{sum} \neq 0$, the normalized scalarization can be obtained using $\tilde{\lambda}_i := \dfrac{\lambda_i}{\lambda_{sum}}$ as weight factors.

The weighted sum scalarization has the following properties (see [36]):

Theorem 10.3.

1. *Let x^* be an optimal solution of the weighted sum scalarization (10.6). Then x^* is weakly Pareto-optimal for the MOP (10.4).*
2. *Let x^* be an optimal solution of the weighted sum scalarization (10.6) with $\lambda_i > 0$, $i = 1, \dots, Q$. Then x^* is Pareto-optimal for the MOP (10.4).*

3. *Let x^* be a unique optimal solution of the weighted sum scalarization (10.6) with $\lambda_i \geq 0$, $i = 1, \ldots, Q$. Then x^* is strictly Pareto-optimal for the MOP (10.4).*

4. *Let X be a convex set and $f_1,\ldots,f_Q$ be convex functions. Let x^* be a weakly Pareto-optimal solution of the MOP (10.4). Then there exist $\lambda_i \geq 0$, $i = 1, \ldots, Q$, such that x^* is the optimal solution of the weighted sum scalarization (10.6).*

5. *Let X be a convex set and $f_1,\ldots,f_Q$ be convex functions. Let x^* be a Pareto-optimal solution of the MOP (10.4). Then there exist $\lambda \in \mathbb{R}^Q_+ \setminus \{0\}$, such that x^* is the optimal solution of the weighted sum scalarization (10.6).*

Then the following relation between the MOP (10.3) and the two-stage problem (10.2) holds:

Theorem 10.4. *The two-stage problem (10.2) is a weighted sum scalarization of the multicriteria problem (10.3).*

Proof. Set $\lambda_0 = 1$ and $\lambda_{i+1} = p_i$, $i = 0, \ldots, l$, where p_i is the probability of scenario i. Then the objective function of problem (10.2) can be formulated as a weighted sum of the objective functions of the MOP:

$$c^\top x + \sum_{i=0}^{l} p_i d^\top y_i = \lambda_0 c^\top x + \lambda_1 d^\top y_0 + \ldots + \lambda_{l+1} d^\top y_l$$

□

10.1.3 Further Scalarizations

Based on the book of M. Ehrgott, see [36], we will present further scalarization techniques and their relevance in the context of the multicriteria problem (10.3).

The Hybrid Scalarization The *hybrid scalarization* is a combination of the weighted sum scalarization and the ϵ-constraint scalarization. The objective of the scalarized problem is a weighted sum function. Additionally, constraints on all objectives are added.

Let x^0 be an arbitrary feasible solution of the MOP (10.4). Then the hybrid scalarization is

$$\begin{aligned} &\min_{x \in X} \sum_{i=1}^{Q} \lambda_i f_i(x) && \\ &\text{s.t. } f_i(x) \le f_i(x^0) && i = 1, \dots, Q, \end{aligned} \tag{10.7}$$

where $\lambda_i \ge 0,\ i = 1, \dots, Q$.

The hybrid scalarization has the following properties (see [36]):

Theorem 10.5. *1. Let $\lambda_i \ge 0,\ i = 1, \dots, Q$ and $\lambda_i > 0$ for at least one i. If a feasible solution $x^* \in X$ is optimal for problem (10.7), then x^* is Pareto-optimal for the multicriteria problem (10.4).*

2. Let x^* be a Pareto-optimal solution for problem (10.4). Then there exists $\lambda_i \ge 0$, $i = 1, \dots, Q$ and $\lambda_i > 0$ for at least one i and a feasible solution x_0 such that x^* is optimal for (10.7).

In the context of the multicriteria supply chain problem (10.3), the hybrid scalarization can be formulated as follows. Let $\lambda_i := p_i$ be the probability of scenario i, $i = 0, \dots, l$. Let $(x^0, y_0^0, \dots, y_l^0)$ be an arbitrary feasible solution of the MOP (10.3). Then a meaningful hybrid scalarization of (10.3) is:

$$\begin{aligned} &\min c^\top x + \sum_{i=0}^{l} p_i d^\top y_i && \\ &\text{s.t. } d^\top y_i \le d^\top y_i^0 && i = 0, \dots, l \\ &\quad Ax \le b && \\ &\quad A_{eq} x = b_{eq} && \\ &\quad Tx + W y_i = h_i && i = 0, \dots, l \\ &\quad x \ge 0 && \\ &\quad y_i \ge 0 && i = 0, \dots, l \end{aligned} \tag{10.8}$$

The objective function of the hybrid scalarization corresponds to the total costs in the supply chain network plus the expected value of the recovery costs. Additionally, the recovery costs for each scenario are restricted by the costs given by the start solution x^0.

The Elastic Constraint Scalarization In practice, the ϵ-constraint scalarization (10.5) is often hard to solve due to the added constraints $f_i(x) \le \epsilon_i$.

The *elastic constraint scalarization* relaxes these constraints and penalizes their violation in the objective function.

Let $k \in \{1, \ldots, K\}$.

Then the elastic constraint scalarization of the MOP (10.4) is

$$\min_{x \in X} f_k(x) + \sum_{\substack{i=1 \\ i \neq k}}^{Q} \mu_i s_i \tag{10.9}$$

$$\text{s.t.} f_i(x) - s_i \leq \epsilon_i \qquad i = 1, \ldots, Q, \quad i \neq k \tag{10.10}$$

$$s_i \geq 0 \qquad i = 1, \ldots, Q, \quad i \neq k$$

where $\mu_i \geq 0,\ i = 1, \ldots, Q, \quad i \neq k$ and $\epsilon_i \in \mathbb{R},\ i = 1, \ldots, Q,\ i \neq k$.

The elastic constraint scalarization has the following properties (see [36]):

Theorem 10.6. 1. *Let (x^*, s^*) be an optimal solution of the elastic constraint scalarization (10.9). Then x^* is weakly Pareto-optimal for the multicriteria problem (10.4).*

2. *Let (x^*, s^*) be a unique optimal solution of the elastic constraint scalarization (10.9). Then x^* is strictly Pareto-optimal for the multicriteria problem (10.4).*

3. *Let x^* be Pareto-optimal for the MOP (10.4). Then for all $k = 1, \ldots, Q$ there exists ϵ, μ and s^* such that (x^*, s^*) is an optimal solution of the elastic constraint scalarization (10.9).*

In the context of the multicriteria supply chain problem (10.3), the elastic constraint scalarization can be formulated as follows.

$$\begin{aligned}
\min\ & c^\top x + \sum_{i=0}^{l} \mu_i s_i \\
\text{s.t. } & d^\top y_i - s_i \le D_i && i = 0,\dots,l \\
& Ax \le b \\
& A_{eq}x = b_{eq} && && (10.11) \\
& Tx + Wy_i = h_i && i = 0,\dots,l \\
& x \ge 0 \\
& y_i \ge 0 && i = 0,\dots,l \\
& s_i \ge 0 && i = 0,\dots,l
\end{aligned}$$

(10.12)

For every scenario i, $i = 0,\dots,l$, upper bounds D_i for the recovery costs are given. The objective function penalizes the exceedance of these upper bound values, where the violation of the desired upper bound values is weighted with a weight factor μ_i for each scenario i.

10.2 Relations between the Scalarizations

In this section, we examine the relations between the ϵ-constraint scalarization and the weighted sum scalarization.

In [36], M. Ehrgott gives the following theorem for a general multicriteria problem $\text{vmin}_{x\in X}(f_1(x),\dots,f_Q(x))$, its weighted sum formulation and the corresponding ϵ-constraint problem $(P_k(\epsilon))$.

Theorem 10.7. *1. Suppose $\hat{x}$ is optimal for $\min_{x\in X} \sum_{i=1}^{Q} \lambda_i f_i(x)$. If $\lambda_k > 0$, $k \in \{1,\dots,Q\}$, there exists $\hat{\epsilon}$ such that $\hat{x}$ solves $(P_k(\hat{\epsilon}))$, too.*

2. Suppose X is a convex set and $f_i : \mathbb{R}^n \to \mathbb{R}$ are convex functions. If $\hat{x}$ is an optimal solution of $(P_k(\hat{\epsilon}))$ for some k and $\hat{\epsilon}$, there exists $\lambda \in \mathbb{R}^Q_+ \setminus \{0\}$ such that $\hat{x}$ is optimal for $\min_{x\in X} \sum_{i=1}^{Q} \lambda_i f_i(x)$.

We want to apply the general theorem 10.7, given by Chankong and Haimes (see [36] and [25]), to the scalarizations of the MOP (10.3) and examine the relation between the LRP (10.1) and the two-stage problem (10.2).

In order to apply the first part of Theorem 10.7 to the scalarizations of the MOP (10.3), we must relax the recovery constraints in the LRP such that for every scenario i, an individual recovery budget D_i can be given, $i = 0, \dots, l$.

Then the following theorem holds:

Theorem 10.8. *Suppose $(\hat{x}, \hat{y}_0, \dots, \hat{y}_l)$ is optimal for the two-stage problem (10.2). Then $(\hat{x}, \hat{y}_0, \dots, \hat{y}_l)$ also solves the LRP (10.1) with $D_i := d^\top \hat{y}_i$, $i = 0, \dots, l$.*

Proof. As $(\hat{x}, \hat{y}_0, \dots, \hat{y}_l)$ is optimal for problem (10.2), we have

$$c^\top x + \sum_{i=0}^{l} p_i d^\top y_i - (c^\top \hat{x} + \sum_{i=0}^{l} p_i d^\top \hat{y}^i) \geq 0 \text{ for all feasible } x. \tag{10.13}$$

Suppose $(\hat{x}, \hat{y}_0, \dots, \hat{y}_l)$ is not optimal for the LRP (10.1) with $D_i := d^\top \hat{y}_i$, $i = 0, \dots, l$. For any $(\tilde{x}, \tilde{y_0}, \dots, \tilde{y_l})$ with $c^\top \tilde{x} < c^\top \hat{x}$ and $d^\top \tilde{y_i} \leq d^\top \hat{y}_i = D_i$, $i = 0, \dots, l$, we have

$$c^\top \tilde{x} - c^\top \hat{x} + \sum_{i=0}^{l} p_i (d^\top \tilde{y_i} - d^\top \hat{y}_i) < 0,$$

which contradicts (10.13). □

Finally, we examine the second part of Theorem 10.7.

Theorem 10.9. *Suppose that $(\hat{x}, \hat{y}_0, \dots, \hat{y}_l)$ is optimal for the LRP (10.1). Let $\bar{\mu}_i$, $i = 0, \dots, l$, be the optimal Lagrange multipliers corresponding to the constraint functions $d^\top y_i \leq D$. Then $(\hat{x}, \hat{y}_0, \dots, \hat{y}l)$ is optimal for the two-stage problem (10.2) with $p_i = \bar{\mu}_i$.*

Proof. Let $f(x, y_0, \dots, y_l) = c^\top x$ and $f_i(x, y_0, \dots, y_l) = d^\top y_i$, $i = 0, \dots, l$. Let $g_j(x, y_0, \dots, y_l) \geq 0$, $j \in J$, denote the inequality constraints in problem (10.3), i.e. $-Ax + b \geq 0$, $x \geq 0$ and $y_i \geq 0$, $i = 0, \dots, l$, line by line. Let $h_s(x, y_0, \dots, y_l) = 0$, $s \in S$ denote the equality constraints in problem (10.3), i.e. $A_{eq} x - b_{eq} = 0$ and $Tx + Wy_i - h_i = 0$, $i = 0, \dots, l$, line by line. Note that these constraints also appear in the problems (10.1) and (10.2) without any modification.

For simplicity, let $\bar{x} := (\hat{x}, \hat{y}_0, \dots, \hat{y}_l)$ in the following.

As (10.1) is a linear program and $\bar{x}$ is optimal for problem (10.1), $\bar{x}$ is a KKT point for problem (10.1). The KKT conditions for problem (10.1) at $(\hat{x}, \hat{y}_0, \dots, \hat{y}_l)$ are:

$$\begin{aligned}
\nabla f(\bar{x}) + \sum_{i=0}^{l} \bar{\mu_i} \nabla f_i(\bar{x}) - \sum_{j \in J} \bar{\nu_j} \nabla g_j(\bar{x}) - \sum_{s \in S} \bar{\xi_s} \nabla h_s(\bar{x}) = 0 & \\
\bar{\mu_i}(D - f_i(\bar{x})) = 0, \quad & i = 0, \dots, l \\
\bar{\nu_j}(g_j(\bar{x})) = 0, \quad & j \in J \\
\bar{\mu_i} \geq 0, \quad & i = 0, \dots, l \\
\bar{\nu_j} \geq 0, \quad & j \in J \\
\bar{\xi_s} \in \mathbb{R}, \quad & s \in S
\end{aligned} \tag{10.14}$$

In contrast to the standard formulation of the KKT conditions, we consider inequality constraints $g_j(x, y_0, \dots, y_l) \geq 0,\ j \in J$ instead of $g_j(x, y_0, \dots, y_l) \leq 0,\ j \in J$. Therefore we have to put a minus sign in front of the sum $\sum_{j \in J} \bar{\nu_j} \nabla g_j(\bar{x})$. Concerning the sum $\sum_{s \in S} \bar{\xi_s} \nabla h_s(\bar{x})$, which corresponds to the equality constraints, a minus or plus sign is not mandatory. In this context, we use the minus sign in order to treat both types of contraints equally.

Now we consider the two-stage problem (10.2) with $p_i = \bar{\mu_i}$:

$$\begin{aligned}
\min\ & c^\top x + \sum_{i=0}^{l} \bar{\mu_i} d^\top y_i \\
\text{s.t. } & Ax \leq b \\
& A_{eq} x = b_{eq} \\
& Tx + W y_i = h^i & i = 0, \dots, l \\
& x \geq 0 \\
& y_i \geq 0 & i = 0, \dots, l
\end{aligned} \tag{10.15}$$

The KKT conditions for problem (10.15) at a point $\tilde{x}$ are:

$$\begin{aligned}
\nabla f(\tilde{x}) + \sum_{i=0}^{l} \bar{\mu_i} \nabla f_i(\tilde{x}) - \sum_{j \in J} \tilde{\nu_j} \nabla g_j(\tilde{x}) - \sum_{s \in S} \tilde{\xi_s} \nabla h_s(\tilde{x}) = 0 & \\
\tilde{\nu_j}(g_j(\tilde{x})) = 0, \quad & j \in J \\
\tilde{\nu_j} \geq 0, \quad & j \in J \\
\tilde{\xi_s} \in \mathbb{R}, \quad & s \in S
\end{aligned} \tag{10.16}$$

Note that we again have minus signs in front of the sums corresponding to the inequality and equality constraints.

Then $\bar{x} = (\hat{x}, \hat{y}_0, \ldots, \hat{y}_l)$ is a KKT point for problem (10.15) with $\tilde{\nu_j} := \bar{\nu_j}$, $j \in J$, and $\tilde{\xi_s} = \bar{\xi_s}$, $s \in S$.

As problem (10.15) is linear and therefore convex, $\bar{x}$ is optimal for problem (10.15). □

In the original two-stage optimization problem, p_i corresponds to the probability of scenario i. In general, the Lagrange multipliers $\bar{\mu_i}$ are not necessarily smaller than 1 and do not sum up to 1 over all scenarios. Let $\bar{M} := \sum_{i=0}^{l} \bar{\mu_i}$. Then the weighted sum problem of Theorem 10.9 corresponds to a two-stage problem with scenario set S with probabilities $\bar{p_i} = \frac{\bar{\mu_i}}{\bar{M}}$, $i = 0, \ldots, l$ and a weight factor $\frac{1}{\bar{M}}$ in the first stage objective function $c^\top x$.

Theorems 10.8 and 10.9 show that the LRP (10.1) and the two-stage problem (10.2) can be transformed into each other under certain conditions: When considering a two-stage problem of the form (10.2), it can be transformed into an LRP with the additional condition that the recovery budget can possibly take a different value for each scenario. When considering an LRP of the form (10.1), it can be transformed into a two-stage problem where the first stage objective function is weighted with a factor $\frac{1}{\bar{M}}$. Then the (transformed) LRP (10.1) can be solved using efficient cutting plane algorithms like the L-shaped algorithm (see Subsection 9.2.1).

10.3 Sensitivity for the e-Constraint Scalarization

In this section, we examine the sensitivity for the LRP (10.1) for modifications of the recovery budget D.

Consider the general ϵ-constraint scalarization $(P_k(\epsilon))$

$$\begin{aligned}
&\min\ f_k(x) && \\
&\text{s.t. } f_i(x) \leq \epsilon_i && i = 1, \ldots, Q,\ i \neq k \\
&\quad\ \ g_j(x) \geq 0 && j = 1, \ldots, p \\
&\quad\ \ h_l(x) = 0 && l = 1, \ldots, q \\
&\quad\ \ x \in \mathbb{R}^n &&
\end{aligned} \tag{10.17}$$

with real-valued functions f_i, $i = 1, \ldots, Q$, g_j, $j = 1, \ldots, p$ and h_l, $l = 1, \ldots, q$. Let x^* be a local optimal solution of the problem $(P_k(\epsilon^*))$ with optimal Lagrange multipliers $\mu_i^* \in \mathbb{R}_+$ for $i = 1, \ldots, Q$, $i \neq k$, $\nu^* \in \mathbb{R}_+^p$ and $\xi^* \in \mathbb{R}^q$. Then we define the following index sets:

$$\begin{aligned}
I^+ &:= \{i \in \{1, \ldots, Q\} \setminus \{k\} | f_i(x^*) = \epsilon_i^*,\ \mu_i^* > 0\} \\
I^0 &:= \{i \in \{1, \ldots, Q\} \setminus \{k\} | f_i(x^*) = \epsilon_i^*,\ \mu_i^* = 0\} \\
I^- &:= \{i \in \{1, \ldots, Q\} \setminus \{k\} | f_i(x^*) < \epsilon_i^*,\ \mu_i^* = 0\}
\end{aligned} \tag{10.18}$$

and

$$\begin{aligned}
J^+ &:= \{j \in J | g_j(x^*) = 0,\ \nu_j^* > 0\} \\
J^0 &:= \{j \in J | g_j(x^*) = 0,\ \nu_j^* = 0\} \\
J^- &:= \{j \in J | g_j(x^*) > 0,\ \nu_j^* = 0\}
\end{aligned} \tag{10.19}$$

Based on the results of V. Chankong and Y.Y. Haimes, see [25], we can state the following theorem and proof.

Theorem 10.10. *Let f_i, $i = 1, \ldots, Q$, g_j, $j = 1, \ldots, p$ and h_l, $l = 1, \ldots, q$ be linear functions. Let $\epsilon^* \in \mathbb{R}^{Q-1}$. Let x^* be a non-degenerate solution of $(P_k(\epsilon^*))$ and let μ_i^* denote the optimal Lagrange multipliers corresponding to the constraint $f_i(x) \leq \epsilon_i^*$, $i \neq k$.*
Then

(i) For all $i = 1, \ldots, Q$, $i \neq k$, it holds that

$$\frac{\partial f_k(x^*)}{\partial \epsilon_i} = -\mu_i^* \tag{10.20}$$

(ii) *Let $x(\epsilon)$ denote the minimal solution of $(P_k(\epsilon))$. Then there exists a neighborhood $N(\epsilon^*)$ of ϵ^* such that*

$$f_i(x(\epsilon)) = \epsilon_i \text{ for all } i \in I^+, \text{ for all } \epsilon \in N(\epsilon^*). \tag{10.21}$$

Proof. Consider a linear instance of $(P_k(\epsilon^*))$. Let $b_{\epsilon^*} := (\epsilon_1^*, \ldots, \epsilon_{k-1}^*, \epsilon_{k+1}^*, \ldots, \epsilon_Q^*, 0, \ldots, 0)^\top$ be the right hand side of the constraints of $(P_k(\epsilon^*))$.

Let x^* be a non-degenerate optimal solution of $(P_k(\epsilon^*))$ with optimal basis B_{ϵ^*} and π^* the corresponding shadow prices (also called dual prices or simplex multipliers, see [30]).

$$\pi^* = (-\mu_1^*, \ldots, -\mu_{k-1}^*, -\mu_{k+1}^*, \ldots, -\mu_Q^*, \nu_1^*, \ldots, \nu_p^*, \xi_1^*, \ldots, \xi_q^*). \tag{10.22}$$

As μ_i^*, $i = 1, \ldots, Q$, $i \neq k$ correspond to the constraints $f_i(x) \leq \epsilon_i^*$ in $(P_k(\epsilon^*))$, which are formulated as $\leq$-constraints, μ_i^* have negative signs, $i = 1, \ldots, Q$, $i \neq k$.

Let $x_{B_{\epsilon^*}}$ be the optimal basic variables, let $A_{B_{\epsilon^*}}$ be the coefficient matrix of the optimal basic variables and $c_{B_{\epsilon^*}}$ the cost coefficients corresponding to the optimal basic variables. As $A_{B_{\epsilon^*}}$ is regular, the inverse matrix exists and $\pi^* = c_{B_{\epsilon^*}}^\top A_{B_{\epsilon^*}}^{-1}$.

Then

$$x_{B_{\epsilon^*}} = A_{B_{\epsilon^*}}^{-1} b_{\epsilon^*} \tag{10.23}$$

and consequently

$$f_k(x^*) = c_{B_{\epsilon^*}}^\top x_{B_{\epsilon^*}} = c_{B_{\epsilon^*}}^\top A_{B_{\epsilon^*}}^{-1} b_{\epsilon^*} = {\pi^*}^\top b_{\epsilon^*} \tag{10.24}$$

As x^* is a non-degenerate solution, each component of $x_{B_{\epsilon^*}}$ is nonzero, there exists a neighborhood $N(\epsilon^*)$ of ϵ^* such that for all ϵ in $N(\epsilon^*)$, the optimal set of basic variables does not change. Hence, we have

$$B_\epsilon = B_{\epsilon^*} \tag{10.25}$$

$$A_{B_\epsilon} = A_{B_{\epsilon^*}} \tag{10.26}$$

$$c_{B_\epsilon} = c_{B_{\epsilon^*}} \tag{10.27}$$

$$\pi_{B_\epsilon} = \pi_{B_{\epsilon^*}} := \pi^*, \tag{10.28}$$

where π_{B_ϵ} denotes the simplex multipliers of problem $(P_k(\epsilon))$, corresponding to the optimal basis B_ϵ.

Furthermore, we have

$$x_{B_\epsilon} = A_{B_{\epsilon^*}}^{-1} b_\epsilon. \tag{10.29}$$

Then x_{B_ϵ} is a continuously differentiable linear function in ϵ.

Since $f_k(x_{B_\epsilon}) = \pi^* b_\epsilon$ for all ϵ in $N(\epsilon^*)$, we can conclude that

$$\frac{\partial f_k(x^*)}{\partial \epsilon_i} = -\mu_i^* \qquad \forall i = 1, \dots, Q,\ i \neq k \tag{10.30}$$

and (i) follows.

According to the construction of $N(\epsilon^*)$, for all ϵ in $N(\epsilon^*)$, the optimal set of basic variables does not change. Therefore, for all constraints i of problem $(P_k(\epsilon^*))$ that are active at x^* and where $\mu_i > 0$, i.e. for all $i \in I^+$, the constraints are also active at $x(\epsilon)$ for problem $(P_k(\epsilon))$, as active constraints remain active. Hence, we have

$$f_i(x(\epsilon)) = \epsilon_i \qquad \forall i \in I^+, \qquad \forall \epsilon \in N(\epsilon^*). \tag{10.31}$$

and (ii) follows.

$\square$

Following the interpretation of the sensitivity analysis given in [37], we can now apply the results of Theorem 10.10 on the LRP (10.1).

Let $f(x, y_0, \dots, y_l) = c^\top x$ and $f_i(x, y_0, \dots, y_l) = d^\top y_i,\ i = 0, \dots, l$. Let $g_j(x, y_0, \dots, y_l) \geq 0,\ j \in J$, denote the inequality constraints and

$h_s(x, y_0, \ldots, y_l) = 0,\ s \in S$, the equality constraints in problem (10.1). Let $\bar{x} := (\hat{x}, \hat{y}_0, \ldots, \hat{y}_l)$ be a non-degenerate optimal solution of problem (10.1) with Lagrange multipliers $\bar{\mu}_i$ to the constraints $f_i(x) \leq D$ for $i = 0, \ldots, l$.

Now we consider problem (10.1) with modified recovery budget $\tilde{D}$. If $\tilde{D}$ lies in an appropriate neighborhood of the original recovery budget D, the following approximation holds for the objective function $f(x)$:

$$f(x(\tilde{D})) \approx f(\bar{x}) + \sum_{i=0}^{l} \frac{\partial f(\bar{x})}{\partial \epsilon_i}(\tilde{D} - D) = c^\top \bar{x} - \sum_{i=0}^{l} \bar{\mu}_i(\tilde{D} - D), \qquad (10.32)$$

where $x(\tilde{D})$ denotes the minimal solution of problem (10.1) with recovery budget $\tilde{D}$. This means that if the recovery budget is increased (decreased) within a neighborhood of the original recovery budget D, the total cost in the objective function is decreased (increased) by the absolute value of the difference in the recovery budget times the sum over $\bar{\mu}_i$.

Furthermore, we have

$$f_i(x(\tilde{D})) = d^\top \hat{y}_i(\tilde{D}) = \tilde{D} \qquad (10.33)$$

for the active constraints of the problem with original recovery budget. This means that for the modified recovery budget, the budget will be fully exploited for all scenarios in which it was fully exploited for the unmodified recovery budget.

10.4 Summary and Conclusions

In this chapter, we drew a link between the recoverable robustness approach for network flow problems with uncertain demands, as it was proposed in Section 8.3, and the two-stage model from Stochastic Programming as presented in Chapter 9.

The two problem formulations are linked via a multicriteria optimization problem: We showed in Section 10.1 that the recoverable robustness problem is an ϵ-constraint scalarization of this MOP, while the two-stage problem is a weighted sum formulation of the same MOP.

In two theorems, we showed in Section 10.2 that the recoverable robustness problem and the two-stage problem can be transformed into each other under certain conditions. That means that if these conditions are fulfilled, the special properties and algorithms of one problem type are applicable to the other problem type, e.g. the (transformed) recoverable robustness problem can be solved using Stochastic Programming techniques like the L-shaped algorithm.

Finally, we examined the sensitivity of the recoverable robustness problem for modifications of the recovery budget in Section 10.3. We showed in which way the total cost value is decreased (increased), if the recovery budget is increased (decreased) in a neighborhood of the original recovery budget.

Chapter 11

Computational Results: Numerical Tests

In this chapter, we evaluate the heuristic methods, which were developed in this thesis, by means of several example networks. In particular, we examine the different implementations of the ant algorithm, i.e. the ant algorithm for the threshold minimum cost flow problem, the ant algorithm for minimum cost flow problems with uncertain costs and the ant algorithm for minimum cost flow problems with uncertain demands.

As introduced in Section 3.1, we focus on network flow problems where the underlying network is a layered network. In particular, the numerical tests are carried through on the basis of randomly generated fully layered networks.

11.1 Random Network Generation

For the numerical tests, we need to generate a considerable quantity of layered networks. Therefore, a layered network generator was developed, which allows the random generation of fully layered networks on the basis of several input parameters.

The layered network generator has the following input parameters:

- number of node layers in the layered network
- minimum/maximum number of nodes per layer: For every node layer, the actual number of nodes is a random number within these bounds

- lower/upper cost bound: For every arc, a random cost value within these bounds is generated
- lower/upper capacity bound: For every arc, a random capacity value within these bounds is generated
- number of commodities

As we want to generate single-source single-sink networks, both the first and the last layer contain exactly one node. The number of nodes in the other layers is generated at random, respecting the minimum and maximum number of nodes per layer.

We generate fully layered networks, that means that each node, except the sink node, is connected via an arc with each node of the following layer. Costs and capacities on these arcs are generated at random within the given bounds. In order to avoid infeasible network flow problems, we set the capacity to the maximum value for all outgoing arcs of the source node. Note that this approach does not guarantee the feasibility of the generated network flow problems.

The nodes in the next to last layer are considered as the demand nodes. The (random generated) capacities on the incoming arcs of the sink node determine the demand values of these demand nodes. The supply at the source node is therefore given by the sum over the capacities of the incoming arcs of the sink node.

For the generation of the networks for the specific problem types (TMCFP, MCFP with uncertain costs or uncertain demands), additional input parameters are required and will be specified later.

11.2 Threshold Minimum Cost Flow Problem

In this section, we examine the computation time and performance of the SOS 2 Branch and Bound method, see Section 5.3, and the ant algorithm for the threshold minimum cost flow problem (TMCFP), as proposed in Section 5.1. In order to be able to interpret the results, we calculate the optimal solution of the corresponding minimum cost flow problem for every network, using the Cost Scaling Algorithm for evaluation.

For the numerical tests, 50 random layered networks were generated, grouped in five groups according to the number of layers. The following table lists the node and arc data: number of layers, nodes and arcs, lower and upper cost and capacity bounds, threshold value and number of commodities.

network	#layers	# nodes	# arcs	nodes per layer	cost	capacity	threshold	com
T001	5	18	66	4 – 8	0 – 5	3 – 10	5	1
T002	5	23	117	4 – 8	0 – 5	3 – 10	5	1
T003	5	17	63	4 – 8	0 – 5	3 – 10	5	1
T004	5	20	75	4 – 8	0 – 5	3 – 10	5	1
T005	5	17	60	4 – 8	0 – 5	3 – 10	5	1
T006	5	14	40	4 – 8	0 – 5	3 – 10	5	1
T007	5	17	55	4 – 8	0 – 5	3 – 10	5	1
T008	5	23	117	4 – 8	0 – 5	3 – 10	5	1
T009	5	17	60	4 – 8	0 – 5	3 – 10	5	1
T010	5	22	108	4 – 8	0 – 5	3 – 10	5	1
T011	10	48	241	4 – 8	0 – 10	3 – 20	5	1
T012	10	48	245	4 – 8	0 – 10	3 – 20	5	1
T013	10	48	248	4 – 8	0 – 10	3 – 20	5	1
T014	10	43	177	4 – 8	0 – 10	3 – 20	5	1
T015	10	57	362	4 – 8	0 – 10	3 – 20	5	1
T016	10	46	219	4 – 8	0 – 10	3 – 20	5	1
T017	10	50	249	4 – 8	0 – 10	3 – 20	5	1
T018	10	44	191	4 – 8	0 – 10	3 – 20	5	1
T019	10	49	256	4 – 8	0 – 10	3 – 20	5	1
T020	10	55	337	4 – 8	0 – 10	3 – 20	5	1
T021	15	101	711	5 – 10	0 – 10	3 – 20	5	1
T022	15	101	710	5 – 10	0 – 10	3 – 20	5	1
T023	15	97	637	5 – 10	0 – 10	3 – 20	5	1
T024	15	109	843	5 – 10	0 – 10	3 – 20	5	1
T025	15	83	485	5 – 10	0 – 10	3 – 20	5	1
T026	15	97	665	5 – 10	0 – 10	3 – 20	5	1
T027	15	100	703	5 – 10	0 – 10	3 – 20	5	1
T028	15	107	835	5 – 10	0 – 10	3 – 20	5	1
T029	15	112	860	5 – 10	0 – 10	3 – 20	5	1
T030	15	103	752	5 – 10	0 – 10	3 – 20	5	1
T031	20	141	1054	5 – 10	5 – 20	5 – 20	7	1
T032	20	135	937	5 – 10	5 – 20	5 – 20	7	1
T033	20	139	1026	5 – 10	5 – 20	5 – 20	7	1
T034	20	133	917	5 – 10	5 – 20	5 – 20	7	1
T035	20	137	972	5 – 10	5 – 20	5 – 20	7	1
T036	20	137	974	5 – 10	5 – 20	5 – 20	7	1
T037	20	134	935	5 – 10	5 – 20	5 – 20	7	1
T038	20	126	818	5 – 10	5 – 20	5 – 20	7	1
T039	20	139	970	5 – 10	5 – 20	5 – 20	7	1
T040	20	147	1136	5 – 10	5 – 20	5 – 20	7	1
T041	25	177	1251	5 – 10	5 – 20	5 – 20	10	1
T042	25	188	1449	5 – 10	5 – 20	5 – 20	10	1
T043	25	183	1358	5 – 10	5 – 20	5 – 20	10	1
T044	25	187	1439	5 – 10	5 – 20	5 – 20	10	1
T045	25	176	1269	5 – 10	5 – 20	5 – 20	10	1
T046	25	177	1282	5 – 10	5 – 20	5 – 20	10	1
T047	25	181	1322	5 – 10	5 – 20	5 – 20	10	1
T048	25	175	1213	5 – 10	5 – 20	5 – 20	10	1
T049	25	164	1130	5 – 10	5 – 20	5 – 20	10	1
T050	25	180	1309	5 – 10	5 – 20	5 – 20	10	1

Table 11.1: Random layered networks for the TMCFP

We apply the ant algorithm and the SOS 2 Branch and Bound method to these example problems. In order to evaluate the quality of the results, we additionally state the exact results for the corresponding MCFP, calculated by the Cost Scaling Algorithm.

For the runs of the ant algorithm, we choose the parameter settings as given in the following table:

parameter	setting	parameter	setting
α	1.0	η_{min}	0.2
β	3.0	τ_{min}	1.0
ρ	0.25	τ_{max}	20.0

Table 11.2: TMCFP - ant algorithm: Parameter settings

The choice of the parameters α, β and ρ corresponds to the recommendation given in Subsection 5.1.1. $\eta_{min} = 0.1$ or $\eta_{min} = 0.2$ are standard settings. The limiting values for τ were chosen dependent on the scale of the demand values, which corresponds to the number of ants in the network.

For each TMCFP, we run the ant algorithm 10 times. Out of these runs, we calculate the average running times, costs and threshold violations as representative for the ant algorithm.

For both the ant algorithm and the SOS 2 Branch and Bound method, we measure the running time. Furthermore, we count the number of arcs, where the threshold claim is violated ("threshold violation"), i.e. where the amount of flow is greater than zero but less than a certain threshold value. In addition, we compare the cost values for the ant algorithm, the SOS 2 B&B method and the CSA algorithm.

The following table lists the results for networks T001 to T050: running time in seconds (⏱), total cost and threshold violation.

	⏱		cost			threshold violation		
network	ants	SOS2	ants	SOS2	CSA	ants	SOS2	CSA
T001	0.28	82.50	372	370	342	7	0	18
T002	0.29	10914.21	261	227	215	7	0	12
T003	0.20	11297.68	137	153	121	6	2	8
T004	0.35	7404.40	297	269	260	10	1	17
T005	0.18	5762.81	102	151	89	9	3	10
T006	0.16	220.12	272	274	253	6	0	10
T007	0.27	224.06	300	302	270	8	0	15
T008	0.34	17283.01	166	177	146	6	0	11
T009	0.22	9632.45	308	310	289	6	0	9
T010	0.29	35341.28	304	311	224	10	1	22
T011	1.53	-	1602	-	1278	10	-	21
T012	1.16	-	1305	-	1017	7	-	17
T013	1.29	-	1253	-	905	11	-	17
T014	1.13	-	1890	-	1584	7	-	18
T015	1.66	-	880	-	661	10	-	22
T016	1.17	-	2010	-	1635	9	-	15
T017	1.88	-	2429	-	1825	11	-	22
T018	1.01	-	1512	-	1246	8	-	22
T019	1.36	-	1286	-	925	4	-	13
T020	1.47	-	1155	-	930	10	-	29
T021	8.27	-	2696	-	1692	25	-	45
T022	7.19	-	1585	-	853	9	-	33
T023	7.72	-	2713	-	1884	28	-	42
T024	9.84	-	2165	-	1130	18	-	55
T025	4.26	-	2733	-	1876	15	-	32
T026	7.69	-	1162	-	862	16	-	28
T027	8.18	-	2549	-	1532	17	-	31
T028	9.48	-	2607	-	2032	18	-	43
T029	11.54	-	3355	-	2340	22	-	33
T030	10.18	-	3319	-	2294	23	-	51
T031	24.07	-	14160	-	10759	37	-	89
T032	20.38	-	20818	-	17145	46	-	73
T033	23.10	-	18112	-	13075	38	-	97
T034	18.55	-	17776	-	13883	37	-	74
T035	24.86	-	21168	-	16869	36	-	84
T036	24.44	-	16877	-	12914	38	-	85
T037	21.27	-	17751	-	14042	43	-	62
T038	16.97	-	15925	-	11797	37	-	68
T039	22.06	-	19428	-	15094	46	-	82
T040	26.62	-	12761	-	9363	34	-	85

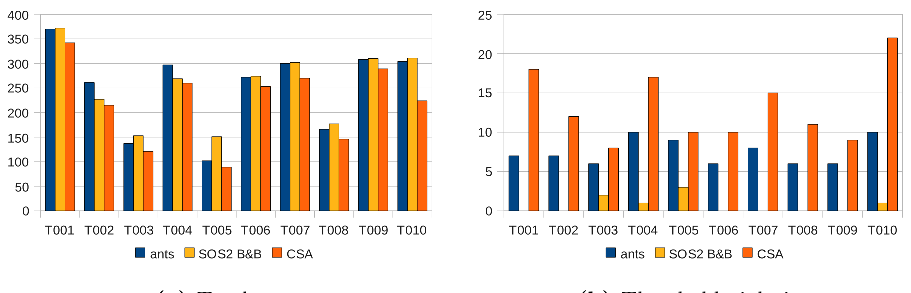
	⏱			cost		threshold violation		
network	ants	SOS2	ants	SOS2	CSA	ants	SOS2	CSA
T041	56.98	-	28605	-	20407	78	-	188
T042	64.90	-	25471	-	17382	78	-	180
T043	62.26	-	29213	-	21304	86	-	196
T044	66.22	-	27945	-	20366	79	-	175
T045	52.92	-	21288	-	15106	61	-	148
T046	52.58	-	22232	-	16383	66	-	149
T047	55.99	-	18067	-	12308	60	-	153
T048	53.14	-	26156	-	19641	77	-	169
T049	39.40	-	11650	-	7650	42	-	110
T050	49.70	-	24805	-	17926	83	-	174

Table 11.3: Numerical tests - threshold problem: Results

The running time is given in seconds.

When applying the SOS 2 Branch and Bound method, we observe that it is only applicable in a reasonable time for the smallest problem instances (five layers). The running time increases immensely with growing problem sizes. Therefore we didn't run the SOS 2 B&B method for networks T011 to T050. However, as the SOS 2 B&B method is an exact method, the obtained results for networks T001 to T010 are optimal.

First, we examine the results for networks T001 to T010 in detail. The following figure illustrates the nominal costs of the solutions of the different optimization methods as well as the corresponding threshold violation.



(a) Total cost **(b)** Threshold violation

Figure 11.1: Numerical tests - threshold problem: Ants vs. SOS2 B&B vs. CSA

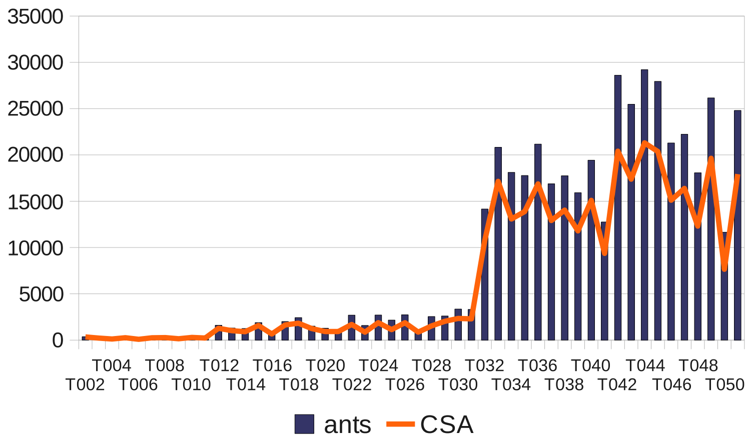
As we would expect, both the ant algorithm and the SOS 2 Branch and Bound method reduce the number of threshold violations, while the total cost increases. However, the SOS2 B&B method reduces the number of threshold violations to the absolute minimum, while the average threshold violation reduction of the ant algorithm is 40%, compared to the threshold violations of the CSA.

In return, the nominal costs of the solutions of the ant algorithm and the SOS 2 B&B method are higher than the costs of the CSA solutions. In 8 out of 10 cases, the solution of the SOS 2 B&B is most expensive. In the remaining cases, the solutions of the ant algorithm are more expensive than the SOS2 B&B solutions.

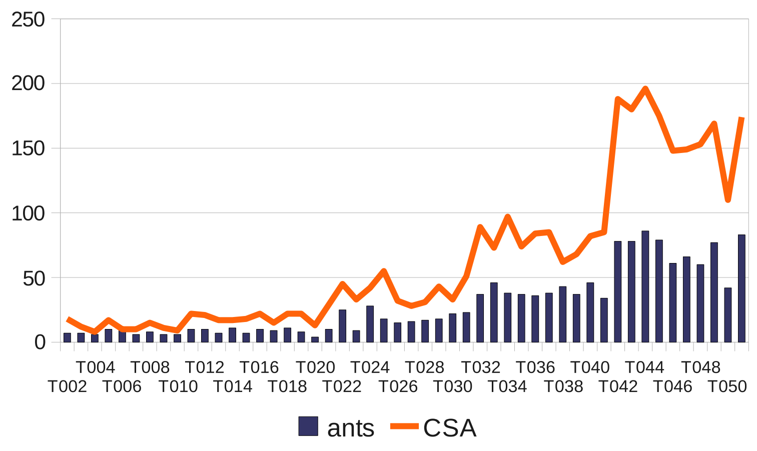
Considering the performance concerning the nominal cost value and the threshold violations, we observe that the ant algorithm can not keep up with the SOS 2 B&B method. In contrast, a comparison of the running time shows that the SOS 2 B&B method is outperformed by the ant algorithm in this respect.

When examining the running time of the SOS 2 Branch and Bound method, we can see that the running time isn't stable for problems with similar problem size: The fastest running time, 82.50 seconds (network T001), is over 400 times faster than the slowest one, 35341.28 seconds (network T010). Therefore, the running time for a problem of a certain size cannot be predicted in advance. In contrast, the running time of the ant algorithm is both stable and fast for a specific problem size and lies within a range of 0.16 to 0.35 seconds for all ten problem instances.

In order to further evaluate the results obtained by the ant algorithm, we take a closer look at the results of the ant algorithm compared with those of the CSA for the networks T001 to T050.



(a) Total cost



(b) Threshold violation

Figure 11.2: Numerical tests - threshold problem: Ants vs. CSA

The average reduction of the threshold violation by the ant algorithm is 50%, compared to the CSA solution. In return, the cost of the ant solution is 34% higher on the average.

The maximum reduction of the threshold violation is 73% for network T022. The maximum increase in cost is 92% for network T024.

Especially for the larger network instances, the percentage of threshold violation reduction is rather high: We observe an average reduction of 57% for the networks with 25 layers (T041 to T050).

We can see that the ant algorithm is able to significantly reduce the number of arcs where the threshold claim is violated. However, the price to pay is a not insignificant increase in the total cost. In practice, the advantages and disadvantages of cheap solutions containing many small amounts of flow and more expensive solutions with considerably less small amounts of flow must be weighted: If small amounts of flow cause large additional costs, which can not be expressed as cost values in the underlying network and therefore are ignored during the optimization, a solution which has higher nominal costs (in the network), but reduces the number of small amounts of flow, might be favorable. If the problem instance is not too large and the running time is not critical, the SOS 2 B&B method might be the best choice.

11.3 Uncertain Costs

In this section, we compare the computation time and performance of the ant algorithm for Gaussian distributed costs, as presented in Subsection 6.3.2, and the results of the multicommodity extension of the algorithm of Bertsimas & Sim for minimum cost flow problems with uncertain costs, as presented in Section 6.2.

Again, we randomly generate fully layered networks for the computational tests. For the random generation of fully layered networks with uncertain cost values, we have one additional input parameter:

- extra cost bound: For every arc, a random extra cost value between 0 and the extra cost bound is generated per commodity

We consider both single-commodity and multicommodity minimum cost flow problems with uncertain cost values.

For the numerical tests, 30 random layered networks were generated, grouped in three groups according to the number of layers:

network	#layers	# nodes	# arcs	nodes per layer	cost	capacity	extra cost	com
C001	5	21	75	4 – 8	0 – 10	3 – 20	5	1
C002	5	16	50	4 – 8	0 – 10	3 – 20	5	1
C003	5	22	90	4 – 8	0 – 10	3 – 20	5	1
C004	5	19	77	4 – 8	0 – 10	3 – 20	5	1
C005	5	16	50	4 – 8	0 – 10	3 – 20	5	1
C006	5	23	117	4 – 8	0 – 10	3 – 20	5	1
C007	5	21	91	4 – 8	0 – 10	3 – 20	5	1
C008	5	18	66	4 – 8	0 – 10	3 – 20	5	1
C009	5	20	90	4 – 8	0 – 10	3 – 20	5	5
C010	5	19	77	4 – 8	0 – 10	3 – 20	5	5
C011	10	56	332	4 – 8	0 – 10	3 – 20	5	1
C012	10	48	243	4 – 8	0 – 10	3 – 20	5	1
C013	10	44	194	4 – 8	0 – 10	3 – 20	5	1
C014	10	51	270	4 – 8	0 – 10	3 – 20	5	1
C015	10	48	254	4 – 8	0 – 10	3 – 20	5	1
C016	10	50	265	4 – 8	0 – 10	3 – 20	5	1
C017	10	49	257	4 – 8	0 – 10	3 – 20	5	1
C018	10	49	254	4 – 8	0 – 10	3 – 20	5	1
C019	10	50	252	4 – 8	0 – 10	3 – 20	5	5
C020	10	48	253	4 – 8	0 – 10	3 – 20	5	5

network	#layers	# nodes	# arcs	nodes per layer	cost	capacity	extra cost	com
C021	15	103	731	5 – 10	5 – 20	3 – 20	10	1
C022	15	96	641	5 – 10	5 – 20	3 – 20	10	1
C023	15	108	843	5 – 10	5 – 20	3 – 20	10	1
C024	15	97	642	5 – 10	5 – 20	3 – 20	10	1
C025	15	93	612	5 – 10	5 – 20	3 – 20	10	1
C026	15	102	734	5 – 10	5 – 20	3 – 20	10	1
C027	15	104	761	5 – 10	5 – 20	3 – 20	10	1
C028	15	94	602	5 – 10	5 – 20	3 – 20	10	1
C029	15	96	637	5 – 10	5 – 20	3 – 20	10	5
C030	15	93	601	5 – 10	5 – 20	3 – 20	10	5

Table 11.4: Random layered networks for the MCFP with uncertain cost

Both the ant algorithm and the algorithm of Bertsimas and Sim provide a robustness parameter Γ to control the level of robustness of the solution. In our test runs, we consider three different robustness levels for each network: $\Gamma = 0$, $\Gamma = 0.5$ and $\Gamma = 1$.

For the runs of the ant algorithm for uncertain cost values, we choose the parameter settings as given in the following table:

parameter	setting	parameter	setting
α	3.0	η_{min}	0.1
β	3.0	τ_{min}	1.0
ρ	0.25	τ_{max}	20.0

Table 11.5: Uncertain costs - ant algorithm: Parameter settings

The settings for the parameters α, β, ρ, η_{min} and τ_{min} are our standard settings for the ant algorithm for uncertain cost values. The upper pheromone bound τ_{max} was chosen dependent on the scale of the demand values, which corresponds to the number of ants in the network.

Differing from the settings given in Subsection 6.3.2 for the random generation of the cost values, the standard deviation σ_{ij}^k for arc (i, j) and commodity k is set to $\sigma_{ij}^k := \frac{1}{20} d_{ij}^k$, where d_{ij}^k is the extra cost value for arc (i, j) and commodity k. This modification is based on the fact that the cost intervals are relatively large. The choice of the factor $\frac{1}{20}$ instead of $\frac{1}{4}$, which was used in Subsection 6.3.2, ensures that the generated cost values are not spread too widely over the whole cost interval.

As before, we consider the average of 10 runs as representative for the ant algorithm.

The following table lists the results of the runs for each network and the three values of the robustness parameter Γ: running time in seconds (⏱), best case and worst case costs for the ant algorithm and the algorithm of Bertsimas and Sim (B&S).

		⏱		best case		worst case	
network	Γ	ants	B&S	ants	B&S	ants	B&S
C001	0	1.06	4.33	921	894	1632	1619
	0.5	0.88	4.78	937	929	1616	1597
	1	1.12	4.62	941	920	1606	1591
C002	0	0.96	2.83	957	953	1462	1460
	0.5	0.59	2.89	968	1042	1460	1435
	1	0.51	2.89	1077	1037	1433	1433
C003	0	1.06	4.51	1113	1092	1843	1841
	0.5	1.00	5.22	1133	1149	1846	1821
	1	1.16	5.34	1170	1149	1839	1821
C004	0	0.44	5.72	627	600	1091	1080
	0.5	0.36	4.76	612	640	1082	1078
	1	0.41	4.71	619	640	1085	1078
C005	0	0.56	3.57	468	467	1098	1098
	0.5	0.43	3.34	468	509	1098	1097
	1	0.43	3.33	514	509	1097	1097
C006	0	0.82	6.63	508	501	1424	1420
	0.5	0.76	7.59	510	606	1421	1420
	1	0.77	7.59	580	606	1421	1420
C007	0	0.66	5.15	561	534	1301	1296
	0.5	0.71	5.18	572	660	1307	1296
	1	0.75	5.18	736	660	1298	1296
C008	0	0.80	3.23	1303	1290	1948	1932
	0.5	0.85	3.77	1318	1312	1949	1928
	1	0.71	3.76	1319	1312	1933	1928
C009	0	8.23	55.98	2955	2481	5265	5042
	0.5	7.17	63.05	3160	2802	5237	5010
	1	6.97	63.06	3576	2802	5329	5010
C010	0	10.50	47.70	4066	3568	7040	6788
	0.5	9.89	45.60	4266	3764	7058	6725
	1	9.78	44.76	4652	3764	7171	6725
C011	0	4.58	52.30	1081	1001	3173	3149
	0.5	2.93	50.58	1063	1624	3146	3157
	1	3.33	53.62	1471	1417	3144	3117
C012	0	2.59	38.85	1324	1269	2886	2873

		⏱		best case		worst case	
network	Γ	ants	B&S	ants	B&S	ants	B&S
	0.5	2.50	35.94	1406	1528	2903	2838
	1	2.34	36.30	1621	1528	2874	2838
C013	0	2.72	20.92	1517	1449	2912	2877
	0.5	1.99	23.27	1529	1630	2890	2838
	1	1.84	23.60	1693	1630	2876	2838
C014	0	1.81	37.14	659	585	1484	1449
	0.5	1.25	34.46	692	734	1462	1444
	1	1.13	34.42	782	734	1459	1444
C015	0	1.97	33.59	1007	837	2396	2325
	0.5	1.58	33.66	907	1125	2318	2310
	1	1.57	33.37	1194	1125	2342	2310
C016	0	2.80	34.77	1494	1331	3071	3034
	0.5	2.27	33.09	1520	1717	3070	3018
	1	2.41	32.74	1841	1717	3038	3018
C017	0	2.79	36.42	917	832	2738	2718
	0.5	2.36	33.20	873	1235	2699	2690
	1	2.39	32.34	1122	1235	2699	2690
C018	0	1.46	40.94	418	364	1462	1462
	0.5	1.20	35.09	394	575	1461	1460
	1	1.25	35.65	559	575	1460	1460
C019	0	64.61	519.78	10430	8205	20288	19433
	0.5	56.08	511.87	11456	9610	20501	19248
	1	53.96	507.05	13432	9610	21101	19248
C020	0	38.80	562.72	6444	4901	13980	13363
	0.5	33.76	534.04	7613	6106	14195	13307
	1	31.97	526.83	9294	6106	14713	13307
C021	0	10.76	166.92	12023	11335	21822	21683
	0.5	10.56	183.88	12132	15570	21741	21528
	1	10.36	157.28	14616	14308	21361	21232
C022	0	6.00	132.89	7478	7151	14100	14009
	0.5	5.83	134.01	7566	9568	14067	13882
	1	5.95	135.71	9229	9316	13815	13714
C023	0	11.34	187.39	12294	11655	22883	22748
	0.5	10.31	204.35	12401	16030	22780	22560
	1	11.17	183.76	15384	14925	22488	22317
C024	0	9.58	121.69	9480	9159	17093	17016
	0.5	8.09	131.93	9809	12011	17036	16864
	1	9.81	127.52	11541	11198	16822	16690
C025	0	6.66	114.40	9873	9329	17720	17571
	0.5	8.84	137.18	10000	11701	17648	17410
	1	7.66	115.62	11429	11244	17426	17305
C026	0	6.48	166.61	6753	6498	12889	12846
	0.5	5.94	175.38	7197	9188	12804	12691
	1	5.78	166.08	8524	8496	12664	12515
C027	0	6.72	177.34	6811	6506	14040	13994
	0.5	5.79	185.84	7196	9737	14013	13819
	1	7.05	163.38	9528	8979	13718	13618
C028	0	5.43	120.68	7552	7335	14821	14780
	0.5	5.32	136.28	7850	10220	14823	14671
	1	5.84	122.66	9488	9428	14568	14482

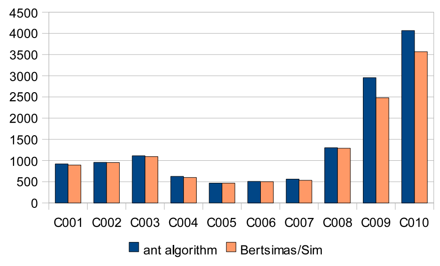
		⏱		best case		worst case	
network	Γ	ants	B&S	ants	B&S	ants	B&S
C029	0	84.52	*)	42505	-	58735	-
	0.5	77.65	*)	46410	-	59408	-
	1	78.91	*)	46917	-	59383	-
C030	0	129.97	*)	55677	-	75780	-
	0.5	126.32	*)	60867	-	76336	-
	1	124.59	*)	62219	-	76718	-

Table 11.6: Numerical tests - uncertain cost problem: Results

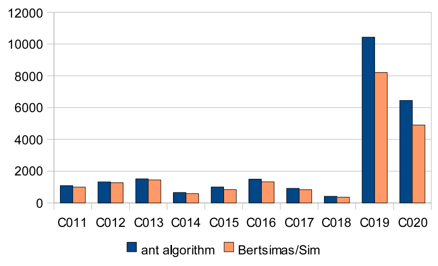
The symbol *) indicates that no solution was found for the networks C029 and C030 for the algorithm of Bertsimas and Sim, as for these problem instances an out-of-memory exception occurred.

First, we examine the results of the ant algorithm and the algorithm of Bertsimas and Sim with regard to the best case and worst case cost.

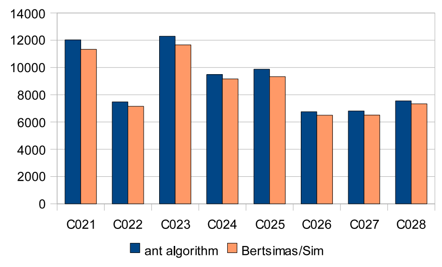
When considering the best case cost, we examine the minimally robust case $\Gamma = 0$. The following figure illustrates the best case cost for the three groups of networks.



(a) 5 layers



(b) 10 layers



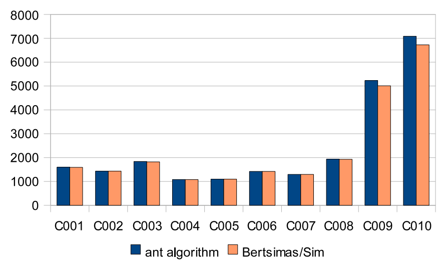
(c) 15 layers

Figure 11.3: Numerical tests - uncertain cost problem: Best case ($\Gamma = 0$)

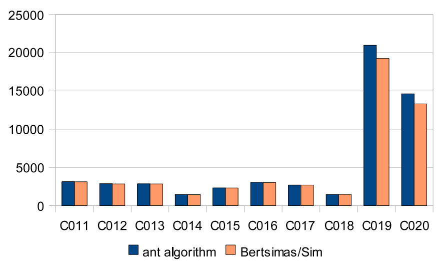
We can see that the best case costs calculated by the algorithm of Bertsimas and Sim are superior to those of the ant algorithm, especially in the multicommodity case. This result meets our expectations, as the ant algorithm is heuristic, while the algorithm of Bertsimas and Sim is an exact method.

On average, the best case cost calculated by the ant algorithm is 6% higher in the single-commodity case and 23% in the multicommodity case. The maximum increase in the best case cost is 15% in the single-commodity case (network C018) and 31% in the multicommodity case (network C020).

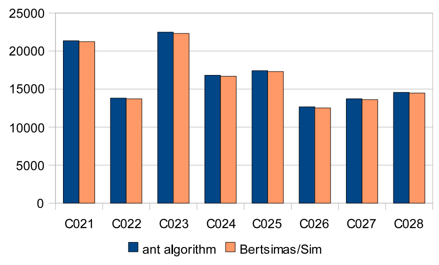
Next, we regard the worst case costs. The worst case costs are examined for the maximally robust case $\Gamma = 1$. The worst case costs are illustrated in the figure below:



(a) 5 layers



(b) 10 layers



(c) 15 layers

Figure 11.4: Numerical tests - uncertain cost problem: Worst case ($\Gamma = 1$)

On average, the worst case cost in the single-commodity case calculated by the ant algorithm is 1% higher than the worst case cost calculated by the algorithm of Bertsimas and Sim. In the multicommodity case, the worst case cost of the ant algorithm is 8% higher, on average.

The maximum increase in the worst case cost is 1% in the single-commodity case (several networks, for example networks C001, C003, C004) and 11% in the multicommodity case (network C020).

We observe that in the single-commodity case, the increase in the best case cost is moderate, while the increase in the worst case cost is very low. In the multicommodity case, however, increases are higher, due to the growing complexity of the problems in the multicommodity case.

Finally, we turn towards the running time comparison. The following figure illustrates the running time of the two algorithms for the test networks C001 to C028.

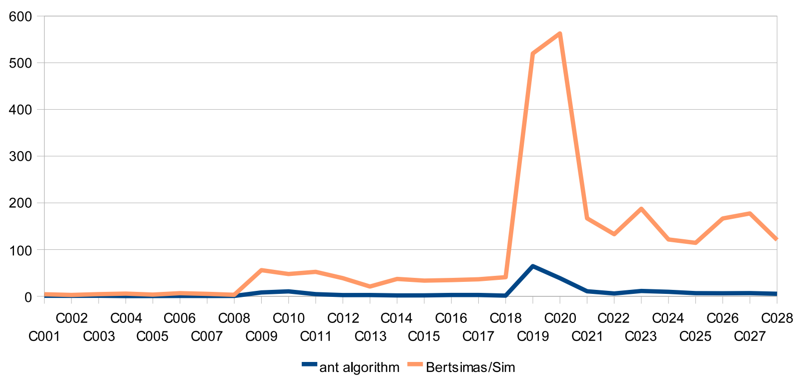


Figure 11.5: Numerical tests - uncertain cost problem: Running time in seconds

We can clearly see how fast the running time of the algorithm of Bertsimas and Sim increases with growing network size and growing number of commodities. The ant algorithm shows a significantly lower running time even for large networks and several commodities.

Note again that the running time for networks C029 and C030 is not stated for the algorithm of Bertsimas and Sim, because of the out-of-memory error. In contrary, the ant algorithm terminated after approximately 82 seconds for network C029 and after approximately 125 seconds for network C030.

To sum up, for smaller problem instances, the algorithm of Bertsimas and Sim seems to be the better choice because of its superiority concerning the solution quality, while for larger problem instances, the ant algorithm provides a reasonable trade-off between computational time and performance.

11.4 Uncertain Demands

In this section, we examine the ant algorithm for network flow problems with uncertain demands as presented in Section 8.2. For each test network, we examine different robustness levels. The aim of the tests is to verify that with an increasing robustness parameter, the solution of the ant algorithm becomes more robust in the sense that more demand units are distributed. Furthermore, we want to evaluate the stability of the ant algorithm, i.e. we want to test whether several runs of the ant algorithm lead to equal results or differ immensely.

For the random generation of layered networks, we have five additional input parameters:

- lower/upper penalty cost bound: For every arc, a random penalty cost value within these bounds is generated
- lower/upper storage cost bound: For every arc, a random storage cost value within these bounds is generated
- additional capacity bound: For every outgoing arc of the nodes in the next to last layer (demand nodes), a random additional capacity value is generated

The additional capacity values on the outgoing arcs of the nodes in the next to last layer, which we consider to be the demand nodes, correspond to the additional demand values for these demand nodes.

We generate ten fully layered networks with uncertain demands, where we consider networks from 5 to 25 node layers with 1 or 3 commodities. The following table lists the information about the generated networks:

network	# layers	# nodes	# arcs	com	nodes p. layer	cost	capacity	extra dem.	penalty cost	storage cost
D001	5	22	104	1	4 – 8	0 – 5	3 – 10	5	10 – 20	1 – 5
D002	5	23	112	3	4 – 8	0 – 5	3 – 10	5	10 – 20	1 – 5
D003	10	68	467	1	5 – 10	0 – 10	3 – 20	5	30 – 50	3 – 5
D004	10	68	458	3	5 – 10	0 – 10	3 – 20	5	30 – 50	3 – 5
D005	15	97	652	1	5 – 10	5 – 20	5 – 20	10	100 – 150	5 – 10
D006	15	114	882	3	5 – 10	5 – 20	5 – 20	10	100 – 150	5 – 10
D007	20	141	1039	1	5 – 10	5 – 20	5 – 20	10	150 – 200	10 – 15
D008	20	147	1121	3	5 – 10	5 – 20	5 – 20	10	150 – 200	10 – 15
D009	25	176	1280	1	5 – 10	10 – 25	10 – 20	12	400 – 500	10 – 20
D010	25	172	1208	3	5 – 10	10 – 25	10 – 20	12	400 – 500	10 – 20

Table 11.7: Random layered networks for the MCFP with uncertain demands

We regard three different robustness levels for each network: $\Gamma = 0$, $\Gamma = 0.5$ and $\Gamma = 1$.

The parameter settings for the ant algorithm are the following:

parameter	setting	parameter	setting
α	1.0	ς^{pher}	2.0
β	1.0	$\varsigma^{storage}_{demand}$	1.0
ρ	0.25	$\varsigma^{penalty}_{demand}$	0.5
η_{min}	0.2	$\varsigma^{storage}_{edge}$	0.1
τ_{min}	1.0	$\varsigma^{penalty}_{edge}$	0.2
τ_{max}	20.0		

Table 11.8: Uncertain demands - ant algorithm: Parameter settings

The settings for the parameters α, β, ρ, η_{min} and τ_{min} are our standard settings for the ant algorithm for uncertain demand values. The limiting value for τ_{max} was chosen dependent on the scale of the demand values. The remaining parameters ς^{pher}, $\varsigma^{storage}_{demand}$, $\varsigma^{penalty}_{demand}$, $\varsigma^{storage}_{edge}$ and $\varsigma^{penalty}_{edge}$ were determined experimentally.

The following tables list the results for each network and each of the 30 runs per network (ten per robustness level) as well as the average of the ten runs per robustness level.

We list the respective robustness parameter, the running time (in seconds), the total sent demand and the sent demand for all demand nodes.

run			total sent	demand node						
no.	Γ	⏱	demand	15	16	17	18	19	20	21
1	0.0	0.63	45	3	3	10	3	10	10	6
2	0.0	0.66	45	3	3	10	3	10	10	6
3	0.0	0.67	45	3	3	10	3	10	10	6
4	0.0	0.64	45	3	3	10	3	10	10	6
5	0.0	0.66	45	3	3	10	3	10	10	6
6	0.0	0.64	45	3	3	10	3	10	10	6
7	0.0	0.61	45	3	3	10	3	10	10	6
8	0.0	0.58	45	3	3	10	3	10	10	6
9	0.0	0.64	45	3	3	10	3	10	10	6
10	0.0	0.66	45	3	3	10	3	10	10	6
average		0.64	45	3	3	10	3	10	10	6
11	0.5	0.75	56	5	4	10	5	14	11	7
12	0.5	0.78	54	5	4	10	3	14	11	7
13	0.5	0.75	51	3	5	10	3	13	10	7
14	0.5	0.77	53	5	4	10	4	14	10	6
15	0.5	0.75	53	5	4	10	4	14	10	6
16	0.5	0.78	54	5	4	11	5	12	10	7
17	0.5	0.77	54	5	4	11	3	14	11	6
18	0.5	0.77	56	5	6	11	3	14	10	7
19	0.5	0.77	57	5	6	11	4	14	11	6
20	0.5	0.70	55	5	6	11	3	14	10	6
average		0.76	54.3	4.8	4.7	10.5	3.7	13.7	10.4	6.5
21	1.0	3.58	60	5	6	12	5	14	11	7
22	1.0	3.64	58	5	5	12	4	14	11	7
23	1.0	3.56	60	5	5	10	8	14	11	7
24	1.0	3.67	60	5	5	12	6	14	11	7
25	1.0	3.53	60	4	6	12	6	14	11	7
26	1.0	3.49	60	5	6	12	5	14	11	7
27	1.0	3.48	60	5	6	12	6	14	11	6
28	1.0	3.61	59	5	6	11	5	14	11	7
29	1.0	3.63	59	4	5	12	7	14	10	7
30	1.0	3.47	60	5	6	12	5	14	11	7
average		3.57	59.6	4.8	5.6	11.7	5.7	14.0	10.9	6.9

Table 11.9: Results for network D001

run				total sent	demand node					
no.	Γ	⏱	com	demand	17	18	19	20	21	22
1	0.0	3.03	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
2	0.0	3.05	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
3	0.0	3.05	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
4	0.0	3.03	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
5	0.0	3.05	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
6	0.0	3.05	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
7	0.0	3.05	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
8	0.0	3.05	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
9	0.0	3.08	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
10	0.0	3.03	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
average		3.05	1	37	4	9	5	9	5	5
			2	36	10	7	7	3	3	6
			3	42	6	10	8	5	7	6
11	0.5	3.81	1	45	8	10	6	9	7	5
			2	39	10	7	7	5	4	6
			3	49	7	11	8	5	10	8
12	0.5	3.72	1	41	7	10	5	9	5	5
			2	41	10	7	7	6	5	6
			3	52	7	13	9	5	10	8
13	0.5	3.61	1	42	8	10	5	9	5	5
			2	40	10	7	8	5	3	7
			3	49	7	10	9	5	10	8
14	0.5	3.7	1	41	8	9	5	9	5	5
			2	40	10	7	7	6	3	7
			3	47	7	11	9	5	7	8
15	0.5	3.77	1	44	8	10	5	9	6	6
			2	40	10	7	7	6	3	7
			3	47	7	10	9	5	8	8
16	0.5	3.78	1	45	8	10	5	10	7	5
			2	40	10	7	7	5	3	8
			3	44	6	10	8	5	7	8

run				total sent	demand node					
no.	Γ	⏱	com	demand	17	18	19	20	21	22
17	0.5	3.81	1	47	8	10	6	10	8	5
			2	37	10	7	7	4	3	6
			3	46	7	10	9	5	9	6
18	0.5	3.94	1	46	8	9	6	12	6	5
			2	41	10	7	7	6	3	8
			3	50	7	11	9	5	10	8
19	0.5	3.75	1	44	8	10	5	9	7	5
			2	43	10	7	8	6	4	8
			3	47	7	10	9	5	8	8
20	0.5	3.73	1	40	6	10	5	9	5	5
			2	40	10	7	7	6	3	7
			3	50	7	11	9	5	10	8
average		3.76	1	43.5	7.7	9.8	5.3	9.5	6.1	5.1
			2	40.1	10.0	7.0	7.2	5.5	3.4	7.0
			3	48.1	6.9	10.7	8.8	5.0	8.9	7.8
21	1.0	5.28	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
22	1.0	5.23	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
23	1.0	5.19	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
24	1.0	5.28	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
25	1.0	5.25	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
26	1.0	5.37	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
27	1.0	5.20	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
28	1.0	5.23	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
29	1.0	5.25	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
30	1.0	5.19	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8
average		5.26	1	51	8	10	6	13	8	6
			2	45	10	7	8	6	6	8
			3	54	7	14	9	5	11	8

Table 11.10: Results for network D002

run			total sent	demand node									
no.	Γ	⏱	demand	58	59	60	61	62	63	64	65	66	67
1	0.0	3.25	117	3	15	6	13	5	16	7	19	17	16
2	0.0	3.63	117	3	15	6	13	5	16	7	19	17	16
3	0.0	3.20	117	3	15	6	13	5	16	7	19	17	16
4	0.0	3.31	117	3	15	6	13	5	16	7	19	17	16
5	0.0	4.00	117	3	15	6	13	5	16	7	19	17	16
6	0.0	3.72	117	3	15	6	13	5	16	7	19	17	16
7	0.0	3.30	117	3	15	6	13	5	16	7	19	17	16
8	0.0	3.28	117	3	15	6	13	5	16	7	19	17	16
9	0.0	3.94	117	3	15	6	13	5	16	7	19	17	16
10	0.0	3.28	117	3	15	6	13	5	16	7	19	17	16
average		3.49	117	3	15	6	13	5	16	7	19	17	16
11	0.5	4.39	124	3	15	7	13	5	17	7	24	17	16
12	0.5	3.67	117	3	15	6	13	5	16	7	19	17	16
13	0.5	4.78	128	4	18	8	15	5	17	8	20	17	16
14	0.5	3.97	129	4	18	6	14	5	16	8	23	17	18
15	0.5	4.39	124	3	15	7	14	5	16	8	22	17	17
16	0.5	4.13	128	5	15	7	15	5	17	8	22	17	17
17	0.5	3.69	126	3	18	7	15	5	16	8	21	17	16
18	0.5	4.52	124	3	15	7	15	5	16	7	21	17	18
19	0.5	3.67	124	3	15	7	15	5	17	7	21	17	17
20	0.5	3.69	124	4	16	8	13	5	17	7	19	17	18
average		4.09	124.8	3.5	16.0	7.0	14.2	5.0	16.5	7.5	21.2	17.0	16.9
21	1.0	6.08	136	6	18	8	15	5	17	8	24	17	18
22	1.0	6.08	136	6	18	8	15	5	17	8	24	17	18
23	1.0	5.89	136	6	18	8	15	5	17	8	24	17	18
24	1.0	5.08	136	6	18	8	15	5	17	8	24	17	18
25	1.0	5.92	136	6	18	8	15	5	17	8	24	17	18
26	1.0	5.03	136	6	18	8	15	5	17	8	24	17	18
27	1.0	5.97	136	6	18	8	15	5	17	8	24	17	18
28	1.0	5.14	136	6	18	8	15	5	17	8	24	17	18
29	1.0	6.06	136	6	18	8	15	5	17	8	24	17	18
30	1.0	5.08	136	6	18	8	15	5	17	8	24	17	18
average		5.63	136	6	18	8	15	5	17	8	24	17	18

Table 11.11: Results for network D003

run				total sent	demand node									
no.	Γ	⏱	com	demand	58	59	60	61	62	63	64	65	66	67
1	0.0	37.18	1	113	11	17	8	12	7	9	14	9	7	19
			2	97	8	14	7	6	17	11	9	6	10	9
			3	99	9	5	19	11	13	3	12	5	19	3
2	0.0	36.42	1	113	11	17	8	12	7	9	14	9	7	19
			2	97	8	14	7	6	17	11	9	6	10	9
			3	99	9	5	19	11	13	3	12	5	19	3
3	0.0	36.38	1	113	11	17	8	12	7	9	14	9	7	19
			2	97	8	14	7	6	17	11	9	6	10	9
			3	99	9	5	19	11	13	3	12	5	19	3
4	0.0	36.38	1	113	11	17	8	12	7	9	14	9	7	19
			2	97	8	14	7	6	17	11	9	6	10	9
			3	99	9	5	19	11	13	3	12	5	19	3
5	0.0	36.48	1	113	11	17	8	12	7	9	14	9	7	19
			2	97	8	14	7	6	17	11	9	6	10	9
			3	99	9	5	19	11	13	3	12	5	19	3
6	0.0	36.57	1	113	11	17	8	12	7	9	14	9	7	19
			2	97	8	14	7	6	17	11	9	6	10	9
			3	99	9	5	19	11	13	3	12	5	19	3
7	0.0	37.32	1	113	11	17	8	12	7	9	14	9	7	19
			2	97	8	14	7	6	17	11	9	6	10	9
			3	99	9	5	19	11	13	3	12	5	19	3
8	0.0	36.03	1	113	11	17	8	12	7	9	14	9	7	19
			2	97	8	14	7	6	17	11	9	6	10	9
			3	99	9	5	19	11	13	3	12	5	19	3
9	0.0	36.58	1	113	11	17	8	12	7	9	14	9	7	19
	0.0		2	97	8	14	7	6	17	11	9	6	10	9
	0.0		3	99	9	5	19	11	13	3	12	5	19	3
10	0.0	36.47	1	113	11	17	8	12	7	9	14	9	7	19
	0.0	36.47	2	97	8	14	7	6	17	11	9	6	10	9
	0.0	36.47	3	99	9	5	19	11	13	3	12	5	19	3
average		36.59	1	113	11	17	8	12	7	9	14	9	7	19
			2	97	8	14	7	6	17	11	9	6	10	9
			3	99	9	5	19	11	13	3	12	5	19	3
11	0.5	41.71	1	123	12	17	13	13	7	10	14	9	8	20
			2	112	9	14	8	9	17	11	14	7	11	12
			3	110	9	7	19	11	13	7	15	6	20	3
12	0.5	41.35	1	127	13	18	13	13	7	11	14	10	8	20
			2	107	9	14	7	9	17	13	10	6	13	9
			3	106	9	6	19	11	13	7	12	7	19	3
13	0.5	42.74	1	127	13	17	13	13	7	11	14	11	8	20
			2	106	9	14	7	9	17	11	13	6	11	9
			3	107	9	7	19	11	13	6	15	5	19	3
14	0.5	41.73	1	121	11	17	11	12	7	11	14	11	7	20
			2	110	9	14	7	9	17	12	14	6	13	9
			3	110	9	7	20	11	13	7	15	5	20	3
15	0.5	42.04	1	125	13	17	11	13	7	11	14	10	9	20
			2	106	9	14	7	9	17	11	11	6	13	9
			3	107	9	6	20	11	13	5	15	6	19	3
16	0.5	41.94	1	125	12	17	12	12	7	11	15	11	8	20
			2	118	9	15	9	9	17	12	14	7	13	13
			3	105	9	6	19	11	13	5	14	6	19	3

run				total sent	demand node									
no.	Γ	⏱	com	demand	58	59	60	61	62	63	64	65	66	67
17	0.5	42.06	1	125	13	17	12	12	7	11	14	11	8	20
			2	109	9	14	7	9	17	13	10	6	13	11
			3	109	10	7	20	11	13	7	14	5	19	3
18	0.5	42.05	1	126	13	17	13	12	7	11	14	9	10	20
			2	113	9	16	8	9	17	11	14	6	13	10
			3	112	10	7	19	11	13	7	15	7	20	3
19	0.5	40.87	1	122	13	17	11	12	7	9	15	11	8	19
			2	112	9	14	9	9	17	15	10	6	13	10
			3	104	9	7	19	11	13	3	13	6	20	3
20	0.5	42.01	1	129	13	18	13	13	7	11	15	10	9	20
			2	110	9	15	8	9	17	11	13	6	13	9
			3	108	9	6	19	11	13	7	14	7	19	3
average		41.85	1	125	12.6	17.2	12.2	12.5	7.0	10.7	14.3	10.3	8.3	19.9
			2	110.3	9.0	14.4	7.7	9.0	17.0	12.0	12.3	6.2	12.6	10.1
			3	107.8	9.2	6.6	19.3	11.0	13.0	6.1	14.2	6.0	19.4	3.0
21	1.0	55.68	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
22	1.0	55.78	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
23	1.0	55.83	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
24	1.0	55.84	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
25	1.0	56.26	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
26	1.0	57.97	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
27	1.0	55.53	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
28	1.0	56.91	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
29	1.0	58.30	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
30	1.0	56.92	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3
average		56.23	1	138	13	22	13	13	7	11	16	11	12	20
			2	124	9	17	9	9	17	16	14	7	13	13
			3	115	11	7	21	11	13	7	15	7	20	3

Table 11.12: Results for network D004

run			total sent	demand node				
no.	Γ	⏱	demand	92	93	94	95	96
1	0.0	2.56	82	17	17	18	12	18
2	0.0	2.66	82	17	17	18	12	18
3	0.0	2.63	82	17	17	18	12	18
4	0.0	2.78	82	17	17	18	12	18
5	0.0	2.73	82	17	17	18	12	18
6	0.0	2.73	82	17	17	18	12	18
7	0.0	2.70	82	17	17	18	12	18
8	0.0	2.64	82	17	17	18	12	18
9	0.0	2.70	82	17	17	18	12	18
10	0.0	2.64	82	17	17	18	12	18
average		2.68	82	17	17	18	12	18
11	0.5	3.00	94	20	20	21	15	18
12	0.5	3.28	93	23	18	19	15	18
13	0.5	3.11	93	19	19	21	16	18
14	0.5	3.09	96	21	20	20	17	18
15	0.5	3.47	96	23	20	19	16	18
16	0.5	3.27	98	20	21	22	17	18
17	0.5	3.41	98	23	20	19	18	18
18	0.5	3.41	94	21	21	20	14	18
19	0.5	3.25	98	22	21	22	15	18
20	0.5	3.09	93	19	21	21	14	18
average		3.24	95.3	21.1	20.1	20.4	15.7	18.0
21	1.0	4.55	106	24	21	22	21	18
22	1.0	4.14	106	24	21	22	21	18
23	1.0	4.17	106	24	21	22	21	18
24	1.0	4.17	106	24	21	22	21	18
25	1.0	4.22	106	24	21	22	21	18
26	1.0	4.19	105	24	21	22	20	18
27	1.0	4.17	106	24	21	22	21	18
28	1.0	4.22	106	24	21	22	21	18
29	1.0	4.20	105	23	21	22	21	18
30	1.0	4.50	106	24	21	22	21	18
average		4.25	105.8	23.9	21	22	20.9	18

Table 11.13: Results for network D005

run				total sent	demand node									
no.	Γ	⏱	com	demand	104	105	106	107	108	109	110	111	112	113
1	0.0	48.81	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
2	0.0	48.23	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
3	0.0	50.33	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
4	0.0	49.14	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
5	0.0	49.36	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
6	0.0	49.52	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
7	0.0	49.31	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
8	0.0	49.22	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
9	0.0	48.07	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
10	0.0	48.82	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
average		49.11	1	112	13	8	17	9	12	10	10	11	15	7
			2	121	10	7	18	13	5	16	9	14	19	10
			3	110	12	11	13	6	11	7	11	16	10	13
11	0.5	69.26	1	142	16	8	18	10	13	15	12	15	23	12
			2	135	12	8	19	13	6	17	10	15	19	16
			3	140	15	12	15	11	15	12	12	19	14	15
12	0.5	67.67	1	135	13	8	18	15	13	14	13	16	17	8
			2	136	12	8	19	17	6	17	10	17	19	11
			3	134	15	15	13	11	13	10	12	19	11	15
13	0.5	68.61	1	132	16	8	18	11	13	13	10	14	21	8
			2	136	12	8	19	16	5	17	9	19	19	12
			3	142	16	12	16	11	14	10	12	19	18	14
14	0.5	68.46	1	141	15	8	18	12	13	13	15	16	21	10
			2	133	12	8	18	14	5	17	10	15	19	15
			3	143	15	14	16	13	17	10	12	18	13	15
15	0.5	68.11	1	141	15	8	18	11	13	15	14	15	21	11
			2	136	12	9	19	18	6	16	10	14	19	13
			3	142	16	16	16	8	16	8	12	19	16	15
16	0.5	68.37	1	135	15	8	18	10	13	13	14	16	19	9
			2	133	11	8	19	14	6	17	10	17	19	12
			3	136	12	12	16	10	17	9	12	18	15	15

run				total sent	demand node									
no.	Γ	⏱	com	demand	104	105	106	107	108	109	110	111	112	113
17	0.5	68.79	1	138	16	8	18	11	13	15	13	16	19	9
			2	136	12	9	19	17	6	17	10	17	19	10
			3	139	15	13	16	6	14	15	12	19	14	15
18	0.5	68.22	1	138	16	8	18	15	13	13	12	13	19	11
			2	138	12	9	19	18	6	17	10	18	19	10
			3	141	15	15	15	10	16	8	12	19	16	15
19	0.5	68.62	1	136	15	8	18	12	13	12	15	16	20	7
			2	135	12	8	19	17	6	17	10	14	19	13
			3	142	17	12	16	10	15	12	12	19	14	15
20	0.5	65.00	1	136	16	8	18	12	13	12	13	15	19	10
			2	140	12	9	19	16	6	17	10	19	19	13
			3	140	18	14	16	9	16	7	12	19	14	15
average		68.11	1	137.4	15.3	8.0	18.0	11.9	13.0	13.5	13.1	15.2	19.9	9.5
			2	135.8	11.9	8.4	18.9	16.0	5.8	16.9	9.9	16.5	19.0	12.5
			3	139.9	15.4	13.5	15.5	9.9	15.3	10.1	12.0	18.8	14.5	14.9
21	1.0	107.46	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
22	1.0	107.84	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
23	1.0	107.06	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
24	1.0	106.89	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
25	1.0	105.53	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
26	1.0	105.43	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
27	1.0	107.64	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
28	1.0	108.53	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
29	1.0	105.81	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
30	1.0	105.23	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15
average		106.74	1	161	16	8	18	18	13	15	18	17	24	14
			2	154	12	9	19	23	6	17	10	19	19	20
			3	170	19	18	16	16	20	16	12	19	19	15

Table 11.14: Results for network D006

run			total sent	demand node				
no.	Γ	⏱	demand	136	137	138	139	140
1	0.0	3.16	58	13	19	10	8	8
2	0.0	2.95	58	13	19	10	8	8
3	0.0	3.06	58	13	19	10	8	8
4	0.0	3.05	58	13	19	10	8	8
5	0.0	3.06	58	13	19	10	8	8
6	0.0	3.08	58	13	19	10	8	8
7	0.0	3.03	58	13	19	10	8	8
8	0.0	3.06	58	13	19	10	8	8
9	0.0	3.00	58	13	19	10	8	8
10	0.0	3.08	58	13	19	10	8	8
average		3.05	58	13	19	10	8	8
11	0.5	3.88	66	13	21	10	11	11
12	0.5	3.88	65	13	22	10	11	9
13	0.5	3.81	66	13	21	10	11	11
14	0.5	3.92	67	13	24	10	10	10
15	0.5	3.77	65	13	22	10	10	10
16	0.5	3.83	65	13	22	10	9	11
17	0.5	3.75	64	13	22	10	10	9
18	0.5	3.91	63	13	24	10	8	8
19	0.5	3.83	65	13	24	10	9	9
20	0.5	3.84	66	13	23	10	8	12
average		3.84	65.2	13.0	22.5	10.0	9.7	10.0
21	1.0	5.02	73	13	24	10	11	15
22	1.0	5.14	73	13	24	10	11	15
23	1.0	5.08	73	13	24	10	11	15
24	1.0	5.06	73	13	24	10	11	15
25	1.0	5.16	73	13	24	10	11	15
26	1.0	5.11	73	13	24	10	11	15
27	1.0	5.08	73	13	24	10	11	15
28	1.0	5.09	73	13	24	10	11	15
29	1.0	5.20	73	13	24	10	11	15
30	1.0	5.03	73	13	24	10	11	15
average		5.1	73	13	24	10	11	15

Table 11.15: Results for network D007

run				total sent	demand node						
no.	Γ	⏱	com	demand	140	141	142	143	144	145	146
1	0.0	50.72	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
2	0.0	50.40	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
3	0.0	50.40	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
4	0.0	50.44	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
5	0.0	51.15	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
6	0.0	51.08	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
7	0.0	50.65	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
8	0.0	50.70	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
9	0.0	51.08	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
10	0.0	51.00	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
average		50.74	1	96	15	16	15	16	16	13	5
			2	102	11	17	19	17	19	9	10
			3	72	6	6	13	20	8	14	5
11	0.5	68.20	1	116	19	20	16	21	18	16	6
			2	123	14	18	22	22	20	14	13
			3	87	8	8	13	23	11	17	7
12	0.5	68.2	1	120	22	20	16	21	19	16	6
			2	124	14	20	22	20	20	13	15
			3	88	8	9	15	23	11	15	7
13	0.5	68.45	1	118	19	18	16	23	19	17	6
			2	124	12	19	25	21	20	12	15
			3	90	8	7	17	24	11	16	7
14	0.5	68.17	1	113	17	19	16	21	18	16	6
			2	124	14	20	25	20	20	12	13
			3	89	7	9	14	24	11	17	7
15	0.5	67.97	1	111	19	18	16	18	18	16	6
			2	124	14	21	23	21	20	13	12
			3	87	6	7	15	24	10	16	9
16	0.5	68.37	1	113	19	20	16	17	19	16	6
			2	123	14	22	22	21	20	12	12
			3	90	7	9	18	24	10	17	5

run				total sent	demand node						
no.	Γ	⏱	com	demand	140	141	142	143	144	145	146
17	0.5	67.37	1	119	16	20	16	21	19	21	6
			2	122	14	19	25	20	20	12	12
			3	85	8	8	16	21	10	17	5
18	0.5	68.26	1	113	17	20	16	23	16	15	6
			2	124	14	20	21	20	20	16	13
			3	88	8	9	18	22	8	17	6
19	0.5	69.42	1	117	19	18	16	21	18	19	6
			2	126	14	22	19	23	20	17	11
			3	91	8	8	17	24	11	17	6
20	0.5	68.18	1	113	16	20	16	20	19	16	6
			2	120	14	20	23	22	20	9	12
			3	88	8	6	17	23	9	17	8
average		68.26	1	115.3	18.3	19.3	16.0	20.6	18.3	16.8	6.0
			2	123.4	13.8	20.1	22.7	21.0	20.0	13.0	12.8
			3	88.3	7.6	8.0	16.0	23.2	10.2	16.6	6.7
21	1.0	106.45	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
22	1.0	104.95	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
23	1.0	105.81	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
24	1.0	103.42	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
25	1.0	104.36	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
26	1.0	105.18	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
27	1.0	104.95	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
28	1.0	106.72	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
29	1.0	103.89	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
30	1.0	103.98	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13
average		105.08	1	133	25	20	16	25	19	22	6
			2	143	14	22	25	26	20	18	18
			3	105	8	9	23	24	11	17	13

Table 11.16: Results for network D008

run			total sent	demand node				
no.	Γ	⏱	demand	171	172	173	174	175
1	0.0	5.28	79	19	15	20	10	15
2	0.0	5.33	79	19	15	20	10	15
3	0.0	5.17	79	19	15	20	10	15
4	0.0	5.17	79	19	15	20	10	15
5	0.0	5.31	79	19	15	20	10	15
6	0.0	4.83	79	19	15	20	10	15
7	0.0	5.28	79	19	15	20	10	15
8	0.0	5.28	79	19	15	20	10	15
9	0.0	5.27	79	19	15	20	10	15
10	0.0	5.25	79	19	15	20	10	15
average		5.22	79	19	15	20	10	15
11	0.5	6.38	87	20	15	25	12	15
12	0.5	6.42	91	24	15	24	12	16
13	0.5	6.34	87	24	15	20	12	16
14	0.5	6.27	87	20	15	24	12	16
15	0.5	6.42	86	24	15	21	10	16
16	0.5	6.44	88	25	15	21	11	16
17	0.5	6.52	85	22	15	22	11	15
18	0.5	6.44	87	21	15	24	12	15
19	0.5	6.11	91	26	15	23	11	16
20	0.5	6.44	88	21	15	24	12	16
average		6.38	87.7	22.7	15.0	22.8	11.5	15.7
21	1.0	8.09	97	26	15	28	12	16
22	1.0	8.16	97	26	15	28	12	16
23	1.0	8.16	97	26	15	28	12	16
24	1.0	7.91	97	26	15	28	12	16
25	1.0	8.08	97	26	15	28	12	16
26	1.0	8.12	97	26	15	28	12	16
27	1.0	8.12	97	26	15	28	12	16
28	1.0	8.05	97	26	15	28	12	16
29	1.0	8.14	97	26	15	28	12	16
30	1.0	8.17	96	25	15	28	12	16
average		8.10	96.9	25.9	15	28	12	16

Table 11.17: Results for network D009

run				total sent	demand node						
no.	Γ	⏱	com	demand	165	166	167	168	169	170	171
1	0.0	64.73	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
2	0.0	64.00	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
3	0.0	64.64	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
4	0.0	60.83	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
5	0.0	63.83	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
6	0.0	66.81	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
7	0.0	61.26	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
8	0.0	64.79	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
9	0.0	64.50	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
10	0.0	64.36	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
average		63.93	1	104	19	11	15	11	13	17	18
			2	120	14	18	18	19	17	19	15
			3	96	15	16	15	12	16	10	12
11	0.5	84.09	1	121	24	15	15	15	15	17	20
			2	132	15	20	18	24	18	19	18
			3	113	16	19	18	14	18	12	16
12	0.5	79.55	1	124	23	13	15	14	18	17	24
			2	131	15	20	18	20	18	20	20
			3	114	17	19	16	14	18	16	14
13	0.5	83.45	1	123	22	15	15	15	17	17	22
			2	131	15	19	18	25	18	21	15
			3	113	17	18	21	15	18	10	14
14	0.5	83.29	1	122	22	15	15	16	16	17	21
			2	132	15	19	18	21	18	21	20
			3	110	16	17	17	13	18	13	16
15	0.5	81.37	1	120	22	12	15	12	20	17	22
			2	132	15	19	18	21	18	21	20
			3	108	17	17	16	13	18	14	13
16	0.5	79.42	1	123	20	15	15	18	16	17	22
			2	129	15	18	18	23	18	21	16
			3	114	15	20	21	16	18	10	14

run				total sent	demand node						
no.	Γ	⏱	com	demand	165	166	167	168	169	170	171
17	0.5	80.01	1	127	27	15	15	15	14	17	24
			2	130	15	20	18	22	18	20	17
			3	111	17	19	16	13	18	12	16
18	0.5	80.59	1	123	20	13	15	16	18	17	24
			2	133	15	20	18	25	18	21	16
			3	114	17	18	20	17	18	10	14
19	0.5	79.59	1	120	21	15	15	17	15	17	20
			2	133	15	19	18	21	18	25	17
			3	118	17	19	20	17	18	11	16
20	0.5	82.14	1	125	24	14	15	18	16	17	21
			2	128	15	19	18	20	18	21	17
			3	111	16	20	18	15	16	12	14
average		81.35	1	122.8	22.5	14.2	15.0	15.6	16.5	17.0	22.0
			2	131.1	15.0	19.3	18.0	22.2	18.0	21.0	17.6
			3	112.6	16.5	18.6	18.3	14.7	17.8	12.0	14.7
21	1.0	119.45	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
22	1.0	114.03	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
23	1.0	114.89	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
24	1.0	115.30	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
25	1.0	119.25	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
26	1.0	113.63	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
27	1.0	119.34	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
28	1.0	117.78	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
29	1.0	121.31	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
30	1.0	115.43	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20
average		117.22	1	140	28	15	15	20	20	17	25
			2	144	15	20	18	25	18	28	20
			3	131	17	20	23	17	18	16	20

Table 11.18: Results for network D010

For all problem instances, the amount of delivered demand units increases with an increasing value of Γ. Hence the objective, that an increasing value of Γ leads to a solution which is more robust is fulfilled.

We can observe that the ant algorithm runs very stable, especially for $\Gamma = 0$ and $\Gamma = 1.0$. This means that when comparing the 10 runs for one problem instance and a fixed value of Γ, the different runs do not differ notably. For $\Gamma = 0$ and $\Gamma = 1.0$, we often experienced 10 identical solutions for the demand delivery. For $\Gamma = 0.5$, differences between the runs occur, but they are still moderate. These observations hold for both the single-commodity and the multicommodity problem instances.

For all problem instances, we can see that the computational time increases with increasing value of Γ. For larger values of Γ, the generated demand values at the demand nodes are larger in general. Therefore, more ants will choose their way through the original network instead of using the additional arcs which directly lead to the sink node. Having fixed capacities on the arcs, it then becomes more difficult for the ants to find a cheap free path to the sink node and therefore the probability of dead lock situations, which lead to a restart of all ants, increases. This leads to larger computational times.

For the largest considered problem instance, network D010 (25 layers, 172 nodes, 1208 arcs, 3 commodities), the average computational time is 117 seconds (case $\Gamma = 1.0$). In view of the problem size, which corresponds to realistic problem sizes in practice, the running time is rather low. This indicates that real-world problem instances are computationally tractable with the ant algorithm.

11.5 Summary and Conclusions

In this chapter, the heuristic methods that were developed in this thesis were evaluated by means of randomly generated fully layered networks.

In Section 11.2, computation time and performance of the SOS 2 Branch and Bound method, see Section 5.3, the ant algorithm for the threshold minimum cost flow problem (TMCFP), as proposed in Section 5.1 and the Cost Scaling Algorithm were compared by means of several TMCFPs.

While the SOS 2 B&B method expectedly has the best performance concerning total cost and threshold violations, its computation time increases immensely with growing problem sizes. The ant algorithm significantly reduces the number of threshold violations, compared to the CSA solutions, and has a faster running times than the SOS 2 B&B method. However, a significant increase in the total cost, compared to the solutions of the CSA, is to be observed.

Therefore, for MCFPs with additional threshold claims, for large problem instances, the advantages and disadvantages of cheap solutions violating the threshold claims and more expensive solutions that violate the threshold claims on significantly less arcs, must be weighted in order to choose either the CSA or the ant algorithm. For small problem instances, we recommend the exact SOS 2 B&B method.

In Section 11.3, computation time and performance of the ant algorithm for Gaussian distributed costs, as presented in Subsection 6.3.2, and the results of the multicommodity extension of the algorithm of Bertsimas & Sim for minimum cost flow problems with uncertain costs, as presented in Section 6.2, were compared by means of several MCFPs with uncertain costs.

As it was to be expected, the exact algorithm of Bertsimas and Sim is superior in solution quality, i.e. the total costs (best case and worst case) of the Bertsimas and Sim solutions are lower than those of the ant solutions. Concerning the running time, the ant algorithm is significantly faster for large problem instances and for multicommodity problems. Furthermore, the trade-off between computational time and performance is reasonable.

In Section 11.4, the ant algorithm for NFPs with uncertain demands, as presented in Section 8.2, was examined, where solution robustness, stability and computational time were in the focus of the test runs.

For all considered problem instances, the ant solutions became more robust with an increasing value of Γ.

Concerning stability, we observed that the ant algorithm runs very stable for extreme values of Γ, i.e. $\Gamma = 0$ and $\Gamma = 1$ and stable with moderate fluctuations for an intermediate value of Γ, i.e. $\Gamma = 0.5$. These observations hold for both the single-commodity and the multicommodity case.

For both single-commodity and multicommodity problem instances, the computational time of the ant algorithm was rather low. Therefore the algorithm might be suitable even for problem instances in real-world size.

Chapter 12

Computational Results: Reference Model

According to [93], a reference model is a model which represents a class of issues and serves as a point of reference for the development of specific models.

In order to construct a reference model for a supply chain network, we focus on the production and transportation stages of a supply chain. We consider a real-world supply chain network with uncertain demands. The aim is to compare the results of different presented optimization approaches for this problem. We refer to this model as “reference model”.

The reference model is based on a real-world network structure and real-world input data given by the industry partner Henkel KGaA.

In the first stage of the supply chain, the resources are mixed to produce different products in eight possible mixing lines in six different production sites. After mixing, an interexchange of the mixed products between the different lines and sites is possible, but not mandatory.

In the next stage, the mixed products are filled into their final containers (bottles, cans, cartons, etc.) in the filling lines of the same six production sites.

Both the filling and mixing lines have a limited capacity which is dependent on the respective commodity.

After filling, the final products are transported to their next destination, to one out of 15 warehouses. At the warehouses, the products are ready for distribution to the customers or for storage.

Several hundred commodities are grouped to four major commodity groups.

Each commodity group can be represented by a characteristic commodity for this group, therefore we focus on four commodities in the following.

First we give all input data, i.e. the network structure and the costs and capacities, for the reference model. Afterwards, we apply different optimization approaches, which we presented in previous chapters, to the reference model.

The following figure illustrates the structure of the underlying network.

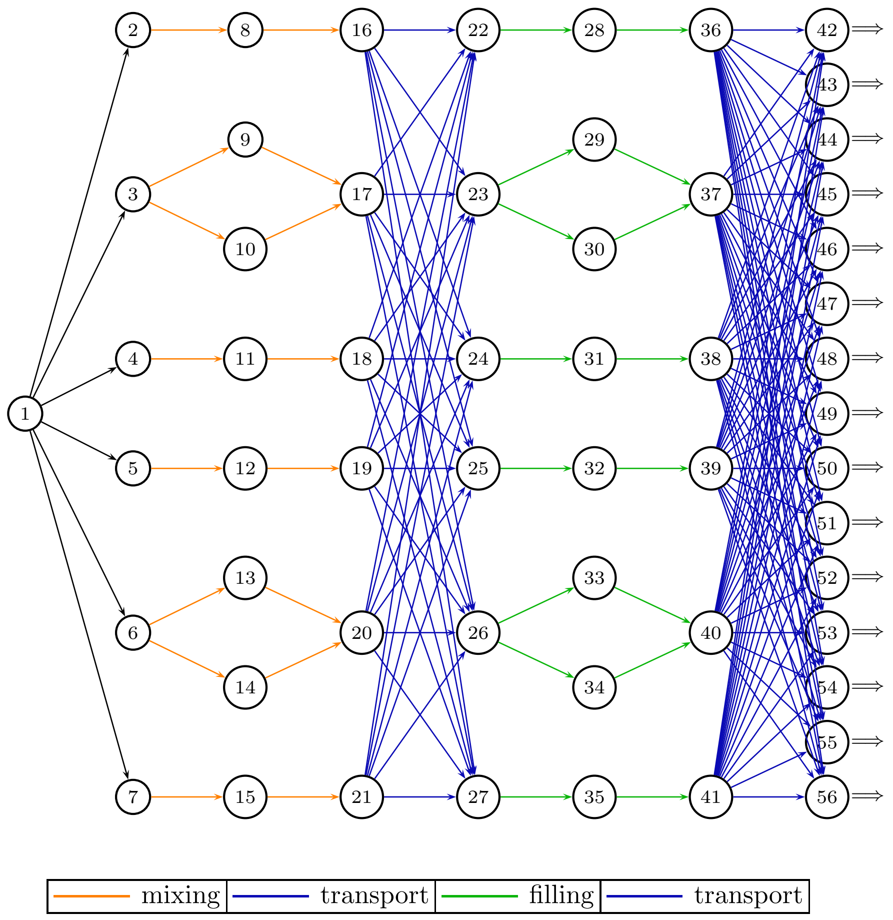


Figure 12.1: Reference model: Multicommodity minimum cost flow problem

The different layers of the network correspond to the stages of the supply chain planning: Mixing, transportation, filling and again transportation of the products. Nodes 8 to 15 correspond to the mixing lines of six different production plants. Nodes 28 to 35 correspond to the filling lines. There is a possibility of transportation between the mixing and the filling process. After the filling, the products are transported to different warehouses, nodes 42 to 56.

We have costs for mixing, filling and transportation. The warehouses have a certain basic demand of the products as well as a possible extra demand. The specific input data is given in the following tables.

arc / commodity	(2, 8)	(3, 9)	(3, 10)	(4, 11)	(5, 12)	(6, 13)	(6, 14)	(7, 15)
1	2340	0	0	1121	897	1872	1872	897
2	1276	0	0	2226	1781	1781	1781	1021
3	0	2197	2197	0	0	977	1757	977
4	0	844	844	0	0	584	584	675

(a) Capacities on "mix" arcs

arc / commodity	(22, 28)	(23, 29)	(23, 30)	(24, 31)	(25, 32)	(26, 33)	(26, 34)	(27, 35)
1	1959	0	0	909	782	1522	1749	948
2	949	0	0	216	1634	1691	1684	872
3	0	1878	1655	0	0	845	1542	956
4	0	781	664	0	0	516	533	753

(b) Capacities on "fill" arcs

Table 12.1: Reference model: Arc capacities

The arc capacity for all not mentioned arcs is infinity.

(2, 8)	(3, 9)	(3, 10)	(4, 11)	(5, 12)	(6, 13)	(6, 14)	(7, 15)
3276	3076	3076	3116	2493	2621	2621	1429

(a) Bundle capacities on "mix" arcs

(22, 28)	(23, 29)	(23, 30)	(24, 31)	(25, 32)	(26, 33)	(26, 34)	(27, 35)
2743	2629	2317	1273	2288	2367	2449	1338

(b) Bundle capacities on "fill" arcs

Table 12.2: Reference model: Bundle capacities

The bundle capacity for all not mentioned arcs is infinity.

arc / commodity	(2, 8)	(3, 9)	(3, 10)	(4, 11)	(5, 12)	(6, 13)	(6, 14)	(7, 15)
1	2	100000	100000	1	2	2	1	2
2	1	100000	100000	1	1	2	2	1
3	100000	1	1	100000	100000	1	1	2
4	100000	1	2	100000	100000	1	1	1

(a) Costs on “mix” arcs

arc / commodity	(22, 28)	(23, 29)	(23, 30)	(24, 31)	(25, 32)	(26, 33)	(26, 34)	(27, 35)
1	9	100000	100000	5	12	8	5	5
2	10	100000	100000	8	12	10	5	12
3	100000	7	6	100000	100000	11	10	6
4	100000	4	7	100000	100000	5	5	3

(b) Costs on “fill” arcs

node / node	22	23	24	25	26	27
16	0	0	9	36	19	34
17	0	0	9	36	19	34
18	9	9	0	34	18	22
19	36	36	34	0	3	22
20	19	19	18	3	0	23
21	34	34	22	22	23	0

(c) Costs on “transportation” arcs for commodities 1 and 2

node / node	22	23	24	25	26	27
16	0	0	5	17	9	16
17	0	0	5	17	9	16
18	5	5	0	16	9	10
19	17	17	16	0	2	10
20	9	9	9	2	0	11
21	16	16	10	10	11	0

(d) Costs on “transportation” arcs for commodities 3 and 4

node / node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
36	3	11	14	10	34	12	4	5	12	22	36	35	43	36	5
37	10	36	6	9	34	22	34	10	11	22	36	3	36	34	5
38	7	19	54	9	11	12	18	40	18	22	25	16	66	25	31
39	10	34	3	11	14	19	22	19	18	23	24	34	36	23	28
40	9	34	10	36	6	16	22	22	17	20	24	34	42	36	34
41	9	11	7	19	54	6	7	12	3	53	35	42	34	23	24

(e) Costs on “transportation” arcs for commodities 1 and 2

node / node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
36	2	5	7	5	16	6	2	2	6	10	17	17	21	17	3
37	5	17	3	5	16	11	16	5	5	10	17	2	17	16	3
38	4	9	26	4	5	6	9	19	9	11	12	8	32	12	15
39	5	16	2	5	7	9	10	9	9	11	12	16	17	11	13
40	5	16	5	17	3	8	11	10	8	9	11	17	20	17	16
41	4	5	4	9	26	3	4	6	2	26	17	20	16	11	12

(f) Costs on “transportation” arcs for commodities 3 and 4

Table 12.3: Reference model: Arc costs

The arc cost for all not mentioned arcs is zero.

Note that some mix and fill arcs have a cost value of 100000 for certain commodities. As it can be seen in Table 12.1, these arcs have a capacity of 0 for the respective commodities and are therefore not feasible for these commodities.

commodity \ node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
1	103	101	104	103	104	78	77	107	115	97	138	103	92	89	90
2	80	83	77	78	81	67	62	88	89	75	119	86	74	72	74
3	80	79	75	73	79	58	58	87	85	73	107	77	74	64	66
4	16	17	17	16	17	13	13	19	19	17	24	17	16	14	15

(a) Nominal demand

commodity \ node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
1	30	30	31	30	31	23	23	32	34	29	41	31	27	26	26
2	24	24	23	23	24	20	18	26	26	22	35	25	22	21	22
3	23	23	22	22	23	17	17	26	25	21	32	23	22	19	19
4	4	5	5	5	5	3	4	5	5	5	7	5	4	4	4

(b) Extra demand

Table 12.4: Reference model: Demands

The demand for all not mentioned nodes is zero. The maximum demand per demand node (warehouse) and commodity is the sum of nominal and extra demand.

For all demand nodes, we have storage and penalty costs.

commodity \ node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
1	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
2	49	49	49	49	49	49	49	49	49	49	49	49	49	49	49
3	57	57	57	57	57	57	57	57	57	57	57	57	57	57	57
4	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82

(a) Penalty costs

commodity \ node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
2	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
4	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6

(b) Storage costs

Table 12.5: Reference model: Penalty and storage costs

In the following sections, we apply the ant algorithm for uncertain demands, the transformation algorithm of uncertain demands to uncertain costs, the recoverable robustness algorithm and the multicommodity L-shaped algorithm to the reference model.

12.1 Ant Algorithm for Uncertain Demands

In order to apply the ant algorithm for uncertain demands, we extend the network to a single-source-single-sink network as described in Section 8.2.1.

We use the following parameter settings:

parameter	setting
α	3.0
β	1.0
ρ	0.5
η_{min}	0.2
τ_{min}	1.0
τ_{max}	110

parameter	setting
ς^{pher}	2.0
$\varsigma^{storage}_{demand}$	1.0
$\varsigma^{penalty}_{demand}$	0.5
$\varsigma^{storage}_{edge}$	0.1
$\varsigma^{penalty}_{edge}$	0.2

Table 12.6: Reference model - ant algorithm: Parameter settings

The settings for the parameters α, β, ρ, η_{min} and τ_{min} are our standard settings for the ant algorithm for uncertain demand values, cf. Section 11.4. The upper pheromone bound τ_{max} was chosen experimentally dependent on the scale of the demand values. The settings for the parameters ς^{pher}, $\varsigma^{storage}_{demand}$, $\varsigma^{penalty}_{demand}$, $\varsigma^{storage}_{edge}$ and $\varsigma^{penalty}_{edge}$ were taken over from the numerical tests in Section 11.4.

In order to draw meaningful conclusions, we do not only consider one run of the ant algorithm, but the average of 20 runs. This is important because the results of the random-based ant algorithm slightly differ from run to run.

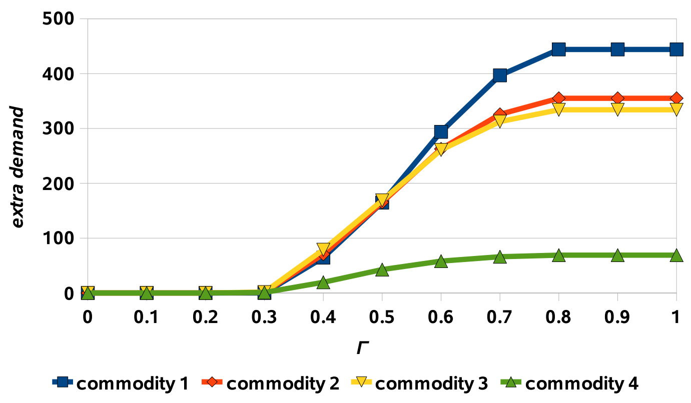
First, we consider the extra demand at the warehouses (nodes 42 to 56). As the robustness parameter Γ controls the level of protection against data uncertainty, we expect that for $\Gamma = 0$, no extra demand is delivered to the warehouses, for $\Gamma = 1$ the full extra demand is delivered to the warehouses and for increasing Γ, the amount of delivered extra demand is also increasing.

We examine the results for $\Gamma = 0$, $\Gamma = 0.1$, …, $\Gamma = 1$. The following table presents the total delivered extra demand at the warehouses for these values of Γ.

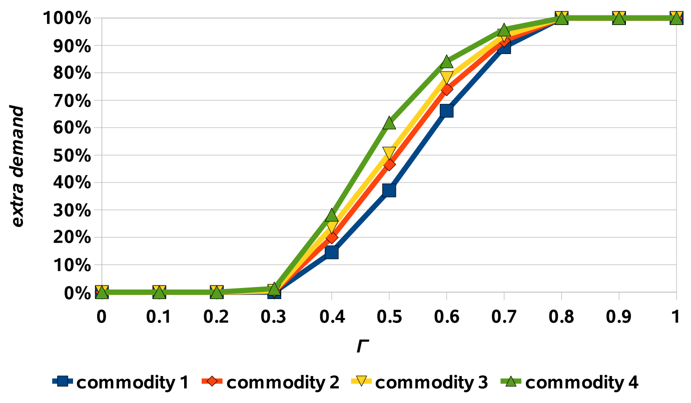
Γ commodity	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1	0	0	0	0	64.5	164.8	293.8	396.8	444	444	444
2	0	0	0	1.1	70.8	165.3	262.7	325.7	355	355	355
3	0	0	0	1.6	78.9	168.8	260.8	312.5	334	334	334
4	0	0	0	0.9	19.5	42.7	58.1	66.1	69	69	69

Table 12.7: Reference model - ant algorithm: Delivered extra demand for different values of Γ

The following figure illustrates the table data.



(a) Total extra demand



(b) Percentaged extra demand

Figure 12.2: Reference model - ant algorithm: Delivered extra demand for different values of Γ

The increase of the amount of delivered extra demand with increasing Γ corresponds to our expectations. But we have to note that the first units of extra demand are not delivered until $\Gamma = 0.3$ and that already for $\Gamma = 0.8$, the full extra demand is delivered.

As the penalty costs are increasing from commodity 1 to commodity 4, i.e. commodity 4 has the highest penalty cost, we can observe that for a fixed value of Γ, where $0.3 \leq \Gamma \leq 0.8$, most extra demand (in percent) is delivered for commodity 4 and least for commodity 1.

In the following, we consider the medium robust case $\Gamma = 0.5$ in detail.

Increased Storage Costs Now we want to examine the change in the delivered extra demand for increased storage cost. For this purpose, we increase the storage cost of commodity 2 from 6 to 24 at all warehouses.

The following table lists the delivered demand at warehouses (wh) 1 to 15 for $\Gamma = 0.5$ and all commodities (com).

com \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	124.2	109.6	110.8	111	110.3	85.1	93.8	123.4	142.9	104.4	144.2	113.2	99.3	96.1	97.5
2	101.9	92.6	87.4	86.9	90.6	73.9	72.3	109.1	112	82.1	126.8	95.1	81.7	77.3	80.6
3	103	86	85	80.3	85.6	66.6	62.4	113	90.7	86.8	125.1	85	79.6	82.7	72
4	20	19.5	19.7	19.1	19.4	14.9	15.5	24	20.6	19.8	28.8	19.3	18.3	18	16.8

(a) Original storage costs

com \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	129.4	109.3	114.3	111.3	112.8	89.1	96.5	127	145.7	106.6	144.9	113.7	99.9	95.7	98
2	99.9	90.3	82.2	82.8	87.7	72.4	68.2	105.2	113	78.9	123.3	92	79.9	76.4	79.4
3	103	92.4	89.9	85.2	86.4	70.1	66.6	113	95.9	82.3	120	87.9	86.5	83	77.2
4	20	21	20.1	18.8	20.2	15.9	16.2	24	22.7	19.2	27.3	20.7	18.7	18	17.6

(b) Increased storage costs

Table 12.8: Reference model - ant algorithm: Delivered demand for original and increased storage costs for commodity 2

As we would expect, the amount of delivered demand of commodity 2 is reduced, as surplus units of this commodity could probably cause higher storage costs. The amount of delivered demand of commodities 1, 3 and 4 is increased, as the storage of these products is comparatively cheaper now.

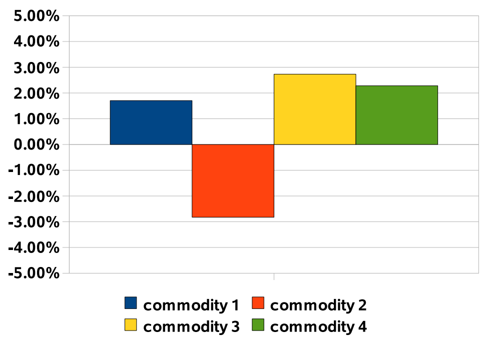


Figure 12.3: Reference model - ant algorithm: Change in the delivered extra demand for increased storage costs for commodity 2

Figure 12.3 shows the change in the delivered extra demand for the case that the storage costs for commodity 2 are increased. Here the average change per commodity is considered.

Increased Penalty Costs Moreover, we examine an increase of the penalty costs for warehouses 11 to 15 and commodities 1 and 2 only. We expect that for these warehouses and these commodities, the amount of delivered extra demand will be increased, as possibly missing units would cause higher penalty costs than before. Again, we consider the medium robust case only, i.e. $\Gamma = 0.5$. The penalty costs for commodities 1 and 2 and warehouses 11 to 15 are doubled.

wh \ com	1	2	sum
11	144.2	126.8	271
12	113.2	95.1	208.3
13	99.3	81.7	181
14	96.1	77.3	173.4
15	97.5	80.6	178.1

(a) Original penalty cost: Commodities 1 and 2

wh \ com	1	2	sum
11	148.9	132.1	281
12	114.7	97.2	211.9
13	110.2	81.9	192.1
14	96.3	81.7	178
15	98.2	83	181.2

(b) Increased penalty costs: Commodities 1 and 2

com / wh	3	4	sum
11	125.1	28.8	153.9
12	85	19.3	104.3
13	79.6	18.3	97.9
14	82.7	18	100.7
15	72	16.8	88.8

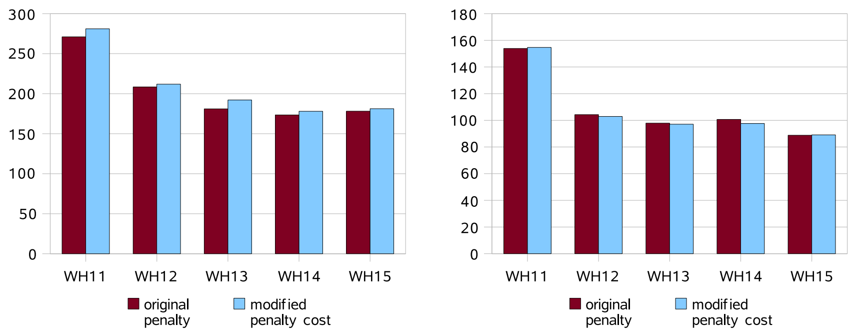
(c) Original penalty costs: Commodities 3 and 4

com / wh	3	4	sum
11	125.1	29.6	154.7
12	83.5	19.4	102.9
13	78.9	18.2	97.1
14	79.9	17.7	97.6
15	72.8	16.3	89.1

(d) Increased penalty costs: Commodities 3 and 4

Table 12.9: Reference model - ant algorithm: Delivered extra demand for original and increased penalty costs for commodities 1 and 2

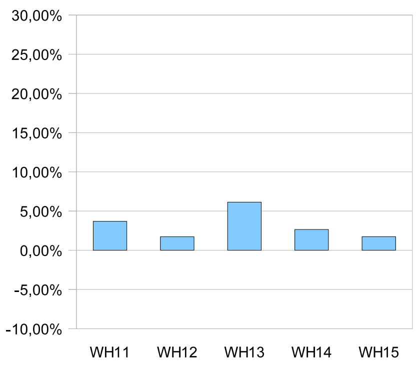
For warehouses 1 to 10, the amount of delivered demand units remains constant. The following figures illustrates the table data.



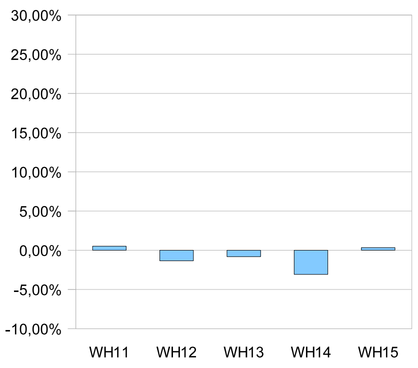
(a) Commodities 1 and 2

(b) Commodities 3 and 4

Figure 12.4: Reference model - ant algorithm: Change in the delivered extra demand for increased penalty costs for commodities 1 and 2 at warehouses WH11, WH12, WH13, WH14 and WH15



(a) Commodities 1 and 2



(b) Commodities 3 and 4

Figure 12.5: Reference model - ant algorithm: Percentual change in the delivered extra demand for increased penalty costs for commodities 1 and 2 at warehouses WH11, WH12, WH13, WH14 and WH15

We can see that for warehouses 11 to 15, the amount of delivered extra demand for commodities 1 and 2 is slightly higher than before the increase of the penalty costs. For commodities 3 and 4, however, the amount of delivered extra demand is nearly equal or marginally lower than before.

12.2 Transformation of Uncertain Demands to Uncertain Costs

In this section, we apply the transformation of uncertain demands to uncertain costs algorithm which we presented in Section 7.1. In order to solve the resulting uncertain cost problem, we apply the ant algorithm for uncertain cost values, as presented in Section 6.3.

The parameter setting for the ant algorithm is the following:

parameter	setting
α	1.0
β	1.0
ρ	0.5
η_{min}	0.001

parameter	setting
τ_{min}	1.0
τ_{max}	110
ς^{pher}	5.0

Table 12.10: Reference model - transformation algorithm: Ant algorithm parameter settings

The settings for the parameters α, β and τ_{min} are our standard settings for the ant algorithm for uncertain cost values. We decided to increase the pheromone evaporation coefficient ρ to 0.5, such that it is easier for the ants to drop information about solutions which were only valueable for the current iteration's cost values. As the additional arcs have rather high cost values, we had to decrease the lower bound for the visibility values τ_{min}, because otherwise, all additional arcs would have the same visibility τ_{min}. The setting for the upper pheromone bound τ_{max} was taken over from Section 12.1. In order to avoid early stagnation behavior, we introduce a parameter ς^{pher} as defined in Section 8.2 and set it to 5.0.

Differing from the settings given in Subsection 6.3.2 for the random generation of the cost values, the standard deviation σ_{ij}^k for arc (i,j) and commodity k is set to $\sigma_{ij}^k := \frac{1}{40} d_{ij}^k$, where d_{ij}^k is the extra cost value for arc (i,j) and commodity k. This modification is based on the fact that the cost intervals for the reference model are very large. The choice of the factor $\frac{1}{20}$ instead of $\frac{1}{4}$, which was used in Subsection 6.3.2, ensures that the generated cost values are not "too far" from the expected value $c_{ij}^k + \Gamma d_{ij}^k$.

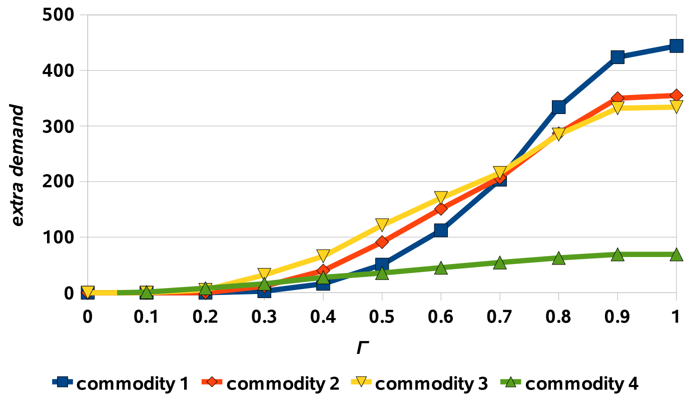
After the transformation of uncertain demands to uncertain costs, we apply the ant algorithm to the new robust optimization problem for different values of Γ, specifically $\Gamma = 0, 0.1, \dots, 1$.

In the following table, the delivered extra demand units in the optimal solution are listed.

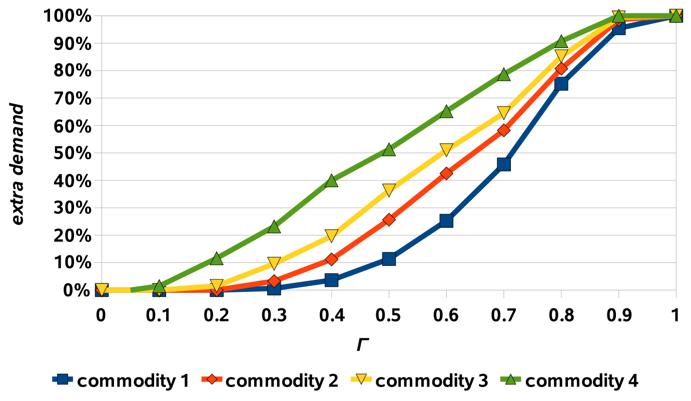
commodity \ Γ	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	0	0	0	2.7	16.0	50.4	112.1	203.4	334.0	423.8	444
2	0	0	0	11.6	39.7	91.0	150.9	206.7	286.6	350.0	355
3	0	0	5	32.0	65.5	121.2	170.3	215.4	284.6	332.0	334
4	0	1	8	16.0	27.6	35.4	45.0	54.3	62.6	69.0	69

Table 12.11: Reference model - transformation algorithm: Delivered extra demand for different values of Γ

The following figure illustrates the table data.



(a) Total extra demand



(b) Percentaged extra demand

Figure 12.6: Reference model - transformation algorithm: Delivered extra demand for different values of Γ

As we expected, the amount of delivered extra units increases with increasing Γ. In contrast to the ant algorithm for uncertain demands, cf. Figure 12.2, the number of delivered extra units increases more uniformly in the whole interval $0 \leq \Gamma \leq 1$.

We can see that the different commodities are delivered with extra demand in decreasing order of the corresponding penalty costs, i.e. commodity 4, which has the highest penalty costs, receives (percentaged) most extra units while commodity 1, which has the lowest penalty costs, receives (percentaged) least extra units. This behavior matches matches the behavior of the ant algorithm for uncertain demands, cf. Figure 12.2.

Again, we consider the medium robust case $\Gamma = 0.5$ in detail.

Increased Storage Costs In order to examine the change in the delivered extra demand for increased storage cost, we increase the storage cost of commodity 2 from 6 to 24 at all warehouses. The following tables lists the delivered extra units for original and modified storage costs.

com \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	107.3	101.7	105.9	103.8	107.5	79.6	81.1	119.8	125.2	98.7	141.9	104.8	92.7	91.4	90
2	90	84.4	85	82.1	87.3	73.7	66.2	101	100.9	82.1	127.7	88.4	76.3	76.9	74
3	92.8	83.9	84.8	79.5	85.8	64.5	63	102.3	93.6	83.7	123.7	81.8	77.2	73.3	66.3
4	18.6	18.8	19.9	18.2	19.9	14.4	15	22.2	21.3	20.3	29.4	18.9	17.2	16.4	15.9

(a) Original storage costs

com \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	111.6	101.1	107.3	103.2	106	80.8	81.7	119.8	125.2	97.7	144.7	105	93.7	91	90
2	88.6	83.8	82.8	80.2	85.9	71.2	63.2	100.7	99.2	79.3	122.9	87.9	73.9	73.1	73
3	92	83.3	85.4	80.3	86.3	66.1	63.4	104	93.4	83	123.6	83.1	79.4	73.2	66.1
4	18.8	18.2	19.2	18.3	19.6	15.1	15.3	22.9	21	20.5	28.6	19.3	17.8	16.9	16.4

(b) Increased storage costs

Table 12.12: Reference model - transformation algorithm: Delivered demand for original and increased storage costs for commodity 2

The following illustrates the table data.

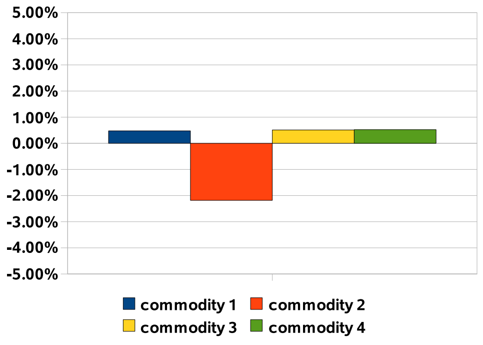


Figure 12.7: Reference model - transformation algorithm: Change in the delivered extra demand for increased storage costs for commodity 2

For increased storage costs for commodity 2, the delivered extra demand for commodity 2 is approx. 2.2% lower than for the original penalty costs. For commodities 1, 3 and 4, the delivered extra demand is approx. 0.5% higher than before. These results corresponds with our expectations and are in general comparable to the behavior of the ant algorithm for uncertain demands, cf.

Figure 12.3. However, for the transformation algorithm, the change in the number of delivered extra demand units is lower.

Increased Penalty Costs Moreover, we consider a doubling in the penalty costs for commodities 1 and 2 at warehouses 11 to 15. We obtain the following results:

wh \ com	1	2	sum
11	141.9	127.7	269.6
12	104.8	88.4	193.2
13	92.7	76.3	169
14	91.4	76.9	168.3
15	90	74	164

(a) Original penalty costs: Commodities 1 and 2

wh \ com	1	2	sum
11	154	138.4	292.4
12	106	92.8	198.8
13	99.2	82.6	181.8
14	98.2	79.4	177.6
15	90	76.9	166.9

(b) Increased penalty costs: Commodities 1 and 2

wh \ com	3	4	sum
11	123.7	29.4	153.1
12	81.8	18.9	100.7
13	77.2	17.2	94.4
14	73.3	16.4	89.7
15	66.3	15.9	82.2

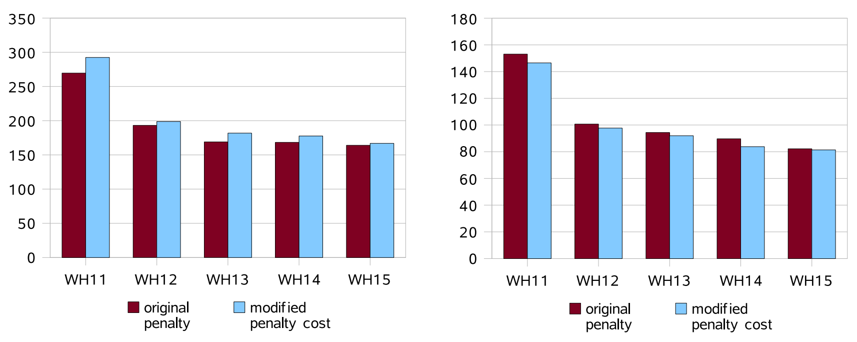
(c) Original penalty costs: Commodities 3 and 4

wh \ com	3	4	sum
11	118.6	27.9	146.5
12	78.8	18.9	97.7
13	74.8	17.2	92
14	68	15.8	83.8
15	66.1	15.3	81.4

(d) Increased penalty costs: Commodities 3 and 4

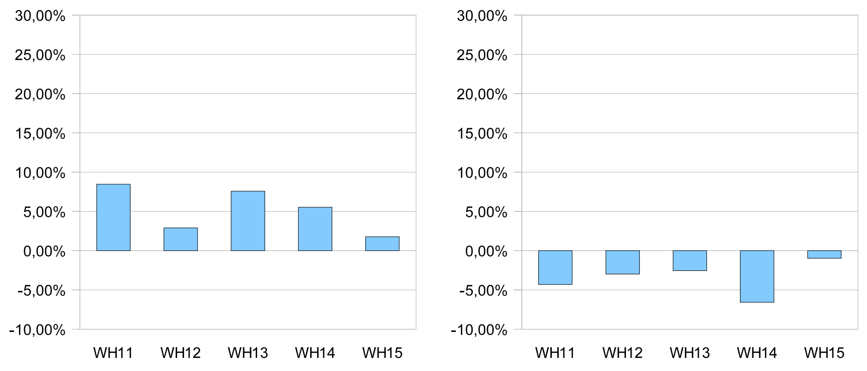
Table 12.13: Reference model - transformation algorithm: Delivered extra demand for original and increased penalty costs for commodities 1 and 2

The following figures illustrates the table data.



(a) Commodities 1 and 2 **(b)** Commodities 3 and 4

Figure 12.8: Reference model - transformation algorithm: Change in the delivered extra demand for increased penalty costs for commodities 1 and 2 at warehouses WH11, WH12, WH13, WH14 and WH15



(a) Commodities 1 and 2 **(b)** Commodities 3 and 4

Figure 12.9: Reference model - transformation algorithm: Percentual change in the delivered extra demand for increased penalty costs for commodities 1 and 2 at warehouses WH11, WH12, WH13, WH14 and WH15

For commodities 1 and 2, the delivered demand at warehouses 11 to 15 is higher for the modified penalty costs than for the original penalty costs, while the delivered extra demand at warehouses 11 to 15 is lower than before.

The change in the delivered extra demand is comparable to the results of ant algorithm for uncertain demands, cf. Figures 12.4 and 12.5, with only small differences in the amounts of delivered extra units.

In general, the results of the transformation algorithm are similar to those of the ant algorithm for uncertain demands in Section 12.1. However, small differences concerning the influence of the robustness parameter can be recognized. For the ant algorithm for uncertain demands, the increase of the number of sent extra demand units is limited to the interval from $\Gamma = 0.3$ to $\Gamma = 0.8$. For the transformation algorithm, the increase of sent additional demand units is distributed over the whole interval $[0,\ 1]$ for Γ. However, the relation between the curves for the four different commodities is similar for both approaches.

12.3 Recoverable Robustness

In this section, we apply the theory of recoverable robustness (see Section 8.3) to the reference model.

We choose a translated demand scenario set, as proposed in Section 8.3.3, based on the lower and upper demand bounds given at the beginning of this chapter. As a fixed value for the recovery budget is not given in advance, we add the recovery budget D as an optimization variable with weight 1 to the objective function, such that the objective function corresponds to the total cost $c^\top x + D$.

The delivered demand in the optimal solution of the LRP is listed in the following table. The recovery budget in the optimal solution is 15500.

com \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	110.5	108.5	111.8	110.5	111.8	83.8	82.8	137.7	123.5	104.3	147.5	110.8	98.8	95.5	96.5
2	110	89	82.8	83.8	87	72	66.5	120.5	95.5	80.5	127.8	92.3	79.5	77.3	79.5
3	108.8	84.8	87.5	78.5	84.8	69.0	62.3	93.5	91.3	85.1	122.2	82.8	79.5	87.8	70.8
4	17	18.3	18.3	17.3	18.3	13.8	14	21	20.3	18.3	25.8	18.3	17	15	16

Table 12.14: Reference model - recoverable problem: Delivered demand at warehouses 1 to 15

The following figure illustrates the delivered extra demand in percent at the warehouses 1 to 15.

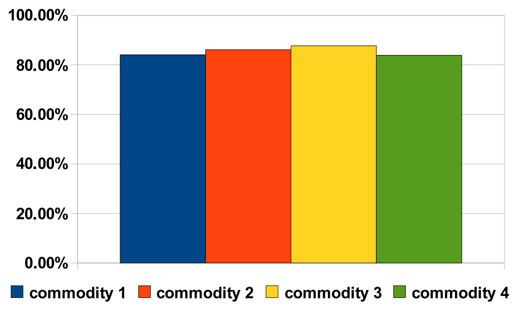


Figure 12.10: Reference model - recoverable problem: Delivered extra demand in percent

Figure 12.10 shows the delivered extra demand, where the average change per commodity is considered.

As we would expect, the percentaged extra demand increases from commodity 1 to commodity 3, since the penalty costs also increase from commodity 1 to 3. However, for commodity 4, which has even higher penalty costs than commodity 3, the percentaged delivered extra demand is less than for the other commodities. This can be explained by the fact that the total number of demand units is much less for commodity 4 than for the other commodities, while the recovery budget is fixed over all commodities. Hence for commodity 4, the percentage of unavailable units that can be cushioned by the recovery budget is higher than for the other commodities.

As in the recoverable robustness approach, the robustness of the solution is not controllable via a robustness parameter, a detailled comparison with the results of the ant algorithm for uncertain demands, cf. Section 12.1, and the transformation algorithm, cf. Section 12.2, is rather difficult. However, we note that the low amount of delivered extra demand units for commodity 4 is a special characteristic of the recoverable robustness approach.

Increased Storage Costs Now we consider the LRP with increased storage costs for commodity 2. The optimal recovery budget is now 16503.

com \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	110.5	108.5	111.8	110.5	111.8	83.8	82.8	115	123.5	104.3	147.5	110.8	98.8	95.5	96.5
2	86	89	82.75	83.8	87	72	66.5	119.4	95.5	80.5	127.8	92.3	79.5	77.3	79.5
3	108.8	84.8	83.9	78.5	84.8	65.6	62.3	93.5	91.3	81.5	118	82.8	79.5	87.8	70.8
4	17	18.3	18.3	17.3	18.3	13.8	14	21	20.3	18.3	25.8	18.3	17	15	16

Table 12.15: Reference model - recoverable problem: Delivered demand for increased storage costs for commodity 2

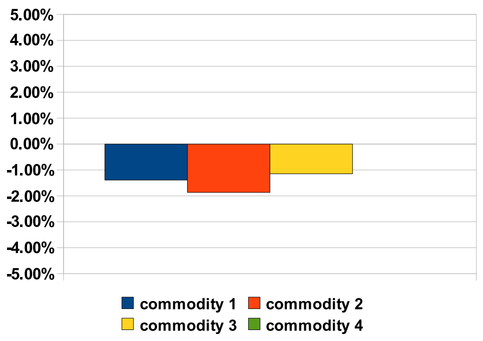


Figure 12.11: Reference model - recoverable problem: Change in the delivered extra demand for increased storage costs for commodity 2

As before for the ant algorithm for uncertain demands, see Figures 12.3 and 12.7, the amount of delivered demand of commodity 2 is reduced, as this could probably cause higher storage costs. The amount of delivered demand of commodities 1 and 3 is also reduced due to the higher recovery budget, which implicates that a higher total sum of recovery costs can be coped with. Note again that the limited recovery budget is a special constraint for the recoverable robustness approach. For commodity 4, there is no further reduction as for this commodity already fewer extra units were delivered than for all other commodities.

Increased Penalty Costs Finally, we examine the effects of an increase of the penalty costs for commodities 1 and 2 at warehouses 1 to 15. The optimal recovery budget is 13121.

wh \ com	1	2	sum
11	147.5	127.8	275.3
12	110.8	92.3	203
13	98.8	79.5	178.3
14	95.5	77.3	172.8
15	96.5	79.5	176

(a) Original penalty costs: Commodities 1 and 2

wh \ com	1	2	sum
11	189.5	162.8	352.3
12	141.78	117.3	259
13	125.8	101.5	227.3
14	121.5	98.3	219.8
15	96.5	79.5	176

(b) Increased penalty costs: Commodities 1 and 2

wh \ com	3	4	sum
11	122.3	25.8	147.9
12	82.8	18.3	101
13	79.5	17	96.5
14	87.8	15	102.8
15	70.8	16	86.8

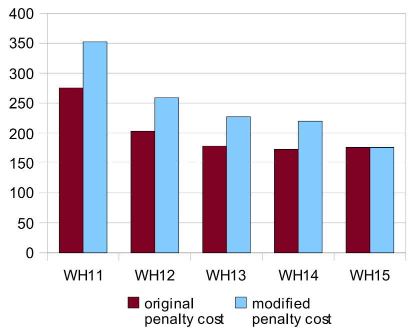
(c) Original penalty costs: Commodities 3 and 4

wh \ com	3	4	sum
11	147	25.8	172.8
12	82.8	18.3	101
13	79.5	17	96.5
14	87.8	15	102.8
15	70.8	16	86.8

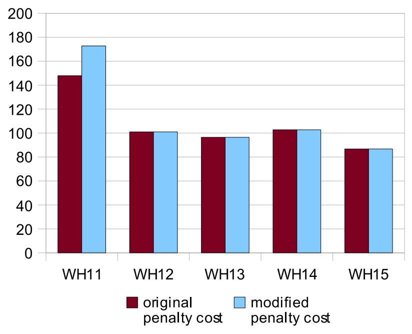
(d) Increased penalty costs: Commodities 3 and 4

Table 12.16: Reference model - recoverable problem: Delivered extra demand for original and increased penalty costs for commodities 1 and 2

The change in the delivered extra demand is illustrated in the following figures.

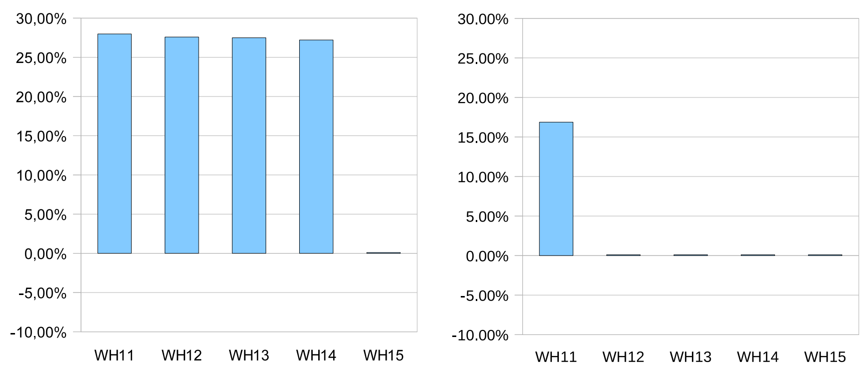


(a) Commodities 1 and 2



(b) Commodities 3 and 4

Figure 12.12: Reference model - recoverable problem: Change in the delivered extra demand for increased penalty costs for commodities 1 and 2 at warehouses WH11, WH12, WH13, WH14 and WH15



(a) Commodities 1 and 2

(b) Commodities 3 and 4

Figure 12.13: Reference model - recoverable problem: Percentual change in the delivered extra demand for increased penalty costs for commodities 1 and 2 at warehouses WH11, WH12, WH13, WH14 and WH15

For commodities 1 and 2 and for warehouses 11 to 14, the number of delivered units increases, for warehouse 15, it remains stable. For commodities 3 and 4, the number of delivered units remains stable at warehouses 12 to 15 and increased for warehouse 11. Though these results differ from those given in Figures 12.4, 12.5, 12.8 and 12.9 for the ant algorithm for uncertain demands and the transformation algorithm, the general tendency of the changes in extra demand units is similar.

The decrease of the recovery budget can be explained by the fact that higher penalty costs lead to a higher amount of produced units in order to avoid these costs. Then the number of possible unavailable units decreases and the recovery budget can be decreased.

12.4 Stochastic Two-Stage Linear Recourse Problem

Finally, we consider the given supply chain problem as stochastic two-stage linear recourse problem. In order to solve the resulting optimization problem, we apply the multicommodity version of the L-shaped algorithm, which we developed in Section 9.2.

12.4.1 Single Time Period

For the two-stage linear recourse problem, we consider not only the lower and upper bounds for the uncertain demand values, but three different scenarios (sc) with a probability of $\frac{1}{3}$ each. The demand data for scenarios 1, 2 and 3 are given in the table below.

sc \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	133	131	135	133	135	101	100	139	149	126	179	134	119	115	116
2	118	131	110	106	130	90	77	130	132	111	137	105	119	110	92
3	103	101	104	103	104	78	77	107	115	97	138	103	92	89	90

(a) Commodity 1

sc \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	104	107	100	101	105	87	80	114	115	97	154	111	96	93	96
2	104	99	80	90	105	80	64	88	110	90	140	111	74	81	85
3	80	83	77	78	81	67	62	88	89	75	119	86	74	72	74

(b) Commodity 2

sc \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	103	102	97	95	102	75	75	113	110	94	139	100	96	83	85
2	81	90	77	76	100	66	60	109	100	80	107	88	96	83	77
3	80	79	75	73	79	58	58	87	85	73	107	77	74	64	66

(c) Commodity 3

sc \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	20	22	22	21	22	16	17	24	24	22	31	22	20	18	19
2	20	18	17	20	22	13	17	22	22	19	30	18	16	15	19
3	16	17	17	16	17	13	13	20	19	17	24	17	16	14	15

(d) Commodity 4

Table 12.17: Reference model - two-stage problem: Demand scenarios

These demand scenarios were also given by the industry partner Henkel KGaA. We have the following relation between the interval data for the demand uncertainty, as given at the beginning of this chapter, and the demand scenarios given above: The demand value of the first scenario corresponds to the upper bound of the uncertainty interval. The demand value of the third scenario corresponds to the lower bound of the uncertainty interval. The demand value of the second scenario is any value of the uncertainty interval, which corresponds to a likely scenario in practice. The demand value of the second scenario therefore determines, which region of the uncertainty interval is more likely. This differs from the robust approaches, where all interval values are implicitly assumed to be equally likely.

As a recourse model, we consider simple recourse with the given penalty and storage costs.

In contrast to the algorithms considered in Section 12.1 and in Section 12.2, we don't have a robustness parameter for the two-stage model. We apply the multicommodity L-shaped algorithm, which we developed in Section 9.2 to the two-stage problem and obtain the following solution after 112 iterations:

com \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	118	131	110	106	130	90	77	130	132	111	138	105	119	110	90
2	104	99	80	90	105	80	64	114	110	90	140	111	74	81	74
3	103	102	97	95	102	75	75	113	100	94	139	100	96	83	77
4	20	22	22	21	22	16	17	24	24	22	31	22	20	18	19

Table 12.18: Reference model - two-stage problem: Delivered demand at warehouses 1 to 15

The following figure illustrates the table data.

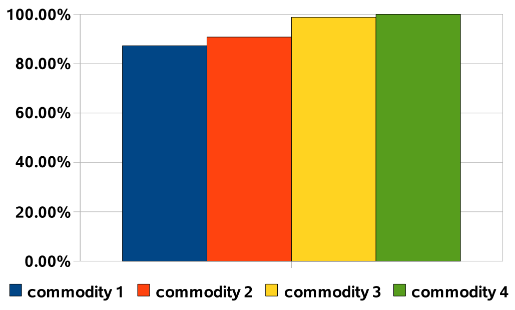


Figure 12.14: Reference model - two-stage problem: delivered extra demand in percent

Figure 12.14 shows the delivered extra demand, where the average change per commodity is considered.

We can see that for commodity 4, which has the highest penalty costs, the full extra demand is delivered, while for commodity 1, which has the lowest penalty costs, only 87% of the extra demand is delivered. Due to capacity restrictions, it is not possible to send the full demand for all commodities.

When comparing the results in Figure 12.14 with those of the recoverable robustness approach, see Figure 12.10, we observe that for both approaches, for all commodities more than 80% of the extra demand units are actually devliered. When comparing these approaches, we have to keep in mind that the two-stage approach is scenario-based, while the recoverable robustness approach is based on interval data for the uncertain demand values.

Increased Storage Costs As before, we examine increased storage costs for commodity 2. The costs are increased from 6 to 24. The following table lists the delivered demand for the modified storage costs.

com \ wh	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	118	131	110	106	130	90	77	130	132	111	138	105	119	110	90
2	104	99	80	90	105	80	64	88	110	90	140	111	74	81	74
3	103	102	97	95	102	75	75	113	100	94	139	100	96	83	77
4	20	22	22	21	22	16	17	24	24	22	31	22	20	18	19

Table 12.19: Reference model - two-stage problem: Delivered demand for increased storage costs for commodity 2

The change in the delivered extra demand for increased storage costs for commodity 2 is illustrated in the following figure.

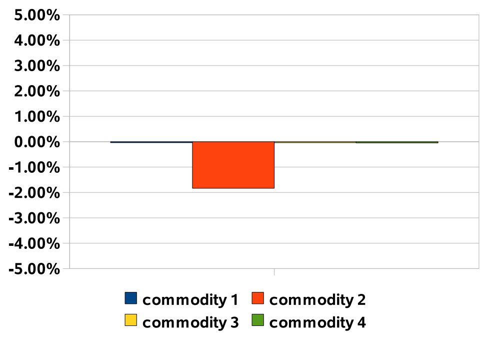


Figure 12.15: Reference model - two-stage problem: Change in the delivered extra demand for increased storage costs for commodity 2

The delivered extra demand for commodity 2 is decreased, as a result of the increased storage costs. However, in contrast to the previous results, see Figures 12.3, 12.7 and 12.11, the delivered extra demand for commodities 1, 3 and 4 remains unchanged. Differences to the ant algorithm for uncertain demands and the transformation algorithm, where the resulting problems are also solved by an ant algorithm, the L-shaped algorithm calculates its results based on the absolute values of the input data, while the ant algorithms calculate arc probabilities based on a ratio of the different cost values of different commodities. For the recoverable robustness approach, the modification of the penalty costs leads to a change in the recovery budget, which directly has effects on all commodities.

Increased Penalty Costs Finally, we examine increased penalty costs for commodities 1 and 2 at warehouses 11 to 15. The following tables list the delivered extra demand for original and increased penalty costs.

wh \ com	1	2	sum
11	138	140	278
12	105	111	216
13	119	74	193
14	110	81	191
15	90	74	164

(a) Original penalty costs: Commodities 1 and 2

wh \ com	1	2	sum
11	179	154	333
12	134	111	245
13	119	96	215
14	115	93	208
15	92	85	177

(b) Increased penalty costs: Commodities 1 and 2

wh \ com	3	4	sum
11	139	31	170
12	100	22	122
13	96	20	116
14	83	18	101
15	77	19	96

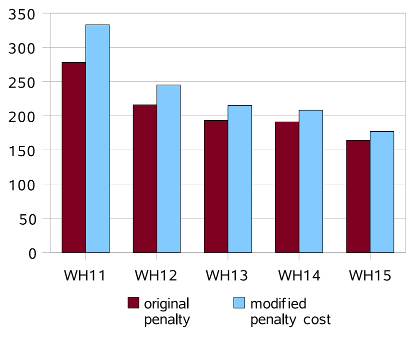
(c) Original penalty costs: Commodities 3 and 4

wh \ com	3	4	sum
11	139	31	170
12	100	22	122
13	96	20	116
14	83	18	101
15	77	19	96

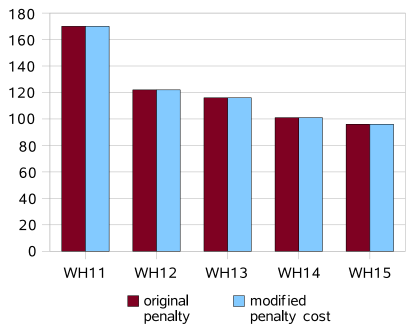
(d) Increased penalty costs: Commodities 3 and 4

Table 12.20: Reference model - two-stage problem: Delivered extra demand for original and increased penalty costs for commodities 1 and 2

The following figures illustrates the table data.

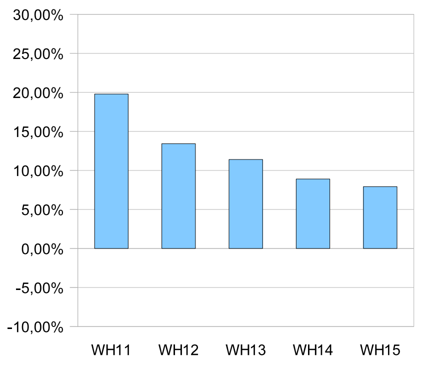


(a) Commodities 1 and 2

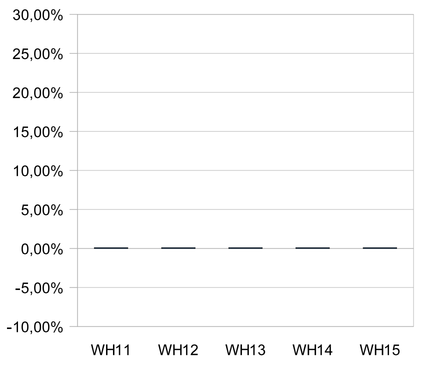


(b) Commodities 3 and 4

Figure 12.16: Reference model - two-stage problem: Change in the delivered extra demand for increased penalty costs for commodities 1 and 2 at warehouses WH11, WH12, WH13, WH14 and WH15



(a) Commodities 1 and 2



(b) Commodities 3 and 4

Figure 12.17: Reference model - recoverable problem: Percentual change in the delivered extra demand for increased penalty costs for commodities 1 and 2 at warehouses WH11, WH12, WH13, WH14 and WH15

For commodities 1 and 2, the delivered extra demand is increased at the concerned warehouses. For commodities 3 and 4, the delivered extra demand remains unchanged. Again, the L-shaped algorithm delivers different results than the other considered optimization models, see Figures 12.4, 12.5, 12.8, 12.9, 12.12 and 12.13, in this context. As we already explained in the context of increased storage costs, differences to the ant algorithms can be explained by the fact that for the ant algorithms the calculation of arc probabilities is based on cost ratios, while the L-shaped algorithm calculates its results based on absolute values. For the recoverable robustness approach, the effects of a modified recovery budget has to be kept in mind.

12.4.2 Recourse by Interexchange

Now we add the possibility of recourse by interexchange to the reference model network. Here we want to apply the interexchange recourse approach for two-stage problems, which was given in Section 9.3.

As before, we consider three demand scenarios as given in Subsection 12.4.1.

The following tables list the total transport to the demand nodes before interexchange, the interexchange for scenarios 1,2 and 3, as well as the total delivered amounts for scenarios 1, 2 and 3.

node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
transport	133	45	104	52	104	101	102	216	132	148	179	146	147	89	0
interexchange 1	0	0	8	17	22	0	-2	-77	0	-22	0	-12	-28	20	74
interexchange 2	-15	86	6	54	26	-11	-25	-86	0	-37	-42	-41	-28	21	92
interexchange 3	-20	56	0	51	0	-16	-20	-20	0	-10	-40	-40	-31	0	90
total 1	133	45	112	69	126	101	100	139	132	126	179	134	119	109	74
total 2	118	131	110	106	130	90	77	130	132	111	137	105	119	110	92
total 3	113	101	104	103	104	85	82	196	132	138	139	106	116	89	90

(a) Commodity 1

node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
transport	116	79	96	77	83	82	60	183	121	98	150	93	63	100	0
interexchange 1	-12	20	0	9	0	0	0	-69	-6	-1	0	0	20	-7	46
interexchange 2	-12	20	-16	13	22	-2	4	-95	-11	-8	-10	18	10	-19	85
interexchange 3	-20	4	-2	1	0	0	2	-20	-15	-14	0	0	11	-20	74
total 1	104	99	96	86	83	82	60	114	115	97	150	93	83	93	46
total 2	104	99	80	90	105	80	64	88	110	90	140	111	74	81	85
total 3	96	83	93	78	83	82	62	163	105	84	150	93	74	80	74

(b) Commodity 2

node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
transport	113	102	97	95	102	75	75	113	100	94	139	100	96	102	66
interexchange 1	-10	0	0	0	0	0	0	0	10	0	0	0	0	-19	19
interexchange 2	0	0	0	-11	0	0	0	0	0	0	0	0	0	0	11
interexchange 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
total 1	103	102	97	95	102	75	75	113	110	94	139	100	96	83	85
total 2	113	102	97	84	102	75	75	113	100	94	139	100	96	102	77
total 3	113	102	97	95	102	75	75	113	100	94	139	100	96	102	66

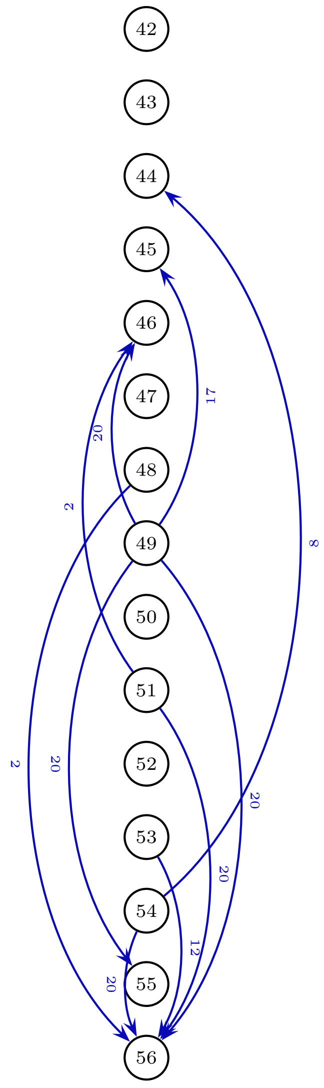
(c) Commodity 3

node	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
transport	33	22	22	21	22	16	17	24	19	22	31	22	16	18	15
interexchange 1	-13	0	0	0	0	0	0	0	5	0	0	0	4	0	4
interexchange 2	0	0	-4	0	0	-3	0	0	3	0	0	0	0	0	4
interexchange 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
total 1	20	22	22	21	22	16	17	24	24	22	31	22	20	18	19
total 2	33	22	18	21	22	13	17	24	22	22	31	22	16	18	19
total 3	33	22	22	21	22	16	17	24	19	22	31	22	16	18	15

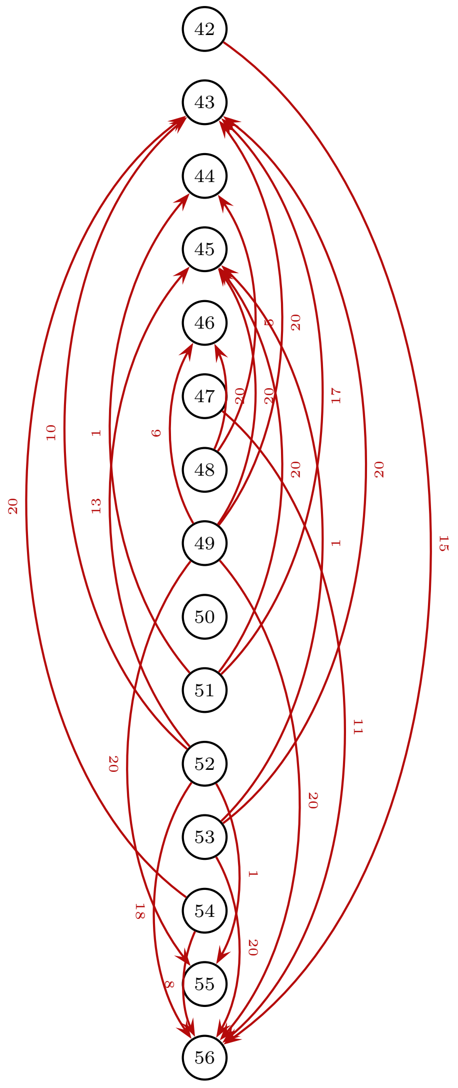
(d) Commodity 4

Table 12.21: Reference model - two-stage problem: Interexchange

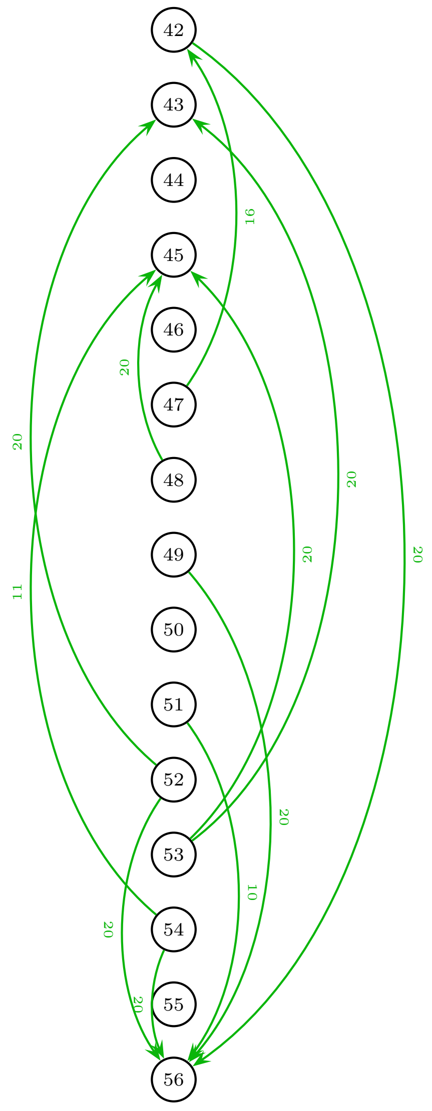
The following figure illustrates the interexchange for commodity 1 in detail.



(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

Figure 12.18: Reference model - two-stage problem: Interexchange for commodity 1

The total cost (production, transportation, penalty and storage costs) of the reference model with the possibility of interexchange is 78043. Without the possibility of interexchange, the total cost is 81023. Clearly, the additional possibility of recourse by interexchange reduces the total cost of the respective optimal solution.

12.4.3 Inventory Management: Multiple Time Periods

Now we want to consider the reference model network for multiple time periods. Here we want to apply the inventory management approach for two-stage problems, which was given in Section 9.4.

We consider four time periods with increasing demands at the warehouses from period 1 to period 4. The increasing demands could be due to a wide-area marketing campaign, for example. As we do not consider the time periods separately, we are able to produce goods in advance and store them until the effects of the marketing campaign take place. This is called inventory management.

Again, we consider three demand scenarios per time period. The following tables list the demand scenarios for warehouses 1 to 15 for four time periods.

com	sc	WH1	WH2	WH3	WH4	WH5	WH6	WH7	WH8	WH9	WH10	WH11	WH12	WH13	WH14	WH15
1	1	35.4	34.8	36	35.7	35.7	26.7	26.7	36.9	39.6	33.3	47.7	35.7	31.8	30.6	30.9
	2	35.4	34.8	36	35.7	35.7	26.7	26.7	36.9	39.6	33.3	47.7	35.7	31.8	30.6	30.9
	3	24.6	24.3	25.2	24.9	24.9	18.6	18.6	25.8	27.6	23.4	33.3	24.9	22.2	21.3	21.6
2	1	27.9	28.5	26.7	27	28.2	23.1	21.6	30.3	30.6	26.1	41.1	29.7	25.8	24.9	25.5
	2	47.4	48.6	45.3	45.9	48	39.3	36.6	51.3	52.5	44.4	70.2	50.4	43.8	42.6	43.5
	3	47.4	48.6	45.3	45.9	48	39.3	36.6	18	18.3	15.6	24.6	17.7	15.3	42.6	15.3
3	1	27.6	27.3	25.8	25.2	27.3	20.1	20.1	30	29.4	25.2	36.9	26.4	25.5	22.2	22.5
	2	46.8	46.5	44.1	43.2	46.5	34.2	34.2	51.3	50.1	42.9	63	45	43.8	37.8	38.7
	3	44.1	43.8	41.4	40.5	43.8	32.4	32.4	48	47.1	40.2	59.4	42.3	41.1	35.7	36.3
4	1	5.4	5.7	6	5.7	5.7	4.5	4.5	6.6	6.6	6	8.1	6	5.4	4.8	5.1
	2	3.3	3.3	3.6	3.3	3.3	2.7	2.7	6.6	6.6	6	8.1	6	5.4	2.7	5.1
	3	5.4	5.7	6	5.7	5.7	4.5	4.5	6.6	6.6	6	8.1	6	5.4	4.8	5.1

(a) Time period 1

com	sc	WH1	WH2	WH3	WH4	WH5	WH6	WH7	WH8	WH9	WH10	WH11	WH12	WH13	WH14	WH15
1	1	127.6	130.9	125.4	129.8	127.6	104.5	96.8	145.2	146.3	118.8	187	130.9	121	114.4	107.8
	2	127.6	130.9	125.4	129.8	127.6	104.5	96.8	145.2	146.3	118.8	187	130.9	121	114.4	107.8
	3	89.1	91.3	86.9	90.2	89.1	72.6	68.2	101.2	102.3	83.6	130.9	91.3	84.7	80.3	75.9
2	1	97.9	100.1	99	100.1	101.2	79.2	80.3	108.9	108.9	96.8	149.6	104.5	94.6	91.3	88
	2	167.2	170.5	169.4	170.5	172.7	135.3	136.4	185.9	184.8	165	254.1	178.2	161.7	155.1	149.6
	3	167.2	170.5	169.4	170.5	172.7	135.3	136.4	64.9	64.9	58.3	89.1	62.7	57.2	155.1	52.8
3	1	100.1	97.9	101.2	100.1	96.8	74.8	80.3	106.7	111.1	93.5	144.1	102.3	85.8	88	90.2
	2	170.5	166.1	171.6	170.5	166.1	127.6	137.5	181.5	190.3	158.4	244.2	173.8	146.3	150.7	154
	3	160.6	156.2	161.7	160.6	156.2	119.9	128.7	170.5	179.3	149.6	229.9	163.9	137.5	141.9	144.1
4	1	20.9	19.8	20.9	20.9	20.9	16.5	16.5	24.2	24.2	20.9	30.8	20.9	19.8	18.7	19.8
	2	12.1	12.1	12.1	12.1	12.1	9.9	9.9	24.2	24.2	20.9	30.8	20.9	19.8	11	19.8
	3	20.9	19.8	20.9	20.9	20.9	16.5	16.5	24.2	24.2	20.9	30.8	20.9	19.8	18.7	19.8

(b) Time period 2

com	sc	WH1	WH2	WH3	WH4	WH5	WH6	WH7	WH8	WH9	WH10	WH11	WH12	WH13	WH14	WH15
1	1	226	222	232	226	228	176	186	258	262	232	334	232	212	206	214
	2	226	222	232	226	228	176	186	258	262	232	334	232	212	206	214
	3	158	156	162	158	158	122	130	180	182	162	234	162	148	144	150
2	1	190	178	190	180	182	142	148	198	208	180	266	202	170	158	168
	2	324	304	324	308	310	242	252	338	354	306	452	344	290	268	286
	3	324	304	324	308	310	242	252	118	124	108	160	120	102	268	100
3	1	182	180	174	174	176	134	146	194	204	168	252	194	164	156	154
	2	312	306	296	296	298	230	250	328	346	286	430	330	280	264	264
	3	294	288	278	278	282	216	236	310	326	270	404	310	264	250	248
4	1	38	40	38	38	36	30	30	40	42	38	58	40	36	34	36
	2	22	24	22	22	22	18	18	40	42	38	58	40	36	20	36
	3	38	40	38	38	36	30	30	40	42	38	58	40	36	34	36

(c) Time period 3

com	sc	WH1	WH2	WH3	WH4	WH5	WH6	WH7	WH8	WH9	WH10	WH11	WH12	WH13	WH14	WH15
1	1	324.8	310.8	305.2	336	308	252	260.4	341.6	358.4	313.6	462	333.2	294	291.2	288.4
	2	324.8	310.8	305.2	336	308	252	260.4	341.6	358.4	313.6	462	333.2	294	291.2	288.4
	3	226.8	218.4	212.8	235.2	215.6	176.4	182	238	249.2	218.4	322	232.4	204.4	204.4	201.6
2	1	249.2	260.4	252	254.8	257.6	196	212.8	285.6	274.4	249.2	355.6	277.2	246.4	226.8	240.8
	2	422.8	445.2	428.4	436.8	439.6	336	361.2	487.2	464.8	425.6	604.8	473.2	417.2	386.4	411.6
	3	422.8	445.2	428.4	436.8	439.6	336	361.2	170.8	162.4	148.4	212.8	165.2	145.6	386.4	142.8
3	1	252	257.6	243.6	252	252	201.6	190.4	266	266	229.6	344.4	249.2	232.4	224	221.2
	2	428.4	436.8	411.6	428.4	428.4	341.6	327.6	453.6	453.6	392	585.2	425.6	397.6	380.8	378
	3	403.2	411.6	389.2	403.2	403.2	322	308	428.4	425.6	369.6	551.6	400.4	375.2	358.4	355.6
4	1	56	56	53.2	53.2	53.2	42	44.8	58.8	61.6	50.4	81.2	53.2	50.4	47.6	50.4
	2	33.6	33.6	30.8	30.8	30.8	25.2	25.2	58.8	61.6	50.4	81.2	53.2	50.4	28	50.4
	3	56	56	53.2	53.2	53.2	42	44.8	58.8	61.6	50.4	81.2	53.2	50.4	47.6	50.4

(d) Time period 4

Table 12.22: Reference model - multiple time periods: Demand scenarios

Now we apply the multicommodity L-shaped algorithm to the two-stage inventory management problem, which terminates after 1319 iterations.

In the following tables, the delivered units for warehouses 1 to 15 are listed per time period and per commodity. The number of delivered units already contains the units which were produced in previous time periods and stored until needed.

com	WH1	WH2	WH3	WH4	WH5	WH6	WH7	WH8	WH9	WH10	WH11	WH12	WH13	WH14	WH15
1	35.4	34.8	36	35.7	35.7	26.7	26.7	36.9	39.6	33.3	47.7	35.7	31.8	30.6	21.6
2	47.4	48.6	45.3	45.9	48	39.3	36.6	51.3	30.6	26.1	41.1	29.7	25.8	42.6	15.3
3	46.8	46.5	44.1	43.2	46.5	34.2	34.2	51.3	47.1	42.9	63	45	43.8	37.8	36.3
4	5.4	5.7	6	5.7	5.7	4.5	4.5	6.6	6.6	6	8.1	6	5.4	4.8	5.1

(a) Time period 1

com	WH1	WH2	WH3	WH4	WH5	WH6	WH7	WH8	WH9	WH10	WH11	WH12	WH13	WH14	WH15
1	127.6	130.9	125.4	129.8	127.6	104.5	96.8	145.2	146.3	118.8	187	130.9	121	114.4	0
2	167.2	170.5	169.4	170.5	172.7	135.3	136.4	185.9	108.9	96.8	149.6	104.5	94.6	155.1	52.8
3	170.5	166.1	171.6	170.5	166.1	127.6	137.5	181.5	179.3	149.6	229.9	173.8	146.3	150.7	144.1
4	20.9	19.8	20.9	20.9	20.9	16.5	16.5	24.2	24.2	20.9	30.8	20.9	19.8	18.7	19.8

(b) Time period 2

com	WH1	WH2	WH3	WH4	WH5	WH6	WH7	WH8	WH9	WH10	WH11	WH12	WH13	WH14	WH15
1	226	222	232	226	228	176	186	258	262	162	234	232	212	206	0
2	324	178	324	308	310	142	163.1	338	208	180	266	120	102	268	0
3	312	306	296	296	298	230	250	328	326	270	404	330	264	264	248
4	38	40	38	38	36	30	30	40	42	38	58	40	36	34	36

(c) Time period 3

com	WH1	WH2	WH3	WH4	WH5	WH6	WH7	WH8	WH9	WH10	WH11	WH12	WH13	WH14	WH15
1	324.8	218.4	305.2	235.2	308	252	260.4	341.6	358.4	218.4	322	316.8	294	291.2	0
2	249.2	260.4	252	436.8	439.6	196	212.8	487.2	274.4	148.4	212.8	165.2	145.6	226.8	0
3	403.2	411.6	389.2	403.2	403.2	322	308	453.6	425.6	369.6	551.6	400.4	375.2	358.4	221.2
4	56	56	53.2	53.2	53.2	42	44.8	58.8	61.6	50.4	81.2	53.2	50.4	47.6	50.4

(d) Time period 4

Table 12.23: Reference model - multiple time periods: Optimal solution

The following table lists the number of units that are stored due to inventory management reasons. Negative numbers represent numbers of units that are produced in the current time period and stored. Positive numbers represent numbers of units that are stored from previous time periods up to the current time period.

per / wh	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	-103	103
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	-162	162	0
11	0	-275.1	234	41.1
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0

(a) Commodity 1

per / wh	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	-47	47	0
7	-123.9	123.9	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	-311.6	149.6	162	0
12	0	0	0	0
13	0	-108.9	49.1	59.8
14	0	0	0	0
15	0	0	0	0

(b) Commodity 2

per / wh	1	2	3	4
1	0	0	-241.4	241.4
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	-149.6	149.6	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0

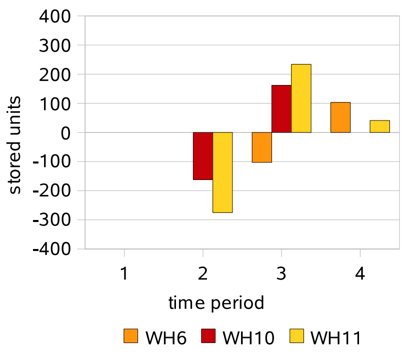
(c) Commodity 3

per / wh	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0

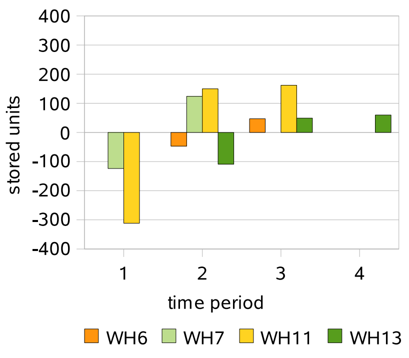
(d) Commodity 4

Table 12.24: Reference model - multiple time periods: Inventory management

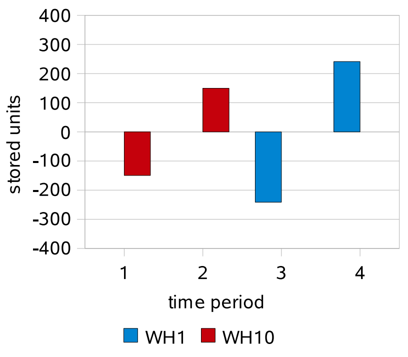
The following figure illustrates the table data.



(a) Commodity 1



(b) Commodity 2



(c) Commodity 3

Figure 12.19: Reference model - multiple time periods: Inventory management

As we have relatively low demand values in time period 1, average demand values in time period 2 and relatively high demand values in time period 3 and 4, we have a surplus production in the early time periods and store these units for later time periods with higher demands. Note that due to capacity restrictions in the production process, only a limited number of extra units can be produced in each time period.

The total cost (production, transportation, penalty and storage costs) for the four-period problem is 725480. If we optimize the four single-period problems separately without a possibility of inventory management, the total cost is 728135.

12.5 Summary and Conclusions

In this chapter, we developed a reference supply chain example, modeling the production and transportation process, which was based on real-world input data. We focused on data uncertainties in the demand values at the warehouses at the end of the supply chain process.

We applied four different algorithms to the reference model: the ant algorithm, the transformation algorithm, the recoverable robustness algorithm and the L-shaped algorithm. Therefore we could validate that all these algorithms are applicable to real-world supply chain problems.

In general, all algorithms except the recoverable robustness algorithm lead to a similar result: The increasing penalty costs from commodity 1 to commodity 4 are respected in the way that the percentaged delivered extra demand increases from commodity 1 to commodity 4. The results for commodity 4 slightly differ in the recoverable robustness algorithm, as commodity 4 only contributes with a small total number of units and the recovery robustness algorithm is the only one with a fixed recovery budget.

When considering increased storage costs for one commodity at all warehouses, all algorithms decrease the delivered extra demand for this commodity and all algorithms except the recoverable robustness approach either increase or keep the amount of delivered extra demand for all other commodities stable.

The increase of penalty costs for two commodities at five out of fifteen warehouses leads to an increase of delivered extra demand for these commodities at these warehouses, while the amount of delivered extra demand remains more or less stable or is decreased for the other commodities.

Two algorithms - the ant algorithm and the transformation algorithm - provide a robustness parameter, which allows to control the level of conservatism of the solution. For both algorithms, we can see that for an increasing robustness parameter, the amount of delivered extra demand increases, which corresponds to our expectations of an increasingly robust solution.

The question arises, which algorithm to choose in practice. This depends on the given situation:

The ant algorithm is suitable for large problems, as it is a fast heuristic algorithm. The additional feature of the ant algorithm, the avoidance of small flows, is unique for the ant algorithm. A disadvantage of the ant algorithm as heuristic method is the fact that, depending on the stopping criterion, not every run will result in the optimal solution and not even in the same solution.

The transformation offers the possibility to apply one of the algorithms for optimization problems with uncertain costs, for example the algorithm of Bertsimas and Sim or the ant algorithm for uncertain costs, in order to obtain an exact or a heuristic solution of the transformed problem.

Both the ant algorithm for uncertain demands and the transformation algorithm (where the corresponding optimization problem can be solved using either the ant algorithm for uncertain costs or the algorithm of Bertsimas and Sim) allow to control the level of conservatism of the solution by a robustness parameter. The other algorithms do not offer this feature.

The recoverable robustness algorithm is the only algorithm that has the feature of limited recovery (or compensation) costs. No matter which scenario of the uncertain input data is realized, the recovery costs cannot surpass the fixed recovery budget.

For scenario-based input data, the two-stage model is suitable, as it respects the given scenarios and calculates the exact solution for the considered Stochastic Programming model. It has to be considered that the expected value of the compensation costs is minimized, which means that in the worst case, these costs can be rather high. However, in case that the number of scenarios is very high, the performance of the L-shaped algorithm will be too slow. In this case, an approximation of the scenario data by interval data and the usage of one of the other algorithms is recommended.

Chapter 13

Final Conclusion and Future Research

In this thesis, we examined various approaches for modeling and optimizing supply chain problems with uncertain input data, where the main focus was on uncertain cost values and uncertain demand values. The resulting algorithms were validated on test data and a real-world supply chain network.

In Chapter 3, Section 3.4, four major challenges in supply chain management with uncertain input data were identified. In the following, it is explained how these challenges were met in this thesis:

Avoidance of small amounts of flow In Chapter 5, both an exact method and a heuristic method were developed for addressing the problem of avoiding small amounts of flow in supply chain problems. In MCFPs, the optimal solution possibly contains many small amounts of flow on diverse arcs. In practice, however, small amounts of flow are often not desired, such that a more expensive solution, where for every arc the flow is either zero or surpasses a certain threshold value, is preferred. We developed two solution methods: An exact method, most suitable for small scale supply chain problems, which is based on a Branch and Bound method with a special branching scheme, and a heuristic method that is based on the ACO metaheuristic.

This ACO based optimization concept was extended to ACO based algorithms for supply chain problems with uncertain costs, cf. Chapter 6, and algorithms for supply chain problems with uncertain demands, cf. Chapter 8. These algorithms combine ideas of the ACO metaheuristic with the concept of Robust

Optimization. For both types of these ant algorithms, it is possible to control the level of robustness via a robustness parameter. A special feature of the ant algorithm for uncertain costs is the extensibility to different probability distributions of the uncertain cost values, e.g. Gaussian distribution or triangular distribution.

Extension of existing approaches to multicommodity problems Two major existing approaches in optimization under uncertainty were extended from single-commodity to multicommodity problems. In Chapter 6, the Robust Optimization approach for network flow problems with uncertain cost values of D. Bertsimas and M. Sim ([14]) was extended such that it can now cope with more than one commodity. An optimization algorithm based on a bisection method was proposed. In contrast to the developed ant algorithms for uncertain cost values, this algorithm does not consider the distribution of the uncertainty values.

In Chapter 9, the L-shaped algorithm (see [18]), which is one of the standard algorithms to solve two-stage linear problems, was modified to exactly suit multicommodity supply chain problems with uncertain demand values. In the Stochastic Programming approach, information about the distribution of the uncertain input data is used. Correctness and convergence of the multicommodity L-shaped algorithm were proven. Furthermore, two extensions concerning recourse by interexchange and inventory management problems were developed.

In addition, the recoverable robustness model proposed by S. Stiller et al. (see [64], [91]) was extended to supply chain problems with uncertain demands in Chapter 8. The presented approach is applicable to both single-commodity and multicommodity problems. The optimization process of the recoverable robustness approach consists of a planning phase and a recovery phase. The special feature of this approach is the fact that the costs for compensation in the recovery phase are limited in advance. We showed that in practice often more demand scenarios than originally intended are covered by the optimal solution of a recoverable robustness problem. This effect is called *coincidental covering.*

In Chapter 10, a link between the recoverable robustness approach for network flow problems with uncertain demands and the two-stage model from Stochastic Programming was drawn. The two approaches are linked via a multicriteria optimization problem: The recoverable robustness problem is an ϵ-constraint scalarization of this problem, while the Stochastic Programming problem is a weighted sum formulation of the same MOP. Furthermore, it was shown that the recoverable robustness problem and the two-stage problem can be transformed into each other under certain conditions.

Inventory management strategies We proposed a concept for considering inventory management problems in Chapter 9. The process of inventory management introduces the possibility of preproduction and storage of goods in case that it is not possible to produce enough goods just in time in time periods with very high demand. Limiting factors are storage capacities and storage costs.

The proposed inventory management approach is based on Stochastic Programming. The underlying single period network is multiplied and the single networks for each time period are connected by arcs representing possible storage from one time period to the next. Cost and capacity values on the connecting arcs represent storage costs and capacities per time period.

The inventory management problem is examined in the context of uncertain demand values in all time periods. The complexity of the resulting linear optimization problem increases rapidly with the number of considered time periods and the number of demand scenarios, such that the use of special purpose algorithms like the L-shaped algorithm instead of standard linear programming algorithms are advisable.

The presented inventory management approach is applicable to both single-commodity and multicommodity problems. Furthermore, a combination with the proposed interexchange recourse scheme, see Section 9.3, is possible.

Applicability to real-world problems In Chapter 11, the heuristic ant algorithms, which were proposed in Chapters 5, 6 and 8, were evaluated numerically. For this purpose, various fully layered networks were generated at random. By means of these test runs, we examined computational time, performance and stability of the ant algorithm for the TMCFP, as proposed in Section 5.1, the ant algorithm for Gaussian distributed costs, as presented in Subsection 6.3.2 and the ant algorithm for NFPs with uncertain demands as presented in Section 8.2. Results were compared with the SOS 2 Branch and Bound method, see Section 5.3, and the CSA for the TMCFP. For the NFP with uncertain costs, results were compared with the multicommodity extension of the algorithm of Bertsimas & Sim, see Section 6.2. For all considered problem categories, the ACO metaheuristic based algorithms provide a reasonable trade-off between solution quality and computational time. Furthermore, the ant algorithms run stable for both single-commodity and multicommodity problems.

In Chapter 12, the proposed algorithms for supply chain problems with uncertain demand data were validated by means of a reference model with real-world network structure and real-world input data. In the reference model, the filling, mixing and transportation stages of a supply chain were modeled. For the demand nodes that represent the warehouses, uncertainty in the demand values was considered. Different robustness levels as well as effects of modifications of penalty and storage costs were examined. The results showed that real-world problems are computationally tractable by our algorithms. Furthermore, advantages and disadvantages of the different algorithms concerning different problem types were identified: The network size, the number of scenarios, the control of the level of robustness by means of a robustness parameter and the limitation of the compensation costs influence the selection of the most suitable algorithm.

Future Research Still there remain many topics of interest for future research. Different directions for future research are suggested:

This thesis is focused on uncertainties in cost and demand data. Another possible source of uncertainty lies in grid failure, i.e. the failure of nodes and arcs in the supply chain network. In real-world problems, production machines can be temporarily unavailable, which corresponds to a failure of a node. Furthermore, transportation arcs can be unavailable due to weather conditions or technical failure of the transportation means.

Moreover, deviations in the production or transport capacities can occur. If for example only two out of three identical production machines are available, the production capacity is decreased to two thirds of the original capacity. As different types of machines have different failure probabilities, a consideration of this type of capacity deviation can be of interest for supply chain optimization.

Another topic of interest is the modeling and design of a supply chain. Instead of optimizing an existing supply chain with uncertain input data, the possible data fluctuations can already be taken into account when designing a supply chain network, like this is done in the context of fuzzy minimum cost flow problems in [49], for example. Then failure probabilities, fluctuations of costs as well as possible production and transport volumes are regarded during the modeling process und will therefore lead to a robust network design.

In order to handle large instances of supply chain optimization problems, we have developed a heuristic algorithm based on the ACO metaheuristic. In principle, this algorithm can be applied to time expanded networks without additional changes, such that inventory management problems can be solved. However, the ant algorithm does not consider the special structure of the multiple period network. Future research could focus on finding an even more specialized algorithm for inventory management problems, which better exploits the data structure of the time expanded networks.

Acknowledgements

First of all, I would like to thank my supervisor, Prof. Dr. Kathrin Klamroth, for her support and guidance throughout the research project and especially for her invaluable suggestions during this work.

Furthermore I would like to recognize many valuable contributions from my research partner in Erlangen, Alexander Keimer. A big thank you to Petra Dächert for her advice concerning Stochastic Programming and her selfless assistance in all kinds of problems.

I thank the cooperation partners of Axxom Software AG, Dr. Mirko Eickhoff and Dr. Tobias Gerken, and Henkel KGaA, Dr. Marco Lescher and Axel Richter, for their input concerning real-world supply chain problems and relevant questions in supply chain practice. Thank you to the Bayerische Forschungsstiftung, who financially supported our common research project. I wish to express my gratitude to Prof. Dr. Günter Leugering, who enabled this research work at the Institute of Applied Mathematics at the University of Erlangen-Nürnberg.

In this context, I would like to give a special thank you to Kerstin Dächert, who proposed me as a member of the research team. Thank you to Sebastian Stiegler and my employer, the IABGmbH, for giving me the opportunity to take part in the research project.

I want to thank Sebastian Stiller for the inspiring discussions concerning recoverable robustness. Thank you also to Dr. Gabriele Eichfelder for her advice concerning Multicriteria Optimization Problems.

A heartfelt thank you to Marc Layer for introducing me into the fascinating world of rock climbing, which turned out to be the perfect method to clear my mind and to regain my strength in stressful times.

Finally I want to thank my family. The encouragement and support from my beloved husband Thorsten was a powerful source of inspiration and energy. A special thought is devoted to my parents for their never-ending support.

Höhenkirchen in 2010,
Simone Gast

List of Abbreviations

ACO	Ant Colony Optimization
B&B	Branch and Bound
bc	best case
com	commodity
CSA	Cost Scaling Algorithm
FFA	Feasible Flow Algorithms
LP	Linear Program
LRP	Linear Recovery Robust Problem
MCFP	Minimum Cost Flow Problem
MFP	Multicommodity Flow Problem
MOP	Multicriteria Optimization Problem
NFP	Network Flow Problem
OIFA	Optimal Infeasible Flow Algorithms
per	(time) period
RROP	Recovery Robust Optimization Problem
sc	scenario
SCM	Supply Chain Management
SOS2	Special Ordered Set of Type 2
TMCFP	Threshold Minimum Cost Flow Problem
wc	worst case

wh	warehouse
wlog	without loss of generality

Bibliography

[1] E. Adida and G. Perakis. A robust optimization approach to dynamic pricing and inventory control with no backorders. *Mathematical Programming*, 107(1–2):97–129, 2006.

[2] R.K. Ahuja, J.L. Batra, and S.K. Gupta. A parametric algorithm for the convex cost network flow and related problems. *European Journal of Operational Research*, 16:222–235, 1984.

[3] R.K. Ahuja, A.V. Goldberg, J.B. Orlin, and R.E. Tarjan. Finding minimum-cost flows by double scaling. Technical report, Princeton University, 1988.

[4] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[5] M. Aigner. *Diskrete Mathematik.* Vieweg, 2001.

[6] A.I. Ali, R. Padman, and H. Thiagarajan. Dual algorithms for pure network problems. *Operations Research*, 37:159–171, 1989.

[7] K.-P. Arnold. *Stochastische Transportprobleme.* Kovač, 1986.

[8] P. Balaprakash, M. Birattari, T. Stützle, Z. Yuan, and M. Dorigo. Estimation-based ant colony optimization and local search for the probabilistic traveling salesman problem. *Swarm Intelligence*, 3(3):223–242, 2009.

[9] E.M.L. Beale and J.J.H. Forrest. Global optimization using special ordered sets. *Mathematical Programming*, 10:52–69, 1976.

[10] E.M.L. Beale and J.A. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In J. Lawrence, editor, *Proceedings of the Fifth International Conference on Operations Research*, pages 447–454, London, 1970. Tavistock Publications.

[11] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Operations Research*, 23:769–805, 1998.

[12] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25:1–13, 1999.

[13] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Program*, A 88:411–424, 2000.

[14] D. Bertsimas and M.Sim. Robust discrete optimization and network flows. *Mathematical Programming*, B 98:49–71, 2003.

[15] D. Bertsimas and M.Sim. The price of robustness. *Operations Research*, 52:35–53, 2004.

[16] D. Bertsimas and A. Thiele. A robust optimization approach to supply chain management . volume 3064 of *Lecture Notes in Computer Science*, pages 145 – 156. Springer, 2004.

[17] M. Birattari, P. Balaprakash, and M. Dorigo. The ACO/F-race algorithm for combinatorial optimization under uncertainty. In *Metaheuristics*, volume 39 of *Operations Research/Computer Science Interfaces Series*, pages 189 – 203. Springer, 2007.

[18] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer, 1997.

[19] R.G. Bland and D.L. Jensen. On the computational behavior of a polynomial-time network flow algorithm. Technical report, School of Operations Research and Industrial Engineering, Cornell University, 1985.

[20] R.G. Bland and D.L. Jensen. On the computational behavior of a polynomial-time network flow algorithm. *Mathematical Programming*, 54(1):1 – 39, 1992.

[21] C. Blum. Beam-aco - hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Computers and Operations Research*, 32:1565 – 1591, 2005.

[22] J. Böttcher. *Stochastische lineare Programme mit Kompensation.* Athenäum Verlag, 1989.

[23] J. Caldeira, R. Azevedo, C.A. Silva, and J.M. Sousa. Beam-ACO distributed optimization applied to supply-chain management . In *Foundations of Fuzzy Logic and Soft Computing*, volume 4529 of *Lecture Notes in Computer Science*, pages 799 – 809. Springer, 2007.

[24] F.T.S. Chan and N. Kumar. Effective allocation of customers to distribution centres: A multiple ant colony optimization approach. *Robotics and Computer-Integrated Manufacturing*, 25:1–12, 2009.

[25] V. Chankong and Y.Y. Haimes. *Multiobjective Decision Making.* North-Holland, 1983.

[26] S.-W. Chioua. A combinatorial approximation algorithm for supply chain network flow problems. *Applied Mathematics and Computation*, 186(2):1526–1536, 2007.

[27] P. Dächert. Produktions- und Transportnetzwerke mit Bedarfsschwankungen: Eine Anwendung der Stochastischen Optimierung. Diploma thesis, University of Erlangen-Nürnberg, 2008.

[28] P. Dächert and S. Gast. Erweiterungen des Two-Stage Modells für Supply Chain Probleme. Technical report, No. 22, University of Erlangen-Nürnberg, 2009.

[29] P. Dächert, S. Gast, and A. Keimer. Modellierung und Betrachtung von Supply Chains mit schwankenden Eingangsgrößen anhand verschiedener Referenzmodelle. Technical report, No. 23, University of Erlangen-Nürnberg, 2009.

[30] U. Diwekar. *Introduction to Applied Optimization.* Springer Netherlands, 2003.

[31] K. Doerner, R.F. Hartl, and M. Reimann. A hybrid ACO algorithm for the full truckload transportation problem. Technical report, University of Vienna, 2001.

[32] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:29–41, 1996.

[33] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, 2004.

[34] R.C. Dorsey, T.J. Hodgson, and H.D. Ratliff. A network approach to a multi-facility, multi-product production scheduling problem without backordering. *Managment Science*, 21:813–822, 1975.

[35] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[36] M. Ehrgott. *Multicriteria Optimization.* Lecture Notes in Economics and Mathematical Systems. Springer, 2000.

[37] G. Eichfelder. *Adaptive Scalarization Methods in Multiobjective Optimization.* Springer, 2008.

[38] H.A. Eiselt and C.-L. Sandblom. *Integer Programming and Network Models.* Springer, 2000.

[39] L. El-Ghaoui and H. Lebret. Robust solutions to least-square problems to uncertain data matrices. *SIAM Journal on Matrix Analysis and Applications*, 18:1035–1064, 1997.

[40] L. El-Ghaoui, F. Oustry, and H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9:33–52, 1998.

[41] J.R. Evans. Some network flow models and heuristics for multiproduct production and inventory planning. *AIIE Transactions*, 9:75–81, 1977.

[42] M. Fischetti and M. Monaci. Light robustness. Technical report, ARRIVAL-TR-0119, ARRIVAL project, 2008.

[43] L. Fleischer and M. Skutella. Minimum cost flows over time without intermediate storage. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 66 – 75. Society for Industrial and Applied Mathematics, 2003.

[44] L.R. Ford and D.R. Fulkerson. Constructing maximum dynamic flows from static flows. *Operations Research*, 6:419–433, 1958.

[45] S. Gast. Applying robust optimization to network flow problems with uncertain demand. Technical report, No. 19, University of Erlangen-Nürnberg, 2008.

[46] S. Gast and A. Keimer. Robust optimization in network flows. Technical report, No. 20, University of Erlangen-Nürnberg, 2009.

[47] E. Gawrilow and M. Joswig. polymake. www.polymake.de.

[48] C. Geiger and C. Kanzow. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben.* Springer, 1999.

[49] M. Ghateea and S.M. Hashemi. Application of fuzzy minimum cost flow problems to network design under uncertainty. *Fuzzy Sets and Systems*, 160(22):3263–3289, 2009.

[50] A.V. Goldberg and R.E. Tarjan. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[51] S. Göttlich, H. Herty, C. Kirchner, and A. Klar. Optimal control for continuous supply network models. *Networks and heterogeneous media*, 1(4):675–688, 2006.

[52] H. Gunnarsson and M. Rönnqvist. Solving a multi-period supply chain problem for a pulp company using heuristics – an application to Södra Cell AB. *International Journal of Production Economics*, 116(1):75–94, 2008.

[53] W.J. Gutjahr. A Converging ACO Algorithm for Stochastic Combinatorial Optimization. In *Stochastic Algorithms: Foundations and Applications*, volume 2827 of *Lecture Notes in Computer Science*, pages 10 – 25. Springer, 2003.

[54] H. W. Hamacher and S. Tifecki. On the use of lexicographic min cost flows in evacuation modeling. *Naval Research Logistics*, 34:487–503, 1987.

[55] H.W. Hamacher and K. Klamroth. *Lineare und Netzwerk-Optimierung - Linear and Network-Optimization.* Vieweg, 2000.

[56] V. Jayaraman and A. Ross. A simulated annealing methodology to distribution network design and management. *European Journal of Operational Research*, 144(3):629–645, 2003.

[57] W.S. Jewell. Warehousing and distribution of a seasonal product. *Naval Research Logistics Quarterly*, 4:29–34, 1957.

[58] P. Kall and S.W. Wallace. *Stochastic Programming.* Wiley, Chichester, 1995.

[59] J.L. Kennington and R.V. Helgason. *Algorithms for Network Programming.* Wiley, New York, 1980.

[60] M. Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14:205–220, 1967.

[61] E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. volume 2461 of *Lecture Notes in Computer Science*, pages 49–56. Springer, 2002.

[62] P. Kouvelis and G. Yu. *Robust Discrete Optimization and its Applications.* Kluwer Academic Publishers, Amsterdam, 1997.

[63] U. Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik.* Vieweg, 2005.

[64] C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. Recoverable robustness. Technical report, ARRIVAl-TR-0066, ARRIVAL-Project, 2007.

[65] J. Linderoth and S. Wright. Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications*, 24:207–250, 2003.

[66] P.D. Loewen. *Convex Analysis with Applications.* Lecture Notes, 2001.

[67] S. Afshin Mansouri. A simulated annealing approach to a bi-criteria sequencing problem in a two-stage supply chain. *Computers & Industrial Engineering*, 50(1–2):105–119, 2006.

[68] A. Martin, M. Möller, and S. Moritz. Mixed integer models for the stationary case of gas network optimization. *Mathematical Programming*, 105:563–582, 2006.

[69] J.T. Mentzer. *Supply Chain Management.* Sage Publications, 2001.

[70] K. Miettinen. *Nonlinear Multiobjective Optimization.* Kluwer, 2004.

[71] S. Moritz. *A Mixed Integer Approach for the Transient Case of Gas Network Optimization.* PhD thesis, Technische Universität Darmstadt, 2006.

[72] J.M. Mulvey, R.J. Vanderbei, and S.A. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43:264–281, 1995.

[73] D. Naso, M. Surico, B. Turchiano, and U. Kaymak. Genetic algorithms for supply-chain scheduling: A case study in the distribution of ready-mixed concrete. *European Journal of Operational Research*, 177(3):2069–2099, 2007.

[74] G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J.Todd. *Optimization.* Elsevier, 1989.

[75] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1988.

[76] J.B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41:338–350, 1993.

[77] M. Padberg. Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters*, 27:1–5, 2000.

[78] A. Papoulis. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, 1984.

[79] W.B. Powell and H. Topaloglu. Stochastic Programming in Transportation and Logistics. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science.* Elsevier Science, 2003.

[80] A. Prékopa. Probabilistic Programming. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science.* Elsevier Science, 2003.

[81] P. Radhakrishnan, V.M. Prasad, and M.R. Gopalan. Optimizing inventory using genetic algorithm for efficient supply chain management. *Journal of Computer Science*, 5(3):233–241, 2009.

[82] H. Röck. Scaling techniques for minimal cost network flows. Discrete Structures and Algorithms, pages 181 – 191. Hanser, 1980.

[83] R.T. Rockafellar. *Convex Analysis.* Princeton University Press, 1970.

[84] A.P. Ruszczynski and A. Shapiro. Stochastic Programming Models. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science.* Elsevier Science, 2003.

[85] A. Schöbel and A. Kratz. A bicriteria approach for robust timetabling. *Robust and Online Large-Scale Optimization*, pages 119 – 144, 2009.

[86] R. Schultz, L. Stougie, and M.H. van der Vlerk. Solving stochastic problems with integer recourse by enumeration: A framework using Gröbner basis reductions. *Mathematical Programming*, 83:229–252, 1998.

[87] C.A. Silva, J.M.C. Sousa, T.A. Runkler, and J.M.G. Costa. Distributed supply chain management using ant colony optimization. *European Journal of Operational Research*, 199:349 – 358, 2009.

[88] D. Simchi-Levi, Xin Chen, and J. Bramel. *Logic Of Logistics. Theory, Algorithms, and Applications for Logistics Management.* Springer, 2004.

[89] M. Skutella. An introduction to network flows over time. Research Trends in Combinatorial Optimization, pages 451–482. Springer, 2009.

[90] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21:1154–1157, 1973.

[91] S. Stiller. *Extending Concepts of Reliability. Network Creation Games, Real-time Scheduling, and Robust Optimization.* PhD thesis, Technische Universität Berlin, 2008.

[92] T. Stützle and H.H. Hoos. Max-min ant system. *Journal of Future Generation Computer Systems*, 16:889–914, 2000.

[93] O. Thomas. Das Referenzmodellverständnis in der Wirtschaftsinformatik. *IWi – Veröffentlichungen des Instituts für Wirtschaftsinformatik im Deutschen Forschungszentrum für Künstliche Intelligenz*, 187:1–35, 2006.

[94] J.P. Vielma, S. Ahmed, and G.L. Nemhauser. Mixed-integer models for nonseparable piecewise linear optimization: unifying framework and extensions. *Operations Research*, 58(2):303–315, 2010.

[95] J.P. Vielma, A.B. Keha, and G.L. Nemhauser. Nonconvex, lower semicontinuous piecewise linear optimization. *Discrete Optimization*, 5:467–488, 2008.

[96] S.J. Wright. *Primal-Dual Interior-Point Methods.* Society for Industrial and Applied Mathematics (SIAM), 1997.

[97] C. Yildiz. *Knickminimales Orthogonales Zeichnen Planarer Graphen im Kandinsky Modell.* PhD thesis, Technische Universität Wien, 2005.

[98] C.-S. Yu and H.-L. Li. A robust optimization model for stochastic logistic problems. *International Journal of Production Economics*, 64(1–3):385–397, 2000.

[99] S. Zeiner. Vergleichende Untersuchung von mathematischen Verfahren zur Optimierung des Betriebs verfahrenstechnischer Systeme. Diploma thesis, University of Erlangen-Nürnberg, 2006.