Rheinisch-Westfälische Technische Hochschule Aachen

Lehrstuhl für Operations Research

Schleusen trotz Verspätungen. Robuste Modelle, Algorithmen und Rechenstudien

Locking with delays. Robust models, algorithms and computational studies.

Masterarbeit

von

Ilja Schwanke

Matr.-Nr. 285313

Erstgutachter: Prof. Dr. Marco Lübbecke

Zweitgutachter: Prof. Dr. Ir. Arie M.C.A. Koster

Betreuer: Dipl.-Math. Florian Dahms

Aachen, 26. September 2014

Erklärung über die selbständige Abfassung einer schriftlichen Arbeit

Hiermit versichere ich, ..., Matrikel-Nr. .....
die vorgelegte schriftliche Arbeit zum Thema
.....
.....
selbständig verfasst zu haben und ausschließlich die angegebenen Quellen und Hilfsmittel verwendet, sowie aus diesen entnommene Gedanken und Formulierungen in angemessener Form gekennzeichnet zu haben. Des Weiteren versichere ich, diese Arbeit weder in dieser noch in modifizierter Form bereits in einer anderen Lehrveranstaltung zum Erwerb eines Leistungsnachweises eingereicht zu haben. Mir ist bekannt, dass eine Arbeit, die nachweislich ein Plagiat darstellt, als schwerer Verstoß gegen die Prüfungsordnung gewertet wird und in jedem Fall als mit ungenügend bewertet gilt.

Aachen, den ....
(Unterschrift)

Inhaltsverzeichnis

1 Einführung **4**
1.1 Motivation 4
1.2 Aufbau der Arbeit 5

2 Schleusenproblem **6**
2.1 Schleusen mit festen Ankunftszeiten 8

3 Robuster Ansatz für das Schleusenproblem **12**
3.1 Robustes Schleusenproblem 12
3.1.1 Szenariodefinition 14
3.2 Kompexität des robusten Schleusenproblems 15

4 Max-Szenario-Problem **17**
4.1 Kombinatorischer Ansatz 19
4.2 Dynamisches Programm 26

5 IP-Formulierung **30**
5.1 Erweiterter max-Szenario Graph 30
5.2 IP-Formulierung mit Flusserhaltungsbedingung 34
5.3 Dualisierung und Linearisierung 38

6 Benderschnitt Ansatz **43**
6.1 Einführung Benderschnitt 43
6.2 Master- und Subproblem 44
6.3 MILP-Initialproblem 47
6.4 Master-Subproblem-Initialproblem 49

7 Benders Dekompositionsalgorithmus **52**
7.1 Allgemeine BDA-Struktur 52
7.2 BDA auf Schleusenproblem 53
7.2.1 Fall der Unzulässigkeit des Subproblems 57
7.3 Version 2 des BD-Algorithmus. 59
7.4 50/50-Version des BD-Algorithmus 62

8 Rechenstudie **67**
8.1 Implementierung 67
8.2 Daten für die Rechenstudie 67

8.3 Startlösung 68
8.4 Weitere Bedingungen für die Schiffszuweisungen 69
8.5 BDA-Implementierung 71

9 Ergebnisse und Modellanalyse **73**
9.1 Anzahl der Schiffe und Partitionen 73
9.2 bigM-Formulierung 76
9.3 Implementierung der 50/50-Variante 77
9.4 Zusammenfassung 78

10 Fazit **80**

11 Notationen **82**

1 Einführung

1.1 Motivation

Wasserwege gehören zu den bedeutendsten und effizientesten Verkehrswegen, um Güter und Personen zu transportieren. Insbesondere in der Zeit der Globalisierung weist Intensität und Komplexität der Schifffahrt eine ständige Zunahme auf. Diese sind eine wichtige Voraussetzung für die Konkurrenzfähigkeit und ein Wirtschaftswachstum einzelner Staaten und Regionen. So werden zum Beispiel über 90 % des Welthandels, fast 95% des Außenhandels der Europäischen Union und nahezu 70% des deutschen Im- und Exports über den Seeweg abgewickelt. Rund 170 Staaten betreiben weltweit etwa 90.000 Handelsschiffe. Davon werden 42.000 in der internationalen Seeschifffahrt eingesetzt, Tendenz steigend [11]. Die wirtschaftliche Bedeutung der Kreuzfahrten spielt ebenso eine immer größere Rolle. Die großen Transportmengen und die Anzahl der Schiffe führen zu der Notwendigkeit der besseren Planung der neuen Transportnetze und deren Optimierung, welche zum Beispiel im Rahmen des „Marco-Polo-II-Programms“ der EU-Kommission gefördert werden, das unter anderem sogenannte „Meeresautobahnen“ vorsieht [12].
Die Maßnahmen sind auf Effektivität und Reduzierung der Transportdauer und dadurch speziell auf Senkung der Transportkosten gerichtet. Während auf offener See große Flächen für Wasserfahrzeuge existieren, sind die Kapazitäten der Häfen sowie der Binnenwasserstraßen begrenzt. Problemstellen sind unter anderem Kanäle und insbesondere Schleusen, weil sie einen Engpass darstellen und dadurch Wartezeiten beim Passieren dieser entstehen, die finanzielle und strukturelle Konsequenzen mit sich bringen.

1.2 Aufbau der Arbeit

In der vorliegenden Masterarbeit wird ein Schleusenablauf modelliert. Das Ziel der Arbeit ist eine systematische Untersuchung und Aufstellung des Optimierungsproblems einer Schleusensteuerung und Implementierung eines Lösungansatzes basierend auf Benders Dekompositionsalgorithmus [3].
Im folgenden Kapitel werden notwendige Begriffe eingeführt und Annahmen für die Vereinfachung des Problems getroffen. Außerdem werden bekannte Ergebnisse präsentiert und Lösungsansätze aufgeführt. In Kapitel 3 wird das Zielmodel aufgebaut und das robuste Schleusenproblem diskutiert.
In Kapitel 4 wird das Max-Szenario-Problem betrachtet, unter anderem um eine Startlösung für das Hauptproblem zu erhalten, die im weiteren verbessert werden soll. Zwei mögliche Lösungswege werden vorgeschlagen und verglichen. Außerdem werden die minimalen Anforderungen an die Partitionierungen und die Zeit gegeben, die das Lösen des Problems vereinfachen sollen.
Weiter erfolgt eine Aufstellung der ganzzahligen linearen Problemformulierung, wobei das ursprüngliche Problem aus Kapitel 4 dualisiert und linearisiert wird. Benders Dekompositionsalgorithmus wird in Kapitel 6 allgemein beschrieben und dessen Ansatz als ein robustes Problem in Kapitel 7 erklärt.
Anschließend werden die Ergebnisse vorgestellt, diskutiert und Möglichkeiten für die weitere Entwicklung des Ansatzes zusammengefasst.

2 Schleusenproblem

Eine Schiffsschleuse, kurz Schleuse genannt, ist ein Bauwerk, welches Wasserfahrzeugen ermöglicht, Wasserstandunterschiede zwischen einzelnen Abschnitten einer Wasserstraße zu überwinden. Den Vorgang der Passage eines Wasserfahrzeuges durch eine Schleuse wird als Schleusung oder Schleusen bezeichnet. Als Beispiel wird eine Schleuse in einem Kanal betrachtet, die die Überwindung des Wasserstandunterschieds ermöglicht. Die zu schleusenden Schiffe kommen von beiden Seiten an, um die Schleuse zu passieren. Eine schematische Darstellung eines Schleusenprozesses ist in Abb. 1 zu sehen. Das Erstellen eines Ablaufplanes der Schleusung ist das Ziel der Untersuchung. Der Ablaufplan beinhaltet die ermittelten Zeitpunkte der Schleusung für jedes Schiff.

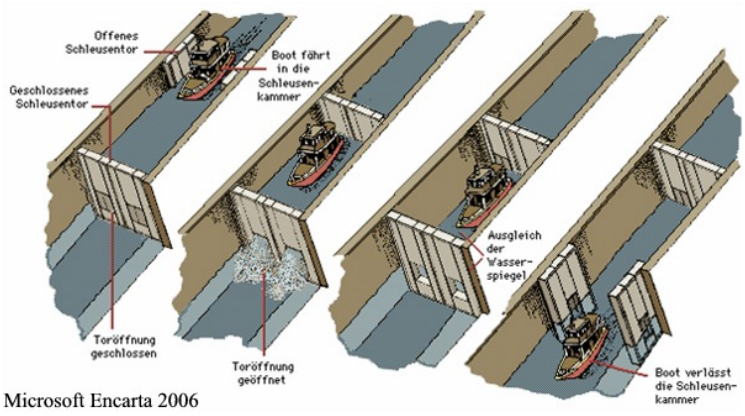

Abb. 1. Schema einer Schleuse (Quelle: Microsoft Encrata

Da das Schleusenproblem sehr umfangreich ist, werden in der vorliegenden Masterarbeit die folgenden Annahmen zur Vereinfachung des Problems getroffen: Die erste Vereinfachung des Problems bezieht sich auf die Situation, in der eine Schleuse nur aus einer Kammer besteht (s. Abb. 1). Außerdem werden am Anfang de-

terministische *Ankunftszeiten* der Schiffe an der Schleuse angenommen, d.h die Zeitpunkte $r_i \in \mathbb{R}_{\geq 0}$, mit $r_1 \leq r_2 \leq \cdots \leq r_n$, zu denen die Schiffe $i = 1, \ldots, n$ an der Schleuse ankommen, seien bekannt und fest. Im Weiteren wird die deterministische Eigenschaft aufgehoben (s. Kapitel 3). Ein weiterer wichtiger Parameter in dem Modell ist die *Dauer der Schleusung* F, d.h. die Zeitspanne zwischen dem Zeitpunkt, in dem die Schleuse schließt, so dass kein Schiff mehr hineinfahren kann und dem Zeitpunkt, wenn diese sich wieder öffnet.

Mit den gegebenen Daten wird eine Schleusungstrategie bestimmt, d.h. die Zeitpunkte, wann die Schleuse hoch- bzw. herunterfahren soll, werden ermittelt. Damit wird die Aufteilung der Schiffe auf die Schleusen festgelegt, wodurch die gesamte Wartezeit aller Schiffe minimiert wird. Die Wartezeit der einzelnen Schiffe ist als Zeitabstand zwischen Ankunftszeit an der Schleuse und dem Zeitpunkt, wann das Schiff die Schleuse verlässt, definiert.

Zusätzlich wird eine unbegrenzte Größe der Schleusenkammer angenommen, so dass beliebig viele Schiffe gleichzeitig in eine Schleuse hereinpassen und transportiert werden können. Damit wird zum Beispiel auch der Fall der zeitgleichen Ankunft mehrerer Schiffe aus einer Richtung vereinfacht; diese werden in einem Vorgang zusammen geschleust.

Da nur die Wartezeiten von Bedeutung sind, werden auch andere äußere Bedingungen vernachlässigt. So ist die verbrauchte Wassermenge nicht relevant und die monetären Kosten der einzelnen Schleusungen werden nicht berücksichtigt. Damit können beliebig viele Schleusungen im gegebenen Zeitraum durchgeführt werden, was in der Praxis nicht immer der Fall ist.

2.1 Schleusen mit festen Ankunftszeiten

In dem Abschnitt wird eine formale Darstellung des Modells entwickelt, wobei die deterministischen Ankunftszeiten der Schiffe $\{r_1, \ldots, r_n\}$ wie oben beschrieben angenommen werden. Unter der Modellprämissen lassen sich Bedingungen für die Zulässigkeit einer Schleusungstrategie schließen:

1. Die Zeitpunkte für das Hoch- bzw. Herunterfahren müssen alternieren.
2. Die Zeitpunkte für das Hoch- bzw. Herunterfahren müssen mindestens einen Abstand F haben.

Eine einfache zulässige Strategie nach diesen Bedingungen wäre zum Beispiel, die Schleuse hoch- und herunterfahren zu lassen, so dass die Schiffe transportiert werden können, die bis zu dem Start der Schleusung aus passender Richtung ankommen. Es kann in diesem Fall auch zu Leerläufen kommen. Des Weiteren wird nicht auf die Ankunft eines Schiffes gewartet. Diese triviale Strategie wird später für das Finden einer Startlösung im Algorithmus verwendet, die eine obere Schranke für die Lösung des Problems bildet.
Die Abbildung 2 soll zum besseren Verständnis des Schleusenproblems beitragen: Jedes Schiff j wird durch die Ankunftszeit r_j repräsentiert. Die obererhalb und unterhalb der horizontalen Zeitachsen liegende Punkte bezeichnen die Schiffe, die aus zwei verschieden Richtungen kommen. Diese bilden die Schiffsmengen J_1 und J_2. Die Menge J enthält alle Schiffe, womit gilt $J = J_1 \cup J_2$. Die Zeitskala verläuft horizontal, so dass Schiffe die links liegen früher als die rechts liegenden ankommen. Die Schiffe, die in einem Schleusenvorgang bearbeitet werden, liegen innerhalb einer elliptischen Fläche. Diese Menge der zusammengehörigen Schiffe wird als Partition bezeichnet. So gehört das Schiff j der Partition R_i. Jede blaue Linie stellt die Dauer des i-ten Schleusenvorgangs dar, der von t_i bis $t_i + F$ andauert. Somit ist

die Wartezeit eines einzelnen Schiffes $j \in R_i$ gleich $t_i + F - r_j$ und ist rot auf der Abbildung dargestellt. Die gesamte Wartezeit ist die Summe der Wartezeiten der Schiffe j. Es wurden im Beispiel vier Schleusenvorgänge benötigt, um 10 Schiffe passieren zu lassen.

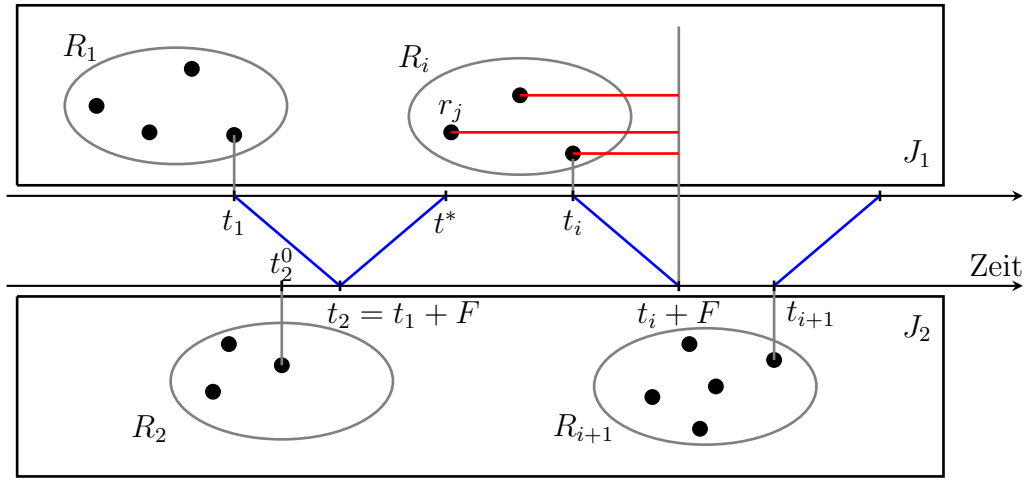

*Abb.*2 Partitionierung der Schiffe

Es lässt sich erkennen, dass obwohl alle Schiffe der Partition R_2 zu dem Zeitpunkt t_2^0 angetroffen sind, deren Schleusung erst zu dem Zeitpunkt $t_1 + F$ möglich ist, d.h erst dann, wenn der Schleusenvorgang der Partition R_1 abgeschlossen ist. Andererseits, im Fall der Partition R_i wäre die Schleusung für das Schiff j auch früher als Zeitpunkt t_i möglich, da die Schleuse schon im Zeitpunkt $t^* = t_1 + 2F$ bereitstand. Das kann eventuell auch die gesamte Wartezeit verringern, ohne Schleusezeiten der Partition R_{i+1} zu beeinflussen (nicht in Abb. 2 zu sehen). Diese Frage, wann und welche Schiffe am besten zu schleusen sind, wird durch das Lösen des deterministischen Schleusenproblems beantwortet.
Zusammenfassung der bis jetzt eingeführten Bezeichnungen und Definitionen in der formalen Beschreibung des Schleusenproblems mit festen Ankunftszeiten der Schiffe:

Deterministisches Schleusenproblem

Gegeben:

- Die Menge der Schiffe $J = J_1 \cup J_2 = \{1, \ldots, n\}$. Die Teilmengen J_1 bzw. J_2 bezeichnen jeweils die Schiffe, die aus beiden Richtungen zur Schleuse kommen.
- Die Ankunftszeit $r_j \in \mathbb{R}_{i\geq 0}$ für jedes Schiff $j \in J$.
- Die Dauer des Schleusenvorgangs $F \in \mathbb{R}_{\geq 0}$.

Finde:

- Die Menge der Partitionen $R_i \subseteq J,\ i = 1, \ldots, 2k$, mit $J = \cup R_i$, so dass $R_i \subseteq J_1$ für ungerade i und $R_i \subseteq J_2$ für gerade i gilt
- Minimiere die gesamte Wartezeit $w(R_i)$

$$w(R_i) = \sum_{i=1}^{2k} \sum_{j \in R_i} (t_i + F - r_j)$$

mit $t_i = max\{t_{i-1} + F, max_{j \in R_i} r_j\}$ die frühestmögliche Schleusezeit der Partition i und $t_0 = -F$

Es gelten die im vorigen Teil getroffene Annahmen bekannter fester Ankunftszeiten der Schiffe $\{r_1, ...r_n\}$ und unbeschränkter Schleusengröße. Für die Definition einer Partitionierung $R \in \mathcal{R}$ reicht dann aus, die Schleusungzeiten t_i für jeden Schleusenvorgang i zu fixieren ($i = 1, \ldots, 2k$ und ein passendes $k \in \mathbb{N}$, das durch $2n$ nach oben beschränkt ist). Die Anzahl der Partitionen ist somit gerade und gleich $2k \in \mathbb{N}$. Für mehr Übersicht startet die Schleuse immer auf der Seite mit hohem Wasserpegel, d.h. mit der Teilmenge der Schiffe J_1, fährt k mal herunter- und hoch, so dass die Kammer sich am Ende der Schleusung wieder in der Anfangsposition befindet. Dabei sind dazwischen auch Leerfahrten erlaubt, d.h. $R_i = \emptyset$ ist zulässig.

Die Verteilung der Schiffe auf die einzelne Partitionen wird durch eine einfache Strategie erhalten: zum Zeitpunkt t_i werden alle Schiffe geschleust, die seit der letzten Schleusung aus gleicher Seite zum Zeitpunkt t_{i-2} die Schleusenanlage erreicht haben. Die Menge der Zeiten $\{t_i\}_{i\in\{1,\dots,2k\}}$ gibt die folgende Partitionierung $R = \cup R_i$ an:

$$R_i = \{j \in J \mid t_{i-2} \leq r_j < t_i,\ j \in J_1 \text{ für } i \text{ ungerade bzw. } j \in J_2 \text{ für } i \text{ gerade}\}$$

für alle $i = 1, \dots, 2k$. Damit die Schiffe beachtet werden, die bis zum Beginn der ersten Schleusung R_1 die Schleusenanlage erreicht haben, wird $t_{-1} = -1 = t_0$ angenommen.

Ein optimaler Ablaufplan kann in diesem Fall mit Hilfe dynamischer Programmierung gefunden werden. Wichtig dabei ist die Beobachtung, dass jede durchführbare Ablaufplan die Eigenschaft erfüllt:

$$t_i \in \{r_j + b \cdot F \mid j \in \{1, \dots, n\},\ b \in \{1, \dots, n\}\},\ \forall\ i \in \{1, \dots 2k\}$$

Die Menge der möglichen Schleusungszeiten hat für alle i eine polynomielle Größe in $n = |J|$, also können die Zeiten t_i auch in polynomieller Zeit berechnet werden. Das deterministische Problem ist unter anderem bei Coene und Spieksma beschrieben [1]. Sie haben in ihrer Arbeit gezeigt, dass es ein dynamisches Programm mit einer Laufzeit von $O(n^5)$ für die Lösung des Problems existiert. Des Weiteren haben sie Erweiterungen des Problems untersucht, wie beschränkte Kapazitäten oder konstante Anzahl der Schleusen.

Die deterministischen Ankunftszeiten sind in der Praxis sehr selten; Deswegen ist der Fall viel wichtiger, wenn von vornherein nicht bekannt ist, wann genau die Schiffe die Schleuse erreichen. Die feste Aufteilung der Schiffe mit nicht festen Ankunftszeiten auf die Partitionen bildet das so genannte robuste Problem, das im nächsten Kapitel diskutiert wird.

3 Robuster Ansatz für das Schleusenproblem

In diesem Kapitel wird die Möglichkeit modelliert, worin die Ankunftszeiten der Schiffe r_j nicht bekannt sind und eine gewisse Varianz auftritt. Damit werden in der Lösung des Schleusenproblems die Verspätungen der Schiffe, die durch äußere Bedingungen, wie zum Beispiel das Wetter hervorgerufen werden, einbezogen. Die Eigenschaft der Stabilität der Lösung wird Robustheit genannt. Im Folgenden wird die Bezeichnung „robustes Schleusenproblem“ in dem Sinne verwendet, dass die aus dem Problem erhaltene Lösung stabil ist. In dem Problem treten unsichere Ankunftszeiten auf, die für alle Schiffe durch eine Menge der möglichen Szenarien $\mathcal{S}$ gegeben sind. Die Bezeichnung „Szenario“ beschreibt dabei eine Menge aller möglichen Ankunftszeitpunkte einzelner Schiffe, deren Definition von der Problemstellung und den Zielen abhängt.

3.1 Robustes Schleusenproblem

Bei einem robusten Schleusenproblem wird die Verteilung der Schiffe mit nicht deterministischen Ankunftszeiten auf die Schleusenvogänge R_i mit $i = 1, \ldots, 2k$ gesucht. Nach dem Eintreten eines Szenarios $S \in \mathcal{S}$ sind die Ankunftszeiten aller Schiffe r_j^S mit $j = 1, \ldots, n$ bekannt. Die Schleusung R_i erfolgt zu dem Zeitpunkt t_i^S, wenn alle zu der Partition gehörenden Schiffe $j \in R_i$ in dem gegebenem Szenario S zu dem Zeitpunkt angekommen sind und die Schleuse für den Vorgang bereit ist. Änderungen der Zuweisungen der Schiffe zu den Partitionen sind in diesem Rahmen nicht zulässig, d.h. dass ein Schiff immer zu dem gleichen Schleusenvorgang gehört, unabhängig von den äußeren Umständen und den tatsächlichen Ankunftszeiten anderer Schiffe. Die Änderung der Partition in der Abhängigkeit vom auftretenden Szenario kann in der so genannten recoverable robusten Version des Schleusenproblems modelliert werden, die aber nicht Teil dieser Arbeit ist.

Analog zu dem deterministischen Schleusenproblem folgt eine formale Beschreibung einer robusten Lösung des Problems:

Robustes Schleusenproblem

Gegeben:

- Menge der n Schiffe $J = J_1 \cup J_2 = \{1, \ldots, n\}$. Die Teilmengen J_1 bzw. J_2 bezeichnen jeweils die Schiffe, die aus beiden Richtungen zur Schleuse kommen.
- Menge der Szenarien $\mathcal{S}$. Jedes Szenario $S \in \mathcal{S}$ gibt Ankunftszeiten r_j^S für $j \in \{1, \ldots, n\}$ an.
- Dauer des Schleusenvorgangs $F \in \mathbb{R}_{\geq 0}$.
- Frühest mögliche Schleusungzeit $t_0^S = -F$.
- Wartezeit $w^S(R)$ für eine Partitionierung $R \in \mathcal{R}$ bei einem Szenario $S \in \mathcal{S}$:

$$w^S(R) = \sum_{i=1}^{2k} \sum_{j \in R} (t_i^S + F - r_j^S)$$

Finde:

- Zulässige Schleusenstrategie $R \in \mathcal{R}$ mit minimaler worst-case Wartezeit $\max_{S \in \mathcal{S}} w^S(R)$:

$$\min_R \max_{S \in \mathcal{S}} w^S(R)$$

Die Lösung des Problems gibt eine Partitionierung der Schiffe R und die gesamte Wartezeit aller Schiffe $\max_{S \in \mathcal{S}} w^S(R)$ an, sodass für alle mögliche Szenarien dieser Wert nicht überschritten wird. Dieser stellt damit die untere Grenze der gesamten Wartezeiten für das schlechteste Szenario der übrigen Partitionierungen $R' \in \mathcal{R} \setminus R$ dar. Es ist wichtig noch einmal zu erwähnen, dass nur die Wartezeiten des schlechtesten Falls von Bedeutung sind. Die tatsächliche Wartezeit hängt vom

eintretenden Szenario ab und kann sich für die jeweilige Partitionierung auffallend von diesem Wert unterscheiden.

3.1.1 Szenariodefinition

Die Angabe der Intervallszenarienmenge $\mathcal{S}_I$ stellt eine Möglichkeit der Szenariodefinition dar. In diesem Fall werden für jedes Schiff $j \in J$ die früheste und späteste Ankunftszeit $\underline{r}_j$ und $\overline{r}_j$ gegeben. Es ist vorab nicht bekannt, zu welchem Zeitpunkt aus diesem Intervall $[\underline{r}_j, \overline{r}_j]$ das Schiff j die Schleuse erreicht. Jedes Szenario $S \in \mathcal{S}_I$ bestimmt die Ankunftszeiten $r_j^S \in [\underline{r}_j, \overline{r}_j]$ für $j = 1, \ldots, n$, andererseits existiert für alle mögliche Werte $r_j \in [\underline{r}_j, \overline{r}_j]$ ein Szenario $S \in \mathcal{S}_I$ mit $r_j^S = r_j$. Eine graphische Darstellung des Problems mit analogen Bezeichnungen wie oben erfolgt in Abbildung drei.

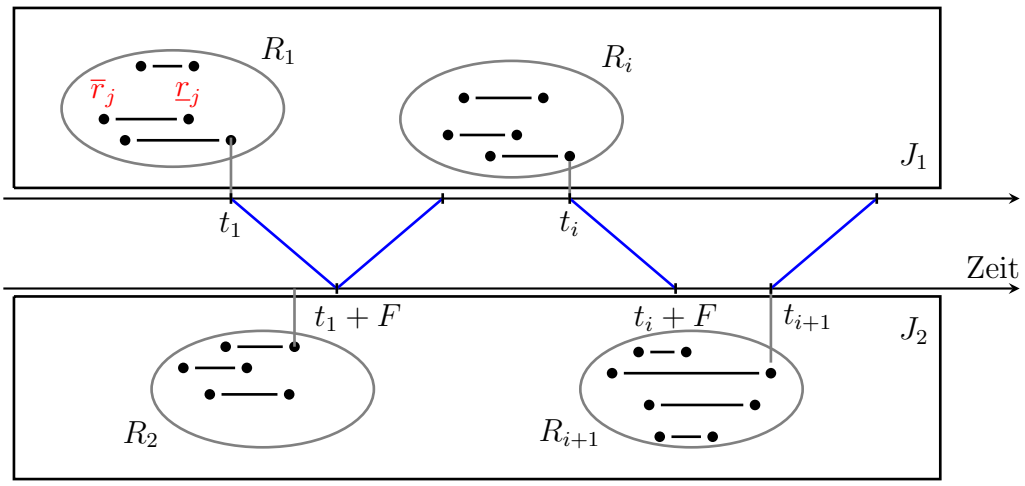

*Abb.*3 Robustes Schleusenproblem

Einzelne Schiffe werden in der Abbildung durch die Ankunftszeitintervalle bezeichnet. Unter einer Verspätung wird ein Ereignis verstanden, wenn das Schiff j

nicht zur frühestmöglichen Ankunftszeit, d.h. unterer Intervallgrenze $\underline{r}_j$, bei der Schleuse erscheint. Beispielweise erfolgt die Schleusung R_1 in der Abbildung drei zu dem Zeitpunkt t_1, wenn alle Schiffe der Partition für alle Szenarien die Schleuse erreicht haben. Das Maximumszenario mit $\min_R \max_{S \in \mathcal{S}} w^S(R)$ könnte in dem Fall eventuell auch dann auftreten, wenn alle Schiffe der Partition R_1 rechtzeitig ankommen würden, obwohl das die Summe der Wartezeiten der Partition R_1 reduzieren würde. Andererseits, auch wenn alle Schiffe der Partition zu bestimmter Zeit angekommen sind, wie im Beispiel der Partition R_2, müssen diese auf die Schleuse bis zum Zeitpunkt $t_1 + F$ warten, die noch im Prozess der vorangehenden Schleusung R_1 ist. Dies verdeutlicht, dass die Wirkung der Verspätung der Schiffe sowohl für die Schleusungzeiten eigener Partition, als auch auf die später erfolgenden Schleusungen relevant ist und untersucht werden muss.

3.2 Kompexität des robusten Schleusenproblems

Löst man das robuste Schleusenproblem exakt, kann dessen Komplexität mit dem deterministischen Fall verglichen und somit der Preis der Robustheit untersucht werden. Für das robuste Schleusenproblem gibt es eine kompakte Formulierung, d.h. polynomiell in der Anzahl der Schiffe, deren detaillierte Darlegung im Kapitel 5 erfolgt.

Bei einer gegebenen zulässigen Partitionierung ist es möglich die maximale Wartezeit in polynomieller Zeit auszurechnen. Das Problem wird Maximales-Szenario-Problem genannt, kurz max-Szenario, und kann durch das Finden des längsten Weges in einem azyklischen Graphen in $\mathcal{O}(n^2)$ (z.B. mit Dijkstra-Algorithmus) [14] gelöst werden. Daneben wird ein dynamisches Programm zum Lösen des Problems in $\mathcal{O}(n^3)$ unten aufgeführt. Außerdem gibt es eine gemischt-ganzzahlige Problemformulierung. Das robuste Schleusenproblem ist damit in der **NP**-Komplexitäts-

klasse.

Das max-Szenario-Problem wird in dem später definierten Algorithmus zur Lösung des robusten Problems benutzt ist der Gegegenstand des nächsten Kapitels.

4 Max-Szenario-Problem

Das robuste Schleusenproblem $\min_R \max_{S \in \mathcal{S}} w^S(R)$ wird vereinfacht, indem ein Teil der zu bestimmenden Parameter schon als bekannt angenommen wird. Seien eine zulässige Partitionierung $R = \{R_i\}_{i \in I}$ der Schiffsmenge $J = J_1 \cup J_2$ und die Menge der möglichen Intervallszenarien $\mathcal{S}$ gegeben. $Max\text{-}Szenario - Problem$ ist damit die Bestimmung der Wartezeiten für das schlechteste Szenario bei gegebenen Daten. D.h. die Schleusungszeiten sind so zu bestimmen, dass die gesamte Wartezeit maximiert wird.
Es folgt die formale Beschreibung des Problems:

max-Szenario Schleusenproblem

Gegeben:

- Menge der Schiffe $J = J_1 \dot{\cup} J_2 = \{1, \ldots, n\}$. Eine zulässige Partitionierung $R \in \mathcal{R}$, s.d. $R_i \subseteq J_1$ für ungerade i, $R_i \subseteq J_2$ für gerade i.
- Frühestmögliche Schleusungzeit $t_0^S = -F$.
- Szenarienmenge $\mathcal{S}$ mit $r_j^S \in [\underline{r}_j, \overline{r}_j]$, $\forall j \in J$, $S \in \mathcal{S}$.

Finde:

- Szenario $S \in \mathcal{S}$, so dass die Wartezeit $W(S)$ maximiert wird:

$$W(S) = \sum_{i=1}^{2k} \sum_{j \in R_i} (t_i^S + F - r_j^S)$$

mit $t_i^S = max\{t_{i-1}^S + F, max_{j \in R_i} r_j^S\}$.

t_i^S definiert die frühestmögliche Schleusungszeit der Partition i bei dem Szenario S, die erst dann erfolgen kann, wenn die Schleuse die vorangehende Schleusung $i - 1$

erledigt hat und alle Schiffe bei dem Szenario S zu dem Zeitpunkt die Schleuse erreicht haben. Die Szenariowartezeit $W(S)$ kann durch die Summe der Wartezeiten der einzelnen Partitionen formuliert werden:

$$W(S) = \sum_{i=1}^{2k} W(R_i),$$

wobei gilt

$$W(R_i) = \sum_{j \in R_i} (t_i^S + F - r_j^S),\ i = 1, \ldots, 2k.$$

Abbildung vier stellt eine Partition mit vier Schiffen dar. Blaue Punkte an einem Ende jedes Intervalls zeigen, wann das jeweilige Schiff ankommt. Für diese Partition ist $W(R_a)$ gleich der Summe der roten Balken, die die Wartezeiten $t_1 + F - r_j$ einzelner Schiffe $j \in J$ verbildlichen sollen.

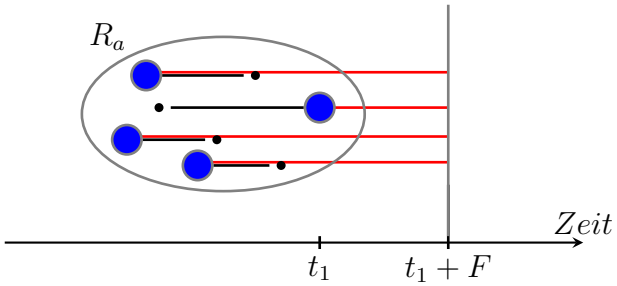

Abb. 4 Partition R_a mit Wartezeiten $W(R_a)$.

Man nennt ein Szenario $S \in \mathcal{S}$ mit dem größten Wert $W(S)$ ein *max-Szenario*. Dieses ist in polynomieller Zeit berechenbar. Eine der Möglichkeiten stellt ein graphentheoretischer Ansatz dar, der die Lösung in $\mathcal{O}(n^2)$ ermöglicht [14]. Anderenfalls kann man für das Problem ein dynamisches Programm aufstellen, das den gesuchten Wert in $\mathcal{O}(n^3)$ ausrechnet, darauf wird später eingegangen.

4.1 Kombinatorischer Ansatz

Bevor das dynamische Programm definiert wird, werden wichtige Definitionen und Eigenschaften der optimalen Lösung des max-Szenario-Problems eingeführt.

Definition 1. *Man nennt ein Schiff j für ein Szenario S verspätet, wenn $r_j^S > \underline{r}_j$ gilt, d.h. dass das Schiff nicht zu seiner möglichst frühen Ankunftszeit die Schleuse erreicht.*

Beachte, dass im Folgenden in diesem Kapitel strikt nicht deterministische Ankunftszeiten $\underline{r}_j < \overline{r}_j$ für alle $j \in J$ angenommen werden. Eine weitere Annahme besteht darin, dass jedem Schleusenvorgang mindestens 2 Schiffe gehören, d.h. $|R_i| \geq 2$ für alle $i = 1, \ldots, 2k$. Als erstes wird die Anzahl der verspäteten Schiffe für jede Partition begrenzt, dazu Lemma:

Lemma 1. *Sei S ein max-Szenario. Dann gibt es für jede Partition R_i, $i = 1, \ldots, 2k$, höchstens ein Schiff $j^* \in R_i$, das sich verspätet, d.h. es gilt $r_{j^*}^S = \overline{r}_{j^*}$ und $r_j^S = \underline{r}_j$ für alle $j \in R_i \setminus \{j^*\}$.*

Beweis. Die Ankunftszeit des j-ten Schiffes r_j^S für ein max-Szenario S nur zwei Werte aus dem gegebenen möglichen Intervall $[\underline{r}_j, \overline{r}_j]$ annehmen kann. Sie ist gleich einem der Interwalgrenzwerte $r_j^S \in \{\underline{r}_j, \overline{r}_j\}$, da die definierte Kostenfunktion $W(S)$ stückweise linear ist und anderenfalls die Wartezeit kleiner ausfällt.
Sei S ein max-Szenario. Angenommen, das Schiff $j' \in J$ aus der Partition $R_{i'}$ verspätet sich nicht (sonst wird direkt die obere Intervallgrenze $\overline{r}_{j'}$ angenommen) und kommt nicht zu frühester Ankunftszeit $\underline{r}_{j'}$, d.h. $\underline{r}_{j'} < r_{j'}^S < \overline{r}_{j'}$. Formuliere ein Szenario S^* mit $r_{j'}^{S^*} = \underline{r}_{j'}$ und $r_j^{S^*} = r_j^S$, $\forall j \in J \setminus \{j'\}$. Die Schleusungszeiten

$t_i^{S^*} = t_i^S$ definieren eine gültige Schleusungstrategie. Es gilt damit:

$$\begin{aligned}
W(S) &= \sum_{i=1}^{2k} \sum_{j \in R_i} (t_i^S + F - r_j^S) \\
&= \sum_{i=1}^{2k} \sum_{j \in R_i \setminus j'} (t_i^S + F - r_j^S) + (t_{i'}^S + F - r_{j'}^S) \\
&< \sum_{i=1}^{2k} \sum_{j \in R_i \setminus j'} (t_i^S + F - r_j^S) + (t_{i'}^S + F - \underline{r}_{j'}) \\
&= \sum_{i=1}^{2k} \sum_{j \in R_i \setminus j'} (t_i^{S^*} + F - r_j^{S^*}) + (t_{i'}^{S^*} + F - r_{j'}^{S^*}) \\
&= \sum_{i=1}^{2k} \sum_{j \in R_i} (t_i^{S^*} + F - r_j^{S^*}) = W(S^*).
\end{aligned}$$

Die Annahme ist damit widerlegt. Es bleibt die Eigenschaft zu zeigen, dass es genau ein Schiff sich verspäten darf. Angenommen es kommen zwei Schiffe j_1 und j_2 der gleichen Partition verspätet an (bei mehreren Schiffen ist gleiche Argumentation anwendbar).

Es gelte OBdA $\overline{r}_{j_1} \geq \overline{r}_{j_2}$. Ein alternatives Szenario S' wird definiert. Nach diesem kommt das Schiff j_2 früher als in dem Szenario S, d.h. zu der frühestmöglichern-Zeit an; die restlichen Daten ändern sich nicht: $r_j^{S'} = r_j^S$ für alle $j = J_1 \cup J_2 \setminus \{j_2\}$ und $r_{j_2}^{S'} = \underline{r}_{j_2}$. Die Schleusungszeit der Partition i ändert sich nicht, es gilt $t_i^S = t_i^{S'}$ durch die Wahl des verspäteten Schiffes j_1 und damit

$$
\begin{aligned}
W(S) &= \sum_{i=1}^{2k} \sum_{j \in R_i} (t_i^S + F - r_j^S) \\
&= \sum_{i=1}^{2k} \sum_{j \in R_i} (t_i^S + F - r_j^S) + (\underline{r}_{j_2} - \underline{r}_{j_2}) \\
&\overset{(a)}{=} \sum_{i=1}^{2k} \sum_{j \in R_i} (t_i^{S'} + F - r_j^{S'}) + (\underline{r}_{j_2} - \underline{r}_j^S) \\
&\overset{(b)}{\leq} \sum_{i=1}^{2k} \sum_{j \in R_i} (t_i^{S'} + F - r_j^{S'}) \\
&= W(S'),
\end{aligned}
$$

mit (a) wegen $r_{j_2}^{S'} = \underline{r}_{j_2}$ und (b) mit $\underline{r}_{j_2} < r_j^S$.
Widerspruch zur Annahme, dass S max-Szenario ist. Damit darf höchstens ein Schiff in Partition verspätet an die Schleusen ankommen. □

Nach dieser Eigenschaft kann ein Szenario S durch die Menge der verspäteten Schiffe J^S repräsentiert werden, d.h. dass ein Szenario $S(J^S)$ eindeutig die Zeiten $r_j^S = \underline{r}_j$ für $j \notin J^S$ bzw. $r_j^S = \overline{r}_j$ für $j \in J^S$ definiert. Für die Bestimmung der Schleusenreihenfolge reicht dann aus, die Menge der verspäteten Schiffe für das Problem zu kennen.

Definition 2. *Es wird angenommen, dass alle Schiffe j einer Partition R_i möglichst früh ankommen und die Schleuse zu diesem Moment für den Schleusenvorgang bereit steht. Die früheste Startzeit des Schleusens der Partition R_i ist gleich dem frühesten Zeitpunkt, zu dem alle Schiffe der Partition angekommen sind und wird mit t_i^0 bezeichnet:*

$$t_i^0 = max_{j \in R_i} \underline{r}_j.$$

Die früheste Startzeit t_i^0 ist logischerweise unabhängig von dem Szenario, da es nur von den möglichen Ankunftszeiten der Schiffe abhängt (s. Abb. 5). Da mehrere zu einer Partition gehörende Schiffe verschiedene Ankunftszeiten haben, entstehen Wartezeiten. Die Wartezeiten sind unabhängig von dem Szenario und hängen direkt mit den frühesten Startzeit einer Partition zusammen.

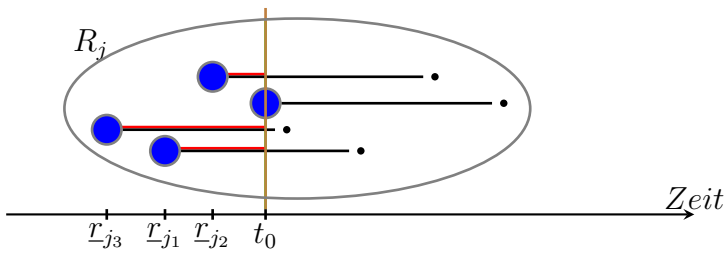

Abb. 5: Früheste Startzeit t_i^0 und unverzichtbare Wartezeit $vol(R_i)$

Definition 3. *Für jedes sinnvolle Szenario S ist die unverzichtbare Wartezeit der Schiffe einer Partition R_i gleich der Summe der Wartezeiten ab der frühestmöglichen Ankunftszeit $\underline{r}_j$, $j \in R_i$ bis zur frühesten Startzeit t_i^0 der Partition R_i :*

$$vol(R_i) = \sum_{i \in R_i} t_i^0 - \underline{r}_j \tag{4.1}$$

Folgende zwei Lemmata beschreiben die Eigenschaften der Mengen der verspäteten Schiffe bei einem max-Szenario S und deren Zusammenhang mit der frühesten Startzeit der Schleusungen. Diese Merkmale ermöglichen die Formulierung eines polynomiellen Lösungansatzes des max-Szenario-Problems, der in Kapitel 4.2 definiert wird.

Lemma 2. *Sei S ein max-Szenario für ein Schleusenproblem mit der Schiffsmenge J. Sei weiter J^S die Menge der verspäteten Schiffe in S. Außerdem gelte für zwei verspätete Schiffe $j \in R_i \cap J^S$ und $j' \in R_{i+l} \cap J^S$ die Ungleichung $\overline{r}_j + lF > t^0_{i+l}$. Dann gilt $\overline{r}_{j'} > \overline{r}_j + lF$.*

Lemma 2 besagt, dass es ein Schiff in der Partition R_{i+l} gibt, das in dem max-Szenario nicht zu seiner frühesten Ankunftszeit geschleust wird, wenn die Schleuse später als zum Zeitpunkt t^0_{i+l} für die Schleusung der Partition R_{i+l} bereitsteht.

Beweis. Angenommen das ist für ein max-Szenario S^* und zwei Schiffe j und j' nicht der Fall und es gelte $\overline{r}_{j'} \leq lF + \overline{r}_j$. Definiere ein alternatives Szenario S' mit $J^{S'} = J^{S*} \setminus \{j'\}$; die übrigen Eigenschaften werden bei S^* übernommen.
$t^S_i = \max\{\max_{j \in R_i} r^S_j | t^S_{i-1} + F\}$ bezeichnet die Schleusungzeit der Menge R_i im Szenario S. Da nach der Annahme $\overline{r}_{j'} \leq \overline{r}_j + lF$ gilt, folgt $t^{S^*}_i = t^{S'}_i$ für alle $i = 1, \ldots, 2k$. Das Verhältnis der Wartezeiten dieser zwei Szenarien lässt sich ausrechnen:

$$\begin{aligned} W(S^*) &\leq W(S^*) + t_{i+l} + F - \underline{r}_{j'} + \overline{r}_{j'} \\ &= W(S') \end{aligned}$$

$\Rightarrow$ Widerspruch zu der Annahme des max-Szenario, da die Wartezeit $W(S')$ der konstruierter Situation S' größer als $W(S*)$ wird.

□

Darüber hinaus lässt sich eine Anforderung an die späteste Ankunftszeiten der Schiffe formulieren, die verspätet in zwei verschiedenen Partitionen ankommen:

Lemma 3. *Sei S ein max-Szenario und J^S die Menge der verspäteten Schiffe in S. Sei weiter $j \in R_i \cap J^S$ und $\overline{r}_j + lF < t^0_{i+l}$ mit minimalem $l \in \mathbb{N}$. Dann existiert ein Schiff $j' \in \cup_{t=1}^{l} R_{i+t} \cap J^S$.*

Das heißt in einem max-Szenario S muss zwischen R_i und R_{i+l} ein verspätetes Schiff vorkommen. Dazu Beweis und die Abbildung 6:

Beweis. Angenommen es gäbe kein solches Schiff j' in der Menge $\cup_{t=1}^{l} R_{i+t} \cap J^S$ und sei $j^* \in R_{i+t}$ ein Schiff mit $\underline{r}_{j*} = t^0_{i+l}$. Dann kann ein Szenario S' mit $J^{S'} = J^S \cap j^*$ definiert werden, für dessen Wartezeit $W(S')$ gilt:

$$W(S') = W(S) + \overline{r}_{j*} = t^0_{i+l}(|R_{i+1}| - 1),$$

ein Widerspruch zur max-Szenario-Annahme. □

In Abb. 6 ist eine Partitionierung für neun Schiffe dargestellt. Es gilt $t^0_3 > t_1 + 2F$, wobei t_1 die späteste Ankunftszeit des verspäteten Schiffes der Partition R_1 gleich ist. Es wird versucht, für diese Partitionierung die Wartezeit zu maximieren. Nach der Aussage von Lemma 3 muss es ein verspäteter Schiff in der Partition R_2 oder R_3 geben. In dem max-Szenario alle Schiffe der Partition 2 kommen möglichst früh an, das vergrößert die Wartezeit, da die Partition nicht früher als zur Zeit t_2 geschleust werden kann. Es ist auch ersichtlich, dass die gesamte Wartezeit vergrößert wird, wenn das Schiff 3 in der Partition R_3 sich verspätet, und die Partition R_3 zu dem Zeitpunkt $t_3 = \overline{r}_j$ geschleust wird.

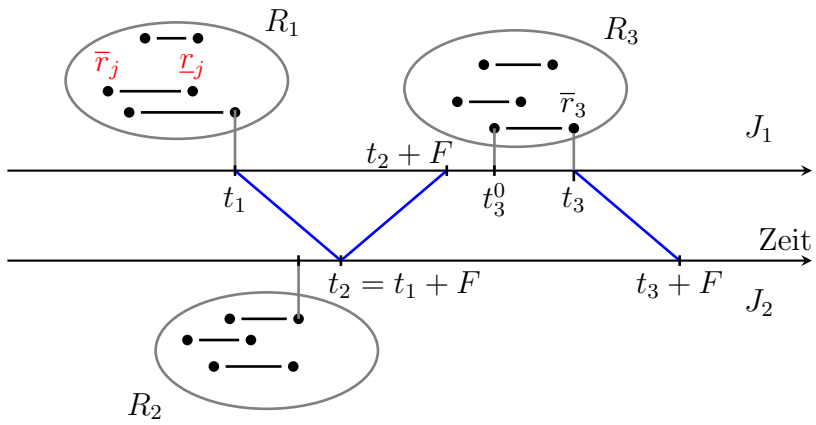

*Abb.*6. Lemma 3

Bis jetzt wurden die unverzichtbaren Zeiten der Partitionen $vol(R_i)$ unabhängig von dem Umstand betrachtet, ob die Schleuse zu dem Zeitpunkt bereit stand. Die erweiterten unverzichtbaren Wartezeiten $vol(t, R_i)$ schließen diese Lücke. Neben dem Warten auf Ankunft aller Schiffe, die zusammen geschleust werden müssen, entstehen Wartezeiten auf die freie Schleuse, die sich im Prozess der Schleusung der vorangehenden Partition befindet.

Definition 4. *Die unverzichtbare Wartezeit $vol(t, R_i)$ der Partition R_i, wenn die korrespondierende Schleusung i nicht vor dem Zeitpunkt t erfolgen kann, ist gleich der Summe der Wartezeiten der Schiffe ab dem frühest möglichen Ankunftszeit bis zu dem frühesten Zeitpunkt, wenn beide Bedingungen erfüllt sind: alle Schiffe sind angekommen, d.h. die früheste Startzeit t_i^0 überschritten ist und der gegebene Zeitpunkt t erreicht wurde:*

$$vol(t, R_i) = \sum_{j \in R_i} (max\{t, t_i^0\} - \underline{r}_j) \tag{4.2}$$

Die Definition vier kann erweitert werden, in dem das verspätete Schiff der Partition festgelegt wird.

Definition 5. *Die unverzichtbare Wartezeit $vol_j(t, R_i)$ der Partition R_i, wenn die korrespondierende Schleusung i nicht vor dem Zeitpunkt t erfolgen kann und das Schiff $j \in R_i$ das einzige verspätete Schiff in der Partition R_i ist, wird auf folgende Weise definiert:*

$$vol_j(t, R_i) = \sum_{j' \in R_i \setminus \{j\}} (max\{t, t_i^0\} - \underline{r}_{j'}) + max\{t, t_i^0, \overline{r}_j\} - \overline{r}_j \tag{4.3}$$

Die Wartezeit $vol_j(t, R_i)$ ist analog zu dem Wert $vol(t, R_i)$. Nur der Summand $max\{t, t_i^0\} - \underline{r}_j$ für die Wartezeit des Schiffes j wird in $max\{t, t_i^0, \overline{r}_j\} - \overline{r}_j$ transformiert, da die Schleusung nicht früher als die angenommene Ankunftszeit $\overline{r}_j$ des Schiffes j stattfinden kann.

4.2 Dynamisches Programm

Mit diesen Erkenntnissen ist es möglich eine Rekursionsformel für das max-Szenario-Problem zu definieren. Betrachte ein max-Szenario Problem mit gegebener Schiffsmenge J und Partitionierung $R = \{R_1, .., R_{2k}\}$; die mögliche Ankunftszeiten der Schiffe sind durch Intervalle $[\underline{r}_j, \overline{r}_j]$, $\forall j \in J$ vorgegeben. Der Wert $W(t, i)$ findet dabei Wartezeiten der Partitionen R_i für das schlechteste Szenario bei den unten bestimmten Bedingungen.

Dynamisches Programm:

$$W(t,i) = \begin{cases} \max\left\{ vol(t, R_i), \max_{j \in R_i} vol_j(t, R_i) \right\} + F & \text{wenn } i = 2k, \\ \max\left\{ vol(t, R_i) + W(\max\{t, t_i^0\} + F, i+1), \right. & \\ \qquad \left. \max_j \{ vol_j(t, R_i) + W(\max\{t, t_i^0, \overline{r}_j\} + F, i+1) \} \right\} + F & \text{anderenfalls.} \end{cases}$$

Lemma 4. *Sei eine zulässige Partitionierung $R = \{R_1, \dots r_{2k}\}$ gegeben. Der resultierende Wert $W(t,i)$ ist gleich der maximalen Wartezeit eines Szenarios für die gegebenen Partitionen $\{R_s\}_{s=1,\dots,R_{2k}}$ unter der Bedingung, dass die i-te Schleusung nicht vor dem Zeitpunkt t starten kann ($t_i \leq t$).*

Beweis. Der Beweis erfolgt durch Induktion nach $i \in \mathbb{N}$.

Induktionsanfang : Da nach Satz 1 das Szenario mit der höchsten Wartezeit unter allen Möglichkeiten gefunden ist, gilt die Aussage für $i = 2k$ und alle $t \geq 0$, was die Basis der Induktion darstellt.

Induktionsschritt : Für den Induktionsschritt nimmt man an, dass die Aussage für ein bestimmtes $i \leq 2k$, $i \in I$ und alle $i' \in I$ mit $i' \geq i$ gilt.
Betrachte die Partition R_{i-1} und ein $t \geq 0$. Nach Lemma 1 kommt bei einem max-Szenario höchstens ein Schiff in der Partition verspätet an. Wenn keines der Schiffe sich verspätet, hat die Partition R_i die Wartezeit $vol(t, R_{i-1})$. Die nächste Schleusung fängt damit nicht früher als zu dem Zeitpunkt $\max\{t, t_{i-1}^0\} + F$ an und das rekursive Aufrufen des $W(\max\{t, t_{i-1}^0\} + F, i)$ ergibt die maximale Wartezeit für alle folgende Schleusungsschritte, wenn der nächste Vorgang nicht vor dieser Zeit starten kann. Die Summe der beiden gibt die maximale Wartezeit an, wenn

keines der Schiffe verspätet ankommt.
Wenn das Schiff $j \in R_{i-1}$ sich verspätet, kann die nächste Schleusung nicht vor der Zeit $\max\{t, t^0_{i-1}, \overline{r}_j\}+F$ starten und eine analoge Berechnung geführt werden kann. Am Ende liefert die Verspätungssituation mit größter Wartezeit die maximal mögliche Wartezeit für die Partitionen $R_i, \ldots, R_{2k}$, wenn die Schleusung der Partition R_{i-1} nicht vor der Zeit t erfolgen kann. Damit ist die Richtigkeit der Induktionsaussage für $i-1$ gezeigt.

Nach dem Prinzip der vollständigen Induktion gilt die Aussage für alle $s \leq i$, $k \in \mathbb{N}$. □

Das dynamische Programm kann man für die Lösung des max-Szenario Problems anwenden.

Satz 1. *Das Max-Szenario-Problem kann in polynomieller Zeit durch dynamische Programmierung gelöst werden.*

Beweis. Mit den Ergebnissen aus Lemma 4 kann das Aufrufen des Wertes $W(0,i)$ für die Berechnung der maximalen Wartezeit eines Szenarios genutzt werden. Man speichert alle verspäteten Schiffe während der rekursiven Kalkulation. Nach der Konstruktion werden nur $W(t,i)$ mit $t \in \{a \cdot \underline{r}_j + b \cdot \overline{r}_j + c \cdot F |\ a,b \in \{0,1\}$ und $j,c \in \{1,\ldots,n\}$ ausgegeben, was höchstens $\mathcal{O}(n^3)$ mögliche Argumente ergibt (da die Anzahl der benötigten Schritte in n linear ist) [1]. □

Das max-Szenario-Problem stellt ein Teilproblem im weiter entwickelten Algorithmus dar. Die Aufteilung der Schiffe auf die Partitionen erfolgt für die Startlösung

heuristisch und durch das Lösen des definierten Masterproblems im weiteren Verlauf des Algorithmus. Dazu mehr in Kapitel 6.

5 IP-Formulierung

Im robusten Ansatz für das Schleusenproblem werden für die Partitionierungsmengen $R \in \mathcal{R}$ die maximal möglichen Wartezeiten für ein $S \in \mathcal{S}$ vergliechen. Die Lösung des robusten Schleusenproblems ist dann eine Partitionierung R mit minimaler Wartezeit $W(R, S)$ für das schlechteste Szenario S im Sinne von Zeitaufwand. Das Ziel dieses Kapitels ist eine Definition des Schleusenproblems als minmax nichtlineares ganzzahliges Problem zu finden. Sie wird als IP-Formulierung (IP, integer program) bezeichnet. Die Abhängigkeit der gesamten Wartezeit von einer Partitionierung R und eines möglichen Szenarios S muss dabei veranschaulicht werden. Die resultierende Zielfunktion soll damit die Form haben:

$$\min_{R \in \mathcal{R}} \max_{S \in \mathcal{S}} W(R, S)$$

Diese wird später durch Dualisierung und Linearisierung in ein lineares ganzzahliges Optimierungsproblem (ILP, integer linear program) modifiziert, wobei ein max-Szenario-Graph zwecks IP-Formulierung definiert und gebraucht wird.
In dem Minimierungsteil des Problems wird die Zuordnung von Schiffen zu Partitionen in Matrix $X = \{x_{ij}\}_{ı \in I, j \in J}$ fixiert und die dazugehörende minimale Wartezeit auf einem erweitertem max-Szenario Graphen $G = (V, A)$ bestimmt. In dem Maximierungsteil wird nach dem längsten Weg in dem erstellten max-Szenario-Graphen gesucht.

5.1 Erweiterter max-Szenario Graph

Ein erweiterter max-Szenario-Graph G für ein Schleusenproblem $P(I, J)$ mit n Schiffen und $2k$ Partitionen (d.h. $|I| = 2k$, und $|J| = n$) stellt einen gerichteten Schichtengraph dar (s. Abb. 6). Dieser Graph besteht aus $2k$ Schichten, dabei

enthält jede Schicht $n + 1$ Knoten: einen Partitionsknoten und n so genannte Schiffsknoten für jedes Schiff des Problems. Die Partitionsknoten modellieren den Gebrauch der Partitionen, die Schifsknoten die Zuweisung der Schiffe zur jeweiligen Partition. Überdies enthält der Schichtengraphen einen zusätzlichen Partitionsknoten $2k + 1$. Die Schiffsknoten werden als ij mit $i \in \{1, \ldots, 2k\}$, $j \in J$ bezeichnet, die Partitionsknoten werden mit i markiert, $i \in \{1, \ldots, 2k + 1\}$.
Der Schichtengraph enthält die Kanten iji', die jeden Schiffsknoten ij mit allen Partitionsknoten i' mit $i' \geq i + 1$ verbindet. Des Weiteren ist jeder Partitionsknoten $i \in I$ mit allen Knoten ij, $\forall j \in J$ verbunden.
Nach der Konstruktion ist dieser Graph azyklisch und folglich ist der längste Weg in polynomieller Zeit berechenbar [14].

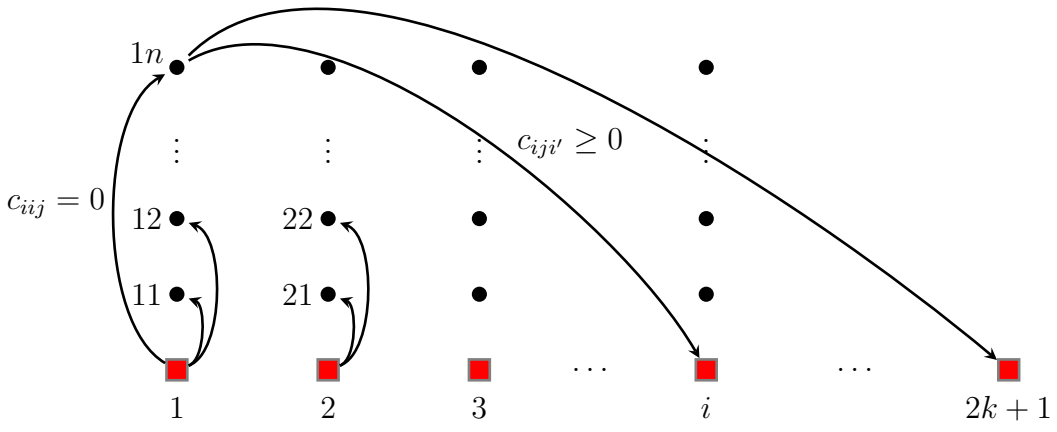

Abb. 6. Schichtengraph G.

Für eine zulässige Partitionierung $\{x_{ij}\}_{ij}$ der Schiffe $j \in J$ zu Partitionen $i \in \{1, \ldots, 2k\}$ sind die Kantenkosten c, die Wartezeiten modellieren, folgendermaßen beschrieben:

- Alle Kanten (i, ij) bekommen Kosten von null zugewiesen:

$$c_{iij} = 0,\ \forall i \in I,\ \forall j \in J.$$

  Diese Kanten zeigen an, welches der Schiffe j in der Partition i sich verspätet. Diese Aktion erzeugt keine Kosten. Ferner sind diese Kosten für alle Zuordnungen gleich, also fest im betrachteten Model.

- Kosten der Kanten (iji') für $i' \geq i+1$ und $i' \in \{1, \ldots, 2k\}$ werden mit M groß genug definiert und werden im Weiteren bigM-Bedingungen genannt:

$$c_{iji'} \geq \sum_{g=0}^{i'-i-1} \left(\sum_{j' \in J} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) x_{(i+g)j'} \right) - (\overline{r}_j - \underline{r}_j) x_{ij} - M(1 - x_{ij}) \tag{5.1}$$

Diese Ungleichung ist mit der Gleichung erfüllt, soweit die rechte Seite nicht negativ ist. Sonst wird die allgemeine Eigenschaft $c_{iji'} \geq 0, \forall i \in I,\ i' \geq i+1,\ j \in J$ in dem Problem vorgegeben. Die Gleichheit gilt, wenn $(1 - x_{ij})$ ungleich null ist, was bei $x_{ij} = 0$ erfüllt ist, also:

- Ist Schiff j nicht zur Partition i zugeordnet, d.h. $j \notin R_i$, dann gilt $c_{iji'} = 0$, $\forall i \in I,\ i' \geq i+1, j \in J, i' \in \{1, \ldots 2k\}$

Die Kosten einer Kante $c_i ji'$ hängen damit nur von der Partitionierung und nicht von den Kosten anderen Kanten ab. Für die restlichen Kanten sollen folgende Kosteneigenschaften erfüllt werden:

- Die Verspätung des Schiffes $j \in J$ in Partition $R_l, l \in I$ hat die Auswirkungen auf die Schleusungszeiten aller Partitionen $i, \ldots, i'-1$ mit:

$$\overline{r}_j + (l-i)F \geq \max_{j' \in R_l} \underline{r}_{j'},\ \forall l \in I,\ i \leq l \leq i'-1, \tag{5.2}$$

wobei $R_l = \{j \in J \mid x_{lj} = 1\}$. Demzufolge enthält $c_{iji'}$ die Wartezeiten, die durch die Verspätung des Schiffes j in der Partitionen i bis $i' - 1$ verursacht werden.

Der Beweis für die Bedingung (5.2) an die Kosten wird hier skizziert:

Beweis. Die gesamte Wartezeit w_i aller zur Partition i gehörenden Schiffe ist gleich

$$\begin{aligned} w_i &= \underbrace{\sum_{j' \in R_i,\ j' \neq j} (\overline{r}_j + F - \underline{r}_{j'})}_{Wartezeit\ aller\ Schiffe\ nach\ j} + \underbrace{F}_{Wartezeit\ von\ j} \\ &= \sum_{j \in R_i} (\overline{r}_j + F - \underline{r}_{j'}) - (\overline{r}_j + \underline{r}_j) \end{aligned}$$

Die Wartezeit w_{i+1} aller Schiffe der Partition $i + 1$ ist gegeben durch:

$$w_{i+1} = \sum_{j' \in R_{i+1}} (\overline{r}_j + 2F - \underline{r}_{j'}) \tag{5.3}$$

Dies gilt ebenfalls für alle $i' \geq i + 2$. Summiert man die Werte für beliebig viele Partitionen, so erhält man:

$$c_{iji'} = \sum_{g=0}^{i'-i-1} \left(\sum_{j' \in R_{i+g}} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) \right) - (\overline{r}_j - \underline{r}_j) \tag{5.4}$$

Der Wert $M(1 - x_{ij})$ und das Produkt von $(\overline{r}_j + \underline{r}_j)$ mit der Variable x_{ij} werden von diesem Zwischenergebnis abgezogen, damit die in der Summe über alle $j' \in R_i$ die für das Schiff j beachteten Kosten abgezogen werden, in der Summe für die systematische Schreibweise beachtet wurden. Somit ist die Behauptung bewiesen. □

Für den erstellten, gerichteten bewerteten Graphen wird der längste mögliche Weg gesucht. Die Längen der einzelnen Kanten sind gleich den berechneten Kosten. Man kann ihn mit Hilfe des Algorithmus in $O(\overline{n}^2)$ finden, mit einer Anzahl von $\overline{n}$ Knoten in G [14].

5.2 IP-Formulierung mit Flusserhaltungsbedingung

In dem ursprünglichen Schleusenproblem ist die Verteilung der Schiffe auf die Partitionen nicht bekannt, weswegen das Modell in der Form nicht ganz zu dem betrachtetem Fall passt. Es lässt sich aber in eine Flusserhaltungsformulierung für einen in diesem Kapitel beschriebenen Graphen G transformieren, das weitere Untersuchung ermöglicht.

Die Größe des Flusses wird zu 1 gesetzt, da nur der längste Weg gesucht wird. Zu dieser Flussstärke wird eine minimale Lösung, mit den Kosten der Kanten, die oben definiert wurden, gesucht. Die Verteilung der Schiffe auf die Schleusenvorgänge ist ebenso Teil der Lösung. Die Kosten c sind damit nicht fest, sondern hängen von der Verteilung x ab und sind durch $\theta^T_{iji'}$ gegeben. Der Zusammenhang von Schiffspartitionierung und entsprechenden Kosten wurde in den bigM-Bedingungen (5.2) beschrieben und hier für jede mögliche Partitionierung über $\theta_{iji'}$ symbolisch dargestellt.

Die Formulierung des Flusserhaltungsproblems für einen Schichtengraphen $G = (V, A)$ mit der gesuchten Partitionierungsmatrix $x = \{x_{ij} \mid i \in I, j \in J\}$ und der Kantenkostenmenge $c = \{c_{iji'} | i \in \{1, \ldots, 2k\, j \in J, i' \geq i + 1, i' \in \{1, \ldots 2k + 1\}\}$ sieht dann folgendermaßen aus:

Flusserhaltungsproblem

$$\min_{x,c} \max c^T f \tag{1}$$

$$\sum_{a\in\delta^-(v)} f_a - \sum_{a\in\delta^+(v)} f_a = \begin{cases} -1, & \text{wenn } v = 1, \\ 1, & \text{wenn } v = 2k+1, \ \forall v \in V \\ 0, & \text{sonst.} \end{cases} \tag{2}$$

$$c_{iji'} \geq \theta^T_{iji'} x \qquad \forall i \in \{1,\dots,2k\}, j \in J, i' \geq i+1 \tag{3}$$

$$\sum_{i\in\{1,\dots,2k\}} x_{ij} = 1 \qquad \forall j \in J \tag{4}$$

$$x_{ij} = 0 \qquad \forall j \in J_1, \ i \text{ gerade} \tag{5}$$

$$x_{ij} = 0 \qquad \forall j \in J_1, \ i \text{ ungerade} \tag{6}$$

$$f_a \geq 0 \tag{7}$$

$$c_{iji'} \geq 0 \qquad \forall i \in \{1,\dots,2k\}, j \in J, \ i' \geq i+1, \ i' \in \{1,\dots,2k+1\} \tag{8}$$

$$x \in \{0,1\}. \tag{9}$$

Die Kosten des Flusses werden in dem Problem minimiert, wobei die Kantenkosten c so minimal gewählt werden, dass in der Lösung die Bedingungen (3) mit Gleichheit erfüllt sind. Die Bedingungen in (2) stellen den Flusserhalt sicher. In (4) erfolgt eine eindeutige Zuordnung der Schiffe zu den Partitionen. Nichtnegativitätsbedingungen werden für den Fluss f in (7) und für die Kosten $c_{iji'}$ für $i \in \{1,\dots,2k\}, \ j \in J, \ i' \geq i+1, \ i' \in \{1,\dots,2k+1\}$ in (8) eingebracht. Wie bei der Definition des erweiterten Schichtengraphen erwähnt wurde, sind die meisten Kosten gleich Null; diese Tatsache ist in die Bedingung (3) eingeflossen.

Aus dem Verlauf eines Weges in dem Graphen G kann damit eine zulässige Parti-

tonierung der Schiffe für das Schleusenproblem abgeleitet werden. Es werden dabei die Partitionen R_i belegt, wenn der Weg in G durch die Partitionsknoten i verläuft. Des Weiteren kann die Menge der verspäteten Schiffe und deren Verteilung auf die Partitionen aus den Schiffsknoten abgelesen werden. Verläuft der Weg über den Knoten ij, so ist j das einzige verspätete Schiff der Partition R_i.

Der längste $(1, 2k+1)$-Weg P (d.h. ein Pfad von Knoten 1 zu Knoten $2k+1$) in dem aufgestellten Graphen $G = (V, A)$ entspricht dem max-Szenario S des Schleusenproblems bei gegebenen Partitionierung. Die Länge des Weges P ist gleich der gesamten Wartezeit dieses Szenarios. Dazu der nächste Satz.

Satz 2. *Die Flusserhaltungsformulierung modelliert das robuste Schleusenproblem.*

Beweis. Sei p^{max} der längste $(1, 2k+1)$-Weg in G und sei $(i_x j_x, i_{x+1})$ eine Kante in p^{max}. Als erstes wird die Tatsache gezeigt, dass $r_{j_{i_x}} + (l - i_x)F \geq \max_{R_l} \underline{r}_j$ für alle $i_x \leq l \leq i_{x+1} - 1$ erfüllt ist. Dabei definiert $r_{j_{i_x}}$ die Ankunftszeit des Schiffes j, das zu der Partition i bei der Verteilung x gehört.

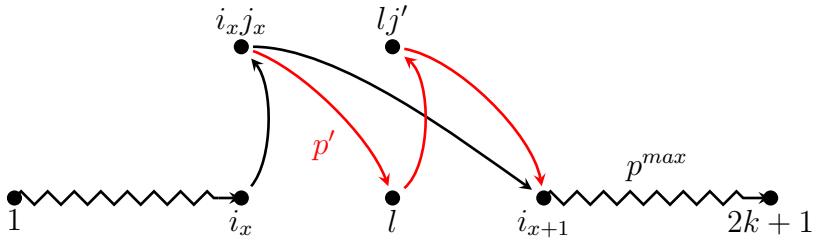

Abb. 7. Wegtransformation

Die obige Aussage wird durch einen Widerspruch gezeigt und für den Beweis des Hauptsatzes verwendet.

Zwischenbeweis: Angenommen das ist nicht der Fall und es gibt einen Knoten l mit $i_x \leq l \leq i_{x+1} - 1$ und $j' \in R_l$, so dass $\underline{r}_{j'} > \overline{r}_{j_{i_x}} + (l - i_x)F$. Betrachte den Weg $p' = p^{max}_{|[1,i_x j_{i_x}]}(i_x j_{i_x}, l)(l, lj')(lj', i_x + 1)p^{max}_{|[i_{x+1},2k+1]}$.
Für die Zuordnung x wird gezeigt, dass p' ein längerer Weg als p^{max} ist. Die Wege p^{max} und p' unterscheiden sich dabei nur in Kanten $c_{i_x j_x l}$, $c_{lj' i_{x+1}}$ und $c_{i_x j_x i_{x+1}}$, wie in Abbildung 7 zu sehen ist (schwarz ist der längste Weg p^{max}, mit rot wurde der alternative Weg markiert. Es gilt weiterhin:

$$c(p') - c(p^{max}) = c_{i_x j_x l} + c_{lj' i_{x+1}} - c_{i_x j_{i_x} i_{x+1}}.$$

Für die Kosten $c_{i_x j_{i_x} i_{x+1}}$ folgt dann:

$$\begin{aligned}
c_{i_x j_{i_x} i_{x+1}} &= \sum_{g=0}^{i_{x+1}-i_x-1} \left(\sum_{j \in R_{i_x+g}} (\overline{r}_{j_{i_x}} + (g+1)F - \underline{r}_j) \right) - (\overline{r}_{j_{i_x}} - \underline{r}_{j_{i_x}}) \\
&= \sum_{g=0}^{l-i_x-1} \left(\sum_{j \in R_{i_x+g}} (\overline{r}_{j_{i_x}} + (g+1)F - \underline{r}_j) \right) - (\overline{r}_{j_{i_x}} - \underline{r}_{j_{i_x}}) \\
&+ \sum_{g=l-i_x-1}^{i_{x+1}-i_x-1} \left(\sum_{j \in R_{i_x+g}} (\overline{r}_{j_{i_x}} + (g+1)F - \underline{r}_j) \right) \\
&\overset{(a)}{=} c_{i_x j_x l} + \sum_{g'=0}^{i_{x+1}-l-1} \left(\sum_{j \in R_{l+g'}} (\overline{r}_{j_{i_x}} + (l - x_i)F + (g'+1)F - \underline{r}_j) \right) \\
&\overset{(b)}{<} c_{i_x j_x l} + \sum_{g'=0}^{i_{x+1}-l-1} \left(\sum_{j \in R_{l+g'}} (\underline{r}_{j'} + (g'+1)F - \underline{r}_j) \right) \\
&\overset{(c)}{\leq} c_{i_x j_x l} + \sum_{g'=0}^{i_{x+1}-l-1} \left(\sum_{j \in R_{l+g'}} (\overline{r}_{j'} + (g'+1)F - \underline{r}_j) \right) - (\overline{r}_{j'} - \underline{r}_{j'}) \\
&= c_{i_x j_x l} + c_{lj' i_{x+1}}
\end{aligned}$$

mit (a) substituiert $g = g' + l - i_x$,

(b) da $j' \in R_l$ und $\underline{r}_j > \overline{r}_{j_x} + (l - i_x)F$,

(c) da $\overline{r}_j \geq \underline{r}_j$ für alle $j \in J$.

Die Rechnung ergibt $c_{i_x j_{i_x} i_{x+1}} < c_{i_x j_x l} + c_{l j' i_{x+1}}$ und damit entsprechend ist der neue Weg länger: $c(p^{max}) < c(p')$, was zu zeigen war. Es ist ein Widerspruch zu der Annahme des längsten Weges p^{max}. Ende Zwischenbeweis.
Die Flusserhaltungsformulierung beschreibt damit den längsten möglichen Weg für das Problem.
Es wird die Gleichwertigkeit der zwei Formulierungen festgestellt: Es kann höchstens ein Schiff aus der Partition verspätet ankommen. Im Graphen G verläuft ein $(1, \ldots, 2k+1)$-Weg durch höchstens einen Knoten ij aus der Schicht i. Wenn ein Schiff sich verspätet, dann ist es möglich die maximale Wartezeit über das Längste- Weg-Problem zu modellieren: die maximale Wartezeit kann dann partitioniert werden in die durch das Schiff j in der Partition i induzierte Wartezeit in den Partitionen i bis i', in der das nächste Schiff verspätet ankommt.
Benutzt man diese Eigenschaften und die oben beschriebenen Beobachtungen über das Max-Szenario-Problem für das robuste Schleusenproblem, dann folgt die Behauptung des Satzes. □

5.3 Dualisierung und Linearisierung

Die Zielfunktion des Flusserhaltungsproblems ist in einer nicht linearen Form $\min \max c^T f$, was die Lösung des Systems erschwert. Die Umformulierung in reines Minimierungs- oder Maximierungsproblem ist damit erstrebenswert. In diesem Kapitel wird ein reines Minimierungsproblem für das robuste Schleusenproblem gebildet. Dazu wird zunächst das innere Maximierungsproblem des Flusses $\max c^T f$ bei bekannten Kosten c betrachtet:

Primales Problem

$$\max \sum_{i\in\{1,\dots,2k\}} \sum_{i'\in\{1,\dots,2k+1\}} \sum_{j\in J} c_{iji'} f_{iji'} \tag{5.5}$$

$$\sum_{j\in J} f_{11j} \leq 1 \qquad \underline{d:\pi_1}$$

$$\tag{5.6}$$

$$\sum_{i'\leq i-1} \sum_{j\in J} f_{i'ji} - \sum_{j\in J} f_{iji} = 0 \quad \forall i \in \{1,\dots,2k\}\backslash\{1,2k+1\}. \quad \underline{d:\pi_i}$$

$$\tag{5.7}$$

$$f_{iij} - \sum_{i'\geq i+1} f_{iji'} = 0 \qquad \forall i \in \{1,\dots,2k\}, j \in J. \quad \underline{d:\pi_{ij}}$$

$$\tag{5.8}$$

$$\sum_{i'\leq i-1} \sum_{j\in J} f_{i'j(2k+1)} \leq 1. \qquad \underline{d:\pi_{2k+1}}$$

$$\tag{5.9}$$

$$f_{iji'} \geq 0 \qquad \forall j \in J, i \in \{1,\dots,2k\}, i' \geq i+1$$

$$\tag{5.10}$$

$$f_{iij} \geq 0 \qquad \forall i \in \{1,\dots,2k\}, j \in J.$$

Die Bedingungen (5.5) bis (5.8) sichern den Flusserhalt und enthalten nur die Variablen $f_{iji'}, \forall\, j \in J, i \in \{1,\dots,2k\}, i' \geq i+1$. Die Knoten-Kanten-Inzidenzmatrix f des Problems ist als Inzidenzmatrix eines gerichteten Graphen unimodular. Damit kann das innere Maximierungsproblem dualisiert werden, ohne die Menge der zulässigen Lösungen zu ändern [5]. Nach den Eigenschaften der Dualisierung, für die nichtnegativen Variablen $f_{iji'}$ in dem primalen LP entsprechen die Ungleichungen (12), (13) in dem dualen LP, den Ungleichungen (5.5) und (5.9) − nichtnegative Variablen π_1 und π_{2k+1} den Gleichungen (5.6) und (5.7) − nicht vorzeichenbe-

schränkte Variablen π_i, $i \in \{1, \ldots, 2k\}$ und π_{ij}, ı $\in \{1, \ldots, 2k\}$, $j \in J$. Das duale lineare Programm (LP) sieht dann dementsprechend aus:

Duales Problem

$$\min \pi_{2k+1} + \pi_1 \tag{5.11}$$

$$-\pi_i + \pi_{ij} \geq 0 \qquad \forall i \in \{1, \ldots, 2k\}, j \in J \tag{5.12}$$

$$-\pi_{ij} + \pi_{i'} \geq c_{iji'} \qquad \forall i \in \{1, \ldots, 2k\}, j \in J, i' \in \{1, \ldots, 2k+1\}, i` \geq i+1 \tag{5.13}$$

$$\pi_1, \pi_{2k+1} \geq 0 \tag{5.14}$$

$$\pi_i, \pi_{ij} \in \mathbb{R} \qquad \forall i \in \{1, \ldots, 2k\} \backslash \{1\}, j \in J \tag{5.15}$$

Man ersetzt das Maximierungsproblem $\sum c^T f$ in dem ursprünglichen min-max-Problem der IP-Formulierung mit dem erhaltenen dualen Minimierungsproblem $\min \pi_{2k+1} + \pi_1$. Desweiteren werden die von der Θ abhängigen Kostenbedingungen mit den Ergebnissen für Kantenkosten des Schichtengraphs substituiert. Es kommen die natürlichen Zuweisungsbedingungen $x_{ij} = 0$ für jedes $j \in J_1$ bei geradem $i \in I$, bzw. $x_{ij} = 0$ für alle $j \in J_2$, wenn i ungerade ist. Damit werden die unmöglichen Zuweisungen vornherein gleich null gesetzt und damit ausgeschlossen. So bekommt man eine gemischt-ganzzahlige lineare Problemformulierung (mixed integer linear problem, kurz MILP) mit reiner Minimierungszielfunktion:

Robustes Schleusenproblem, MILP-Formulierung

$$\min_{x,c} \pi_{2k+1} + \pi_1$$

$$-\pi_i + \pi_{ij} \geq 0 \qquad \forall i \in \{1,\dots,2k\}, j \in J$$

$$-\pi_{ij} + \pi_{i'} \geq c_{iji'} \qquad \forall i \in \{1,\dots,2k\}, j \in J,$$
$$i' \in \{1,\dots,2k+1\}, i' \geq i+1$$

$$\sum_{g=0}^{i'-i-1} \left(\sum_{j' \in J} (\overline{r}_j + (g+1)F - \underline{r}_j) x_{(i+g)j'} \right)$$

$$-(\overline{r}_j - \underline{r}_j) x_{ij} - M(1 - x_{ij}) \leq c_{iji'} \qquad \forall i \in \{1,\dots,2k\}, j \in J,$$
$$i' \geq i+1, i' \in \{1,\dots,2k+1\}$$

$$\sum_{i \in \{1,\dots,2k\}} x_{ij} = 1 \qquad \forall j \in J$$

$$x_{ij} = 0 \qquad \forall j \in J_1, i \text{ } gerade$$

$$x_{ij} = 0 \qquad \forall j \in J_2, i \text{ } ungerade$$

$$c_{iji'} \geq 0 \qquad \forall i \in \{1,\dots,2k\}, j \in J,$$
$$i' \in \{1,\dots,2k+1\}, i' \geq i+1$$

$$\pi_1, \pi_{2k+1} \geq 0$$

$$\pi_i, \pi_{ij} \in \mathbb{R} \qquad \forall i \in \{1,\dots,2k\} \backslash \{1\}, j \in J$$

$$x \in \{0,1\}$$

Diese MILP-Formulierung für das robuste Schleusenproblem stellt die Basis für die weitere Optimierung. Die binären Variablen erhöhen die Komplexität des Problems. Es wird im Weiteren untersucht, wie man die Lösungsberechnung beschleunigen und eventuell die Problematik der Binarität umgehen kann. Als mögliche Methode wird das Bendersschnitt-Verfahren zur Lösung des Problems gewählt,

das Problem dabei exakt gelöst wird. Es wird dabei erwartet, dass die Lösungsdauer dadurch geringer ausfällt als die Dauer der direkten Lösung des originalen MILP-Problems. Die Beschreibung des Verfahrens erfolgt im nächsten Kapitel.

6 Benderschnitt Ansatz

Benders Dekompositionsalgorithmus ist eine gängige Methode schwierige gemischt-ganzzahlige lineare Probleme zu lösen [2][3]. In analogen Problemen bringt es deutliche Zeitersparnisse [13], was auch für das robuste Schleusenproblem erwartet wird. In den nächsten zwei Kapiteln folgt die Beschreibung des iterativen Benderschnitt-Ansatzes für die Lösung des im vorigen Kapitel definierten MILP-Formulierung des Schleusenproblems, Analyse dessen Effektivität und $Bottle-Neck$-Stellen.

6.1 Einführung Benderschnitt

In der Einführung erfolgt die Definition eines Benderschnitts anhands des Beispiels eines gemischt-ganzzahliges Problems MIP (mixed integer program), das mit der Notation aus [6] mit ganzzahligem Variablenvektor x und reelle Variablenvektor y die Form hat:

MIP

$$
\begin{aligned}
z^* := \min c^T x + d^T y \\
s.d.\ Fx \leq g, \\
Mx + Ay \geq b, \\
Dy \geq e, \\
x_j \in \{0,1\}, j \in B, \\
y_i \in \mathbb{R}, i \in G,
\end{aligned}
$$

Die ganzzahligen Variablen x werden *complicating* Variablen genannt, da sie die

Lösung des Problems besonders erschweren. Um MIP zu vereinfachen, wird es geteilt und die Werte der Variablen x werden für ein Teilproblem als bekannt angenommen. Die Aufteilung des Problems erfolgt in das so genannte *Master-* und *Subproblem*. Die Bezeichnungen werden [8] entnommen.

6.2 Master- und Subproblem

Man transformiert das MIP-Probelm so, dass die neue Zielfunktion nur von x abhängt. Das geschieht wie folgt:

Alternative Formulierung vom MIP

$$\min_x c^T x + \alpha(x) \tag{6.1}$$

so dass

$$Fx \leq g, \tag{6.2}$$

$$x_j \in \{0,1\}, j \in B, \tag{6.3}$$

wobei

$$\alpha(x) = \min_y d^T y, \tag{6.4}$$

so dass

$$Ay \geq b - Mx, \tag{6.5}$$

$$Dy \geq e, \tag{6.6}$$

$$y_i \in \mathbb{R}, i \in G. \tag{6.7}$$

Das neue System ist äquivalent zum MIP, da eine Aufteilung von einem x und y abhängigen Minimierungsproblem auf zwei Minimierungsprobleme stattfindet.

Das erste beinhaltet die Zeilen (6.1)-(6.3), ist nur von der Variablen x abhängig und wird das Masterproblem genannt. Das zweite bezeichnet man als Subproblem; diese wird durch die Zeilen (6.4)-(6.7) definiert und hängt von den Variablen x und y ab. Die Relation von x und y ist in der Bedingung (6.5) festgelegt.
Die Funktion $\alpha(x)$ im MIP gibt den Wert der optimalen Lösung des Subproblems (6.4)-(6.7) an. Die Eigenschaften der Funktion $\alpha(x)$, unter anderem Konvexivität der Wertebereiche, sind in [8] beschrieben.

Unter der Annahme der gegebenen Werte der *complicating* Variablen $x_j^{(k)}$, $\forall j \in B$ wird ein Subproblem gebildet. Das Problem ist einfacher als das MIP, da alle Variablen stetig sind. Es stellt außerdem eine Instanz des Originalproblems dar, da die Lösungsbereich durch die als Parameter gegebene Variablen $x_j^{(k)}$, $\forall j \in B$ stärker beschränkt ist.

Subproblem im k-ten Iterationsschritt

$$\min_y d^T y, \tag{6.8}$$

so dass:

$$Ay \geq b - Mx^{(k)} \tag{6.9}$$

$$Dy \geq e \tag{6.10}$$

$$x_j = x_j^{(k)} \qquad : \underline{\lambda}_j \qquad \forall j \in B \tag{6.11}$$

$$y_i \in \mathbb{R}, i \in G \tag{6.12}$$

Die Lösung des Subproblems sind die Werte $y_i^{(k)}$, $\forall i \in G$. Die optimalen Werte der dualen Variablen, die mit der Gleichung (6.11) assoziiert sind, in denen die Festsetzung der Werte der Variablen x_j, $\forall j \in B$ erfolgt, werden mit $\lambda_j^{(k)}$, $\forall j \in B$

bezeichnet. Mit den neu berechneten Werten ist es möglich, eine erweiterte Version des Problems (6.1)-(6.3) der alternativen Formulierung des MIP aufzustellen:

Masterproblem im k-ten Iterationsschritt

$$\min_{x,\alpha} c^T x + \alpha, \tag{6.13}$$

so dass:

$$Ay^{(k)} + \lambda^{(k)}(x - x^{(k)}) \leq \alpha,\ k = 1, \ldots, t \tag{6.14}$$

$$Fx \leq g, \tag{6.15}$$

$$x_j \in \{0, 1\}, j \in B \tag{6.16}$$

$$\lambda \in \mathbb{R}^{|B|} \tag{6.17}$$

$$\alpha \geq \alpha^{down} \tag{6.18}$$

In dem Masterproblem wird der Wert $c^T x + \alpha$ minimiert, wobei α eine Variable und α^{down} ihre untere Grenze ist, die passend zu dem Problem angenommen werden darf. Die Variablen x im Masterproblems sind am Anfang weniger als in der MIP-Formulierung beschränkt, dadurch wird die MIP-Lösung mit dem optimalen Wert des Masterproblems von unten approximiert.
Es kommt eine neue Bedingung (6.14) hinzu, die als Benderschnitt genannt wird. Dieser schneidet eine nicht optimale Lösung ab, die vorher in dem Subproblem auf die Optimalität geprüft wurde. Das Masterproblem wird mit dem Einfügen neuer Benderschnitte besser eingegrenzt; diese werden durch erneuten Aufrufen des Subproblems mit definiert.
Für den Zusammenhang zwischen der Zulässigkeit und Optimalität der beiden Probleme siehe [9]; hier werden nur einige wichtige Eigenschaften ausgeführt. Sei $P(x, y)$ ein MIP-Problem wie oben Formuliert. Für dessen Masterproblem $MP(x)$

und Subproblem $SP(x^*, y)$, wobei x^* eine aus einer MP erhaltene Lösung ist, gilt Folgendes:

Eigenschaft 1 : Masterproblem $MP(x)$ unzulässig $\Rightarrow$ $P(x, y)$ unzulässig.

Eigenschaft 2 : Sei x^* eine optimale Lösung des Problems $P(x, y)$:

- Subproblem $SP(x^*)$ hat eine Lösung $y^* \Rightarrow (x^*, y^*)$ optimale Lösung von $P(x, y)$.
- Subproblem $SP(x^*)$ unzulässig $\Rightarrow$ x^* unzulässig für $P(x, y)$.

Für den Fall der Unzulässigkeit können auch Unzulässigkeitsschnitte hinzugefügt werden, wobei in diesem Fall die unzulässigen anstatt der nicht optimalen Lösungen abgeschnitten werden [16].

6.3 MILP-Initialproblem

Im Laufe des letzten Kapitels wurde die MILP-Formulierung für ein robustes Schleusenroblem mit $|J| = n$ Schiffen und $|I| = 2k$ Partitionen gefunden. Das Problem wird in eine Master-Subproblem-Darstellung umformuliert, wobei die ganzzahligen, bzw. binären Variablen $x = \{x_{ij}\}_{i \in I, j \in J}$ als *complicating* Variablen gewählt werden und die übrigen π und c Variablen reel bzw. positiv sind (mit Mengen $c = \{c_{iji'} \mid \forall i \in \{1, \ldots, 2k\},\ j \in J,\ i' \geq i + 1,\ i' \in \{1, \ldots, 2k + 1\}\}$ und $\pi = \{\pi_i |\ \forall i \in I\} \cup \{\pi_{ij} |\ \forall I \in I,\ j \in J\}$). Dann sieht das MIP-Problem in der Standardform entsprechend aus:

Alternative Formulierung für das Schleusenproblem

$$\min_{x} \quad \sum 0x_{ij} + \alpha(x) \tag{6.19}$$

$$\sum_{i\in\{1,\dots,2k\}} x_{ij} = 1 \qquad \forall j \in J \tag{6.20}$$

$$x_{ij} = 0 \qquad \forall j \in J_1, i \text{ } gerade \tag{6.21}$$

$$x_{ij} = 0 \qquad \forall j \in J_2, i \text{ } ungerade \tag{6.22}$$

$$x \in \{0,1\} \tag{6.23}$$

$$\alpha(x) = \min_{\pi,c} \ \pi_{2k+1} + \pi_1 \tag{6.24}$$

$$\tag{6.25}$$

$$\sum_{g=0}^{i'-i-1} \left(\sum_{j'\in J} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) x_{(i+g)j} \right)$$

$$-(\overline{r}_j - \underline{r}_j)x_{ij} + Mx_{ij} - c_{iji'} \le M \qquad \forall i \in \{1,\dots,2k\}, i' \ge i+1,$$

$$i' \in \{1,\dots,2k+1\}, j \in J \tag{6.26}$$

$$-\pi_i + \pi_{ij} \ge 0 \qquad \forall i \in \{1,\dots,2k\}, j \in J \tag{6.27}$$

$$-\pi_{ij} + \pi_{i'} - c_{iji'} \ge 0 \qquad \forall i \in \{1,\dots,2k\}, j \in J, i` \ge i+1,$$

$$i' \in \{1,\dots,2k+1\} \tag{6.28}$$

$$\pi_1, \pi_{2k+1} \ge 0$$

$$\pi_i, \pi_{ij} \in \mathbb{R} \qquad \forall i \in \{1,\dots,2k\}\backslash\{1\}, j \in J$$

$$c_{iji'} \ge 0 \qquad \forall i \in \{1,\dots,2k\}, i' \ge i+1$$

$$i' \in \{1,\dots,2k+1\}, \ j \in J$$

In der *bigM*-Ungleichung (6.26) des Initialproblems kommen reellwertige und *complicating* Variablen vor. Hier wird analog zu der Ungleichung (6.5) der Zusammenhang zwischen den Variablen ausgedrückt. Diese Bedingungen werden nach der Aufteilung in dem Subproblem genutzt.

6.4 Master-Subproblem-Initialproblem

Das Initialproblem wird wie oben für das Beispielmodel beschreiben in zwei Probleme aufgeteilt. Die Bedingungen, die nur von *complicating* Allokationsvariablen x abhängen, gehören zu dem Masterproblem. Die Ungleichungen, in denen reellwertige Variablen vorkommen, gehen in das Subproblem herein.
Für die Lösung des Subproblems werden die Werte der Zuweisungvariablen $x_{ij}^{(t)}$ als bekannt angenommen. Nach dieser Aufteilung entspricht das Subproblem der oben beschriebenen max-Szenario-Formulierung. Entsprechend wird hier zu der gegebenen Partitionierung x die schlechteste Situation der Ankünfte der Schiffe gefunden, man löst also das Subproblem $\min_{\pi,c} \pi_{2k+1} + \pi_1$ nach reellen Variablen π und c, wobei die dualen Variablen λ ebenso gefunden werden:

Subproblem

$$\alpha(x) = \min_{\pi,c} \pi_{2k+1} + \pi_1$$

$$\begin{aligned}
\pi_i - \pi_{ij} &\leq 0 && \forall i \in \{1,\dots,2k\}, j \in J \\
\pi_{ij} - \pi_{i'} - c_{iji'} &\leq 0 && \forall i \in \{1,\dots,2k\}, j \in J, i' \in \{1,\dots,2k+1\}, i` \geq i+1
\end{aligned}$$

$$\begin{aligned}
-c_{iji'} \leq M - \sum_{g=0}^{i'-i-1} \Bigg(\sum_{j' \in J} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) x_{(i+g)j'} \Bigg) \\
+ (\overline{r}_j - \underline{r}_j) x_{ij} - M x_{ij} \qquad \forall i \in \{1,\dots,2k\}, j \in J, i' \geq i+1
\end{aligned}$$

$$\begin{aligned}
\pi_1, \pi_{2k+1} &\geq 0 \\
\pi_i, \pi_{ij} &\in \mathbb{R} && \forall i \in \{1,\dots,2k\} \backslash \{1\}, j \in J \\
c_{iji'} &\geq 0 && \forall i \in \{1,\dots,2k\}, j \in J, i \in \{1,\dots,2k+1\}, i' \geq i+1 \\
x_{ij} = x_{ij}^{(0)} &: \underline{\lambda_{ij}^{(0)}} && i \in \{1,\dots,2k\}, j \in J.
\end{aligned}$$

Die Lösung $\pi_{2k+1}^{(0)} + \pi_1^{(0)} \in \mathbb{R}$ und die Menge der dualen Variablen λ werden an das Masterproblem weitergegeben. Als zweite Teilaufgabe ist ein Masterproblem in Abhängigkeit von *complicating* Variablen x mit der Zielfunktion $\min_x \sum 0 x_{ij} + \alpha(x)$ zu lösen; Die erzeugten Benderschnitte sollen die nicht optimale Lösung $x^{(0)}$ mit dem das Subproblem gelöst wurde, von dem beachteten Bereich der möglichen Lösungen abschneiden:

Masterproblem

$$\min_x \sum 0x_{ij} + \alpha(x) \tag{6.29}$$

$$\sum_{i\in\{1,\dots,2k\}} x_{ij} = 1 \qquad \forall j \in J \tag{6.30}$$

$$x_{ij} = 0 \qquad \forall j \in J_1, i \; gerade \tag{6.31}$$

$$x_{ij} = 0 \qquad \forall j \in J_2, i \; ungerade \tag{6.32}$$

$$\pi_{2k+1} + \pi_1 + \sum_{i,j} \lambda_{ij}^{(0)}(x_{ij} - x_{ij}^{(0)}) \leq \alpha \tag{6.33}$$

$$x \in \{0,1\} \tag{6.34}$$

Im Unterschied zum Subproblem, das immer die gleiche Anzahl der Variablen und Bedingungen besitzt, ist die Anzahl der Bedingungen im Masterproblem nicht fest, da die Benderschnitte (6.33) für jede nicht optimale Lösung erschaffen werden können.

Zusammenfassung: Nach dieser Aufteilung werden die Probleme separat gelöst. Beide Probleme sind einfacher als die ursprüngliche MILP-Formulierung, da sie jeweils von weniger Variablen abhängen. Mit der Lösung des Subproblems werden die Daten für einen Benderschnitt erhalten, der in das Masterproblem hinzugefügt wird.
Diesen rekursiven Ansatz für das Problem benutzt man im Benders Dekompositionsalgorithmus, der im nächsten Kapitel erläutert wird.

7 Benders Dekompositionsalgorithmus

Benders Dekompositionsalgorithmus ist ein rekursiver, auf dem Benderschnitt-Ansatz aufbauende Algorithmus. Er stellt in jeder Iteration die obere und untere Schranke für die Lösung des Problems auf. Durch die Schließung der Lücke zwischen den Schranken wird der optimale Wert gefunden [8].
Im Weiteren wird die Bezeichnung BDA oder BD-Algorithmus verwendet. Die Master-und Subproblemdarstellung des MIP-Problems von oben stellt die Eingabe für den Algorithmus dar. Im Laufe des Algorithmus wird abwechselnd nach den Lösungen beider Teilprobleme gesucht, bis die Gleichheit deren Zielfunktionen erreicht ist; dabei sind die *complicating* Variablen für das Schleusenproblem von besonderem Interesse, da diese die Zuweisungen der Schiffe darstellen.

7.1 Allgemeine BDA-Struktur

Die allgemeine Vorgehensweise bei Benders Dekompositionsalgorithmus sieht folgendermaßen aus: Man löst abwechselnd die Master- und Subprobleme. Nach jeder Iteration findet eine Konvergenzkontrolle der erhaltenen Zielfunktionswerte statt. Wenn die Konvergenzbedingung nicht erfüllt ist, erzeugt man eine zusätzliche Nebenbedingung für die erhaltene nicht optimale Lösung und fügt diese in das Masterproblem ein. Die formale Struktur sieht wie folgt aus:

Benders Dekompositionsalgorithmus

Input: Ein MIP-Problem mit *complicating* Variablen x und Variablen y, ein kleiner Wert $\epsilon \in \mathbb{R}$ für die Konvergenzkontrolle.

Output: Die Lösung des Problems: Werte der Variablen x, y

Schritt 0: Initialisierung

Führe Schritte 1 bis 3 durch, bis Konvergenzbedingung in 2 erfüllt ist:

Schritt 1: Lösung des Subproblems
Schritt 2: Konvergenzkontrolle
Schritt 3: Lösung des Masterproblems

Es kann auch der Fall mehrerer Subprobleme mit den dualen $\lambda_{i,s}^{(t)}$ vorkommen [8]. Der Fall ist besonders effizient, weil die so erhaltenen kleineren Probleme viel einfacher als das große Subproblem zu lösen sind. Mit den erhaltenen dualen Variablenwerten wird analog zum allgemeinen Fall ein Benderschnitt gebildet, der diese einzelnen Teilprobleme beachtet und zum Masterproblems im nächsten Schritt hinzugefügt wird.
Außerdem muss die mögliche Unzulässigkeit des Subproblems beachtet werden [8]. Dafür wird eine immer zulässige Formulierung des Subproblems entwickelt, oder ein Unzulässigkeitsschritt in das Masterproblem eingefügt, der diese unzulässige Lösung des Masterproblem des vorigen Schrittes abschneidet [16].

7.2 BDA auf Schleusenproblem

Der BD-Algorithmus soll auf das Schleusenproblem MILP anwendet werden. Die Aufteilung in ein *Sub-* und ein *Masterproblem* wurde oben gezeigt. In diesem Kapitel wird die Vorgehensweise in Schritten null bis drei ausführlicher beschrieben.

Schritt 0: Initialisierung

Initialisiere den Zähler $v = 1$. Löse das Masterproblem:

$$\min_x \sum 0x_{ij} + \alpha \tag{7.1}$$

$$x_{ij} = 0 \qquad \forall j \in J_1, i \; gerade \tag{7.2}$$

$$x_{ij} = 0 \qquad \forall j \in J_2, i \; ungerade \tag{7.3}$$

$$x \in \{0, 1\} \tag{7.4}$$

$$\alpha \geq \alpha^{down} \tag{7.5}$$

Die Problemzielfunktion ist unabhängig von Variablen x. Damit sind binäre $x_{ij}^{(1)}$ soweit frei wählbar, sodass die Bedingungen (7.2) und (7.3) erfüllt bleiben. Wie die passende Startpartitionierung eingestellt wird, wird später in dem Teil über die Startlösungen diskutiert. Folgend gilt $\alpha^{(1)} = \alpha^{down}$. Für ein jeweiliges Problem wird α^{down} sinnvoll angenommen und als bekannt in die Eingabe geschrieben.

Schritt I: Lösung des Subproblems

Nachdem die optimale Lösung des Masterproblems gefunden wurde, löse mit den Werten $x_{ij}^{(v)}$ das Subproblem:

$$\min_{\pi,c} \pi_{2k+1} + \pi_1$$

$$\pi_i - \pi_{ij} \leq 0 \quad \forall i \in \{1,\ldots,2k\}, j \in J.$$

$$\pi_{ij} - \pi_{i'} - c_{iji'} \leq 0 \quad \forall i \in \{1,\ldots,2k\}, j \in J,$$
$$i' \in \{1,\ldots,2k+1\}, i' \geq i+1.$$

$$\sum_{g=0}^{i'-i-1} \left(\sum_{j' \in J} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) x_{(i+g)j'} \right)$$
$$-(\overline{r}_j - \underline{r}_j)x_{ij} + Mx_{ij} - c_{iji'} \leq M \quad \forall i \in \{1,\ldots,2k\}, j' \in J,$$
$$i' \geq i+1, i' \in \{1,\ldots,2k+1\}.$$

$$\pi_1, \pi_{2k+1} \geq 0$$

$$\pi_i, \pi_{ij} \in \mathbb{R} \quad \forall i \in \{1,\ldots,2k\}\backslash\{1\}, j \in J.$$

$$c_{iji'} \geq 0 \quad \forall i' \in \{1,\ldots,2k\}, j' \in J,$$
$$i \in \{1,\ldots,2k+1\}, i \geq i'+1.$$

$$x_{ij} = x_{ij}^{(v)} : \lambda_{ij} \quad i \in \{1,\ldots,2k\}, j \in J.$$

Die Lösung des Problems sind die Werte $\pi_{ij}^{(v)}, \pi_i^{(v)}, c_{iji'}^{(v)}$ und der dualen Variablen $\lambda_{ij}^{(v)}$ für alle $i \in \{1,\ldots,2k\}$, $j' \in J$, $i' \geq i+1$, $i' \in \{1,\ldots,2k+1\}$.

Schritt II: Konvergenzkontrolle

Man vergleiche obere und untere Schranken der objektiven Funktion der MILP-Formulierung, für die man die Lösungen des Master- bzw. Subproblems einsetzt.

Die obere Schranke der objektiven Funktion des Originalproblems ist gleich dem

Zielfunktionswert des Subproblems:

$$z_{up}^{(v)} = \sum 0x_{ij}^{(v)} + \pi_{2k+1}^{(v)} + \pi_1^{(v)}$$

Die entsprechende untere Schranke des optimalen Wertes der objektiven Funktion des Originalproblems ist der aktuelle Wert des Masterproblems:

$$z_{down}^{(v)} = \sum 0x_{ij}^{(v)} + \alpha^{(v)}$$

Gilt es $z_{up}^{(v)} - z_{down}^{(v)} < \epsilon \Rightarrow$ STOP. Eine optimale Lösung ist für die geforderte Genauigkeit durch $x_{ij}^{(v)}$ und $\pi_{ij}^{(v)}, \pi_i^{(v)}, c_{iji'}^{(v)}$, für alle $i \in \{1, \ldots, 2k\}$, $j' \in J$, $i' \geq i+1$, $i' \in \{1, \ldots, 2k+1\}$ gegeben.

Anderenfalls gehe wieder zur Lösung des neuen Masterproblems über, wobei man eine zusätzliche Benderschnitt-Bedingung in das Masterproblem einfügt.

Alternativ kann der Abstand $\pi_{2k+1}^{(v)} + \pi_1^{(v)} - \alpha^{(v)}$ überprüft werden und analog über die Notwendigkeit des nächsten Schrittes entschieden werden.

Schritt III: Lösung des Masterproblems

Wenn die Konvergenzbedingung nicht erfüllt ist, wird zu dem Masterproblem ein neuer Bender-Schnitt hinzugefügt und das System nach α und x gelöst, wobei der Wert α^{down} während des letzten Aufrufs des Masterproblems geändert wurde. Also werden drei Teilschritte ausgeführt:

1. Aktualisiere $(v) \leftarrow (v+1)$

2. Löse das Masterproblem:

$$\min_{x,\alpha} \sum 0 x_{ij} + \alpha$$

$$\pi_{2k+1}^{(t)} + \pi_1^{(t)} + \sum_{i,j} \lambda_{ij}^{(t)} (x_{ij} - x_{ij}^{(t)}) \leq \alpha \qquad t = 1, \ldots, v-1$$

$$\sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J$$

$$x_{ij} = 0 \qquad \forall j \in J_1, i \; gerade$$

$$x_{ij} = 0 \qquad \forall j \in J_2, i \; ungerade$$

$$x \in \{0, 1\}$$

$$\alpha \geq \alpha^{down}$$

3. Mit der gefundenen Lösung $x_{ij}^{(v)}$ und dem neuen Wert $\alpha^{down} = \alpha^{(v)}$ gehe zu Schritt I.

Die Schritte (I)-(III) des BD-Algorithmus werden so lange wiederholt, bis die Konvergenzbedingung erfüllt ist. Damit wird die optimale Lösung der Schleusenproblems gefunden und die Partitionierung $x_{ij}^{(v)}$ und die dazugehörige gesamte Wartezeit $\alpha^{(v)}$ ausgegeben.

7.2.1 Fall der Unzulässigkeit des Subproblems

Falls der Fall der Unzulässigkeit des Subproblems vorkommen sollte, gibt es eine alternative Darstellung des Problems, die immer zulässig ist:

$$\min_{\pi,c} \pi_{2k+1} + \pi_1 + \overline{M} \sum_{i,j,i'} (v_{iji'} + w)$$

$$\pi_i - \pi_{ij} = 0 \quad \forall i \in \{1,\dots,2k\}, j \in J.$$

$$\pi_{ij} - \pi_{i'} - c_{iji'} = 0 \quad \forall i \in \{1,\dots,2k\}, j \in J,$$
$$i' \in \{1,\dots,2k+1\}, i \geq i'+1.$$

$$\sum_{g=0}^{i'-i-1} \left(\sum_{j' \in J} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) x_{(i+g)j'} \right)$$
$$-(\overline{r}_j - \underline{r}_j) x_{ij} + M x_{ij} - c_{iji'} + v_{iji'} - w = M \quad \forall i \in \{1,\dots,2k\}, j \in J,$$
$$i' \geq i+1, i' \in \{1,\dots,2k+1\}$$

$$\pi_1, \pi_{2k+1} \geq 0$$

$$\pi_i, \pi_{ij} \in \mathbb{R} \quad \forall i \in \{1,\dots,2k\} \backslash \{1\}, j \in J.$$

$$c_{iji'} \geq 0 \quad \forall i \in \{1,\dots,2k\}, j \in J,$$
$$i' \in \{1,\dots,2k+1\}, i' \geq i+1.$$

$$x_{ij} = x_{ij}^{(v)} : \lambda_{ij} \quad i \in \{1,\dots,2k\}, j \in J.$$

$$0 \leq v_l \leq v_l^{up},$$

$$0 \leq w \leq w^{up},$$

wobei $\overline{M}$ eine positive Konstante, groß genug ist, und v für jede bigM-Bedingung konstruierte Variablen ist, die mit der Variable w die Problemzulässigkeit sichern. Nachdem eine Lösung des Problem gefunden wurde, arbeitet man mit dem ursprünglichen Masterproblem wie davor weiter.

7.3 Version 2 des BD-Algorithmus.

Im Laufe der Implementierung kommt man zu der Erkenntnis, dass die Formulierung der Schnitte des BD-Algorithmus einfach ist, die Werte der dualen Variablen λ aber betragsmäßig sehr groß ausfallen können, was Instabilität der Rechnungen mit sich bringt. Das Subproblem und die Benderschnitte im Masterproblem werden umformuliert und im Rahmen des BD2-Algorthmus definiert. Die einzelnen Schritte sind analog zur ersten Version des Algorithmus, hier folgt eine kurze Beschreibung.

Schritt I: Lösung des Subproblems

Nachdem die optimale Lösung des Masterproblems gefunden wurde, löse mit den Werten $x_{ij}^{(v)}$ das Subproblem:

Subproblem

$$\min_{\pi,c} \pi_{2k+1} + \pi_1$$

$$-\pi_i + \pi_{ij} \geq 0 \qquad \forall i \in \{1,\dots,2k\}, j \in J.$$

$$-\pi_{ij} + \pi_{i'} - c_{iji'} \geq 0 \qquad \forall i \in \{1,\dots,2k\}, j \in J,$$
$$i' \in \{1,\dots,2k+1\}, i' \geq i+1.$$

$$c_{iji'} \geq \sum_{g=0}^{i'-i-1} \left(\sum_{j' \in J} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) x_{(i+g)j'}^{(t)} \right)$$
$$- (\overline{r}_j - \underline{r}_j) x_{ij}^{(t)} + M x_{ij}^{(t)} - M \qquad \underline{d: \quad \lambda_{iji'}}.$$
$$\forall i \in \{1,\dots,2k\},\ j \in J,\ i' \geq i+1,\ i' \in \{1,\dots,2k+1\}$$

$$
\begin{aligned}
\pi_1, \pi_{2k+1} &\geq 0 \\
\pi_i, \pi_{ij} &\in \mathbb{R} \quad \forall i \in \{1, \ldots, 2k\} \backslash \{1\}, j \in J. \\
c_{ij',i'} &\geq 0 \quad \forall i \in \{1, \ldots, 2k\}, j \in J, \\
& \qquad i' \in \{1, \ldots, 2k+1\}, i' \geq i+1.
\end{aligned}
$$

Die Lösung des Problems sind die Werte $\pi_{ij}^{(v)}, \pi_i^{(v)}$ und die Werte der dualen Variablen $\lambda_{iji'}$.

Schritt II: Konvergenzkontrolle

Die Konvergenzkontrolle geschieht wie in dem BD-Algorithmus:
Gilt $z_{up}^{(v)} - z_{down}^{(v)} = \pi_{2k+1}^{(v)} + \pi_1^{(v)} - \alpha^{(v)} < \epsilon \Rightarrow$ STOP, eine optimale Lösung ist für die gegebene Genauigkeit durch $x_{ij}^{(v)}$ und $\pi_{ij}^{(v)}$, $\pi_i^{(v)}$, $c_{iji'}^{(v)}$ gegeben.

Anderenfalls gehe wieder zur Lösung des neuen Masterproblems über, wobei man eine zusätzliche Bendersschnitt-Bedingung in das Masterproblem einfügt.

Schritt III: Lösung des Masterproblems

Ist die Konvergenzbedingung nicht erfüllt, wird zu dem Masterproblem ein neuer Bendersschnitt hinzugefügt und das System nach α und x gelöst, wobei der Wert α^{down} während des letzten Aufrufens des Masterproblems geändert wurde. Also werden wie auch in dem ersten Algorithmus drei Teilschritte ausgeführt:

1. Aktualisiere $(v) \leftarrow (v+1)$

2. Löse das Masterproblem:

Masterproblem

$$\min_{x,c,\alpha} \sum 0x_{ij} + \alpha$$

$$\sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J$$

$$\sum_{i,j,i'} \lambda_{iji}^{(t)} \Bigg[\sum_{g=0}^{i'-i-1} \Bigg(\sum_{j' \in J} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) x_{(i+g)j'} \Bigg) - (\overline{r}_j - \underline{r}_j) x_{ij} + M x_{ij} \Bigg] + \alpha \geq M \sum_{i,j,i'} \lambda_{iji'}^{(t)}$$

$$t = 1, \ldots, v-1$$

$$x_{ij} = 0 \qquad \forall j \in J_1, i \; gerade$$

$$x_{ij} = 0 \qquad \forall j \in J_2, i \; ungerade$$

$$x \in \{0,1\}$$

$$\alpha \geq \alpha^{down}$$

3. Mit der gefundenen Lösung $x_{ij}^{(v)}$ und dem neuen Wert $\alpha^{down} = \alpha^{(v)}$ gehe zu Schritt 1.

Der BD2-Algorithmus ist eine Alternative Formulierung der BDA, die gleiche Benderschnitte für das Masterproblem erzeugen sollten, da das Subproblem bei der erhaltenen nicht optimalen Lösung diese aus dem Optimalitätsbereich ausgrenzt. Da sich die Werte der dualen Variablen stark unterscheiden, und das Problem der großen Lambda-Werte in dem BD-Algorithmus auftritt, können sich die beiden Algorithmen im Zeitaufwand und der Anzahl der benötigten Schritte unterschei-

den.

7.4 50/50-Version des BD-Algorithmus

Bei dem Benderschnittansatz erfolgt die Aufteilung der Variablen in *normale* und *complicated*. Die Vereinfachung des Master- bzw. Subproblems wirkt in diesem Falle gegeneinander. D.h. je einfacher das Masterproblem, desto komplizierter das Subproblem und andersherum. In diesem Teil betrachten wir eine Version des BD-Algorithmus, in dem die Hälfte der Zuweisungsvariablen x als *complicated* gewählt werden, die andere Hälfte gehört zum Subproblem.

Das Masterproblem hängt damit von den Variablen x_{ij} mit $i \leq k$, $j \in J$, wobei $2k = |I|$ die Partitionsanzahl bezeichnet. Der Algorithmus ist dann im Wesentlichen dem oben beschriebenem gleich, es unterscheiden sich nur die Variablen, nach denen die einzelnen Probleme gelöst werden.
Es folgt die Beschreibung des Algorithmus.

Schritt I: Lösung des Subproblems

Nachdem die optimale Lösung des Masterproblems gefunden wurde, löse mit den Werten $x_{ij}^{(v)}$ für $i \leq k$, $j \in J$ das Subproblem:

Subproblem t

$$\min_{\pi,c} \pi_{2k+1} + \pi_1$$

$$x_{ij} = 0 \qquad \forall j \in J_1, i \; gerade, i > nparts/2$$

$$x_{ij} = 0 \qquad \forall j \in J_2, i \; ungerade, i > nparts/2$$

$$\sum_{i \in \{1,\ldots,2k\}, i>k} x_{ij} = 1 - \sum_{i \in \{1,\ldots,2k\}, i \leq k} x_{ij}^{(t)}, \quad d : \lambda_j \qquad \forall j \in J$$

$$-\pi_i + \pi_{ij} \geq 0 \quad \forall i \in \{1,\ldots,2k\}, j \in J.$$

$$-\pi_{ij} + \pi_{i'} - c_{iji'} \geq 0 \quad \forall i \in \{1,\ldots,2k\}, j \in J,$$

$$i' \in \{1,\ldots,2k+1\}, i` \geq i+1.$$

$$\sum_{g=0}^{i'-i-1} \left(\sum_{j' \in J} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) x_{(i+g)j'} \right)$$

$$-(\overline{r}_j - \underline{r}_j) x_{ij} - M(1 - x_{ij}) \leq c_{iji'} \quad \underline{d : \lambda_{iji'}} \quad \forall i \in \{1,\ldots,2k\}, j \in J, i` \geq i+1$$

$$mit \; x_{ij} = x_{ij}^{(t)}, \forall i \leq k$$

$$\pi_1, \pi_{2k+1} \geq 0$$

$$\pi_i, \pi_{ij} \in \mathbb{R} \quad \forall i \in \{1,\ldots,2k\} \backslash \{1\}, j \in J.$$

$$c_{ij',i'} \geq 0 \quad \forall i \in \{1,\ldots,2k\}, j \in J,$$

$$i' \in \{1,\ldots,2k+1\}, i' \geq i+1.$$

Die Lösung des Problems sind die Werte $\pi_{ij}^{(v)}, \pi_i^{(v)}$ und der dualen Variablen $\lambda_{iji'}$ und λ_j.

Schritt II: Konvergenzkontrolle

Die Konvergenzkontrolle geschieht wie beim BD-Algorithmus:
Gilt es $z_{up}^{(v)} - z_{down}^{(v)} = \pi_{2k+1}^{(v)} + \pi_1^{(v)} - \alpha^{(v)} < \epsilon \Rightarrow$ STOP. Eine optimale Lösung ist

für die gegebener Genauigkeit gegeben durch $x_{ij}^{(v)}$ und $\pi_{ij}^{(v)}, \pi_i^{(v)}, c_{i'j'i}^{(v)}$.

Anderenfalls gehe wieder zur Lösung des neuen Masterproblems über, wobei man eine zusätzliche Bendersschnitt-Bedingung in das Masterproblem einfügt.

Schritt III: Lösung des Masterproblems

Ein neuer Bendersschnitt wird hinzugefügt und das System nach den Variablen α und zu der Problem gehörende Hälfte der Variablen x gelöst, wobei der Wert α^{down} während des letzten Aufrufens des Masterproblems geändert wurde. Es wird $(v) \leftarrow (v+1)$ aktulisiert und das Masterproblem gelöst:

Masterproblem v

$$\min_{x,c,\alpha,i\leq k} \sum 0x_{ij} + \alpha$$

$$\sum_{i\in\{1,\ldots,2k\}} x_{ij} \leq 1 \qquad \forall j \in J$$

$$\sum_{j,i\leq k} \lambda_j^{(t)} x_{ij} + \sum_{i,j,i'} \lambda_{iji'}^{(t)} \Bigg[\Big(\sum_{g=0}^{i'-i-1} \Big(\sum_{j'\in J} (\overline{r}_j + (g+1)F - \underline{r}_{j'}) x_{(i+g)j'} \Big) - (\overline{r}_j - \underline{r}_j) x_{ij} + M x_{ij} \Bigg] + \alpha \geq \sum_{i,j,i'} \lambda_{iji'}^{(t)} M + \sum_j \lambda_j^{(t)}, \quad t = 1,\ldots,v-1$$

$$x_{ij} = 0 \qquad i > k,\ |I| = 2k$$

$$x_{ij} = 0 \qquad \forall j \in J_1, i\ gerade, i \leq nparts/2$$

$$x_{ij} = 0 \qquad \forall j \in J_2, i\ ungerade, i \leq nparts/2$$

$$x \in \{0,1\}$$

$$\alpha \geq \alpha^{down}$$

Mit der gefundenen Lösung $x_{ij}^{(v)}$, $i \leq k$, $j \in J$ und dem neuen Wert $\alpha^{down} = \alpha^{(v)}$ gehe zum Schritt 1.

Da im Subproblem die binäre Variablen enthalten sind und das Dualisieren ohne Weiteres für gemischt-ganzzahlige Programme nicht möglich ist [5], ist es ein Approximativer Algorithmus. Für das Subproblem ist die entstehende Matrix nicht immer unimodular, sodass der Wert dessen dualen Problems nicht immer gleich der optimalen Subproblemlösung ist. Mit dem ausgeführter Version des BD-Algorithmus wird eine schnellere Lösung des Schleusenproblems erhofft, bei der die

Genauigkeit im akzeptablen Rahmen bleibt. Eine Implementierung und Ausführen des Algorithmus wird die Übersicht schaffen, inwiefern es zutrifft.

8 Rechenstudie

In diesem Kapitel wird die Implementierung und die Auswertung des entworfenen Dekompositionsalgorithmus nach Bender in drei oben vorgestellten Versionen beschrieben. Es werden dazu im Laufe der Arbeit gemachte Beobachtungen präsentiert und die erhaltenen Ergebnisse analysiert.
Die Implementierung erfolgte in C++, dabei wurde SCIP [15] für die MIP-Lösungen genutzt. Die Experimente wurden auf einem Rechner mit Intel Core i7-2600 CPU 8*3,4 GHz und 16 GB RAM unter Linux durchgeführt. Der Programmcode, die Probleminstanzen und die Ergebnisse sind unter [20] zu finden.

8.1 Implementierung

8.2 Daten für die Rechenstudie

Als Daten werden generierte Instanzen verwendet, die auf dem Beispiel der realen Schleusungszeiten auf dem Kielkanal basieren. Die Instanzen werden jeweils so verändert, dass die verschiedenen Eigenschaften der Schiffsankunftszeiten auf die Stabilität und den Zeitaufwand im Benders Dekompositionsalgorithmus Einfluss nehmen. Die Probleminstanzen sind als Textdateien gespeichert und enthalten die für die Schleusungpartitionierung benötigten Daten: Anzahl der Schiffe als Dimension des Problems, Dauer eines Schleusungsverfahrens und die Schiffsdaten. Diese beinhalten für jedes Schiff die Nummer, die Richtung, aus der das Schiff kommt und die möglichen Ankunftszeiten, die durch die Angabe der oberen und unteren Intervallgrenze gegeben sind. Die Zahlen sind auf zwei Nachkommastellen beschränkt, um die Rechnung zu vereinfachen. Dies ermöglicht noch eine minutengenaue Zeitangabe, was für die Problemstellung zufriedenstellend ist.

Nach dem Einlesen der Problemdaten wird eine heuristische Zuweisung für die Schiffsmenge J und die Partitionsmenge I als eine mögliche Startlösung gefunden.

8.3 Startlösung

Im Initialisierungschritt des Algorithmus wird eine zulässige Partitionierung gesucht, die möglichst gute Werte $\{x_{ij}\}_{i\in I, j\in J}$ besitzt. Diese Zuweisung muss die Mindestanforderungen der Partitionierung erfüllen. Des Weiteren sollte diese Partitionierung einen möglichst kleinen Wert für das max-Szenario-Problem haben. Gegeben sind dabei die unsicheren Ankunftszeiten der Schiffe und die Richtungen, aus denen diese ankommen. Das Schleusenproblem wird aufgestellt, wobei Annahmen getroffen werden können, dass keines der Schiffe sich verspätet oder dass alle Schiffe verspätet ankommen. Alternativ kann die mittlere Ankunftszeit jedes Schiffs als fest angenommen werden.
In der Implementierung wird angenommen, dass jedes Schiff zu seiner spätesten Ankunftszeit an der Schleuse erscheint. Damit werden die Ankunftszeiten deterministisch und gleich $\overline{r}_j$ für jedes Schiff j. Die Verteilung der Schiffe auf die Schleusenvorgänge wird dann folgendermaßen ausgeführt: Die Schleuse startet auf der Seite mit dem höheren Wasserstand, also mit der Schiffsteilmenge J_1. Sie fährt ununterbrochen herunter und hoch. Alle bis zu dem Zeitpunkt der jeweiligen Schleusungstart angetroffene Schiffe werden geschleust. Das kleinste und das größte Element der Menge der Schiffsankunftszeiten $\{\overline{r}_1, \ldots, \overline{r}_n\}$ werden mit $\overline{r}_{kl}$ bzw. $\overline{r}_{gr}$ bezeichnet. Man setzt die Anzahl der Schleusungen gleich $(\overline{r}_{gr} - \overline{r}_{kl})/F + 4$, wobei F die oben definierte Dauer des Schleusenvorgangs ist.
Die Implementierung der Startlösung erfolgt in der Funktion *startsolution*(). Deren Ergebnisse werden gespeichert und bei der Lösung des ersten Subproblems im Laufe des BD-Algorithmus benutzt.

8.4 Weitere Bedingungen für die Schiffszuweisungen

In die Startlösung werden zusätzliche Bedingungen integriert, die für jede zulässige Partitionierung erfüllt werden müssen. Die Eigenschaften einer Partitionierung, die auch für den deterministischen Fall gültig sind, hängen von den Ankunftszeiten der Schiffe ab und werden für die unsicheren Ankunftszeiten definiert.

Eigenschaft 1: Betrachte ein Schiff l, das bei allen Szenarien $S \in \mathcal{S}$ später als Schiff k und früher als Schiff m ankommt, mit $l, m, k \in I$. Wenn die Schiffe k und m in einer Partition R_i mit $i \in I$ liegen, dann ist das Schiff l in der gleichen Partition mit k und m. Formal:

$$\overline{r}_m^S \leq r_l^S \leq \underline{r}_m^S \; \forall S \in \mathcal{S} \text{ und } m, k \in R_i, i \in I \Rightarrow l \in R_i$$

Dazu die Abbildung 7:

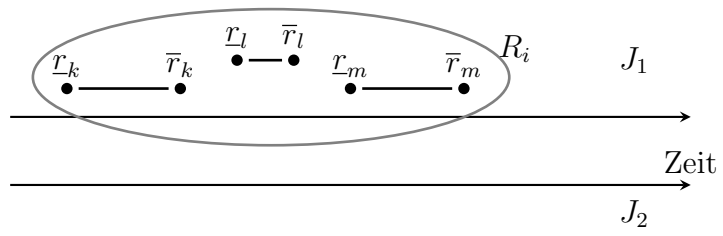

Abb. 7: Eigenschaft 1

Beweis. Für alle Szenarien $S \in \mathcal{S}$ gilt $r_k^S \leq r_l^S \leq r_m^S$. Es wird gezeigt, dass die Schiffe k, l, n bei jedem Szenario S für die Partitionierung $R \in \mathcal{R}$ mit minimalen Wartezeiten zusammen in einer Partition R_i, $i \in I$ geschleust werden.

Betrachte eine Partitionierung $R^* \in \mathcal{R}$, so $x_{ij}^R = x_{ij}^{R^*}$ fur alle $i \in I$, $i \neq l$ und alle $j \in J$. Damit wird das Schiff l nicht in einer Partition R_i zusammen mit k und m geschleust, sondern zu späterem Zeitpunkt in einer der Partitionen $R_{i+v}, v \in \mathbb{N}$ und $i + v \in I$. Die Ankunftszeit r_j hat keinen Einfluss auf die Partitionen $R_i + v$,

da r_j für die angenommene Partitionierung R für alle Szenarien früher als der Zeitpunkt t^0_{i+v} liegt. Dann ist die Wartezeit des Schiffes l gleich $t_{i+v} + F - r^S_l$. Dieser Wert ist größer als die Wartezeit $t_i + F - r^S_l$, wenn das Schiff zu der i-ten Partition gehört. Mit der gleichen Argumentation kann gezeigt werden, dass die Verschiebung der Schiffe m und k in eine spätere Partition $R_i + v$ mit $v \in \mathbb{N}$ und $i + v \in I$ die gesamte Wartezeit für jedes Szenario erhöht. □

Eigenschaft 2: Sei der Abstand zwischen der spätesten Ankunftszeit eines Schiffes und der frühesten Ankunftszeit eines anderen Schiffes größer oder gleich $4F$. Zwischen diesen Zeitpunkten befinden sich keine Ankunftszeiten eines anderen Schiffes. Dann kann man das Problem in zwei Teilprobleme zerlegen. Die Lösung des Problems ist die Zusammensetzung der Lösungen der zwei kleineren Probleme [1].

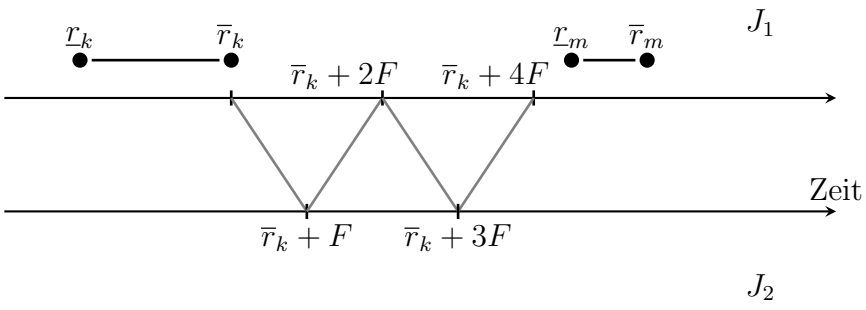

Abb. 8: Eigenschaft 2

Beweis. Angenommen das Schiff $k \in J$ hat die Ankunftszeit $\overline{r}_k$. Kein anderes Schiff erscheint in dem Zeitintervall $(\overline{r}_k, \overline{r}_k + 4F]$ und mindestens ein Schiff $m \in I$ kommt nach dem Zeitpunkt $\overline{r}_k + 4F$ an. Sei J_k eine Schiffsteilmenge $J_k \subset J$, die alle Schiffe enthält, die bis zum Zeit $\overline{r}_k$, $\overline{r}_k$ eingeschlossen, ankommen. In der Teilmenge J_{k+1} sind die Schiffe enthalten, deren Ankunftszeit nach $\overline{r}_k + 4F$ liegt.

Betrachte eine optimale Lösung für J_k. Die spätest mögliche optimale Schleusungszeit für das Schiff, das in $\overline{r}_k$ angekommen ist, ist $\overline{r}_k + 2F - \epsilon$ für $\epsilon \in \mathbb{R}$ klein genug. Damit ist jede Lösung, in der das Schiff zu der Zeit $t \geq \overline{r}_k + 2F$ geschleust wird, durch eine Lösung dominiert, in der dieses Schiff in $\overline{r}_k - 2aF$ geschleust wird, wobei $a \in \mathbb{Z}$ ist, so dass $t - 2aF \in [\overline{r}_k, \overline{r}_k + 2F)$ gilt. Alle Schiffe in J_k werden also bis $\overline{r}_k + 3F$ abgearbeitet, eingeschlossen F Zeiteinheiten, die für den Transfer des $k - ten$ Schiffes nötig sind, das zur Zeit $\overline{r}_k$ ankommt.
Betrachte die optimale Lösung der Teilmenge J_{k+1}. Dabei kann sich die Schleuse zu dem Zeitpunkt in beliebiger Position befinden, und die Schleusung der entsprechenden Richtung anfangen. $\overline{r}_k + 4F$ ist gleich der frühestmöglichen Ankunftszeit eines Schiffes in J_{k+1}. Man kann die Schleuse in diese Position spätestens ab dem Zeitpunkt $\overline{r}_k + 3F$ bringen. Dafür werden nicht mehr als F Zeiteinheiten gebraucht. Daraus folgt, dass die optimalen Lösungen für die Teilmengen J_k und J_{k+1} zusammengesetzt werden können, was die optimale Lösung des Problems ergibt. □

Nachdem die Startlösung gefunden ist, folgen die Iterationen des Benders Dekompositionsalgorithmus bis die optimale Lösung gefunden ist.

8.5 BDA-Implementierung

Für jedes einzelne Master-, Sub- bzw. Dualproblem werden drei Funktionen benötigt: In der ersten wird dazu ein SCIP-Problem aufgestellt. In der zweiten wird die Lösung des SCIP-Problems gefunden. Die dritte speichert die Lösung für die weitere Nutzung ab. Dabei kann die Lösung des Subproblems in der ersten und zweiten Version des Algorithmus entfallen, da Werte der dualen Probleme den optimalen Werten der jeweiligen Subprobleme entsprechen. Nachdem die Lösung des jeweiligen Teilproblems gefunden ist, wird das dazugehörige SCIP-Problem gelöscht und

der Speicher geleert.

Sobald die optimale Lösung der Instanz können folgende Daten neben der optimalen Partitionierung die Ausgabe des Programms darstellen: die Laufzeit des Algorithmus, die Anzahl der durchgeführten Iterationen, die Anzahl der Partitionen bzw. nicht leeren Partitionen in der optimalen Lösung, die Anzahl der benötigten Schleusungsvorgänge und die Entwicklung der Werte der oberen und unteren Grenzen für den optimalen Wert. Anhand dieser Daten wird die Effizienz des Algorithmus bewertet und die *bottle-neck*-Stellen festgestellt.

9 Ergebnisse und Modellanalyse

In diesem Kapitel werden die Ergebnisse aus der Rechenstudie ausgewertet und Empfehlungen für die weitere Entwicklung des Ansatzes formuliert. Die umfangreichen Instanzlösungen können unter [20] abgerufen werden. Als Referenz für die Wartezeiten und die optimale Lösung werden die Werte der direkten MILP-Problemlösung durch SCIP herangezogen.
Im Laufe der Implementierung des Algorithmus treten deutliche Probleme auf. Wie im Modell selbst, als auch in den SCIP-Lösungen kommt es zu Zeitverlusten, so dass die Laufzeit des Benders Dekompositionsalgorithmus länger als das Lösen des aufgestellten MILP-Problems im SCIP ist. Die 50/50-Version des Algorithmus ist deutlich schneller, wobei dessen approximative Eigenschaft in den Experimenten bestätigt wird.

9.1 Anzahl der Schiffe und Partitionen

Eine wichtige Rolle spielt bei dem Algorithmus die Anzahl der Schiffe in dem Problem. Während die Lösung für kleine Instanzen mit fünf oder 7 Schiffen noch relativ schnell erfolgt und die Rechenzeit mit der direkten Lösung im SCIP vergleichbar ist, kommt es bei mehr als 15 Schiffen zu beträchtlich höherem Zeitaufwand. Besonders zeitintensiv ist das Finden der Masterproblemlösung. Der Zeitaufwand für ein Masterproblem steigt mit der Anzahl der Benderschnitte deutlich.
Die Anzahl der Variablen und Ungleichungen in dem MILP-Schleusenproblem ist linear in der Anzahl der Schiffe n; die wachsende Schiffsanzahl kann dazu die Anzahl der für die Schleusung benötigten Partitionen vergrößern, da die Menge der möglichen Ankunftszeiten in dem Problem vergrößert wird. Die Reduzierung der Problemdimension durch die Aufteilung des Problems, die im vorigen Kapitel über Startlösungen beschrieben wurde, kann den Rechenaufwand reduzieren. Auf den

stark befahrenen Wasserstraßen kommt es jedoch zu solchen Situationen selten, was aus den Daten vom Kielkanal entnommen werden kann.
Ein einfacher Fall tritt auf, wenn das Ankunftszeitenintervall eines Schiffes gleich dem Ankunftszeitenintervall eines anderen ist, dann gehören die Schiffe zu einer Partition in der optimalen Lösung und können als ein Schiff betrachtet werden.

Instanz	$\|I\| = n$	$J = 2k$	t_{ges}	t_{sub}	t_{mast}	N_{Iter}	WZ_{min}
cr5os-3	5	8	1	0	1	23	36
cr6os-5	6	10	2	1	1	34	69
schiffe7a	7	16	34	6	27	113	58
schiffe7	7	6	1	0	1	25	42
cr8a-5	8	12	46	13	33	161	78
d10a	10	16	2540	43	2485	526	67
d10	10	14	617	31	584	293	64
cr10-3	10	22	5221	137	5077	730	5.52
cr10-5	10	14	262	14	246	219	9.5
cr10a-5	10	16	542	16	525	299	97
cr10os-3	10	22	4644	115	4523	593	69
schiffe11	11	16	613	26	586	226	79.5

Tabelle 1: Abhängigkeit der Laufzeit von Schiffs- und Partitionenanzahl

In der Tabelle 1 sind die Ergebnisse der Ausführung des BD-Algorithmus für neun Instanzen präsentiert. Die einzelnen Spalten enthalten die Instanznamen, die Anzahl der Schiffe und der Partitionen, die gesamte Laufzeit t_{ges} des Algorithmus. Des Weiteren werden die summierten Zeiten für die Lösung aller Subprobleme durch t_{sub} und analog t_{master} für die Summe der Lösungsdauer des Masterproblems gegeben. Alle Laufzeitangaben sind in Sekunden. Die Anzahl der benötigten Iterationen wird in der siebten Spalte gegeben und die Lösung des Problems in der letzten Spalte.
Aus diesen Ergebnissen ist zu entnehmen, dass bereits für 10 Schiffe die Lösung des

Problems mehr als eine Stunde dauert. Mit wachsender Anzahl der Schiffe wächst die Anzahl der Iterationsschritte. Die Abhängigkeit ist aber nicht eindeutig festzustellen, da die anderen Faktoren wie die Anzahl der Partitionen eine wichtige Rolle spielen. Zum gleichen Ergebnis kommt man mit den Daten der 50/50-Version. Für mehr Instanzen und Lösungen siehe [20].

Die Anzahl der Partitionen spielt eine noch wichtigere Rolle als die Anzahl der Schiffe. Zum Beispiel sind die Daten der Instanzen cr10-3 und der cr10-5 gleich, nur die Schleusungsdauer F in dem ersten Fall gleich drei bzw. gleich 5 im zweiten Fall. Durch die Vergrößerung dieses Wertes wird die maximale Anzahl der Schleusungen von 22 auf 14 verkleinert. Die Lösungszeit wird dadurch drastisch gesenkt. Die Anzahl der Variablen und der originalen Nebenbedingungen ist quadratisch in der Größenordnung der Partitionen. Aus den einbezogenen Ergebnissen ist es zu schließen, dass die Anzahl der nicht leeren Partitionen der Lösung sich deutlich von der angenommenen Maximalanzahl unterscheidet. In dem Model wurde diese Anzahl für den schlechtesten Fall mit $4 * n + 4$ beschränkt. Andere obere Schranke wäre die Anzahl der Partitonen der Startlösung, wenn die Schleuse wie oben beschrieben ununterbrochen herunter und hoch fährt.
Darüber hinaus ist die Anzahl der Bedingungen und Variablen der MILP-Formulierung quadratisch in der Größenordnung der Partitionsmenge J. In den meisten Fällen wurde eine viel kleinere Anzahl der Partitionen benötigt, als der schlechteste Fall es vorsieht. In den Beispielinstanzen kommt im Durchschnitt im optimalen Ergebnis $1.5 * n$ als Anzahl der benötigten Partitionen, wobei n die Anzahl der Schiffe des Problems ist. Anderenfalls, wenn zu wenige Partitionen zur Verfügung stehen, wird die optimale Lösung nicht gefunden. In diesem Fall werden die Schiffe für die gegebene Partition wartezeitminimierend verteilt. Damit ist eine bessere Definition der oberen Schranke der Partitionsanzahl ein wichtiges Ziel für die weitere Verbesserung des Modells des Schleusenproblems.

9.2 bigM-Formulierung

In Subproblemen treten die bigM-Bedingungen auf. Der Wert M sollte groß genug sein, sodass dieser größer als die Summe aller anderen auftretende Werte ist. Andererseits sollte diese Größe möglichst klein gewählt werden. Nach den erhaltenen Daten wurde bestätigt, dass die Größe des bigM direkt die Lösungsdauer des Problems beeinflusst, wie man es am Beispiel der vier Probleminstanzen sehen kann. Für mehr siehe Ergebnisse siehe [20].

Instanz	n	2k	t_{ges}	t_{sub}	t_{mast}	N_{Iter}	WZ
cr9-3	9	14	161	28	133	305	4.76
cr9-5	9	14	107	15	88	203	8.84
cr9a-3	9	14	1174	74	1092	719	66
cr9os-3	9	14	176	16	159	247	65

Tabelle 1: BD-Algorithmus mit M=100000

Instanz	n	2k	t_{ges}	t_{sub}	t_{mast}	N_{Iter}	WZ
cr9-3	9	14	441	58	377	554	4.76
cr9-5	9	14	331	30	299	345	8.84
cr9a-3	9	14	3725	116	3592	1204	66
cr9os-3	9	14	729	43	682	538	65

Tabelle 2: BD-Algorithmus mit M=10000

Die Experimente im Umfang von 20 Instanzen wurde unter sonst gleichen Bedingungen durchgeführt; lediglich der Wert bigM wurde geändert. In allen Fällen wurde mit der Verkleinerung der Größe von M die Reduzierung der Laufzeit des Algorithmus beobachtet.

Die Abhängigkeit der Größe des bigM sowohl von der Anzahl der Schiffe und Partitionen, als auch von den Zeitangabenwerten kann untersucht werden. Als Mögliche Ergebnis

könnte eine Funktion entwickelt werden, die den Wert von bigM geschickt von oben beschränkt.

9.3 Implementierung der 50/50-Variante

Im unterschied zu den BD- und BD2-Algorithmen, die ungefähr gleiche Effizienz im Sinne der Laufzeit aufweisen, ist die 50/50-Variante wesentlich schneller. In der Tabelle 7 werden die Ergebnisse für die Laufzeiten der 50/50-Version des Algorithmus mit der direkter Lösung und des BD2-Algorithmus verglichen. In den Klammern werden neben den Lösungsdauer die Anzahl der Iterationen gegeben.

Problem	SCIP	opt. Lösung	BD-Algorithmen		
			BD2-Version	50/50-Version	appr. Lösung
schiffe4	1.1	17	6(31)	2(2)	17
schiffe5	13.1	21	33(71)	5(4)	22
schiffe7a	26	58	153(120)	11(2)	58
schiffe10	152	74	457(216)	64(10)	76
schiffe11	88	79.5	674(234)	37(2)	79.5
schiffe15	466	117.5	3350(226)	91(3)	117.5
schiffe20	1653	153	-	172(2)	159.3

Tab. 7: Approximativer Algorithmus im Vergleich

In den Beispielen kann man erkennen, dass der Approximativer Algorithmus wesentlich weniger Iterationsschritten braucht. Die Lösung wird schneller als in den beiden anderen Methoden gefunden. Die weicht aber von der optimalen Wert ab, was auf die in der Kapitel 7 beschriebene Eigenschaft der Dualisierung zurückzuführen ist.
Die Allokationsvariablen des Problems sind binär und nach wenigen Iterationen und damit erstellten Benderschnitten dauert die Lösung des Masterproblems viel länger als die Lösung des entsprechenden Subproblems. Die Lösung des Subproblems, bzw. des dazu

dualen Problems nimmt für alle Iterationen gleiche Zeit in Anspruch.
Die Idee das Masterproblem durch die Variablenanzahlreduzierung zu vereinfachen führte zu der 50/50-Version des BD-Algorithmus. Für diese Formulierung muss das Subproblem explizit gelöst werden (s. Kapitel 8). Die Aufteilung der Zuweisungsvariablen erfolgte dabei nach der Nummer der Partitionen. Eine weitere Möglichkeit wäre die Aufteilung nach der Schiffsnummer durchzuführen oder eine andere Bedingung für die Zugehörigkeit der Zuweisungsvariablen zu dem Master bzw. Subproblem zu aufzustellen. Es wurde unter anderem eine 70/30-Version des BD-Algorithmus getestet, die analog zu der 50/50-Version ist. Die Zuweisungsvariablenmenge wird zwischen dem Master- und Subproblem aber im Verhältnis von 2:1, anstatt 1:1 wie in der 50/50-Version, aufgeteilt. Im Laufe der Rechenstudie wurde festgestellt, dass die 70/30-Variante keinen Zeitgewinn mit sich bringt.
Bei den Lösungen der 50/50-Variante ist aufgefallen, dass zum Teil Benderschnitte erzeugt wurden, unter denen alle Zuweisungsvariablen x des Masterproblems gleich null gesetzt sein müssen. Um es zu umgehen, wurde eine zusätzliche Bedingung erzeugt, die Mindestanzahl der Zuweisungsvariablen sichert, die für die Lösung der Masterproblem ungleich null ist. Diese Bedingung verlangsamert jedoch die Lösung des Problems, so das der Zeitaufwand auf das Niveau des BD- bzw. BD2-Algorithmus steigt.

9.4 Zusammenfassung

Die Rechenstudie hat gezeigt, dass die BD- und BD2-Algorithmen die optimale Partitionierung des Schleusenproblems berechnen. Dessen Performance ist in der implementieren Form aber nicht zufriedenstellend. So kommen die ersten zwei Versionen des Algorithmus zum optimalen Ergebnis in längerer Zeit als die direkte Lösung des Problems mit Hilfe von SCIP. Es konnte der höchste Zeitverbrauch für die Lösung der Masterprobleme festgestellt werden. Die Binarität der Allokationsvariablen trägt dabei zu der Komplexitätssteigerung bei. Aus dieser Beobachtung kommt der Ansatz der 50/50-Version des

BD-Algorithmus, wobei die Hälfte der Zuweisungsvariablen als *complicated* angenommen wird. Diese Version ist wesentlich schneller im Lösen der Masterprobleme, da es im Vergleich zu dem Masterproblem nur die Hälfte der binären Variablen enthält. Die Dauer der Lösung der einzelnen Subprobleme (und damit auch Dualprobleme) unterscheidet sich unwesentlich von der einfachen Version des Algorithmus. Bei manchen Probleminstanzen schneiden die erzeugten Benderschnitte die optimale Lösung ab, so dass die optimale Partitionierung nicht gefunden werden kann. In den ausgeführten Beispielen konnte aber relativ gute Lösung gefunden werden. Die Frage wäre wichtig zu beantworten, wie man die Größe der Abweichung der so gefundenen Lösung von dem optimalen Wert abschätzen kann, ist aber nicht Teil dieser Arbeit.

Des Weiteren konnte festgestellt werden, dass die Anzahl der möglichen Partitionen einer Partitionierung eine der wichtigsten Rollen in dem Problem spielt. In den meisten Fällen wurde nur Hälfte der zur Verfügung stehenden Partitionen benutzt. Weitere Begrenzung der Partitionenanzahl $|I|$ ist erstrebenswert. Mit wachsender Anzahl der Iterationen und den damit erhaltenen Benderschnitte steigt die Komplexität des Masterproblems dramatisch, was in der Summe die lange Laufzeit des Algorithmus ergibt. Damit wäre das Ziel die Anzahl der Iterationen möglichst klein zu halten, und dadurch die Effektivität der einzelnen Benderschnitte zu erhöhen. Wie es geht, wäre das Ziel weiterer Forschung.

10 Fazit

Im Laufe der Arbeit wurde die Effizienz des Bendersschnitt-Ansatzes am Beispiel des Schleusenproblems untersucht. Ein gemischt-ganzzahliges lineares Problem für das Schleusenproblem wurde definiert, darauf basierend ein rekursiver Algorithmus für die Berechnung der optimaler Lösung entwickelt. In der vorliegenden Masterarbeit wurde unter anderem untersucht, ob das Benderverfahren eine optimale Lösungsweg für die Fragestellung darstellt.

Die Fragestellung wurde in Kapitel drei bis sechs mathematisch formuliert und das Schleusenproblem mit Hilfe eines linearen Programms beschrieben. Die Implementierung des Algorithmus erfolgte im C++ Code (s. Anhang). Die darin erzeugte lineare Teilprogramme wurden mit Hilfe von SCIP gelöst. Im Laufe der Validierung und Evaluierung wurde eine Lösung des MILP-Formulierung des Schleusenproblems in SCIP als Referenz herangezogen. In den Anwendungsfällen konnte die richtige Lösung gefunden werden, allerdings brachte der Algorithmus keinen Zeitgewinn. Im Laufe der Optimierung des Programms ließen sich die Grenzen des Verfahrens herausfinden; den besonders zeitkritischen Punkt für den BD-Algorithmus stellt dabei die Anzahl der Benderschnitte dar und der damit wachsende Zeitaufwand für die Lösung einzelner Masterprobleme. Des Weiteren wurde eine Version des BDA entworfen, die das Problem approximativ löst.

In der vorliegenden Masterarbeit wurde gezeigt, dass die Verwendung vom BD-Algorithmus für die Fragestellung einer Schleuse mit einer unbegrenzten Kapazität grundsätzlich geeignet ist. Die Laufzeit der Algorithmus übersteigt dabei die Dauer der direkten Lösung mit SCIP. Die weitere Optimierung des Problems besteht darin die Anzahl der benötigten Partitionen zu beschränken und eine Definition des Problems ohne bigM zu formulieren. Es ist auch ersichtlich, dass die Aufteilung der ganzzahligen Variablen auf die einzelnen Probleme des BDA-Algorithmus problematisch ist, da die Dualisierung nicht ohne

weiteres möglich ist. Bei dem BDA-Algorithmus führte die Zunahme an Nebenbedingungen zu einem erheblichen Wachstum an Rechenaufwand. Eine Möglichkeit wäre es, die Benderschnitte nicht im Rahmen des BDA-Algorithmus zu verwenden, sondern als Heuristiken in die direkte SCIP-Lösung einzubauen. Damit würden nicht so viele zusätzliche Bedingungen in das Problem einfließen, bzw. diese würden optimaler konstuiert. Die Weiterentwicklung des BDA-Algorithmus für die Schleusenanlagen mit begrenzter Kapazität bzw. mehreren Schleusen stellt die Grundlage für eine weitere Nachforschung dar.

11 Notationen

I	Partitionsmenge
J	Schiffsmenge
r_j	Deterministische Ankunftszeit Schiffes j
$\underline{r}_j, \overline{r}_j$	früheste/späteste Ankunftszeit Schiffes j
F	Schleusungsdauer
t_i^S	Startzeit der Schleusung i beim Szenario S
t_i^0	früheste Startzeit der Schleusung i
$J_1,\ J_2$	aus der linken bzw. rechten Richtung ankommende Schiffsmengen
S	Szenario
R	Partitionierung
$R_i, i \in I$	einzelne Partitionen
$\mathcal{S}$	Menge der möglichen Szenarien
$\mathcal{R}$	Menge der möglichen Partitionierungen
IP	Integer program / Ganzazhliges problem
ILP	Integer linear program / Ganzzahliges lineares Problem
$MILP$	Mixed integer linear program / Gemischt ganzzahliges lineare Problem
BDA	Benders Dekompositionsalgorithmus
$w(R)$	Gesamte Wartezeit bei einer Partitionierung R
$W(S)$	Größte gesamte Wartezeit einer Strategie S
$W(S,R)$	Gesamte Wartezeit einer Strategie S für Partition R
$W(t,i)$	Wartezeit, wenn die i-te Partition nicht bevor dem Zeitpunkt t erfolgen kann.
$vol(t,R_i)$	unverzichtbare Zeit der Partition R_i bei Schleusung nicht bevor dem Zeitpunkt t
$vol(R_i)$	unverzichtbare Wartezeit der Partition R_i
x	Zuweisungsmatrix $\{x_{ij}\}_{i\in I, j\in J}$
c	Kantenkostenmenge $\{c_{iji'}\}, \forall i \in \{1,\ldots,2k\}$, $j \in J$, $i' \in \{1,\ldots,2k+1\}$, $i' \geq i+1$.
π	Variablenmenge $\{\pi_i\}_{i\in I} \cup \{\pi_{ij}\}_{i\in I, j\in J}$.

Literatur

[1] S. Coene, F. Spieksma, Greet Vanden Berghe. The Lockmaster's Problem, (2011).

[2] A. Geoffrion. Generalized benders decomposition. Journal of Opti- mization Theory and Applications 10,, 237-260, (1972).

[3] J.F. Benders, Partitioning Pocedures for Solving Mixed-Variables Programming Problems, Numerische Mathematik 4, 238-252. (1962).

[4] A. M. Georion, Generalized Benders Decomposition, Journal of Optimization Theory and Applications 10, no. 4, 237-260, (1972).

[5] R. Pendavingh, Totally unimodular matrices and otal dual integrality, (2012).

[6] E.S. Thorsteinsson. Branch-and-Check. A Hybrid Framework Integrating Mixed Integer Programming and Constraint Logic Programming, (2001).

[7] J.N. Hooker, G. Ottosson. Logic-based Benders decomposition, (2003).

[8] A.J. Conejo, E. Castillo, R. Mingues, R. Garcia-Bertrandt. Dekomposition Techniques in Mathematical Programming. Engeneering and Science Application, 107-119, (2006).

[9] G. Codato, M. Fischetti, Combinatorial benders' cuts for mixed- integer linear programming. Operations Research 54, 756-766, (2006).

[10] R. Diestel. Graphentheorie, (2000).

[11] Die bank – Zeitschrift für Bankpolitik und Praxis, (12/2008).

[12] Zweites Marco Polo Programm der EU-Kommission, (2007).

[13] J. Verstichel, J. Kinable, P. De Causmaecker, G. Vanden Berghe. A Combinatorial Benders' Decomposition for the Lock Scheduling Problem, (2013).

[14] L. Volkmann. Graphen an allen Ecken und Kanten, (2011).

[15] SCIP. Solving Constraint Integer Programs. http://scip.zib.de/

[16] E. Klvelagen. Benders Decomposition with GAMS, (2005).

[17] J. Verstichel, P. De Causmaecker, F.C.R. Spieksma, G. Vanden Berghe. Exact and heuristic methods for placing ships in locks, (2013).

[18] J. Verstichel, P. De Causmaecker, F.C.R. Spieksma, G. Vanden Berghe. The gene-

ralized lock scheduling problem: An exact approach (Technical Report), (2013).

[19] R.M. Freund. Benders' decomposition methods for structured optimization, including stochastic optimization, (2004).

[20] Link auf den Programmcode: http://1drv.ms/1su3FXZ