Heuristische Baumsuche für Stackingprobleme im Zwischenlager von Containerterminals

Schriftliche Promotionsleistung
zur Erlangung des akademischen Grades
Doctor rerum politicarum

vorgelegt und angenommen
an der Fakultät für Wirtschaftswissenschaft
der Otto-von-Guericke-Universität Magdeburg

Verfasser:	Florian Forster
Geburtsdatum und -ort:	30. Mai 1981, Dachau
Arbeit eingereicht am:	07.03.2014

Gutachter der schriftlichen Promotionsleistung:

PD Dr. Andreas Bortfeldt

Prof. Dr. Gerhard Wäscher

Datum der Disputation: 21.05.2014

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmen-Verzeichnis

Abkürzungsverzeichnis

AGV	Automated Guided Vehicle
ALV	Automated Lifting Vehicle
CPP	Container Positioning Problem
CPMP	Container Pre-Marshalling Problem
CRP	Container Relocation Problem
CRTP	Container Retrieval Problem
CT	Containerterminal
DP	Dynamische Programmierung
DRMG	Double Rail Mounted Gantry Crane
FEU	Forty-Foot Equivalent Unit
IATA	International Air Transport Association
ISO	International Organization for Standardization
LB	Lower Bound
OR	Operation Research
QCAP	Quay Crane Assignment Problem
QCSP	Quay Crane Scheduling Problem
RMG	Rail Mounted Gantry Crane

RTG	Rubber Tired Gantry Crane
SA	Simulated Annealing
SC	Straddle Carrier
SSAP	Storage Space Allocation Problem
TEU	Twenty-Foot Equivalent Unit
ULD	Unit Load Device
UNCTAD	United Nations Conference on Trade and Development

1 Einleitung

1.1 Stackingprobleme in Containerterminals

Container sind wiederverwendbare, größennormierte und stapelbare Behälter, die vorwiegend dem Transport von Stückgütern mit unterschiedlichen Transportmitteln (Schiffe, Bahn, LKW) dienen (vgl. hierzu und im Folgenden Lowe, 2005, S. 10). Der 20-Fuß-Container mit den Abmessungen 6,096 x 2,438 x 2,591 Metern sowie der doppelt so lange 40-Fuß-Container (siehe Abbildung 1.1) sind die weltweit am häufigsten eingesetzten Containertypen.

Abbildung 1.1: 40-Fuß-Container (Quelle: Autor)

Neben den Standardcontainern für reguläre Stückgüter gibt es auch Container mit Lüftung, Kühl- oder Heizaggregaten sowie speziellen Schutzvorkehrungen für Gefahrenguttransporte. In der Zeit von 1986 bis 2008 hat sich die Menge an Waren, die weltweit in Containern transportiert wurde, mehr als versechsfacht (vgl. UNCTAD, 2008, S. 22f.). In der Schifffahrt ist der Trend zum Containertransport besonders ausgeprägt. Das an Containerterminals in Häfen umgeschlagene Containervolumen hat sich im selben Zeitraum fast verzehnfacht (vgl. Günther und Kim, 2006, S. 438). Nach Öltankern und Massengutfrachtern stellen Containerschiffe bereits die drittgrößte Klasse von Transportschiffen in Bezug auf die Gesamttonnage dar (vgl. UNCTAD, 2006, S. 20).

Containerterminals stehen untereinander in einem harten Wettbewerb und sind darauf angewiesen, die für ihre Kunden entscheidenden Leistungskennzahlen permanent zu verbessern. Besondere Bedeutung kommt dabei der Liegezeit für Containerschiffe zu, mit der die Zeitspanne zwischen dem An- und Ablegen eines Schiffes bezeichnet wird. Die Liegezeit eines Schiffes wird am stärksten durch die Be- und Entladedauer beeinflusst. Um diese zu minimieren, muss ein möglichst unterbrechungsfreier Containerfluss zwischen dem Schiff und dem Zwischenlager, in dem sowohl die aus dem Schiff zu entladenden Container eingelagert werden als auch die ins Schiff einzuladenden Container untergebracht sind, sichergestellt werden.

In den meisten Containerterminals sind die Container im Zwischenlager aus Platzgründen in Stapeln aufeinandergelagert, so dass immer nur jeweils der oberste Container auf einem Stapel für einen Kran zugänglich ist. Im Terminalbetrieb tritt nun häufig der Fall auf, dass ein Container aus dem Zwischenlager befördert werden muss, der sich aktuell nicht auf der obersten Position in einem Stapel befindet. Um auf den Container zugreifen zu können, müssen alle Container, die sich über ihm befinden, temporär auf andere Stapel umgeladen werden.

Diese Umladeoperationen sind unproduktiv, zeitaufwändig und lassen den Containerfluss zwischen dem Schiff und dem Zwischenlager ins Stocken kommen. Dadurch wirken sie sich negativ auf die Be- und Entladedauer und schließlich auf die gesamte Abfertigungsdauer aus. Sie lassen sich allerdings aus verschiedenen Gründen nicht immer vollständig vermeiden.

Mehrere Optimierungsprobleme im Zwischenlager von Containerterminals zielen auf die Verringerung der Be- und Entladedauer durch die Reduzierung von unproduktiven Umladeoperationen im operativen Betrieb des Containerterminals ab. Da die Stapelung der Container und die damit einhergehende Zugriffsbeschränkung auf den höchsten Container im Stapel die wesentliche Ursache für unproduktive Umladeoperationen sind, können diese als Stackingprobleme bezeichnet werden. In der Literatur werden insbesondere das Container Pre-Marshalling Problem (CPMP), das Container Relocation Problem (CRP) und das Container Retrieval Problem (CRTP) thematisiert.

Beim CPMP geht es darum, eine Reihe von Stapeln durch möglichst wenige Umladeoperationen so zu gestalten, dass alle Container in der für das Entladen richtigen Reihenfolge gestapelt sind, d.h. dass keine weiteren Umladeoperationen mehr notwendig sind, um die Container zu einem späteren Zeitpunkt aus dem Zwischenlager zu entnehmen. Dies macht aber natürlich nur dann Sinn, wenn vor dem Entladen des Zwischenlagers noch genug Zeit bleibt, die Container entsprechend umzusortieren, was beim CPMP unterstellt wird.

Beim CRP geht man hingegen davon aus, dass keine Zeit für das Umsortieren der Container vorhanden ist. Die Container aus einer Reihe von Stapeln müssen also unmittelbar in einer vorgegebenen Reihenfolge entladen werden. Ziel ist es, dies mit möglichst wenigen unproduktiven Umladeoperationen zu erreichen.

Das CRTP stellt eine Erweiterung des CRP dar. Während beim CRP nur genau eine Reihe von Stapeln berücksichtigt wird, sind beim CRTP mehrere Stapelreihen zu leeren. Das Optimierungsziel ist die Minimierung der Zeit, die der Stapelkran benötigt, um diese Aufgabe zu erfüllen.

1.2 Zielsetzung der Arbeit

Für die zuvor umrissenen Stackingprobleme wurden bereits verschiedene Lösungsansätze publiziert. Darunter finden sich sowohl Verfahren, die die Optimalität der Lösung garantieren, als auch Näherungsverfahren, die mit nicht-willkürlichen Re-

geln arbeiten und damit zu guten Lösungen führen, jedoch kein optimales Ergebnis garantieren (sog. Heuristiken).

Unter den bisher angewendeten Heuristiken finden sich allerdings keine Baumsuche-Heuristiken. Dabei wurden in der jüngeren Vergangenheit Baumsuche-Heuristiken sehr erfolgreich auf zahlreiche kombinatorische Optimierungsprobleme angewendet, speziell auch im Bereich Logistik (siehe Abschnitt 4.4.2).

Ziel dieser Arbeit ist es, die Frage zu klären, ob für die oben genannten Stackingprobleme mit Baumsuche-Verfahren bessere Ergebnisse erzielbar sind als mit den bisher publizierten Lösungsansätzen.

Um diese Frage zu beantworten, sollen unter Berücksichtigung des engen praktischen Zusammenhangs und der ausgeprägten formalen Verwandtschaft der bisher bekannten Stackingprobleme im Zwischenlager eines Containerterminals heuristische Baumsuche-Verfahren für drei Stackingprobleme entwickelt werden, nämlich für das CPMP, das CRP und das CRTP. Die für die Lösung dieser Stackingprobleme bisher veröffentlichten Verfahren sollen in diesem Zusammenhang ebenfalls beschriebenen werden. Schließlich sollen die entwickelten Baumsuche-Verfahren einem Vergleichstest mit den konkurrierenden Verfahren unterzogen werden.

Die Algorithmen sollen generell für in der Praxis auftretende Problemdimensionen in vertretbaren Zeitspannen sehr gute, häufig nahoptimale, Ergebnisse liefern können. Um dies zu belegen, sollen falls notwendig auch neue Probleminstanzen generiert und die Algorithmen mit diesen getestet werden.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in neun Kapitel.

Nach dieser Einleitung werden in Kapitel 2 die grundlegenden Zusammenhänge, die Geschichte und die wirtschaftliche Relevanz des globalen Containertransports beschrieben. Dabei wird ausführlich auf die Bedeutung der Containerschifffahrt eingegangen.

Kapitel 3 beschäftigt sich eingehend mit Containerterminals, die den praktischen Bezugspunkt der in dieser Arbeit untersuchten Optimierungsprobleme bilden. Zunächst werden der Aufbau eines Containerterminals in einem Hafen und die dabei eingesetzten Anlagen und Maschinen detailliert beschrieben. Anschließend wird erläutert, warum eine hohe Produktivität ein wesentlicher Faktor für die Wettbewerbsfähigkeit eines Containerterminals darstellt.

Kapitel 4 widmet sich allgemein Optimierungs- und Entscheidungsproblemen. Nach der Definition einiger wichtiger Grundbegriffe wird in einem gesonderten Abschnitt auf kombinatorische Optimierungsprobleme eingegangen. Darauf folgt eine kurze Betrachtung des komplexitätstheoretischen Hintergrundes. Abschließend werden die wichtigsten und erfolgreichsten Lösungsansätze für kombinatorische Optimierungsprobleme skizziert und der für die vorliegende Arbeit zentrale Ansatz der Baumsuche-Heuristik entsprechend eingeordnet.

In Kapitel 5 werden betriebliche Entscheidungsprobleme beleuchtet, die im operativen Betrieb von Containerterminals auftreten. Den operativen betrieblichen Entscheidungsproblemen im Zwischenlager wird dabei besondere Aufmerksamkeit gewidmet. Dabei werden die im Fokus dieser Arbeit stehenden Stackingprobleme genauer charakterisiert und in die umgebende Problemlandschaft eingeordnet.

Die Kapitel 6 bis 8 bilden den Kern der Arbeit. In diesen Kapiteln werden Baumsuche-Heuristiken für drei verschiedene Stackingprobleme entwickelt, die im Zwischenlager von Containerterminals auftreten: für das Container Pre-Marshalling Problem (CPMP), das Container Relocation Problem (CRP) und das Container Retrieval Problem (CRTP).

In den Kapiteln wird jeweils zunächst das konkrete Stackingproblem beschrieben und formalisiert. Nach Vorstellung der bisherigen Ansätze zur Lösung des Problems wird ein auf einer Baumsuche-Heuristik basierender Algorithmus präsentiert. Den Abschluss eines Kapitels bildet jeweils ein ausführlicher Leistungsvergleich der Baumsuche-Heuristik mit den bisher publizierten Lösungsansätzen sowie eine Evaluierung mit weiteren Testinstanzen. Für das CPMP und beim CRP werden zusätzlich einzelne Komponenten des Lösungsverfahrens separat untersucht.

Kapitel 9 fasst die in dieser Arbeit gewonnenen Ergebnisse zusammen und beschreibt zwei Ansatzpunkte, mit denen zukünftige Forschungsaktivitäten an die Ergebnisse dieser Arbeit anknüpfen können.

2 Globaler Containertransport

2.1 Grundlagen

Die heutige Wirtschaftswelt ist von einer hohen Arbeitsteilung und räumlichen Verteilung geprägt, die immer häufiger über Ländergrenzen hinausgeht (vgl. Gleißner und Femerling, 2007, S. 41f.). Der Welthandel, gemessen am Wert der jährlich von allen Volkswirtschaften exportierten Güter, ist in den Jahren von 1990 bis 2010 im Mittel um jährlich 10 % gewachsen (siehe Abbildung 2.1). Der Welthandel und damit auch das transportierte Gütervolumen wächst seit vielen Jahren schneller als die Weltwirtschaft (vgl. Göpfert und Braun, 2008, S. 3). Diese Entwicklung wird durch den generellen Trend zum Outsourcing und den steigenden Anteil hochwertiger Produkte am exportierten Gütervolumen weiter verstärkt (vgl. hierzu und im Folgenden Lemper, 2009, S. 16f.). Unternehmen globalisieren in zunehmenden Maße ihre Beschaffungs-, Produktions- und Absatzprozesse, gleichzeitig werden Handelsbarrieren weltweit abgebaut. All diese Entwicklungen haben eine steigende Nachfrage nach Transportleistungen für Rohstoffe, Zwischen- und Endprodukte zur Folge.

Die Art und Weise, wie diese Transportleistungen erbracht werden, hängt wesentlich vom Typ des zu transportierenden Guts ab. Grundsätzlich lassen sich zu transportierende Güter zunächst grob, wie in Abbildung 2.2 dargestellt, nach ihrem Aggregatszustand klassifizieren (vgl. Gudehus, 2010, S. 775). Gasförmige und flüssige Güter werden typischerweise in Tanks transportiert. Gase können dabei für den Zeitraum des Transport verflüssigt werden, um volumenökonomischer transportiert werden zu können. Bei den festen Gütern lassen sich Schüttgüter und Stückgüter unterscheiden. Schüttgüter weisen eine granulare Struktur auf (z.B. Zucker, Kohle, Getreide oder Zement) und werden in der Regel unverpackt transportiert. Der Be-

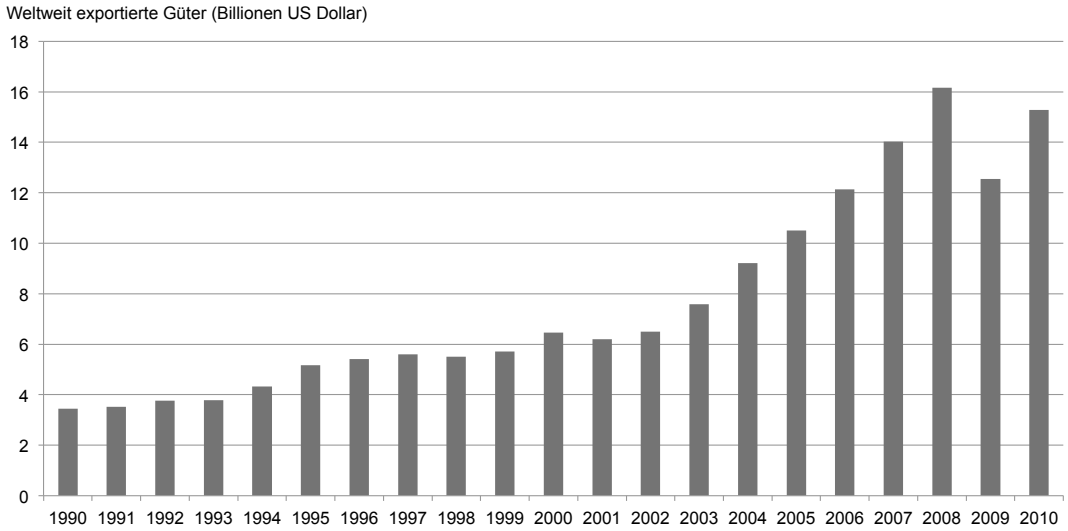


Abbildung 2.1: Entwicklung des Welthandels von 1990 bis 2010 (Eigene Darstellung nach World Trade Organization, 2011)

griff Massengut dient als Sammelbegriff für gasförmige und flüssige Transportgüter sowie Schüttgüter.

Alle nicht-granularen festen Güter werden schließlich als Stückgüter bezeichnet. Der Begriff umfasst demnach so verschiedenartige Produkte wie z.B. Elektronikbauteile, Textilien, Obst und Schuhe. Stückgüter wurden ursprünglich in unterschiedlichsten Behältnissen wie z.B. Kisten, Kartons, Paletten oder Fässern befördert. Heutzutage werden Stückgüter zum überwiegenden Teil in Containern transportiert. Dabei sind die Stückgüter häufig weiterhin in den genannten Behältnissen verpackt, es werden dann aber wiederum mehrere dieser Behältnisse in einem Container verstaut. Ein Container ist ein „[...] standardisiertes Lade- und Transporthilfsmittel in Form eines abschließbaren Transportbehälters“ (Bichler et al., 2011, S. 33). Im Zeitraum von 1980 bis 2010 hat sich der Anteil der in Containern transportierten Stückgüter am gesamten Stückgutverkehr von knapp 10 % auf 61 % erhöht (siehe Abbildung 2.3).

Obwohl bereits im 19. Jahrhundert auf einigen Eisenbahnstrecken in England containerartige Behälter eingesetzt wurden, die in einem Stück entladen werden

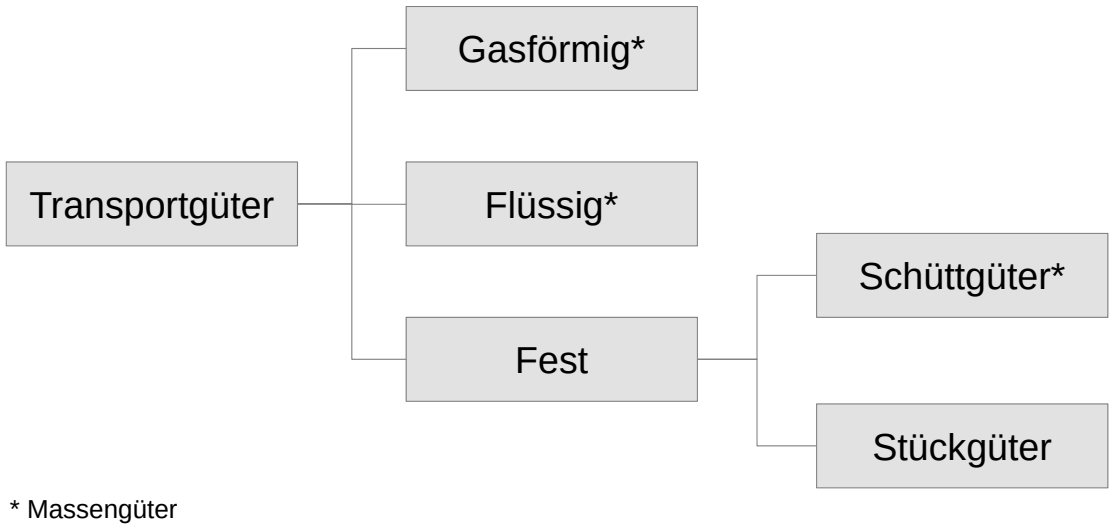


Abbildung 2.2: Klassifikationsschema für Transportgüter (Eigene Darstellung nach Gudehus, 2010, S. 775)

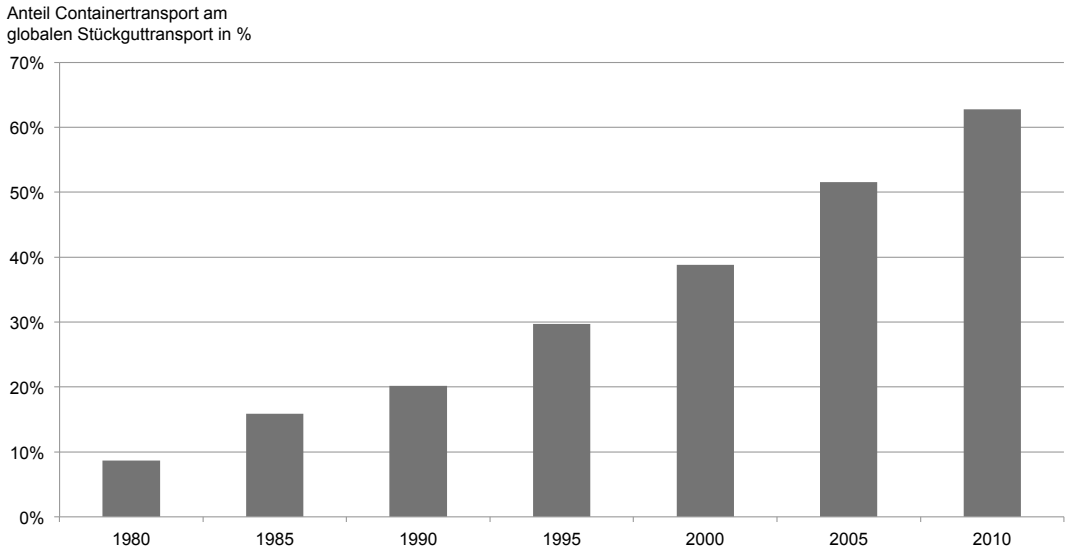


Abbildung 2.3: Anteil des Containertransports am globalen Stückguttransport (Eigene Darstellung nach UNCTAD, 2010, S. 31)

konnten (vgl. Rackwitz, 1969, S. 4f.), gilt das Jahr 1953 als eigentliches Geburtsjahr der Containerisierung (vgl. hierzu und im Folgenden Levinson, 2006, S. 37ff.). Der US-Amerikaner Malcolm McLean war als Inhaber einer Spedition tagtäglich mit den Ineffizienzen des Stückguttransports mit individuellen Verpackungs- und Transportschachtelgrößen konfrontiert. Das Umladen der einzelnen, unterschiedlich dimensionierten Kisten von einem LKW auf ein anderes Transportmittel wie z.B. ein Schiff war ein hochgradig manueller, langwieriger und damit kostenintensiver Prozess. 1953 kam McLean auf die Idee, die Dauer des Umladeprozesses dadurch zu verkürzen, dass die LKW-Anhänger direkt auf die Schiffe geladen wurden. Kurz darauf verbesserte McLean diese Idee dahingehend, dass er nicht mehr die ganzen Anhänger, sondern nur noch die eigentlichen Transportbehälter (also Anhänger ohne Fahrgestell) umladen musste. Dies führte auf zwei Arten zu einer besseren Flächennutzung: Zum einen konnten die Transportbehälter in der Ebene praktisch lückenlos nebeneinander gestellt werden, zum anderen konnte man sie aber auch vertikal stapeln.

In den folgenden zwei Jahrzehnten entwickelten sich auf der ganzen Welt zahlreiche, wegen unterschiedlicher Abmessungen, Materialbeschaffenheiten und Belastungsspezifikationen inkompatible Containersysteme. Ende der 1960er Jahre verabschiedete die internationale Standardisierungsorganisation ISO erstmals global gültige Standards für Transportcontainer, die eine allgemeingültige Terminologie, eine Klassifikation, Abmessungsvorschriften sowie geeignete Testmethoden für Frachtcontainer umfassen. Der heutzutage am weitesten verbreitete ISO-Containertyp ist der Allzweckcontainer (*general purpose dry cargo container*), der in verschiedenen Dimensionen spezifiziert ist. Die zwei dominierenden Dimensionen sind (vgl. Hapag-Lloyd, 2010, S. 5):

- 20-Fuß-Container
  - Länge (Außenmaß): 20 Fuß / 6,058 Meter
  - Breite (Außenmaß): 8 Fuß / 2,438 Meter
  - Höhe (Außenmaß): 8 Fuß, 6 Zoll / 2,591 Meter
  - Leergewicht: 2.330 kg

  - Maximalgewicht (brutto): 24.000 kg

- 40-Fuß-Container
  - Länge (Außenmaß): 40 Fuß / 12,192 Meter
  - Breite (Außenmaß): 8 Fuß / 2,438 Meter
  - Höhe (Außenmaß): 8 Fuß, 6 Zoll / 2,591 Meter
  - Leergewicht: 4.000 kg
  - Maximalgewicht (brutto): 30.480 kg[1]

Zwischen 1985 und 2004 hat sich die Anzahl der weltweit eingesetzten Container von 4,8 Millionen auf 28,5 Millionen erhöht und damit fast versechsfacht. 2004 hatten die 40-Fuß-Container und die 20-Fuß-Container dabei mit 73 % bzw. 18 % den größten Anteil (vgl. Stopford, 2009, S. 511). Die hohe Verbreitung dieser zwei Containertypen führte auch dazu, dass sich deren Volumina als Referenzgrößen etablieren konnten. Die Volumeneinheit TEU (*twenty-foot equivalent unit*) entspricht dem Innenvolumen eines 20-Fuß-Containers. Analog dazu ist die Volumeneinheit FEU (*forty-foot equivalent unit*) definiert, die das Innenvolumen eines 40-Fuß-Containers bezeichnet. Folglich gilt die Beziehung 1 FEU = 2 TEU.

Ein Nachteil der ISO-Spezifikation im täglichen Transport ist die Tatsache, dass die Dimensionen nicht optimal auf die im europäischen Raum weit verbreiteten EURO-Paletten abgestimmt sind (vgl. Beuthe, 2007, S. 78). So bleiben selbst bei einer optimalen Stapelung der EURO-Paletten in einem 20-Fuß-Container knapp 20 % des Volumens ungenutzt (vgl. Vahrenkamp, 2011, S. 281).

Vom Allzweckcontainer existieren Varianten mit Belüftung und mit offenem Dach. Darüber hinaus gibt es angepasste Container für spezielle Ladungen, z.B. isolierte Thermalcontainer (teilweise zusätzlich mit Kühlung oder Heizung) oder auch Container für Massengut (Tankcontainer, Schüttgutcontainer). Die ISO-Container sind im See- und Landtransport allgegenwärtig. Im Bereich der Luftfracht ist dagegen die „Unit Load Device“ (ULD) genannte Norm maßgeblich, die von der internationalen

[1] Das Maximalgewicht eines 40-Fuß-Containers ist nicht wesentlich höher als das eines 20-Fuß-Containers. Dies liegt vor allem daran, dass bei einem wesentlich höheren Gewicht der Landtransport mit LKW nicht mehr möglich wäre.

Luftfahrtorganisation IATA verabschiedet wurde (vgl. Mensen, 2013, S. 99). ISO- und ULD-Container weisen unterschiedliche Dimensionen auf.

Der Transport von Stückgütern in Containern weist gegenüber der konventionellen Art der Beförderung mit unterschiedlichen Verpackungsgrößen und -arten eine Reihe von Vorteilen auf (vgl. hierzu und im Folgenden Obermaier et al., 2007, S. 314ff.):

- In einem Container können in der Regel wesentlich mehr Einheiten der zu transportierenden Ware eingelagert werden als in den ursprünglich verwendeten Verpackungen. Tatsächlich wird die Ware meist komplett in Kisten oder ähnlichen Verpackungsformen im Container gestapelt. Entsprechend müssen über den gesamten Transportweg wesentlich weniger Objekte umgeladen werden bzw. kann der gesamte Umladeaufwand wesentlich reduziert werden.

- Container sind aus Stahl gefertigt und äußerst robust. Dadurch schützen sie zum einen die Ware effektiv vor Beschädigung auf dem Transportweg, sind zum anderen aber auch selbst über viele Jahre verwendbar.

- Durch die genormten Dimensionen, die Quaderform und die hohen Anforderungen an die Druckbelastbarkeit lassen sich Container gut aufeinander stapeln (die ISO-Norm spezifiziert, dass ein Container dem Druck von mindestens 6 vollen Containern standhalten muss). Dadurch lässt sich die Flächennutzung innerhalb der Transportmittel, speziell innerhalb von Schiffen, sowie bei der Zwischenlagerung an Land erheblich verbessern.

- Die stetig steigende Verbreitung der Container macht es ökonomisch sinnvoll, hochspezialisierte Anlagen und Maschinen wie Kräne oder autonom steuernde Transportfahrzeuge für den Transport und das Umladen von Containern zu entwickeln. Damit lassen sich immer mehr Container in immer kürzerer Zeit umschlagen.

- Die Containerisierung ermöglicht den wirtschaftlichen Betrieb einer durchgehenden Transportkette vom Produzenten zum Konsumenten, über unterschiedliche Transportmodalitäten hinweg.

Alle genannten Vorteile führen zu erheblichen Zeit- und Kosteneinsparungen, weswegen der Transport von Waren auch über immer weitere Distanzen wirtschaftlich

interessant wurde. Mittlerweile sind die Transportkosten für die meisten Waren derart reduziert worden, dass sie in Wirtschaftlichkeitsbetrachtungen fast keine Rolle mehr spielen. So liegt der Anteil der Kosten für den Transport eines Fernsehgeräts von Asien nach Europa bei nur 1,4 % der Gesamtkosten (vgl. Bundeszentrale für politische Bildung, 2009, S. 5). Die Containerisierung gilt deshalb als eine wesentliche Triebfeder und der Container als ein Symbol für die als Globalisierung bezeichnete Entwicklung zu weltweit immer enger verflochtenen Volkswirtschaften.

2.2 Containerschifffahrt

Der Siegeszug des Containers hatte auf wohl keine Branche so direkte, strukturverändernde Auswirkungen wie auf die Schifffahrtsbranche (vgl. hierzu und im Folgenden Stopford, 2009, S. 33ff.). In der ersten Hälfte des zwanzigsten Jahrhunderts wurden für den Stückguttransport zur See universelle Transportschiffe eingesetzt, die unterschiedliche Typen von Stückgütern und häufig dazu auch Passagiere befördern konnten.

Beim Passagiertransport machten schließlich die Fluggesellschaften den Schifffahrtsunternehmen zunehmend Konkurrenz und verdrängten sie in den folgenden Jahrzehnten fast vollständig. Einzig im Rahmen von Kreuzfahrten werden heutzutage noch in nennenswertem Umfang Passagiere per Schiff transportiert.

Beim Gütertransport konnten sich die Schiffe dagegen weitgehend als Transportmittel gegenüber Flugzeugen behaupten. Ausnahmen bilden bestimmte Güterklassen, z.B. Güter mit besonderer Eilbedürftigkeit. Eine Verdrängung vollzog sich allerdings innerhalb der Schifffahrtsbranche. Ein wesentlicher Nachteil der universellen Transportschiffe waren die erheblichen Liegezeiten am Hafen, bedingt durch den sehr personalintensiven Umschlagprozess. Als Gegenmaßnahme wurden nun nach und nach immer größere und spezialisiertere Schiffe entwickelt, durch die Skaleneffekte realisiert werden konnten und deren Betrieb, insbesondere bei der Be- und Entladung, wesentlich weniger Arbeitskraft benötigte. Heutzutage lassen sich vier verschiedene Segmente des Gütertransports per Schiff und damit verbundene spezialisierte Schiffstypen unterscheiden:

- Trockene Massengutschifffahrt: Massengutfrachter
- Nasse Massengutschifffahrt: Tanker
- Containerschifffahrt: Containerschiffe
- Sonstige Schifffahrt: Spezialschiffe, z.B. für Forschungsaufgaben oder zum Transport von Stückgütern, die sich aufgrund ihrer Größe oder Schwere nicht für den Transport in Containern eignen.

Der Transport der Container zur See erfolgt durch Containerschiffe und Feederschiffe. Letztere können nur einen Bruchteil der Ladung eines Containerschiffs transportieren, kommen aber immer dann zum Einsatz, wenn ein Containerschiff für eine bestimmte Route wegen des benötigten Tiefgangs nicht eingesetzt werden kann oder wenn wegen der geringen Anzahl umzuschlagender Container der Einsatz eines Containerschiffes ökonomisch nicht sinnvoll wäre. Als Tiefgang wird der Abstand vom tiefsten Punkt des Schiffes bis zur Wasseroberfläche bezeichnet. Somit bestimmt der Tiefgang die minimale Wassertiefe, in der ein Schiff fahren kann, ohne auf Grund zu laufen.

Die ersten Containerschiffe boten nur Kapazitäten von wenigen hundert TEU (vgl. Cudahy, 2006, S. 257f.), während Containerschiffe Mitte der 1980er Jahre bereits mehrere tausend TEU laden konnten (vgl. Lim, 1994, S. 151). Seit Mitte der 1990er Jahre sind die Ladekapazitäten erneut stark angestiegen und erreichen mittlerweile Werte deutlich über 10.000 TEU (vgl. hierzu und im Folgenden UNCTAD, 2010, S. 30f.). Die entsprechenden Containerschiffe weisen eine Länge von knapp 400 Metern, eine Breite von 60 Metern bei einem Tiefgang von 16 Metern auf.

Anfang 2010 bestand die weltweite Containerschifffflotte aus 4.677 Schiffen[2], mit einer Gesamtkapazität von 12,8 Millionen TEU. Im Durchschnitt konnte ein Containerschiff demnach ca. 2.700 TEU transportieren. Verglichen mit dem Jahr 1997 hat sich die Anzahl der Containerschiffe weltweit mehr als verdoppelt, während sich die Gesamtkapazität sogar mehr als vervierfacht hat.

[2]Hier werden nur die zellulären Containerschiffe, also solche Schiffe, die spezielle Ladevorrichtungen für die geordnete Unterbringung der Container vorweisen, gezählt.

3 Containerterminals

Die Containerterminals an Seehäfen sind neuralgische Punkte des Containertransportsystems, an denen der - zeitlich durch das Zwischenlager entzerrte - Containerfluss zwischen Schiffen und Landtransportmitteln hergestellt wird. Neben Containerterminals an Seehäfen gibt es auch Containerterminals im Inland, an denen zum Beispiel der Umschlag zwischen Bahn und LKW realisiert wird. Diese sind ähnlich zu den Containerterminals an Seehäfen organisiert, sind aber in der Regel erheblich kleiner und schlagen auch wesentlich weniger Container um. Im weiteren Verlauf der Arbeit werden ausschließlich Containerterminals an Seehäfen betrachtet.

3.1 Grundlagen

Der wesentliche Zweck eines Containerterminals an einem Seehafen ist es, den für den Containertransport notwendigen Wechsel zwischen See- und Landtransportmitteln zu ermöglichen. Ein Containerterminal kann als System aus drei Komponenten verstanden werden: Der Seeseite, dem Zwischenlager und der Landseite (siehe Abbildung 3.1). An der Seeseite des Containerterminals docken Schiffe an. Dort werden sowohl die für den Import in den Containerterminal vorgesehenen Container (Importcontainer) aus den Schiffen entladen als auch die für den Weitertransport mittels der andockenden Schiffe vorgesehenen Container (Exportcontainer) in das Schiff eingeladen. Das Zwischenlager dient als Puffer für die entladenen und die zu ladenden Container. An der Landseite des Containerterminals werden die Container auf landseitige Transportmittel wie LKW oder Züge geladen bzw. wird deren Ladung entgegengenommen. Der Transport der Container innerhalb des Terminals erfolgt mit Hilfe von Kränen und speziellen Fahrzeugen.

Container, die das Zwischenlager über die Seeseite erreicht haben, können es potenziell nicht nur über die Landseite, sondern auch wieder über die Seeseite verlassen. Der Container wird in diesem Fall bis zu seiner Einlagerung in das Zwischenlager als Importcontainer und danach als Exportcontainer behandelt. Ein Containerterminal, bei dem der Großteil des Containerumschlags ohne Einbezug der Landseite erfolgt, wird auch als Transshipment-Terminal bezeichnet. Transshipment-Terminals werden im Folgenden nicht gesondert thematisiert.

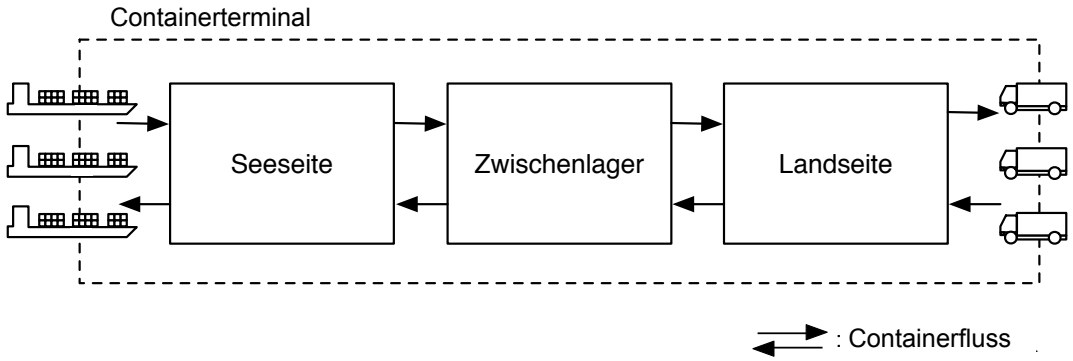


Abbildung 3.1: Komponenten eines Containerterminals

Die Größe eines Containerterminals wird üblicherweise anhand des Volumens der dort jährlich umgeschlagenen Container gemessen. Wie bereits erwähnt dient dabei das Volumen eines 20-Fuß-Containers unter dem Begriff TEU als Basisgröße (*twenty-foot equivalent unit*), auch wenn mittlerweile Container der doppelten Länge gebräuchlicher sind. Ein 40-Fuß-Container fasst dementsprechend 2 TEU oder eine FEU (*forty-foot equivalent unit*).

An Containerterminals werden enorme Mengen von Containern umgeschlagen (vgl. hierzu und im Folgenden The World Bank, 2014). Im Jahr 2011 waren es weltweit 572 Millionen TEU. Seit dem Jahr 2000 hat sich das Volumen der über Containerterminals transportierten Waren mehr als verdoppelt (siehe Abbildung 3.2). Die vier größten Containerterminals, in denen im Jahr 2011 jeweils mehr als 20 Millionen TEU umgeschlagen wurden, befinden sich allesamt in Asien (vgl. hierzu und im Folgenden UNCTAD, 2012, S. 83). Der Port of Shanghai ist der größte Containerterminal der Welt. Dort wurden 2011 über 30 Millionen TEU verladen. Die globale Wirtschaftskrise wirkte sich überaus heftig auf die Containerwirtschaft

aus, so dass 2009 erstmals seit Jahrzehnten ein Rückgang der umgeschlagenen Mengen im Vergleich zum Vorjahr zu beobachten war. In den Folgejahren zeichnete sich allerdings eine rasche Erholung ab.

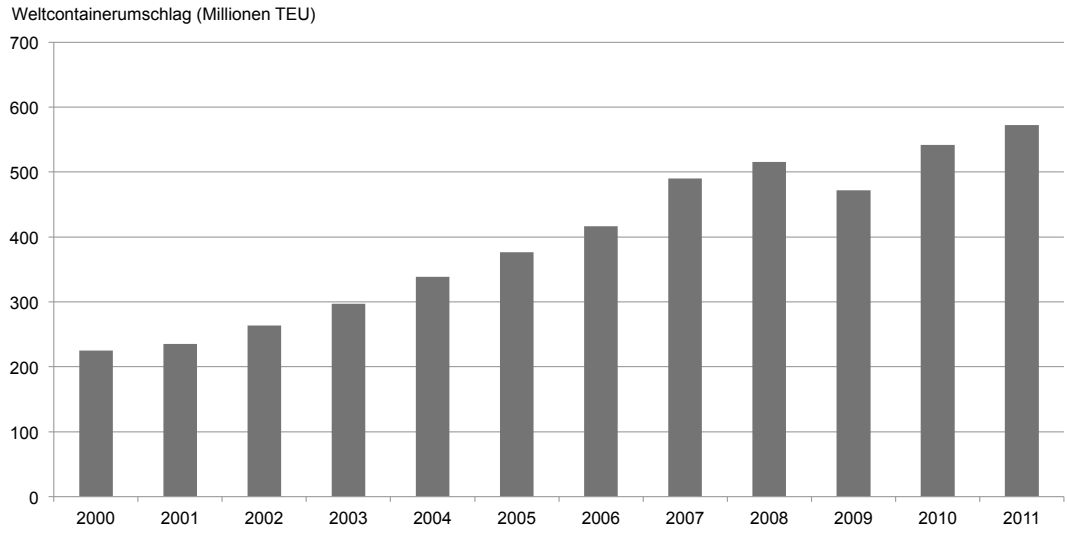


Abbildung 3.2: Entwicklung der weltweit an Containerterminals umgeschlagenen Container (Eigene Darstellung nach The World Bank, 2014)

Um die oben bezifferten Transportleistungen erbringen zu können, sind hochspezialisierte Anlagen und Maschinen sowie fein aufeinander abgestimmte Abläufe notwendig. In den folgenden Abschnitten werden die drei Komponenten eines Containerterminals im Hinblick auf die eingesetzten Anlagen und Maschinen sowie die wesentlichen Prozesse detaillierter beschrieben.

3.2 Komponenten

3.2.1 Seeseite

Die Seeseite stellt die Verbindung zwischen dem Containerterminal und verschiedenen Wasserfahrzeugen her, typischerweise Containerschiffe (*container ships*) und Feeder-Schiffe (*feeder vessels*). Zum Be- und Entladen von Containern steuern die

Schiffe als erstes den ihnen zugewiesenen Liegeplatz (*berth*) am Kai (*quay*) des Containerterminals an. Mittels Containerbrücken (*quay cranes; container cranes*) werden die Container vom Festland auf das Schiff bzw. vom Schiff auf das Festland befördert. Eine Containerbrücke (siehe Abbildung 3.3) ist ein auf Schienen laufender Portalkran mit einem land- und einem seeseitigen Ausleger (vgl. hierzu und im Folgenden Brinkmann, 2005, S. 254). Die Laufkatze (*trolley*) ist eine Komponente des Krans, die sich horizontal an den Auslegern entlang hin- und herbewegen kann. An der Laufkatze ist ein vertikal justierbarer Greifarm (*spreader*) befestigt, mit dem die Container aufgenommen werden können.

Eine typische Containerbrücke bewältigt operativ zwischen 25 und 35 Containerbewegungen pro Stunde. Ein Schiff wird üblicherweise von mehreren Containerbrücken (aktuell bis zu 6 Stück) simultan bedient (vgl. Meisel, 2009, S. 11). Üblicherweise werden die Ent- und Beladevorgänge für eine Bucht des Containerschiffs sequenziell durchgeführt (*single-cycle*). Effizienter kann der Umschlag dadurch erfolgen, dass beim Entladen eines Containers an Land beim Rückweg der Laufkatze zum Schiff nach Möglichkeit ein für den Weitertransport bestimmter Container mitgeführt wird (vgl. Zhang und Kim, 2009, S. 979). Eine weitere Maßnahme zur Erhöhung der Produktivität der Containerbrücken ist das Twinlift-Verfahren, bei dem der Spreader zwei 20-Fuß-Container auf einmal aufnehmen kann. Neuere Containerbrücken erlauben es teilweise sogar, zwei 40-Fuß-Container bzw. vier 20-Fuß-Container auf einmal zu transportieren.

Auf der Landseite werden die Container von den Kränen in der Regel direkt auf wartende Fahrzeuge abgeladen, die die Container ins Zwischenlager transportieren. Alternativ können die Container auch temporär am Boden unter der Containerbrücke abgestellt werden, wo sie von den Transportfahrzeugen - entsprechende Hebevorrichtungen vorausgesetzt - aufgenommen werden.

Zum Transport zwischen den Bereichen eines Containerterminals kommen, je nach Ausrichtung und Design des Containerterminals, unterschiedliche Fahrzeugtypen zum Einsatz (vgl. Günther und Kim, 2006, S. 440f.). Am häufigsten werden spezielle LKW eingesetzt, auf die jeweils ein 40-Fuß-Container bzw. zwei 20-Fuß-Container geladen werden können. Diese LKW werden meistens von menschlichen Fahrern gesteuert, in Ländern mit relativ hohen Arbeitskosten finden sich aber auch vollautomatische Transportfahrzeuge (*automated guided vehicles, AGV*). Bei AGVs

Abbildung 3.3: Containerbrücken beim Be- und Entladen eines Containerschiffs am Containerterminal Laem Chabang 1 (Thailand) (Quelle: Maersk Line)

handelt es sich um unbemannte LKW zum terminalinternen Transport von Containern, die zentral von einer Software koordiniert werden. Zum schnellen Be- und Entladen eines Schiffes wird jeweils eine ganze Flotte von Transportfahrzeugen parallel eingesetzt, im Regelfall ca. 30 AGV pro Schiff (vgl. Meersmans und Dekker, 2001, S. 10).

Den „passiven“ LKW stehen „aktive“ Fahrzeuge gegenüber, die mit Vorrichtungen ausgestattet sind, durch die sie Container heben können. Solche Fahrzeuge kommen also auch ohne Beladung durch einen Kran aus. Dazu gehören Greifstapler (*reach stacker*), die Container mittels eines Teleskoparmes vom Boden oder auch von höheren Ebenen (bis max. 6 Containerhöhen, vgl. Brinkmann, 2005, S. 267) aufgreifen können, sowie Portalhubwagen (*straddle carrier*), die mit ihrem Rahmengestell über einen Container fahren und diesen mit einer Hubvorrichtung emporheben (bis max. 4 Containerhöhen, vgl. Brinkmann, 2005, S. 267). Die aktive Variante der AGVs sind die vollautomatischen Hubfahrzeuge (*automated lifting vehicles, ALV*). Diese Fahrzeuge können Container nicht nur vollautomatisch transportieren, sondern auch emporheben bzw. wieder ablegen.

In allen Fällen transportieren die Fahrzeuge die Container in das Zwischenlager.

3.2.2 Zwischenlager

Im Zwischenlager werden die Container übereinander in Stapeln (*stacks*) aufbewahrt. Durch die Stapelung kann die Fläche des Zwischenlagers besser ausgenutzt werden. Im Gegenzug verschlechtert sich aber die Zugänglichkeit, da immer nur der oberste Container eines Stapels per Kran bewegt werden kann. Je höher die Container gestapelt werden, desto wahrscheinlicher tritt der Fall ein, dass ein abzutransportierender Container von einem oder mehreren anderen Containern blockiert wird. Die Stapelhöhe ist unter anderem durch den zulässigen Bodendruck, abhängig vom Untergrundmaterial und dem Fundament des Zwischenlagers, begrenzt. Eine Alternative zur Bodenstapelung ist die Unterbringung der Container auf Chassis, also Auflegern für Zugmaschinen. Die Container können so ohne Hebevorrichtungen wieder aus dem Zwischenlagern entnommen werden. Weil das Chassissystem wegen der fehlenden Stapelmöglichkeit eine deutlich geringere Flächennutzung sowie zudem Kosten für die Chassis mit sich bringt, wurde es größtenteils durch die Bo-

denstapelung verdrängt. Im Verlauf der Arbeit wird das Chassisystem daher nicht weiter betrachtet.

Mehrere nebeneinander angeordnete Stapel werden als Bucht (*bay*) bezeichnet. In heutigen Containerterminals besteht eine Bucht normalerweise aus 6 bis 8 Containerstapeln (Ng, 2005, S. 3). Nebeneinander liegende Buchten werden wiederum zu Blöcken (*blocks*) zusammengefasst. Ein Block besteht aus ca. 20 Buchten (Murty, 2007, S. 2) und ist ca. 170m bis 180m lang (Kim et al., 2007, S. 690). Mehrere angrenzende Blöcke bilden eine Zone (*zone*).

Abbildung 3.4 visualisiert eine Zone in einem Zwischenlager, die aus drei Blöcken besteht. Jeder Block besteht aus vier Buchten à 8 Containern. Die Container sind so im Zwischenlager ausgerichtet, dass sich die Stapelkräne parallel zu der längeren Seite der Container bewegen können.

Häufig wird das Zwischenlager funktional in zwei Bereiche gegliedert (vgl. Meersmans und Dekker, 2001, S. 13). In einem Bereich werden ausschließlich Importcontainer (für den Weitertransport über Land) gelagert, im anderen nur Exportcontainer (für den Weitertransport per Schiff) aufbewahrt. Zum Zeitpunkt des Einlagerns der Importcontainer im Zwischenlager ist häufig nicht absehbar, wann ein einzelner Container weitertransportiert werden wird. Um die Wahrscheinlichkeit zu verringern, dass beim Abtransport eines Importcontainers dieser von anderen, darüber gestapelten Containern blockiert wird, werden die Importcontainer in der Regel niedriger gestapelt als die Exportcontainer, für die der Zeitpunkt des Weitertransports mit weniger Unsicherheit geplant werden kann. Gelegentlich werden auch dedizierte Pufferzonen vorgesehen, in denen Container temporär gelagert werden können, bevor sie an ihre endgültige Position im Zwischenlager transportiert werden. Damit erreicht man eine höhere Geschwindigkeit beim Entladen von Schiffen, verliert aber im Anschluss Zeit für das erneute Handling der Container.

In Containerterminals, bei denen für das Zwischenlager nur wenig Platz zur Verfügung steht, wie z.B. bei den meisten großen Containerterminals in Asien, wird auf eine funktionale Gliederung des Zwischenlagers und gesonderte Pufferzonen verzichtet, um den vorhandenen Platz besser ausnutzen zu können (vgl. Zhang et al., 2003, S. 6).

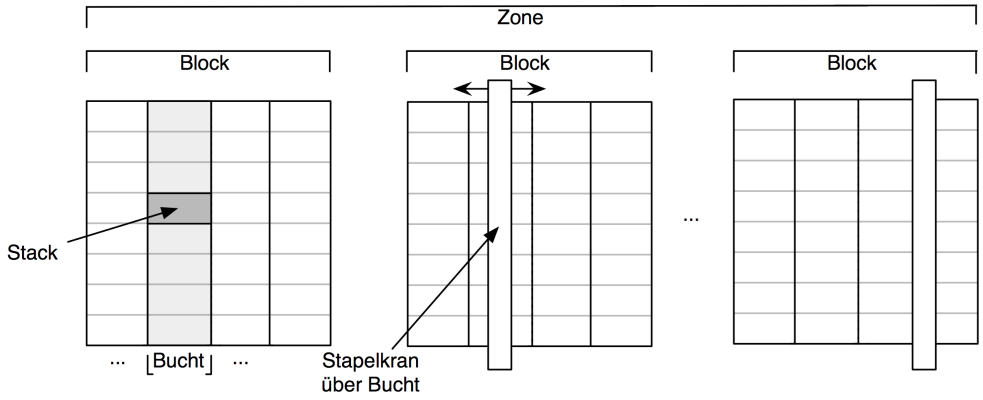


Abbildung 3.4: Schematische Darstellung einer Zone eines Zwischenlagers (Draufsicht)

Je nachdem, welche Anlagen und Maschinen im Containerterminal verwendet werden, lassen sich zwei verschiedene Systeme der Bewirtschaftung des Zwischenlagers unterscheiden (vgl. Moccia et al., 2006, S. 46):

1. Direkte Transfersysteme: Die für den Transport zwischen den Bereichen des Containerterminals verwendeten Fahrzeuge legen die Container direkt auf der für sie vorgesehenen Position im Zwischenlager ab bzw. können sie direkt von der aktuellen Position im Zwischenlager aufnehmen. Dies bedingt, dass für den Transfer aktive Fahrzeuge wie z.B. Greifstapler eingesetzt werden. Das direkte Transfersystem bringt einen erhöhten Flächenbedarf im Zwischenlager mit sich, denn es müssen relativ viele Spuren gezogen werden, damit alle Stapel des Zwischenlagers für die Fahrzeuge zugänglich sind. Die Fläche für die Spuren steht natürlich nicht mehr für die Ablage von Containern zur Verfügung. Direkte Transfersysteme sind hauptsächlich in Europa und Nordamerika verbreitet.

2. Indirekte Transfersysteme: Für das Be- und Entladen der Container im Zwischenlager werden andere Maschinen eingesetzt als zum Transport der Container zwischen Seeseite und Zwischenlager bzw. zwischen dem Zwischenlager und der Landseite. Für das Be- und Entladen der Container kommen Portalkräne (*gantry cranes*, im Folgenden auch als Stapelkräne bezeichnet) zum Einsatz, durch die wesentlich höhere Stapelhöhen möglich sind als beim direkten Transfersystem. Darüber hinaus lassen sich die Kräne im direkten Transfersystem

wesentlich einfacher automatisieren als dies beim indirekten Transfersystem der Fall ist, was eine höhere Produktivität bei geringeren Lohnkosten verspricht. Zusätzlich ist die Flächennutzung gegenüber dem direkten Transfersystem verbessert, da weniger Spuren für die Kräne reserviert werden müssen. Vor allem der letzte Vorteil hat dazu geführt, dass das indirekte Transfersystem vor allem in Asien mit seinen großen Containerterminals das dominierende Bewirtschaftungssystem im Zwischenlager ist.

Wegen der kompakteren Stapelung wird das Zwischenlager im indirekten Transfersystem auch Blocklager genannt, während es im direkten Transfersystem wegen der größeren Flächenausbreitung als Flächenlager bezeichnet wird (vgl. Brinkmann, 2005, S. 240f.).

Die im indirekten Transfersystem eingesetzten Stapelkräne bestehen aus einem Gerüst (siehe Abbildung 3.5), das sich quer zu den Buchten des Zwischenlagers bewegen lässt. An der oberen Verbindungsbrücke befindet sich ein bewegliches Element, das Laufkatze genannt wird (*trolley*), an dem wiederum eine Vorrichtung zum Greifen eines Containers befestigt ist (*spreader*). Ein Stapelkran kann immer nur auf den obersten Container eines Stapels zugreifen sowie einen Container immer nur auf dem obersten Container eines Stapels (bzw. auf dem Boden oder einem Fahrzeug zum Abtransport) ablegen. Dabei wird zunächst das Gerüst zur entsprechenden Bucht gefahren, dann die Laufkatze über dem gewünschten Stapel positioniert und schließlich der Container aufgegriffen bzw. abgeladen.

Die Stapelkräne lassen sich entweder auf Schienen (*rail-mounted gantry crane, RMG*) oder auf Gummireifen (*rubber-tyred gantry crane, RTG*) bewegen. Mit RMGs kann, wegen ihrer großen Dimensionen, zur Zeit mit 1100 TEU pro Hektar die beste Flächennutzung im Zwischenlager erzielt werden (vgl. Günther und Kim, 2006, S. 440). Für RTGs liegen die Auslastungswerte bei 1.000 TEU pro Hektar, während die im direkten Transfersystem eingesetzten Fahrzeuge nur zwischen 500 und 750 TEU pro Hektar erreichen. RMGs können nur zwischen den Blöcken *einer* Zone hin- und herfahren, da die Gleise der Zonen nicht verbunden sind. RTGs können dagegen auch zu Blöcken aus anderen Zonen bewegt werden. Dies wird aber nach Möglichkeit aus zwei Gründen vermieden: Erstens sind derartige Zonenwechsel relativ zeitaufwändig und verschlechtern daher die Produktivität eines RTGs. Zweitens haben RTGs einen sehr großen Wendekreis, so dass sie beim Zonenwechsel zeitwei-

Abbildung 3.5: Ein Stapelkran (RTG) (Quelle: Konecranes)

se die gesamte Fahrspur blockieren und somit auch die Produktivität der internen Transportfahrzeuge negativ beeinflussen können.

Ein Block kann von mehreren Kränen simultan bearbeitet werden, wobei die Kräne dabei in der Regel nicht aneinander vorbeifahren können. Zur Vermeidung von Unfällen wird vielmehr empfohlen, dass die Stapelkräne zu jedem Zeitpunkt einen Sicherheitsabstand von mindestens drei Buchten einhalten. Eine Ausnahme bildet das Double-Rail-Mounted-Gantry-Crane-System (DRMG-System) (vgl. Cao et al., 2008, S. 221). In diesem System wird ein Block von zwei RMGs unterschiedlicher Höhe und Breite bearbeitet, für die jeweils ein separates Schienenpaar zur Verfügung steht. Dadurch kann ein Kran über bzw. unter dem anderen Kran vorbeifahren.

Die Anzahl an Stapeln in einer Bucht sowie die maximale Stapelhöhe werden auch durch die Abmessungen der verwendeten Stapelkräne limitiert. Die in Containerterminals eingesetzten Kräne sind typischerweise zwischen vier und sechs Containerhöhen hoch und sechs bis acht Containerbreiten breit (vgl. Bohrer, 2005, S. 5). Die Maximalwerte liegen für RTGs bei sieben Containerbreiten und sechs Containerhöhen, während RMGs wesentlich größere Spannweiten (fünf bis 24 Containerbreiten) und höhere Stapelhöhen (bis zu neun Containerhöhen) aufweisen (vgl. Brinkmann, 2005, S. 267f.). Bezüglich der Containerhöhe ist zu beachten, dass die maximale Stapelhöhe mindestens eine Containerhöhe niedriger als die Kranhöhe ist, damit ein aufgegriffener Container kollisionsfrei über alle Stapel bewegt werden kann. Ebenso muss mindestens eine Containerbreite frei bleiben, um das Abladen der Container auf ein Fahrzeug zu ermöglichen.

Grundsätzlich steigt mit wachsenden Stapelhöhen die Wahrscheinlichkeit, dass ein abzutransportierender Container von einem oder mehreren anderen Containern blockiert wird. In diesem Fall müssen die blockierenden Container zunächst umgestapelt werden, wodurch die mittlere Umschlagsdauer für einen Container deutlich erhöht wird. Daher sind in der Praxis Vorgaben in Form von Stapelrichtlinien üblich, nach denen - auch bei höheren Kränen - die Stapel auf maximal fünf Containerhöhen begrenzt werden (vgl. Brinkmann, 2005, S. 248).

Eine Kranoperation bezeichnet das Aufheben eines Containers von seiner aktuellen Position und das Ablegen des Containers auf einer neuen Position. Ein Stapelkran benötigt für eine Operation im Mittel zwischen 1 und 2 Minuten (vgl. Kim

	RTGs	RMGs	DRMGs
Geschwindigkeit Krangerüst (m/s)	2,0	4,0	3,0 / 3,5
Geschwindigkeit Laufkatze (m/s)	1,2	0,8 - 1,1	1,0
Geschwindigkeit Spreader (beladen, m/s)	0,3	0,3 - 0,6	1,0
Geschwindigkeit Spreader (unbeladen, m/s)	0,7	0,6 - 1,0	1,0

Tabelle 3.1: Technische Kennzahlen für die gebräuchlichsten Stapelkrantypen (Quelle: Stahlbock und Voß, 2008, S. 31)

und Hong, 2006, S. 2), und absolviert im normalen Betrieb um die 25 Operationen pro Stunde (vgl. Murty, 2007, S. 2). Wie aus Tabelle 3.1 ersichtlich ist, sind RMGs den RTGs in Hinblick auf die für die Dauer einer Operation wichtigen technischen Kennzahlen leicht überlegen. Mit RMGs können also ceteris paribus in der gleichen Zeiteinheit mehr Operationen durchgeführt werden als mit RTGs. DRMGs nehmen in Bezug auf die Krangerüstgeschwindigkeiten eine Mittelposition ein.

Im Zwischenlager eines großen Containerterminals sind in der Regel viele Kräne (auch unterschiedlicher Krantypen) gleichzeitig aktiv (vgl. Brinkmann, 2005, S. 308): Der Hafen von Hong Kong zum Beispiel hat im Zwischenlager eine Gesamtkapazität von 104.000 TEU, kann also 52.000 40-Fuß-Container lagern. Es kommen gleichzeitig bis zu 127 RTGs und 24 RMGs zum Einsatz. Die Kranoperationen sind dabei weitgehend automatisiert.

3.2.3 Landseite

An der Landseite des Containerterminals erfolgt der Containerumschlag von und zu landseitigen Transportmitteln. In den meisten Fällen handelt es sich bei den landseitigen Transportmitteln um LKW, teilweise aber auch um Züge. Der Transport zwischen Landseite und Zwischenlager erfolgt mit den bereits vorher erwähnten terminaleigenen Transportfahrzeugen.

Werden an der Landseite Stapelkräne verwendet, laden diese die Container, die für den landseitigen Weitertransport vorgesehen sind (Importcontainer), direkt auf die LKW ab bzw. nehmen die Container, die für den Weitertransport per Schiff vorgesehen sind (Exportcontainer), von den LKW auf. Werden an der Landseite Greifstapler oder Portalhubwagen verwendet, halten die LKW auf speziellen Park-

plätzen und werden dort dann be- bzw. entladen. Für den Umschlag mit Zügen kommen häufig RMGs zum Einsatz, es können aber auch die gleichen Maschinen wie für den Umschlag bei LKW Verwendung finden.

Abbildung 3.6 veranschaulicht abschließend die drei Komponenten eines Containerterminals mit den dazugehörigen Anlagen und Maschinen im Zusammenhang.

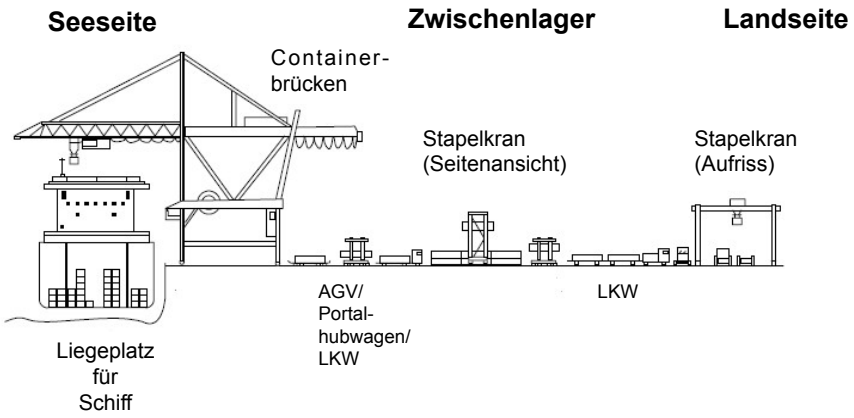


Abbildung 3.6: Anlagen und Maschinen für den Containerumschlag in einem Containerterminal (Quelle: Steenken et al., 2004, S.13)

3.3 Produktivität

Containerterminals stehen miteinander permanent im Wettbewerb um Aufträge und sind darum bemüht, die für die Auftraggeber bedeutenden Leistungskennzahlen zu verbessern (vgl. hierzu und im Folgenden Vacca et al., 2007, S. 4f.). Dabei kann zwischen serviceorientierten und produktivitätsorientierten Kennzahlen unterschieden werden.

Die Auftraggeber interessieren sich vorrangig für die serviceorientierten Kennzahlen, in denen sich der Servicelevel eines Containerterminals widerspiegeln soll. Die wichtigste serviceorientierte Kennzahl ist die Abfertigungsdauer (*turnaround time*)

eines Schiffs. Die Abfertigungsdauer ist die Summe aus vier Zeiten (vgl. Kim und Bae, 1998, S. 655):

- Wartezeit: Die Zeit, die ein Schiff außerhalb des Containerterminals vor Anker liegt, bevor es seinen Liegeplatz ansteuern kann (z.B. weil der Liegeplatz noch von einem anderen Schiff blockiert ist).
- Anlegezeit: Die Zeit, die das Schiff benötigt, um von seiner Warteposition aus zu dem für es vorgesehenen Liegeplatz zu manövrieren.
- Liegezeit: Die Zeit, die das Schiff den Liegeplatz belegt, um be- und entladen zu werden.
- Ablegezeit: Die Zeit, die das Schiff benötigt, um den Liegeplatz wieder freizumachen und den Hafen zu verlassen.

Da ein Containerschiff zwischen 50 % und 66 % der Betriebszeit im Hafen verbringt, wirkt sich die Abfertigungsdauer stark auf die pro Zeiteinheit durch das Schiff transportierte Containermenge und damit auf den Umsatz der Reeder aus (vgl. Hummels, 2007, S. 41). Diese werden sich also bevorzugt für Containerterminals entscheiden, die niedrige Abfertigungsdauern für Containerschiffe realisieren können. Da üblicherweise die Warte- und die Liegezeit betragsmäßig den größten Teil der Abfertigungsdauer repräsentieren, sind diese Werte auch für sich wichtige serviceorientierte Kennzahlen.

Die produktivitätsorientierten Kennzahlen beziehen sich meist auf das Containervolumen, das durch den Containerterminal geleitet wird. Zentrale Kennzahlen sind in diesem Kontext die Anzahl an umgeschlagenen Containern/TEU und die Kranproduktivität (Operationen pro Kranstunde). Es gibt aber auch produktivitätsorientierte Kennzahlen, in denen sich das Containervolumen nicht direkt widerspiegelt, wie z.B. die Auslastung der Liegeplätze (Schiffe pro Jahr und Liegeplatz).

Eine hohe Produktivität ist die Grundlage für eine kurze Liegezeit und damit auch für eine kurze Abfertigungsdauer. Die Erhöhung der Produktivität bringt für Containerterminals, neben der gestiegenen Attraktivität für die Reeder, noch zwei weitere Vorteile mit sich. Zum einen wird dadurch die Kapazität des Containerterminals erhöht, weil mehr Container pro gegebenem Zeitintervall durch das Containerterminal

befördert werden können (vgl. Le-Griffin und Murphy, 2006, S. 1f.). Zum anderen kann ein Containerterminal mit einer höheren Produktivität die gleiche Anzahl an Containern mit weniger Personalstunden umschlagen als ein Containerterminal mit niedrigerer Produktivität. Daher hat ein hochproduktiver Containerterminal entsprechend niedrigere Personalkosten.

Zusammenfassend lässt sich feststellen, dass eine hohe Produktivität im Allgemeinen positiven Einfluss auf wichtige serviceorientierte Kennzahlen hat und damit die Wettbewerbsfähigkeit eines Containerterminals verbessert, die Kapazität des Containerterminals erhöht und die Personalkosten verringert. Entsprechend sind Maßnahmen und Methoden, mit denen die Produktivität von Containerterminals gesteigert werden kann, für diese von hoher ökonomischer Relevanz. Auch bei der Lösung der im Zentrum der vorliegenden Arbeit stehenden Stackingprobleme besteht das übergeordnete Ziel darin, durch eine Beschleunigung der Abläufe im Zwischenlager zu einer höheren Produktivität des Containerterminals beizutragen.

4 Optimierungs- und Entscheidungsprobleme

4.1 Grundlagen

In der Informatik, insbesondere in der Komplexitätstheorie, werden unterschiedliche Kategorien von Problemen betrachtet. Zwei sehr bedeutende Kategorien sind Entscheidungsprobleme und Optimierungsprobleme (vgl. Falkenauer, 1997, S. 4):

- Bei Entscheidungsproblemen soll die Frage beantwortet werden, ob mindestens ein Element s aus einer Menge Ω eine Eigenschaft E besitzt[1]. Die Antwort auf ein Entscheidungsproblem kann also nur entweder Ja oder Nein lauten. Im Fall einer bejahenden Antwort ist auch eine Lösung $s \in \Omega$ zu präsentieren, welche die Eigenschaft E besitzt.

- Bei Optimierungsproblemen möchte man dasjenige Element s aus einer Grundmenge Ω identifizieren, das eine Eigenschaft E besitzt und für das eine gegebene Funktion f, die so genannte Zielfunktion, ihren minimalen (oder maximalen) Funktionswert annimmt.

Schon aus der Ähnlichkeit der Definitionen wird die Verwandschaft der Problemkategorien ersichtlich. Man kann dies ausnutzen, um die Lösung eines Optimierungsproblems durch die Lösung von mehreren Instanzen eines korrespondierenden Entscheidungsproblems zu bestimmen. Dazu führt man eine Schranke für die zu optimierende Größe in die Eigenschaft des Entscheidungsproblems ein. Es sei z.B. ein

[1] Dies umfasst auch den Fall von mehreren Eigenschaften, da sich diese durch Verwendung der booleschen Operatoren UND, ODER und NICHT zu einer kombinierten Eigenschaft zusammenfassen lassen.

Straßennetz gegeben, das verschiedene Orte miteinander verbindet. Die Suche nach einem Weg von einem Ort a zu einem anderen Ort b mit den wenigsten Zwischenstationen in diesem Straßennetz ist ein Optimierungsproblem. Durch sukzessives Lösen von Entscheidungsproblemen der Form „Gibt es einen Weg von a nach b mit weniger als x Zwischenstationen?“ kann man ermitteln, wie viele Zwischenstationen eine optimale Lösung hat. Man beginnt mit $x = 0$ und inkrementiert x solange, bis das Entscheidungsproblem positiv beantwortet wird. x bezeichnet nun die Anzahl der Zwischenstationen in der optimalen Lösung und die Lösung des letzten Entscheidungsproblems ist zugleich eine Optimallösung des Optimierungsproblems.

Bei Optimierungsproblemen geht es darum, für eine gegebene Entscheidungssituation eine oder mehrere Entscheidungen zu treffen, die im Hinblick auf ein oder mehrere Optimierungskriterien (z.B. Umsatz, Kosten oder Zeit) optimal sind. Unterstellt sei zur Vereinfachung, dass nur ein Optimierungskriterium zu betrachten ist. Mathematisch lässt sich dies durch eine reellwertige Funktion mit meist mehreren unabhängigen Entscheidungsvariablen modellieren, wobei jede Entscheidungsvariable einer zu fällenden Entscheidung entspricht. Es muss dann diejenige Variablenbelegung ermittelt werden, für die der Funktionswert minimal bzw. maximal wird (Zielvorschrift). In der Regel werden die gültigen Werte für die Entscheidungsvariablen durch Nebenbedingungen eingeschränkt. Soll eine Entscheidungsvariable x beispielsweise die Anzahl der von einer Maschine zu erzeugenden Produkte eines gewissen Produkttyps repräsentieren, muss durch eine Nebenbedingung $x \geq 0$ sichergestellt werden, dass die Variable keine negativen Werte annehmen kann. Optimierungsprobleme, die mittels linearer Zielfunktion und linearen Nebenbedingungen modelliert werden können, werden als lineare Optimierungsprobleme bezeichnet. Im Folgenden werden nur lineare Optimierungsprobleme behandelt.

Lineare Optimierungsprobleme können unter anderem anhand der Wertebereiche der Entscheidungsvariablen differenziert werden (vgl. Wolsey, 1998, S. 3ff.). Dieser Kategorisierungsdimension folgend, lassen sich die Fälle „kontinuerliche lineare Optimierung“, „(reine) ganzzahlige linerare Optimierung“ und „gemischt-ganzzahlige lineare Optimierung“ unterscheiden:

- Kontinuierliche lineare Optimierung: Alle Variablen können kontinuierliche Werte annehmen, d.h. jede Variable kann alle Werte in mindestens einem (Teil-)Intervall der reellen Zahlen $\mathbb{R}$ annehmen.

- (Reine) Ganzzahlige lineare Optimierung: Alle Variablen müssen ganzzahlige Werte annehmen, d.h. der Wertebereich aller Variablen ist auf die Menge der ganzen Zahlen $\mathbb{Z}$ oder eine Teilmenge von $\mathbb{Z}$ beschränkt.
- Gemischt-ganzzahlige lineare Optimierung: Mindestens eine Variable muss ganzzahlige Werte annehmen und mindestens eine Variable kann kontinuierliche Werte annehmen.

Im weiteren Verlauf des Kapitels werden lineare kombinatorische Optimierungsprobleme, eine spezielle Klasse von linearen Optimierungsproblemen, genauer beleuchtet. Auch bei den im Fokus dieser Arbeit stehenden Stackingproblemen handelt es sich um kombinatorische Optimierungsprobleme. Im Anschluss wird etwas ausführlicher auf das Thema Komplexität von Optimierungs- und Entscheidungsproblemen eingegangen sowie ein Überblick über Verfahrensklassen zur Lösung von kombinatorischen Optimierungsproblemen gegeben.

4.2 Kombinatorische Optimierungsprobleme

Kombinatorische Optimierungsprobleme sind spezielle ganzzahlige Optimierungsprobleme, bei denen der Wertebereich der Entscheidungsvariablen endlich ist (vgl. Papadimitriou und Steiglitz, 1998, S. 2). Dies trifft auf sehr viele praxisrelevante Optimierungsprobleme zu. Hier werden nur lineare kombinatorische Optimierungsprobleme betrachtet.

Formal lässt sich ein kombinatorisches Optimierungsproblem folgendermaßen definieren (vgl. Osman und Kelly, 1996, S. 2): Ein kombinatorisches Optimierungsproblem besteht aus einer Menge von konkreten Probleminstanzen. Für jede Probleminstanz ist ein endlicher Lösungsraum Ω sowie eine Lösungsmenge $S \subseteq \Omega$ definiert. Der Lösungsraum Ω enthält genau alle möglichen Kombinationen der Merkmalsausprägungen der Entscheidungsvariablen. Er ergibt sich also durch das Kreuzprodukt der (endlichen) Mengen von Ausprägungen der (endlich vielen) Entscheidungsvariablen. In der Lösungsmenge S befinden sich genau diejenigen Elemente des Lösungsraums, die die Nebenbedingungen des kombinatorischen Optimierungsproblems erfüllen. Weiterhin ist eine Zielfunktion f gegeben, die jedem Element aus S einen

reellwertigen Kosten- oder Nutzwert zuordnet. Das kombinatorische Optimierungsproblem besteht nun darin, diejenige Lösung $s \in S$ zu identifizieren, für die $f(s)$ minimal (oder maximal) wird.

Ein berühmtes Beispiel für ein kombinatorisches Optimierungsproblem ist das Problem des Handlungsreisenden (*Traveling Salesman Problem*). Dabei soll ein Handlungsreisender, ausgehend von seinem Startpunkt, eine Reihe von Städten besuchen. Als Startpunkt wird eine der n Städte gewählt. Der Handlungsreisende soll jede Stadt genau einmal besuchen und am Ende wieder an seinem Startpunkt ankommen. Je zwei Städte sind durch einen Weg verbunden, dem eine gewisse Reisedauer zugeordnet ist. Es soll nun eine eine Rundreise in Form einer Reihenfolge der zu besuchenden Städte bestimmt werden, mit der die Gesamtreisedauer des Handlungsreisenden minimiert wird. Das Problem des Handlungsreisenden kann als lineares Optimierungsproblem mit ganzzahligen (genauer: binären) Entscheidungsvariablen mit jeweils endlichem Wertebereich modelliert werden und ist demnach ein kombinatorisches Optimierungsproblem (vgl. Diaby, 2007, S. 745ff.).

Da alle Variablen bei kombinatorischen Optimierungsproblemem einen endlichen Wertebereich haben, ist auch die Menge der Lösungen endlich. Jedes kombinatorische Optimierungsproblem lässt sich daher grundsätzlich wie in Algorithmus 1 beschrieben lösen, wobei ohne Beschränkung der Allgemeinheit ein Minimierungsproblem vorausgesetzt wird. Im weiteren Verlauf des Kapitels wird von nun an ohne gesonderte Erwähnung immer von Minimierungsproblemen ausgegangen.

Zunächst wird die Lösungsmenge S konstruiert, indem man alle Kombinationen der Merkmalsausprägungen in Ω mit einer problemspezifischen Funktion $isValid$ auf Gültigkeit prüft und die gültigen Kombinationen in die Menge S aufnimmt. Die Funktion $isValid$ übernimmt also die Prüfung der Nebenbedingungen aus der Problemformulierung. Danach wird mit der Zielfunktion f für jede Lösung in S der jeweilige Zielfunktionswert errechnet. Ist dieser kleiner als derjenige der bisher besten gefundenen Lösung, wird die aktuelle Lösung als neue beste bekannte Lösung s^* gespeichert. Nachdem der Algorithmus beendet ist, enthält die Variable s^* eine optimale Lösung, falls mindestens eine Lösung für das Problem existiert. Ansonsten ist $f_{min} = \infty$. Weil alle Mengen endlich sind, terminiert der Algorithmus auch nach endlich vielen Schritten.

Algorithmus 1 Lösung von kombinatorischen Optimierungsproblemen

```
1: Ω ← alle möglichen Kombinationen der Merkmalsausprägungen der Variablen

   {Identifizieren der Lösungen / gültigen Kombinationen}
2: S ← ∅
3: for all ω ∈ Ω do
4:    if isValid(ω) then
5:       S ← S ∪ {ω}
6:    end if
7: end for

   {Auswahl der Lösung mit niedrigstem Zielfunktionswert}
8: f_min ← ∞
9: for all s ∈ S do
10:   if f(s) < f_min then
11:      s* ← s
12:      f_min ← f(s)
13:   end if
14: end for
15: end.
```

Mit dem Algorithmus 1 aus dem vorherigen Abschnitt lassen sich theoretisch alle kombinatorischen Optimierungsprobleme lösen. Die Laufzeit des Algorithmus hängt im Wesentlichen von der Mächtigkeit der Menge Ω ab. Da Ω durch Aufzählen aller möglichen Variablenkombinationen konstruiert wird, ist die Mächtigkeit von Ω wiederum durch die Anzahl der Variablen und die Mächtigkeit der jeweiligen Definitionsbereiche der Variablen festgelegt.

Das Problem des Handlungsreisenden kann so modelliert werden, dass die Anzahl an Variablen der Anzahl der zu besuchenden Städten entspricht, d.h. der Städte ohne den Start- und Zielort. Die Definitionsmenge der Variablen besteht wiederum jeweils aus der Menge aller zu besuchenden $n-1$ Städte[2].

Wie man unschwer feststellt, ist die Mächtigkeit von Ω $(n-1)^{(n-1)}$ und die Menge S besteht dann aus $(n-1)!$ Elementen. Folglich wächst die Mächtigkeit von Ω und S exponenziell mit der Anzahl der zu besuchenden Städte bzw. der Problemgröße.

[2]Hier wird vom allgemeinen Fall des asymmetrischen Problem des Handlungsreisenden ausgegangen, bei denen die Reisedauer zwischen je zwei Städten A und B unterschiedlich lang sein kann, je nachdem ob man von A nach B oder von B nach A reist.

Möchte man z.B. ein Problem des Handlungsreisenden für 25 Städte lösen, besteht Ω aus ca. 10^{33} Elementen und die Mächtigkeit von S liegt immer noch bei astronomischen 10^{23} gültigen Lösungen. Alle Elemente dieser Menge einzeln zu prüfen, würde mit heutigen Computern viele Millionen Jahre dauern.

Die Vorgehensweise des Algorithmus 1 wird als vollständige und explizite Enumeration bezeichnet. Sie ist dadurch gekennzeichnet, dass einerseits alle Lösungen geprüft werden und andererseits jede Lösung einzeln geprüft wird. Für die meisten kombinatorischen Optimierungsprobleme gilt wie für das Problem des Handlungsreisenden, dass die Anzahl der zulässigen Lösungen mit dem Problemumfang exponenziell wächst, ein Phänomen, das als kombinatorische Explosion bezeichnet wird (vgl. Wolsey, 1998, S.8). Aus diesem Grund können Algorithmen der vollständigen und expliziten Enumeration für praktisch relevante Problemumfänge eines kombinatorischen Optimierungsproblems meist nicht verwendet werden.

4.3 Einige Grundbegriffe der Komplexitätstheorie

Die Komplexitätstheorie ist das Teilgebiet der Informatik, das sich unter anderem mit der Fragestellung beschäftigt, wie viele Ressourcen zur Lösung von Entscheidungs- und Optimierungsproblemen benötigt werden (vgl. hierzu und im Folgenden Wegener, 2005, S. 1 ff.). Bei den zur Lösung notwendigen Ressourcen wird zwischen dem Platzbedarf im Arbeitsspeicher des Computers (Speicherkomplexität) und der Anzahl der auszuführenden elementaren Rechenoperationen (Zeitkomplexität) unterschieden.

Die Zeitkomplexität eines Algorithmus wird auch als Laufzeit des Algorithmus bezeichnet. Dadurch, dass die Laufzeit nicht in Zeiteinheiten (z.B. in Sekunden), sondern anhand der Anzahl der Rechenoperationen bestimmt wird, ist sie unabhängig vom Einsatz eines konkreten Computers. Betrachtet sei nun ein konkretes Entscheidungs- oder Optimierungsproblem. Bei der Zeitkomplexität lassen sich die Best-Case-, Average-Case- und Worst-Case-Zeitkomplexität differenzieren. Die Best-Case-Zeitkomplexität gibt an, wie viele elementare Rechenoperationen ein Algorithmus zur Lösung einer Probleminstanz von gegebenem Umfang mindestens braucht, die Average-Case-Zeitkomplexität bezeichnet die durchschnittlich zu er-

wartende Anzahl an elementaren Rechenoperationen für diese Konstellation und die Worst-Case-Zeitkomplexität bezieht sich auf die maximal benötigte Anzahl von elementaren Rechenoperationen für diese Konstellation. Die Betrachtung der Worst-Case-Zeitkomplexität ermöglicht die Bestimmung einer oberen Schranke für die Rechenzeit.

In der Komplexitätstheorie ist man besonders daran interessiert, wie sich eine wachsende Problemgröße auf die Anzahl der durchzuführenden elementaren Rechenoperationen auswirkt. Entsprechend dieses Zusammenhangs werden Algorithmen auch in unterschiedliche Komplexitätsklassen eingeteilt. Besondere Bedeutung hat die Unterscheidung in polynomielle Algorithmen, bei denen der Zusammenhang zwischen Laufzeit und Problemgröße durch ein Polynom beschrieben werden kann, und exponenzielle Algorithmen, bei denen die Laufzeit exponenziell mit der Problemgröße wächst.

Exponenzielle Algorithmen - so wie das im vorherigen Abschnitt vorgestellte allgemeine Verfahren zur Lösung von kombinatorischen Optimierungsproblemen - sind in der Praxis nur für kleine Probleminstanzen hilfreich, da die Laufzeit ansonsten schnell inakzeptable Werte erreicht. Um mittlere und große Probleminstanzen lösen zu können, sind im Allgemeinen polynomielle Algorithmen erforderlich. Nun lässt sich aber aus der Tatsache, dass es für ein Problem einen Lösungsalgorithmus mit exponenzieller Laufzeit gibt, nicht folgern, dass das Problem per se nur mit exponenzieller Laufzeit lösbar ist. Es könnte ja einen alternativen Algorithmus für das Problem geben, dessen Laufzeit nur polynomial mit der Problemgröße wächst. Spricht man also in der Komplexitätstheorie einem Problem eine gewisse Komplexität zu, so muss dies implizieren, dass kein Algorithmus zur Lösung des Problems eine bessere Worst-Case-Komplexität erreichen kann.

Für Entscheidungsprobleme wird u.a. zwischen folgenden Zeitkomplexitätsklassen unterschieden (vgl. Cook et al., 1998, S. 309ff.):

- P: Die Klasse P beschreibt die Menge der Entscheidungsprobleme, die mit deterministischen Maschinen in polynomieller Laufzeit lösbar sind. Da alle bisher bekannten Computer[3] derartige deterministische Maschinen sind, hat

[3] Mit der Ausnahme von Quantencomputern, die bisher noch im Stadium der Grundlagenforschung sind und für komplexere Berechnungen nicht eingesetzt werden können.

diese Problemklasse eine große Praxisrelevanz, denn sie beschreibt demnach alle Entscheidungsprobleme, für die mit den zur Zeit verfügbaren Rechenmaschinen auch für große Probleminstanzen in akzeptablen Zeiträumen Lösungen gefunden werden können. Dies gilt zumindest dann, wenn die Rechnenzeit mit der Problemgröße nur linear oder quadratisch wächst.

- NP: Die Klasse NP beschreibt die Menge der Entscheidungsprobleme, die mit nicht-deterministischen Maschinen in polynomieller Laufzeit lösbar sind (vgl. Pomberger und Dobler, 2008, S. 382). Nicht-deterministische Maschinen sind bislang nur theoretische Konstrukte, für die es zur Zeit kein reales Äquivalent gibt. Alternativ kann die Klasse NP auch als Menge der Entscheidungsprobleme verstanden werden, für die es ein effizientes Verfahren zur Lösungsverifizierung gibt (vgl. Goldreich, 2010, S. 59ff.). Dazu muss ein Algorithmus existieren, der mit polynomieller Laufzeit prüft, ob es sich bei einer gegebenen potenziellen Lösung für das Entscheidungsproblem tatsächlich um eine gültige Lösung handelt. Wolsey formuliert diesen Umstand wie folgt (Wolsey, 1998, S. 83):

  > NP ist the class of decision problems with the property that: for any instance for which the answer is YES, there is a „short“ (polynomial) proof of the YES.

- NP-vollständig: Der Klasse NP-vollständig werden Entscheidungsprobleme zugeordnet, die in NP liegen und die zusätzlich die Eigenschaft haben, dass sich alle Probleme in NP mit polynomieller Laufzeit auf sie reduzieren lassen. Ein Problem P lässt sich allgemein auf ein Problem Q reduzieren, wenn für jede Probleminstanz von P eine Probleminstanz von Q mit gleicher Antwort (ja oder nein) konstruiert werden kann. Würde man also für nur ein Problem aus NP-vollständig einen Lösungsalgorithmus mit polynomieller Laufzeit finden, könnte man alle Probleme in NP mit polynomieller Laufzeit lösen.

Daneben gibt es mit der Klasse NP-schwer[4] eine für die kombinatorische Optimierung (also auch für die vorliegende Arbeit) besonders wichtige Komplexitätsklasse. Diese Klasse umfasst diejenigen Probleme, auf die sich alle Probleme in NP mit po-

[4] Auch gelegentlich als NP-hart bezeichnet, obwohl es sich hierbei um eine nicht ganz zutreffende Übersetzung des englischen NP-hard handelt

lynomieller Laufzeit reduzieren lassen. Die Klasse enthält demnach alle Probleme aus NP-vollständig, aber neben Entscheidungsproblemen auch Optimierungsprobleme, welche nicht zu NP und NP-vollständig gehören.

Es gilt insbesondere folgender Zusammenhang: Wenn ein Entscheidungsproblem in NP-vollständig liegt, liegt das dazugehörige Optimierungsproblem in NP-schwer (vgl. Wolsey, 1998, S. 88). So ist z.B. die Entscheidungsproblem-Variante des Problem des Handlungsreisenden („Existiert eine Rundreise mit Gesamtlänge $\leq k$?“) ein NP-vollständiges Problem, während die Optimierungsproblem-Variante („Was ist die Rundreise mit geringster Gesamtlänge?“) ein NP-schweres Problem ist. Mittels der Lösung der Optimierungsproblem-Variante kann man offensichtlich in polynomieller Laufzeit das Entscheidungsproblem beantworten, indem man einfach die Länge der Rundreise in der optimalen Lösung ermittelt und dann mit der Schranke k vergleicht.

Da eine nicht-deterministische Maschine ohne Erhöhung des Berechnungsaufwands eine deterministische Maschine nachahmen kann, liegt jedes Problem aus P auch in NP, d.h. es gilt $P \subseteq NP$. Ob nun aber auch $NP \subseteq P$ und damit $P = NP$ gilt, ist bis dato ungeklärt (vgl. Goldreich, 2010, S. 69).

Kombinatorische Optimierungsprobleme finden sich sowohl außerhalb als auch innerhalb von NP-schwer (vgl. Korte und Vygen, 2007, S. 403ff.). So ist zum Beispiel das Finden eines minimal spannenden Baums in einem Graphen oder die Suche nach einem kürzesten Weg in einem Graphen nicht NP-schwer. Andere kombinatorische Optimierungsprobleme mit hoher Praxisrelevanz wie das Mengenüberdeckungsproblem, das Rucksack-Problem oder das Problem des Handlungsreisenden sind dagegen NP-schwer. Für diese wie für alle anderen NP-schweren Optimierungsprobleme wurde bisher noch kein Algorithmus mit polynomieller Laufzeit gefunden. Falls $P \neq NP$ gilt, würde ein solcher auch bewiesenermaßen nicht existieren. Obwohl auch die Beziehung $P \neq NP$ bisher nicht bewiesen wurde, geht man heute praktisch von ihrer Gültigkeit aus, weil zahllose Versuche, einen polynomiellen Algorithmus für ein NP-schweres Problem zu konstruieren, gescheitert sind. Die existierenden exakten Algorithmen für NP-schwere Probleme sind aufgrund ihrer exponenziellen Rechenzeit für größere Probleminstanzen häufig unpraktikabel.

4.4 Verfahrensklassen zur Lösung von kombinatorischen Optimierungsproblemen

Im bisherigen Verlauf der Arbeit wurde davon ausgegangen, dass als Lösung eines kombinatorischen Optimierungsproblems nur eine optimale Lösung akzeptiert werden kann. Als exakte Lösungsverfahren werden genau diejenigen Lösungsverfahren verstanden, die unabhängig von der Problemgröße garantiert optimale Lösungen liefern. Wie im vorherigen Kapitel diskutiert, benötigen diese aber für NP-schwere Probleme (nach heutigem Wissensstand) durchweg eine mit der Problemgröße exponenziell wachsende Rechenzeit.

In der Praxis kann man nun aber durchaus auch mit suboptimalen Lösungen von kombinatorischen Optimierungsproblemen zufrieden sein, insbesondere, wenn diese mit vertretbarem Ressourceneinsatz berechnet werden können. In diesem Fall werden häufig Heuristiken eingesetzt. Ein Vorgehen wird als heuristisch bezeichnet werden, wenn es „ [...] nichtwillkürliche Entscheidungsoperatoren verwendet die bewirken, dass potenzielle Lösungen vom Suchprozess ausgeschlossen werden, und wenn aufgrund des fehlenden Konvergenzbeweises keine Lösungsgarantie gegeben werden kann“ (Streim, 1975, S. 151).

Durch Verzicht auf die Optimalitätsgarantie können auch für NP-schwere Probleme Algorithmen entwickelt werden, die mit polynomieller Laufzeit terminieren. Da viele kombinatorische Optimierungsprobleme NP-schwer sind, haben Heuristiken für die Lösung von kombinatorischen Optimierungsproblemen eine große Bedeutung.

Im Folgenden wird sowohl auf exakte Lösungsverfahren als auch auf heuristische Lösungsverfahren genauer eingegangen.

4.4.1 Exakte Verfahren

Exakte Lösungsverfahren müssen die Optimalität der gefundenen Lösung garantieren. Der zu Beginn des Kapitels vorgestellte Algorithmus stellt dies sicher, indem er alle Elemente des Suchraums in einer Menge vereinigt und jedes Element dieser Menge dediziert überprüft.

Intelligentere exakte Lösungsverfahren dagegen nutzen das Prinzip der impliziten Enumeration (vgl. Colbourn und Colbourn, 1985, S. 74ff.). Dabei wird versucht, den Suchraum systematisch aufzubauen und zu durchsuchen, ohne notwendigerweise jede einzelne mögliche Lösung explizit zu überprüfen.

Ansätze der Baumsuche

Eine Grundform der impliziten Enumeration ist das Backtracking. Beim Backtracking wird der Lösungsraum nach dem Prinzip „Versuch-und-Irrtum“ schrittweise durchsucht und dabei die Lösungsmenge komponentweise aufgebaut (vgl. Pomberger und Dobler, 2008, S. 380f.). Kann nach dem Hinzufügen einer Komponente zu einer unvollständigen Lösung die bisher beste vollständige Lösung nicht mehr übertroffen werden, muss diese unvollständige Lösung nicht weiter untersucht werden. In diesem Fall revidiert man die letzte Entscheidung und versucht, eine andere Komponente zu ergänzen. Dadurch muss man nicht jede mögliche (vollständige) Lösung explizit prüfen. Ein Backtracking-Algorithmus liefert immer das optimale Ergebnis für ein kombinatorisches Optimierungsproblem, hat aber im ungünstigsten Fall trotzdem exponenzielle Laufzeit.

Um alternative Suchwege zu verfolgen, kann man sich zu Nutze machen, dass sich der Suchraum eines kombinatorischen Optimierungsproblems als ein Baum darstellen lässt, also als ein kreisfreier, zusammenhängender Graph (vgl. Pomberger und Dobler, 2008, S. 381). Die inneren Knoten entsprechen partiellen Lösungen, die äußeren Knoten (auch Blätter genannt) stehen für vollständige Lösungen. Der Wurzelknoten repräsentiert die Ausgangssituation bzw. die leere Lösung. Alle inneren Knoten sind durch eine oder mehrere Kanten mit anderen Knoten verbunden, wobei jede Kante genau einer möglichen Entscheidungsalternative entspricht. Um unterschiedliche Kosten für die Entscheidungsalternativen abzubilden, können Kanten auch Gewichte (z.B. in Form von reellen Zahlen) zugeordnet sein. Die Kosten bzw. der Zielfunktionswert für eine vollständige Lösung kann dann dadurch ermittelt werden, dass die Kantengewichte vom Wurzelknoten bis zum entsprechenden äußeren Knoten aufsummiert werden. Das Lösen eines kombinatorischen Optimierungsproblems lässt sich dann als die Suche nach dem Blatt mit geringsten Kosten im dazugehörigen Suchbaum verstehen. Entsprechend ist der Einsatz von Baumsuchalgorithmen

(bzw. Graphensuchalgorithmen) naheliegend. Es gilt hierbei zu beachten, dass auch die Größe der Suchbäume von kombinatorischen Optimierungsproblemen exponenziell mit der Problemgröße wächst. Daher wird in diesem Zusammenhang meistens eine implizite Baumsuche durchgeführt, bei der der Suchbaum nicht erst vollständig konstruiert und dann durchsucht wird, sondern während der Suche Schritt für Schritt aufgebaut wird.

Bei den Baumsuchalgorithmen unterscheidet man zwischen uninformierten und informierten Suchverfahren (vgl. Edelkamp und Schrödl, 2012, S. 47ff.). Die uninformierten Suchverfahren durchsuchen den Baum ohne Anwendung von problemspezifischem- und problemtypspezifischem Wissen, während die informierten Suchverfahren sich dieser Informationen bedienen.

Grundlegende Typen der uninformierten Baumsuche sind (vgl. hierzu und im Folgenden Bortfeldt, 1995, S. 98ff.):

- Tiefensuche: Vom aktuellen Knoten wird der erste diesem direkt untergeordnete (oder nachgeordnete) Knoten untersucht, der bisher noch nicht untersucht wurde. Dieser Ansatz wird solange fortgesetzt, bis für einen Knoten alle direkt untergeordneten Knoten betrachtet wurden. Dann wird dasselbe Verfahren auf den übergeordneten Knoten angewendet. Das Verfahren beginnt beim Wurzelknoten.

- Breitensuche: Vom aktuellen Knoten aus wird nacheinander jeder direkt untergeordnete Knoten betrachtet. Dann wird dasselbe Verfahren auf jeden der direkt untergeordneten Knoten angewendet. Ausgangspunkt ist wieder der Wurzelknoten.

- Suche mit uniformer Bewertung: Die Suche mit uniformer Bewertung geht von einer Kostenfunktion f aus, die jedem Knoten einen reellwertigen Kostenwert zuweist. Ausgehend vom Wurzelknoten werden die Kostenwerte aller den bisher bereits untersuchten Knoten direkt untergeordneten Knoten bestimmt. Die Suche wird dann bei demjenigen Knoten fortgesetzt, der den geringsten Kostenwert aufweist.

Die informierten Suchverfahren bedienen sich problemspezifischer Informationen, die z.B. Schätzungen über noch zu erwartende Kosten für bisher noch nicht durch-

suchte Teile des Suchbaums erlauben. Diese Information kann dann dazu dienen, die Suche immer bei demjenigen Knoten fortzusetzen, für den die zu erwartenden Kosten am niedrigsten sind. Algorithmen, die nach diesem Muster verfahren, werden „Best-first"-Algorithmen genannt, wobei sich „best" hier nicht im Sinne eines tatsächliches Optimums, sondern nur als bester Wert der Schätzfunktion versteht.

Eine besondere Variante dieses Verfahrens ist der A*-Algorithmus (vgl. Hart et al., 1968). Hierbei wird der vielversprechendste Nachfolgeknoten von allen bisher untersuchten Knoten gewählt, nicht nur der vielversprechendste Nachfolgeknoten des letzten untersuchten Knotens. Der vielversprechendste Nachfolgeknoten wird dabei durch eine Kostenfunktion der Form $f(n) = g(n) + h(n)$ ermittelt. Dabei bezeichnet $g(n)$ die Kosten des Pfades vom Ausgangsknoten zum Knoten n mit den geringsten Gesamtkosten, die das A*-Verfahren zum aktuellen Zeitpunkt ermitteln konnte. Die Funktion $h(n)$ ist eine problemspezifische Schätzfunktion für die Kosten, die ein kostenminimaler Pfad vom Knoten n zu einem der Zielknoten (Blatt) benötigt. Wenn garantiert ist, dass $h(n)$ eine untere Schranke der tatsächlichen Kosten darstellt, d.h. die tatsächlichen Kosten für einen Pfad zwischen n und einem Zielknoten niemals niedriger sind als $h(n)$, dann ist die erste von A* gefundene vollständige Lösung automatisch optimal. Die Suche kann also sofort abgebrochen werden, unabhängig davon, wie viele Knoten explizit betrachtet wurden.

Branch-and-Bound-Ansatz

Auch der Branch-and-Bound-Ansatz ist eine spezielle Form der Baumsuche (vgl. Alba et al., 2009, S. 193ff.).

Generell repräsentieren Knoten des Suchbaums Teilmengen zulässiger Lösungen des gegebenen Optimierungsproblems. Blätter stehen für einelementige Teilmengen, also für genau eine zulässige Lösung. Innere Knoten entsprechen mehrelementigen Teilmengen zulässiger Lösungen. Der Wurzelknoten repräsentiert alle zulässigen Lösungen des gegebenen Problems. Ferner gilt, dass die Elternknoten die Vereinigungsmenge der durch die Kinderknoten repräsentierten Mengen darstellen, und dass letztere stets disjunkt sein sollten.

Ein Branch-and-Bound-Verfahren zeichnet sich dadurch aus, dass die Menge der zulässigen Lösungen, beginnend beim Wurzelknoten, schrittweise in immer mindestens zwei echte Teilmengen aufgespalten wird. Dadurch wird der oben beschriebene Suchbaum expandiert. Dieses Aufspalten bezeichnet man als „Branching“. Das Branching kann informiert oder uninformiert erfolgen, ebenso wie die Selektion des als nächsten aufzuspaltenden Knoten.

Um möglichst wenige Knoten des Baums aufspalten zu müssen, wird mittels des so genannten „Bounding“ versucht, Bereiche des Baums zu identifizieren, die man von weiteren Branching-Schritten ausschließen kann. Der Zielfunktionswert der zum jeweiligen Zeitpunkt besten bereits gefundene Lösung des Gesamtproblems dient dazu als obere Schranke. Um zu entscheiden, ob ein Knoten weiter untersucht werden muss, wird mittels einer geeigneten, problemspezifischen Funktion eine untere Schranke des Zielfunktionswertes für jede durch den Knoten repräsentierte zulässige Lösung ermittelt. Ist dieser Wert nun nicht kleiner als die obere Schranke, kann der Knoten von weiteren Betrachtung ausgeschlossen werden, da durch ein Aufspalten des Knotens keine Lösungsverbesserung erzielt werden kann.

Der Algorithmus 2 verdeutlicht den Branch-and-Bound-Ansatz (vgl. Alba et al., 2009, S. 367f.). Die Zielfunktion sei f. Eingabeparameter ist der Wurzelknoten S, der die Menge aller gültigen Lösungen repräsentiert. Mit diesem wird die *open*-Menge der zu expandierenden Knoten initialisiert. Die Variable s^* ist für die beste bisher bekannte Lösung vorgesehen. Zu Beginn wird diese auf den Wert *null* gesetzt um auszudrücken, dass noch keine Lösung bekannt ist. Der Zielfunktionswert von s^* ist in diesem Fall ∞. Der Algorithmus arbeitet nun solange die *open*-Menge ab, bis sie leer ist. In jedem Durchgang wird, gemäß den Vorgaben der Selektionsregel, ein Knoten aus der Liste genommen. Dieser wird dann dem Branching-Schema folgend in mehrere Knoten aufgespalten, die in der Menge C gespeichert werden. Für jeden Knoten c in C wird zuerst geprüft, ob er genau eine Lösung des Minimierungsproblems repräsentiert (es sich also um ein Blatt des Suchbaums handelt), und wenn ja, ob die von ihm repräsentierte Lösung besser ist als die beste bekannte Lösung s^*. Ist dies der Fall, wird s^* aktualisiert. Ansonsten wird geprüft, ob die untere Schranke für c echt niedriger ist als der Zielfunktionswert der aktuell besten bekannten Lösung s^*. Nur wenn dies der Fall ist wird der Knoten wieder der *open*-Menge hinzugefügt und

damit in einem späteren Durchgang weiter untersucht. Ansonsten wird der gesamte durch c repräsentierte Teil des Suchbaums abgeschitten.

Algorithmus 2 Branch-and-Bound

```
   {Eingabe: Menge aller gültigen Lösungen S (Wurzelknoten)}
   {Ausgabe: Optimale Lösung s* (mit minimalem Zielfunktionswert)}

   {Initialisieren der Werte}
 1: open ← S
 2: s* ← null

   {Branch-and-Bound}
 3: while open ≠ ∅ do
 4:    Wähle ein s aus open
 5:    Entferne s aus open
 6:    C ← Spalte s auf (Branching)
 7:    for all c ∈ C do
 8:       if istLösung(c) then
             {Neue beste Lösung?}
 9:          if f(c) < f(s*) then
10:             s* ← c
11:          end if
             {Weiter untersuchen? (Bounding)}
12:       else if untereSchranke(c) < f(s*) then
13:          open ← open ∪ {c}
14:       end if
15:    end for
16: end while
17: end.
```

Kurz hingewiesen sei noch auf den Branch-and-Cut-Ansatz (vgl. Mitchell, 2002), der eine Weiterentwicklung des Branch-and-Bound-Ansatzes darstellt und z.B. erfolgreich auf Probleme der Tourenplanung angewandt wurde (vgl. z.B. Baldacci et al., 2008).

4.4.2 Heuristische Verfahren

Im Gegensatz zu den exakten Lösungsverfahren finden Heuristiken in der Regel keine optimale Lösungen, liefern dafür aber die Ergebnisse im Normalfall wesentlich schneller.

Es lassen sich folgende fünf Kategorien von Heuristiken unterscheiden (vgl. hierzu und im Folgenden Domschke und Scholl, 2006, S. 2ff):

1) Eröffnungsverfahren: Eröffnungsverfahren dienen zur Bestimmung einer (ersten) zulässigen Lösung. Ein bekannter Vertreter dieser Art von Heuristik ist die Greedy-Heuristik (vgl. Blum, 2001, S. 132ff.), bei der die Lösung komponentenweise aufgebaut wird, eine einmal getroffene Entscheidung aber niemals revidiert wird. Pro Komponente wird diejenige Alternative gewählt, die gemäß einem komponentenbezogenen Bewertungskriterium am besten abschneidet.

2) Verbesserungsverfahren: Als Verbesserungsverfahren werden Ansätze bezeichnet, mit denen ausgehend von bereits gefundenen Lösungen in jedem Verfahrensschritt verbesserte Lösungen gefunden werden sollen. Dabei benutzen die Verfahren das Prinzip der Nachbarschaftssuche / lokalen Suche: Beginnend bei einer Ausgangslösung wird eine Menge von weiteren Lösungen (Nachbarschaft) gebildet. Die Nachbarschaft wird durch Modifikation einer oder mehrerer Variablen der aktuellen Lösung erzeugt. Gemäß definierter Auswahlkriterien wird eine Nachfolgelösung aus der Nachbarschaft gewählt, welche die aktuelle Lösung verbessert. Dieser Vorgang wird solange wiederholt, bis die Lösung nicht mehr verbessert werden kann oder ein Abbruchkriterium erreicht wurde.

3) Relaxationsbasierte Verfahren: Bei den relaxationsbasierten Verfahren wird die ursprüngliche Problemformulierung verändert mit dem Ziel, das neu formulierte Problem einfacher lösen zu können. Dies geschieht z.B. durch die Entfernung von Nebenbedingungen oder die Vernachlässigung von Ganzzahligkeits-Anforderungen. Die Lösung für das einfacher zu lösende Problem soll dann dabei helfen, eine gute Lösung für das ursprüngliche Problem zu finden.

4) Unvollständig ausgeführte Optimierungsverfahren: Ein unvollständig ausgeführtes Optimierungsverfahren liegt immer dann vor, wenn ein Optimierungsverfahren durch ein künstliches Abbruchkriterium oder durch eine sonstige

Modifikation vor dem Auffinden einer optimalen Lösung oder ohne Bestätigung der Optimalität der besten gefundenen Lösung abgebrochen wird bzw. terminiert.

5) Metaheuristiken: Als Metaheuristiken werden Verfahren bezeichnet, deren Kernelement ein „iterativer Generierungs-/Erstellungsprozess“ ist, „der eine untergeordnete Heuristik steuert, indem er intelligent verschiedene Konzepte zur Erforschung und Ausnutzung des Suchraums kombiniert, um Informationen so zu strukturieren, dass effizient nah-optimale Lösungen gefunden werden“ (Osman und Kelly, 1996, S. 3). Metaheuristiken sind selbst keine Heuristiken, sondern nur Rahmenverfahren, die zur Konstruktion von problemspezifischen Heuristiken genutzt werden. Ein wesentlicher Unterschied von Metaheuristiken im Vergleich zu Verbesserungsverfahren liegt darin, dass Metaheuristiken auch eine zeitweise Verschlechterung von Lösungen im Lösungsprozess akzeptieren können. Verbreitete Metaheuristiken sind z.B. simulierte Abkühlung (*Simulated Annealing, SA*, vgl. Nikolaev und Jacobson, 2010), Tabu-Suche (*Tabu Search, TS*, vgl. Gendreau und Potvin, 2010), variable Nachbarschaftssuche (*Variable Neighborhood Search, VNS*, vgl. Hansen und Mladenovic, 2001), *Greedy Randomized Adaptive Search Procedure* (*GRASP*, vgl. Resende und Riberio, 2010) und genetische Algorithmen (*Genetic Algorithms, GA*, vgl. Reeves, 2010).

Für die vorliegende Arbeit spielt die Kategorie 4, also Heuristiken auf Basis von unvollständig ausgeführten Optimierungsverfahren, die wesentliche Rolle. Daher wird dieser Ansatz nun für den speziellen Fall der Baumsuche genauer betrachtet.

Baumsucheverfahren als heuristische Vefahren

Auch die im Abschnitt zu den exakten Lösungverfahren vorgestellten Baumsuche-Verfahren lassen sich zu Heuristiken umgestalten. Dabei muss man von der Forderung abrücken, dass der gesamte Suchbaum (implizit) durchsucht werden muss, bevor der Algorithmus terminiert. Am einfachsten kann dies dadurch erreicht werden, dass man einen Baumsuche-Algorithmus nach einer bestimmten Laufzeit beendet und als Ergebnis die beste zu diesem Zeitpunkt gefundene Lösung ausgibt. Dieses Vorgehen eignet sich zur Modifikation eines Backtracking- oder Branch-and-Bound-Verfahrens, kann allerdings nicht sinnvoll auf das A*-Verfahren angewandt werden,

da die erste gefundene Lösung beim A*-Verfahren auch gleich die optimale Lösung ist, und zuvor keine vollständigen Lösungen konstruiert werden.

Alternativ kann man geeignet ausgewählte Teile des Suchbaums von der Suche ausschließen. So ist ein weiterer Nachteil des A*-Verfahrens, dass je nach Größe des Baums sehr viele Knoten im Speicher gehalten werden müssen. Dieses Problem kann dadurch umgangen werden, dass die Anzahl der nach einem Branching-Schritt weiter zu untersuchenden Knoten auf einen Maximalwert - die so genannte Beambreite - beschränkt wird. Derartige Baumsuche-Verfahren werden als Beamsearch-Algorithmen bezeichnet (vgl. Edelkamp und Schrödl, 2012, S. 280). Gilt für die Beambreite z.B. ein Wert von 6 und in einem Branchingschritt werden zehn Knoten aufgespalten, die auch nicht durch Bounding abgeschnitten werden können, müssen vier der zehn Knoten von weiteren Untersuchungen ausgeschlossen werden. Da nicht auszuschließen ist, dass in mindestens einem dieser vier Knoten eine Optimallösung des Problems liegt, wird die Optimalitätsgarantie dadurch aufgegeben.

Algorithmus 3 zeigt ein generisches Beamsearch-Verfahren (vgl. Alba et al., 2009, S. 368f.). Der wesentliche Unterschied zum Branch-and-Bound-Verfahren aus Algorithmus 2 liegt darin, dass die Nachfolgeknoten, die nicht durch die untere Schranke aussortiert werden können, in einer Menge B gesammelt werden. Aus dieser Menge werden dann die besten *bw* Knoten ausgewählt und in die *open*-Menge übertragen, d.h. in späteren Schritten weiter untersucht. Beim Branch-and-Bound werden dagegen alle Nachfolgeknoten, die nicht durch eine untere Schranke aussortiert werden können, direkt in die *open*-Menge übertragen.

Beamsearch-Algorithmen im Speziellen und Baumsuche-Verfahren im Allgemeinen konnten in der Vergangenheit bereits erfolgreich in vielen klassischen Bereichen der kombinatorischen Optimierung eingesetzt werden, so z.B. für das Mixed-Model Assembly Lines Sequencing-Problem (vgl. Leu et al., 1997), das Open Shop Scheduling-Problem (vgl. Blum, 2005) und das Single Maschine Scheduling-Problem (vgl. Rakrouki et al., 2012). Ein Beamsearch-Algorithmus für das 2D Circular Strip Packing-Problem konnte erst kürzlich für zahlreiche Benchmark-Testinstanzen neue Bestwerte aufstellen (vgl. Akeb et al., 2013).

Auch im Kontext der Containerlogistik wird das Verfahren erfolgreich eingesetzt. In Form der Verfahren von Araya und Riff (2014), Fanslau und Bortfeldt (2010)

Algorithmus 3 Beamsearch

{Eingabe: Menge aller gültigen Lösungen S (Wurzelknoten), Beambreite bw}
{Ausgabe: Beste gefundene Lösung s^* (mit minimalem Zielfunktionswert)}

{Initialisieren der Werte}
1: $open \leftarrow S$
2: $s^* \leftarrow null$

{Beamsearch}
3: **while** $open \neq \emptyset$ **do**
4: Wähle ein s aus $open$
5: Entferne s aus $open$
6: $B \leftarrow \emptyset$
7: $C \leftarrow$ Spalte s auf (Branching)
8: **for all** $c \in C$ **do**
9: **if** istLösung(c) **then**
{Neue beste Lösung?}
10: **if** $f(c) < f(s^*)$ **then**
11: $s^* \leftarrow c$
12: **end if**
{Weiter untersuchen? (Bounding)}
13: **else if** $untereSchranke(c) < f(s^*)$ **then**
14: $B \leftarrow B \cup \{c\}$
15: **end if**
16: **end for**
17: $open \leftarrow open \cup$ beste bw Elemente aus B
18: **end while**
19: **end.**

und Zhu und Lim (2012) handelt es sich bei drei der vier aktuell erfolgreichsten Verfahren zur Lösung des Container Loading-Problems um Baumsuche-Heuristiken.

Die Erfolge in diesen Bereichen legen es nahe, die Eignung von (problemspezifisch angepassten) Baumsuche-Verfahren auch für weitere kombinatorische Optimierungsprobleme an Containerterminals zu untersuchen.

5 Betriebliche Entscheidungsprobleme in Containerterminals

Bei der Konzeption und beim Betrieb eines Containerterminals sind zahlreiche Entscheidungen zu treffen, die jeweils auf die Optimierung relevanter Leistungskennzahlen abzielen. Darunter fallen auch Entscheidungen im Rahmen der Lösung der im Zentrum dieser Arbeit stehenden Stackingprobleme. Um diese weitergehend charakterisieren und abgrenzen zu können, werden zunächst der Begriff des wohlstrukturierten Entscheidungsproblems und die verschiedenen Ebenen, auf denen betriebliche Entscheidungsprobleme zu lösen sind, beschrieben.

Ein betriebliches Entscheidungsproblem liegt ganz generell dann vor, wenn in einem Unternehmen eine Entscheidung zu fällen ist, für die mehrere Entscheidungsoptionen vorliegen.

Als wohlstrukturierte Entscheidungsprobleme sollen solche betrieblichen Entscheidungsprobleme verstanden werden, für die „eine bestimmte Anzahl von Lösungsmöglichkeiten, Informationen über deren Auswirkungen (Konsequenzen), klar formulierte Ziele (Prämissen) sowie Regeln (Lösungsalgorithmen) vorliegen, mit deren Hilfe eine eindeutige Präferenzordnung der Alternativen gebildet werden kann.“ (vgl. Heinen, 1985, S. 44). Auf diese Art von betrieblichen Entscheidungsproblemen lassen sich Methoden des Operations Research (OR) anwenden. Im weiteren Verlauf dieses Kapitels werden nur wohlstrukturierte betriebliche Entscheidungsprobleme behandelt.

Betriebliche Entscheidungsprobleme sind auf unterschiedlichen (Planungs-)Ebenen zu treffen. Üblicherweise wird dabei zwischen strategischer, taktischer und operativer

Planungsebene unterschieden (vgl. Günther und Tempelmeier, 2012, S. 25ff.). Die Planungsebenen lassen sich wie folgt charakterisieren (vgl. hierzu und im Folgenden Wäscher, 1998, S. 429ff.):

- Strategische Planungsebene: Entscheidungen auf der strategischen Planungsebene stehen im direkten Zusammenhang mit dem „ [...] Aufbau und Erhalt dauerhafter Wettbewerbsvorteile“ (Wäscher, 1998, S. 429) des Unternehmens. Darunter fallen Fragestellungen zur Unternehmensgesamtstrategie, zu Geschäftsstrategien und zu funktionalen Strategien. Quellen für Wettbewerbsvorteile von Containerterminals sind beispielsweise eine hohe Servicequalität, niedrige Kosten, hohe Verfügbarkeit und die Vorteilhaftigkeit des Standorts (vgl. Yeo et al., 2008, S. 919f.).

- Taktische Planungsebene: Auf der taktischen Planungsebene werden Entscheidungen über die Aufbau- und Ablauforganisation des Unternehmens getroffen, durch die das Unternehmen sich den Vorgaben aus der strategischen Planung entsprechend ausrichten soll. Für Containerterminals ist beispielsweise über die Infrastruktur, die zu verwendenden Betriebsmittel und über die generell einzusetzenden Arbeitskräfte zu entscheiden.

- Operative Planungsebene: Alle weiteren betrieblichen Entscheidungen, die nicht der strategischen oder taktischen Planungsebene zugerechnet werden, fallen in die Kategorie der operativen Planung.

Es gilt grundsätzlich, dass Entscheidungen, die auf höheren Planungsebenen getroffen werden, in der Regel Auswirkungen auf die Problemtypen, die Problemvielfalt und -komplexität in den niedrigeren Planungsebenen haben. Entscheidet man sich z.B. auf taktischer Ebene für den Einsatz von AGVs zum Containertransport im Terminal, müssen in der Zukunft zwangsläufig eine Vielzahl von Entscheidungen auf der operativen Planungsebene getroffen werden, um diese zu koordinieren. Ebenso unmittelbar einleuchtend ist der Zusammenhang zwischen der gewählten Größe des Zwischenlagers und der damit einhergehenden größeren Komplexität (im allgemeinen Sinne des Wortes, nicht notwendigerweise im speziellen Sinne der Komplexitätstheorie) der betrieblichen Entscheidungsprobleme in diesem Bereich.

Im Folgenden wird davon ausgegangen, dass Entscheidungen auf strategischer und taktischer Ebene bereits getroffen sind. Es werden also ausschließlich operative

Entscheidungsprobleme betrachtet. Es soll darüber hinaus festgehalten werden, dass es sich bei den im Zentrum dieser Arbeit stehenden Stackingproblemen um operative, wohlstrukturierte betriebliche Entscheidungsprobleme handelt.

Im weiteren Verlauf des Kapitels wird nun die größere Problemlandschaft beschrieben, in deren Kontext sich die Stackingprobleme befinden. Durch den Einbezug der benachbarten Probleme wird der Gesamtzusammenhang für die Abläufe in Containerterminals hergestellt sowie eine Einordnung und Abgrenzung der Stackingprobleme ermöglicht. Da das Zwischenlager im Fokus dieser Arbeit steht, werden die darin auftretenden betrieblichen Entscheidungsprobleme eingehender beschrieben als die betrieblichen Entscheidungsprobleme an der See- und der Landseite. Insbesondere werden hier auch in größerem Umfang relevante Lösungsansätze für die betrieblichen Entscheidungsprobleme genannt, um die Bandbreite der in diesem Bereich zum Einsatz kommenden Methoden darzulegen.

Die operative Entscheidungsprobleme werden entlang der im Abschnitt 3.2 eingeführten Komponenten eines Containerterminals vorgestellt. Die aufgeführten Forschungsarbeiten zu den verschiedenen betrieblichen Entscheidungsproblemen im Zwischenlager von Containerterminals beziehen sich vorrangig auf den Veröffentlichungszeitraum 2008 bis 2012 und decken diesen weitgehend ab. Es werden allerdings auch einige bedeutende Arbeiten, die vor 2008 veröffentlich wurden, behandelt. Für eine umfassende Übersicht der Forschungsarbeiten in den besagten Bereichen bis zum Jahr 2008 sei erneut auf Steenken et al. (2004) und Stahlbock und Voß (2008) verwiesen.

5.1 Operative Entscheidungsprobleme an der Seeseite

Die operativen betrieblichen Entscheidungsprobleme auf der Seeseite beziehen sich auf das Be- und Entladen der Schiffe. Dabei sind betriebliche Entscheidungsprobleme in den folgenden fünf Bereichen zu lösen (vgl. hierzu und für den gesamten Abschnitt zur Seeseite Meisel, 2009, 18ff.).

Zuordnung der Liegeplätze

Bevor ein Schiff am Containerterminal entladen werden kann, muss es einen Liegeplatz ansteuern. Da die Anzahl der Liegeplätze begrenzt ist, muss ein Plan erstellt werden, aus dem hervorgeht, welches Schiff zu welchem Zeitpunkt an welchem Liegeplatz anzulegen hat (*berth plan*). Bei der Zuordnung der Liegeplätze müssen einige Nebenbedingungen wie die Länge des Schiffs, der Zeitpunkt der Ankunft im Containerterminal oder der benötigte Tiefgang berücksichtigt werden. Typischerweise ist man bestrebt, die Warte- und Liegezeiten der Schiffe zu minimieren. Die Wartezeit ist die Zeitspanne, in der das Schiff außerhalb des Containerterminals vor Anker liegt, bevor es an seinem Liegeplatz anlegen kann. Die Liegezeit bezeichnet die Zeitspanne, in der das Schiff den Liegeplatz belegt. Das Problem wird in der Literatur als Liegeplatz-Zuordnungsproblem (*Berth Allocation Problem, BAP*) bezeichnet und existiert in einer ganzen Reihe von Varianten, bei denen unterschiedliche Annahmen in Bezug auf die Liegeplätze und die Ankunftszeiten getroffen werden (vgl. Imai et al., 2001, 2005).

Zuordnung der Containerbrücken

Nachdem ein Schiff an einem Liegeplatz angelegt hat, muss es von den Containerbrücken ent- bzw. beladen werden. Vorrangig muss entschieden werden, welche Containerbrücke in welchem Zeitraum welchem Schiff zugeordnet sein soll. Einem Schiff können dabei zu einem Zeitpunkt keine, eine oder auch mehrere Containerbrücken zugeordnet sein. Grundsätzlich ist man an einer möglichst hohen Auslastung der teuren Containerbrücken interessiert, muss aber gleichzeitig darauf achten, die Liegezeiten der Schiffe niedrig zu halten. Teilt man einem Schiff sehr viele Containerbrücken zu, kann es in der Regel insgesamt schneller abgefertigt werden als mit weniger Containerbrücken. Die einzelnen Containerbrücken haben in der Regel aber eine unterdurchschnittliche Produktivität (gemessen an den bewegten Containern pro Zeiteinheit), weil es wesentlich häufiger zu Staus von Transportfahrzeugen unterhalb der sich dicht nebeneinander befindlichen Containerbrücken kommt. Zusätzlich muss u.a. beachtet werden, dass die Containerbrücken ein gemeinsames Schienenpaar nutzen und somit nicht aneinander vorbeifahren können. Das Containerbrücken-Zuordnungsproblem (*Quay Crane Assignment Problem, QCAP*)

wird auch Containerbrücken-Aufteilungsproblem (*Crane Split Problem*) genannt. In der Regel wird das Problem so formuliert, dass die Liegezeit und damit die Abfertigungsdauer zu minimieren ist.

Ablaufplanung für die Containerbrücken

Die Aufgabe der Containerbrücken ist es, Importcontainer aus dem Schiff und Exportcontainer auf das Schiff zu laden. Dies geschieht in mehreren Arbeitsschritten (*tasks oder jobs*). Die Suche nach einer Zuordnung der durchzuführenden Arbeitsschritte zu den verfügbaren Containerbrücken, durch die die Produktivität der Containerbrücken maximiert wird, bezeichnet man als Containerbrücken-Ablaufplanungsproblem (*Quay Crane Scheduling Problem, QCSP*). Je nachdem, wie detailliert das Problem modelliert ist, kann ein Arbeitsschritt das Be- und Entladen mehrerer Buchten oder auch nur das Umladen eines bestimmten Containers bedeuten. Eine detailliertere Modellierung des Problems erlaubt es, potenziell bessere Lösungen zu finden, allerdings steigt die Komplexität des QCSP drastisch an (vgl. Meisel, 2009, S. 24).

Stauplanung

Die Stauplanung (*stowage planning*) beschäftigt sich mit der Frage, an welcher Stelle im Stauraum eines Schiffes welcher Container gelagert werden sollte. Eine Planung ist schon deshalb notwendig, da eine ungünstige Lagerung schwerer Container die Stabilität des Schiffes negativ beeinflussen kann. Daneben ist es auch im Interesse der möglichst schnellen Entladung eines Schiffes sinnvoll, Container mit einem früher anzufahrenden Zielhafen über denen mit einem später anzufahrenden Zielhafen zu platzieren, da ansonsten unproduktive Umladeoperationen notwendig sind, um die blockierten Container freizulegen. Desweiteren sind für eine stabile Stapelung natürlich auch die Containerdimensionen zu berücksichtigen. Die Stauplanung erfolgt üblicherweise nicht auf der Ebene der einzelnen Container, sondern auf der Ebene von Containerklassen. Die Klassen fassen Container mit gleichen Eigenschaften zusammen, z.B. mit gleichem Gewicht, mit gleicher Größe, und/oder mit gleichem Zielhafen. Für jeden Stauplatz im Containerschiff wird also nur die Klasse des Containers festgelegt, nicht aber, welcher individuelle Container dieser Klasse dort plat-

ziert werden soll. Grundsätzlich wird bei der Stauplanung versucht, die Anzahl an unproduktiven Umladeoperationen, die beim Be- und Entladen des Schiffes an den verschiedenen angefahrenenen Containerterminals auftreten, zu minimieren, wobei wie erwähnt bestimmte Restriktionen zur Stabilisierung des Schiffes berücksichtigt werden müssen.

Planung des Containertransports vom und zum Zwischenlager

Bei der Planung des Containertransports vom und zum Zwischenlager muss entschieden werden, welches Transportfahrzeug zu welchem Zeitpunkt welche Transportaufgabe zu erledigen hat (*dispatching, scheduling*). Im Falle von automatisierten Fahrzeugen, also AGVs (autonom fahrende Transportfahrzeuge) oder ALVs (autonom fahrende Transportfahrzeuge, die Container aufnehmen und abladen können), müssen auch die von den Fahrzeugen zu fahrenden Routen - unter Berücksichtigung der vorhandenen Fahrspuren - geplant (*routing*) sowie die Steuerung des Verkehrs auf Echtzeitebene gewährleistet werden (*traffic control*). Unter allen Umständen sind Situationen zu vermeiden, in denen sich die fahrerlosen Fahrzeuge gegenseitig blockieren und somit nicht weiterfahren können (*deadlock prevention*). Das Optimierungsziel ist in der Regel die Maximierung der Produktivität, gemessen an den pro Zeiteinheit transportierten Containern. Dies wird wiederum durch Minimierung der Fahr- und ggf. Wartedauer der Transportfahrzeuge erreicht.

Eine detailliertere Klassifizierung und Beschreibung der Optimierungsprobleme an der Seeseite eines Containerterminals findet sich bei Bierwirth und Meisel (2010). Die Autoren weisen dabei auch auf die engen Abhängigkeitsbeziehungen zwischen den Optimierungsproblemen hin und schlagen daher vor, diese in einem integrativen Ansatz zu lösen. Stahlbock und Voß präsentieren eine umfangreiche Übersicht bisher publizierter Lösungsansätze für die genannten Probleme (Stahlbock und Voß, 2008, S. 7ff.).

5.2 Operative Entscheidungsprobleme im Zwischenlager

Die betrieblichen Entscheidungsprobleme im Zwischenlager haben in der Praxis eine besonders hohe Bedeutung, da das Zwischenlager häufig das Nadelöhr in der Transportkette im Terminal darstellt. Eine Erhöhung bzw. eine Verringerung der Produktivität im Zwischenlager spiegelt sich daher in der Regel direkt in einer Verbesserung bzw. Verschlechterung der wichtigsten Leistungskennzahlen, insbesondere der Produktivität der Containerbrücken, wider. Diese hat wiederum einen wesentlichen Einfluss auf die mittlere Liegezeit der Schiffe.

Auf der operativen Ebene sind im Zwischenlager eine ganze Reihe unterschiedlicher Entscheidungsprobleme zu lösen, darunter die Bestimmung von geeigneten Plätzen für die Container im Zwischenlager, die Bereitstellung von Kränen, sowie die Ermittlung der Kranoperationen für die bereitgestellten Kräne. Bei der Bestimmung der Plätze für die Container im Zwischenlager muss zunächst entschieden werden, in welchem Block des Zwischenlagers ein Container untergebracht werden soll. Dann muss die genaue Position des Containers innerhalb des Blocks festgelegt werden. Abbildung 5.1 zeigt eine Komplettübersicht der in dieser Arbeit erwähnten operativen Entscheidungsprobleme im Zwischenlager, inklusive der weiterführenden Literaturstellen.

Bestimmung der Blöcke für Container im Zwischenlager

Zhang et al. (2003) beschäftigen sich mit dem Problem, die Export- und Importcontainer im Zwischenlager möglichst gut auf die Blöcke zu verteilen (*Storage Space Allocation Problem, SSAP*). Dabei streben sie eine Minimierung der Liegezeit der Schiffe bei gleichzeitiger Maximierung des Durchsatzes der Containerbrücken an. Das genaue Layout der Blöcke bleibt unberücksichtigt, es bleibt also ungewiss, wo genau der Container in dem zugewiesenen Block platziert werden soll. Zur Lösung des Problems wenden die Autoren einen zweiphasigen Algorithmus iterativ mit rollierendem Planungshorizont an. In der ersten Phase wird eine rein anzahlmäße Aufteilung der im aktuellen Planungshorizont umzuschlagenden Container auf die Blöcke des Zwischenlagers bestimmt, bei der die Arbeitslast der für den Umschlag eingesetz-

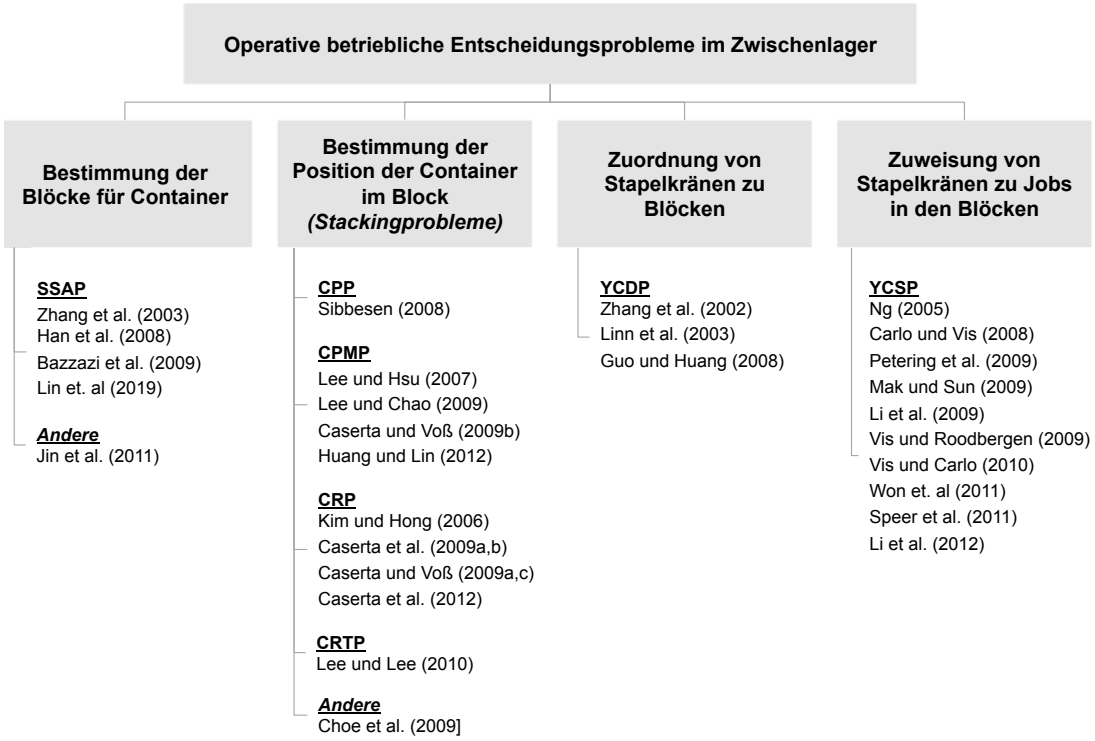


Abbildung 5.1: Übersicht der operativen betrieblichen Entscheidungsprobleme im Zwischenlager

ten Anlagen und Maschinen möglichst gleichmäßig über alle Blöcke verteilt ist. In der zweiten Phase wird bestimmt, welche Container von welchen Schiffen in den einzelnen Blöcken gelagert werden sollen, um das Ent- und Beladen der Schiffe mit möglichst geringen Transportzeiten erledigen zu können.

Bazzazi et al. (2009) erweitern das SSAP, indem sie unterschiedliche Eigenschaften der Container bei deren Zuordnung zu den Blöcken berücksichtigen (z.B. unterschiedliche Größen, Kühlfunktionalität etc.), die zur Formulierung zusätzlicher Einschränkungen genutzt werden können. So kann im erweiterten SSAP zum Beispiel formuliert werden, dass Container mit bestimmten Eigenschaften nur bestimmten Blöcken zugeordnet oder das bestimmte Container nicht auf andere Container gestapelt werden dürfen. Es wird ein genetischer Algorithmus vorgestellt, der Probleme mit in der Praxis üblichen Dimensionen effizient löst, und dessen Lösungen dabei nur geringe Differenzen zur optimalen Lösung aufweisen.

Han et al. (2008) beschäftigen sich mit einer spezifischen Variante des SSAP in einem Transshipmentterminal, der eine auf dem Prinzip der Reservierung basierende Aufteilungsstrategie verfolgt. Sie präsentieren ein gemischt-ganzzahliges lineares Optimierungsmodell, mit dem berechnet wird, in welchen Bereich des Zwischenlagers Container für ein Schiff platziert werden sollten. Dabei ist die Anzahl an Schichten (im Sinne von Arbeitszeiträumen), in denen Terminalbrücken eingesetzt werden müssen, zu minimieren. Die Autoren zeigen, dass sich mit einer auf Tabusuche aufsetzenden Heuristik in der Regel auch für komplexe Szenarien optimale Lösungen finden lassen.

Liu et al. (2010) beschreiben einen Algorithmus zur Lösung des SSAP, der für Zwischenlager maßgeschneidert wurde, bei denen Import- und Exportcontainer in denselben Bereichen gelagert werden. Sie definieren zwei Optimierungsziele: Zum einen soll die Arbeitsauslastung der Stapelkräne möglichst gleichmäßig sein, zum anderen soll die Fahrstrecke der internen Transportfahrzeuge minimiert werden.

Jin et al. (2011) argumentieren, dass die Entscheidung über einen optimalen Speicherort für einen Container im Zwischenlager auch davon abhängt, wie die Zuordnung der Stapelkräne zu den jeweiligen Blöcken geplant wird. Sind z.B. die Stapelkräne statisch immer genau einem Block im Zwischenlager zugeordnet, ist eine gleichmäßige Verteilung der Container über die Blöcke vorteilhaft. Wenn die Sta-

pelkräne den Blöcken dynamisch zugeordnet werden können, kann es sinnvoll sein, Blöcke, die nah an der Kaimauer liegen, dichter mit Containern zu packen, da dann die mittlere Transportdauer für einen Container verringert werden kann. Die Autoren beschreiben ein gemischt-ganzzahliges lineares Optimierungsmodell, das sowohl den Speicherort der umzuschlagenden Container als auch die Zuordnung der Kräne zu den Blöcken festlegt. Mit dem Modell lassen sich die Betriebskosten für den Umschlag der Container in einem begrenzten Planungshorizont minimieren.

Bestimmung der Position/Plätze von Containern in einem Block

Während die im vorherigen Abschnitt besprochenen Probleme nur die Einlagerung der Container ins Zwischenlager betrachten, umfasst das Containerpositionierungsproblem (*Container Positioning Problem, CPP*) den gesamten Zyklus eines Containers im Zwischenlager. Für jeden Container aus der betrachteten Menge sind der Ankunfts- sowie der geplante Weitertransportzeitpunkt bekannt. Das Zwischenlager wird vereinfacht als ein großer, zusammenhängender Block modelliert. In einer Lösung des CPP werden nun jedem Container geeignete Positionen im Block zugewiesen, die sich über den Zeitverlauf auch ändern können. Das Optimierungsziel ist es, die Kosten für den Umschlag der Container zu minimieren. Als Messgröße wird dazu in der Regel die Anzahl der Kranoperationen oder die gesamte Dauer der Kranoperationen verwendet. Eine Übersicht verschiedener Lösungsansätze für das CPP findet sich bei Sibbesen (2008).

Choe et al. (2009) beschäftigen sich mit der Frage, wie ein ganzer Block von Exportcontainern von zwei parallel arbeitenden Stapelkränen so umsortiert werden kann, dass alle Container nach der Sortierung entsprechend einem Stauplan ohne unproduktive Umladeoperationen aus den Buchten entladen werden können. Dabei nehmen sie an, dass der Block aus anfänglich mit Containern gefüllten Quellbuchten sowie aus leeren Zielbuchten besteht und alle Container aus den Quellbuchten zunächst in die Zielbuchten befördert werden, bevor sie schließlich zur Seeseite transportiert werden können. In den Zielbuchten müssen die Container entsprechend ihrer Containerprioritäten so gelagert werden, dass sich Container mit höherer Priorität immer über Containern mit niedrigerer Priorität befinden. Dann sind keine unproduktiven Umladeoperationen beim Entladen der Bucht notwendig. Desweiteren

verbieten die Autoren Umladeoperationen innerhalb der Quellbuchten, so dass die Reihenfolge, in der die Container aus den Quellbuchten in die Zielbuchten geladen werden, teilweise durch das anfängliche Blocklayout vorgegeben ist (partielle Ordnung). Der Ansatz besteht aus zwei miteinander verschränkten Algorithmen. Der erste Algorithmus berechnet ein der Sortierungsbedingung genügendes Layout der Zielbuchten, der zweite Algorithmus ordnet einer Folge von Kranoperationen, die das Layout herstellen, immer genau einen freien Kran zu. In mehreren Iterationen wird in jedem Durchgang das Ziellayout der vorhergehenden Iteration als Grundlage für ein Nachbarschaftssuchverfahren verwendet und im Anschluss geprüft, ob eine das neue Ziellayout herstellende Kranzuordnung eine schnellere Durchführung der Vorsortierung erlaubt als es bei der aktuell besten bekannten Lösung der Fall ist.

Beim Containervorsortierungsproblem (*Container Pre-Marshalling Problem, CPMP*) geht es darum, eine gegebene, mit Containern gefüllte Bucht mit möglichst wenigen Kranoperationen innerhalb der Bucht so umzusortieren, dass alle Container gemäß zugeordneten Prioritäten in den Stapeln liegen. Beim Containerumladeproblem (*Container Relocation Problem, CRP*) wird das CPMP dahingehend modifiziert, dass sukzessive Container mit der aktuell höchsten Priorität aus der Bucht zu entfernen sind, bis die Bucht vollständig geleert ist. Das Containerentladeproblem (*Container Retrieval Problem, CRTP*) ist eine Erweiterung des CRP auf einen ganzen Block des Zwischenlagers.

Alle in diesem Abschnitt genannten Probleme haben gemeinsam, dass die Anzahl bzw. die Gesamtdauer der durchzuführenden Kranoperationen minimiert werden soll. Dies stellt sich nur deshalb als problematisch dar, weil die Container in Stapeln aufeinander gelagert werden, und immer nur der oberste Container in einem Stapel zugreifbar ist. Daher sollen diese Probleme als Stapel- bzw. Stackingprobleme bezeichnet werden. Das CPMP, das CRP und das CRTP stehen im Zentrum der Arbeit und werden in den Kapiteln 6, 7 und 8 ausführlich behandelt.

Zuordnung von Stapelkränen zu Containerblöcken

Wenn die Zielpositionen einer Menge von Containern im Zwischenlager bestimmt wurden, müssen die Container physisch von Stapelkränen an diese Positionen befördert werden. Wegen der hohen Kosten für Stapelkräne sind diese in Zwischenlagern

in der Regel eine knappe Ressource und müssen daher möglichst effizient genutzt werden. Bei den damit korrespondierenden Optimierungsproblemen lassen sich zwei Detailstufen unterscheiden: Bei der Zuordnung von Stapelkränen zu Containerblöcken (*Yard Crane Deployment Problem, YCDP*) werden die Stapelkräne „grobgranular" den Blöcken des Zwischenlagers zugewiesen, in denen sie für eine bestimmte Zeit alle anfallenden Kranoperationen durchführen sollen. Bei der Zuordnung von Stapelkränen zu Jobs (Sequenzen von abzuarbeitenden Kranoperationen) in den Containerblöcken (*Yard Crane Scheduling Problem, YCSP*) werden die Stapelkräne „feingranular" den einzelnen Kranoperationen zugewiesen, d.h. auch wenn mehrere Kräne in einem Block arbeiten, ist immer eindeutig festgelegt, welcher Kran welche Kranoperation durchzuführen hat. Bei beiden Problemen ist es in der Regel das Ziel, die Zeit zu minimieren, in denen sich die Stapelkräne durch das Zwischenlager bewegen, ohne dass sie Container auf bzw. von internen Trucks bewegen.

Zhang et al. (2002) formulieren das Yard Crane Deployment Problem als ein gemischt-ganzzahliges lineares Optimierungsproblem und beschreiben ein auf Lagrange-Relaxation basierendes Lösungsverfahren.

Linn et al. (2003) präsentieren ebenfalls eine Formulierung des Yard Crane Deploymenent Problems als gemischt-ganzzahliges Optimierungsproblem, bei dem allerdings einige weitere praxisrelevante Bedingungen berücksichtigt wurden. Die Autoren testen ihren Ansatz mit Daten eines realen Containerterminals aus Hong Kong und kommen zu dem Schluss, dass die Effizienz der Stapelkräne signifikant verbessert wurde.

Guo und Huang (2008) wenden einen um heuristische Elemente erweiterten A*-Algorithmus zur Lösung des Yard Crane Deployment Problems an. Sie definieren das Optimierungsziel allerdings so, dass die Wartezeit der Transportfahrzeuge zwischen Kaimauer und Zwischenlager minimiert werden soll. Mit numerischen Experimenten zeigen die Autoren, dass der Algorithmus in der Praxis Laufzeiten von wenigen Sekunden haben sollte, und daher auch dann eingesetzt werden kann, wenn sich die äußeren Bedingungen häufig ändern.

Zuweisung von Stapelkränen zu Jobs in den Containerblöcken

Ng (2005) beschreibt einen auf Dynamischer Programmierung basierenden Lösungsansatz für das Yard Crane Scheduling Problem. Als Vereinfachung nehmen die Autoren allerdings an, dass das Zwischenlager aus nur einer Reihe von Blöcken besteht.

Carlo und Vis (2008) präsentieren Heuristiken zur Lösung des Yard Crane Scheduling Problems sowohl für den Fall, dass nur ein Kran in einem Block arbeitet, als auch für den Fall, dass zwei Kräne gleichzeitig einen Block bearbeiten. Für den zweiten Fall werden wiederum zwei Alternativen betrachtet. Bei der ersten Alternative können die Kräne nicht aneinander vorbeifahren, während dies bei der zweiten Alternative möglich ist. Optimierungsziel ist die maximale Gesamtproduktivität der Kräne.

Petering et al. (2009) stellen ein System zur Echtzeit-Zuordnung von RTGs zu Kranoperationen vor. Die Autoren beschränken sich bei ihrer Untersuchung auf Transshipmentterminals. Optimierungsziel ist die Maximierung der Produktivität der RTGs. Mittels diskreter Simulation leiten sie Anforderungen an ein Echtzeit-Zuordnungssystem für RTGs in Transshipmentterminals ab. So zeigt sich als Ergebnis der Simulation, dass die Produktivität der RTGs dadurch erhöht werden kann, dass das Zuordnungssystem nicht nur schon im Zwischenlager wartende, sondern auch auf das Zwischenlager zufahrende Transportfahrzeuge berücksichtigt. Zweitens empfehlen die Autoren, Kranoperationen, die Container aus dem Zwischenlager entfernen, gegenüber solchen, bei denen Container eingelagert werden, zu bevorzugen.

Mak und Sun (2009) lösen das Yard Crane Scheduling Problem für RMGs, also schienengebundene Stapelkräne, mit einem genetischen Algorithmus. Dabei berücksichtigen die Autoren in ihrer Problemformulierung, dass ein Block von mehreren RMGs zeitgleich bearbeitet werden kann, und setzen bei der Zuordnung der Kranoperationen zu den RMGs voraus, dass diese nicht aneinander vorbeifahren können. Bei einem DRMGC-System werden zwei unterschiedlich große RMGs auf parallel laufenden Schienen eingesetzt, so dass der kleinere Kran unter dem größeren Kran passieren kann. Für dieses Szenario präsentieren Cao et al. (2008) eine Heuristik, bei der ein Greedy-Algorithmus mit einem Simulated Annealing-Verfahren kombiniert wird.

Li et al. (2009) stellen ein gemischt-ganzzahliges lineares Optimierungsmodell zur Lösung des Yard Crane Scheduling Problems vor, das sowohl gleichzeitiges Laden und Entladen von Schiffen als auch einen minimalen Sicherheitsabstand zwischen Stapelkränen berücksichtigen kann. Die Autoren zeigen darüber hinaus eine Heuristik mit einem rollierenden Planungshorizont, die innerhalb von wenigen Sekunden Lösungen mit einer hohen Lösungsqualität generieren kann.

Vis und Roodbergen (2009) beschäftigen sich mit Containerterminals, in denen Portalhubwagen zum Betrieb des Zwischenlagers eingesetzt werden. Als Optimierungsziel definieren sie die Minimierung der von den Straddle Carriern ingesamt zurückgelegten Wegstrecken beim Abarbeiten eine gegebenen Menge von Jobs im Zwischenlager. Sie stellen einen Lösungsansatz vor, der eine Kombination aus einem Graphenmatching-Algorithmus und einem DP-Schema darstellt und das Problem optimal löst.

Vis und Carlo (2010) untersuchen Zwischenlager, bei denen jeweils zwei (automatisierte) Portalhubwagen pro Containerblock arbeiten. Sie präsentieren eine SA-Heuristik, die die in einem Block zu erledigenden Jobs derart auf die zwei Portalhubwagen verteilt, dass deren Produktivität maximal wird. Die Autoren weisen nach, dass die Lösungsqualität mit ihrem Ansatz nur zwischen 2 % und 6 % von einer unteren Schranke für die optimale Lösungsqualität abweicht und die Arbeitslast zwischen den Portalhubwagen gleichmäßig aufgeteilt wird.

Won et al. (2011) stellen ein mathematisches Modell vor, mit dem sich eine kostenminimale Konfiguration für den Betrieb des Zwischenlagers berechnen lässt. Dabei werden simultan sowohl die Zuordnung von Containern zu Blöcken im Zwischenlager als auch die Zuordnung der Stapelkräne zu den Blöcken berücksichtigt. Die Autoren beschreiben ebenfalls eine Heuristik, mit der qualitativ gute Lösungen für das Modell in vergleichsweise kurzer Zeit gefunden werden können.

Speer et al. (2011) beschäftigen sich mit einem Zwischenlager, das mit DRMGs betrieben wird. Sie entwickeln einen Branch-and-Bound-Ansatz, der die Stapelkräne derart den Jobs zuweist, dass die Produktivität der Stapelkräne maximiert wird (Yard Crane Scheduling Problem). Die Praktikabilität des Ansatzes wird durch Anwendung auf operative Daten vom Containerterminal Altenwerder (CTA) untermauert.

Li et al. (2012) behandeln ebenfalls das Yard Crane Scheduling Problem und beschreiben eine Modellierung als gemischt-ganzzahliges lineares Optimierungsmodell, das, im Vergleich zu älteren Arbeiten, noch wesentlich mehr Faktoren berücksichtigt. So wird beispielsweise auch die Beschleunigungs- und Bremsgeschwindigkeit der Stapelkräne modelliert. Die Autoren entwickeln auch eine Heuristik, die für das derart modellierte Yard Crane Scheduling Problem mit polynomialer Laufzeit Lösungen findet. Abschließend erweitern die Autoren die Heuristik derart, dass sie auch kurzfristig eingeschobene Jobs noch verarbeiten kann.

5.3 Operative Entscheidungsprobleme an der Landseite

Planung des Containertransports vom und zum Zwischenlager

Beim Transport der Container vom Zwischenlager zu den Transferpunkten für den Umschlag auf die Landtransportmittel treten prinzipiell ähnliche Entscheidungsprobleme auf, wie sie bereits beim Transport der Container von und zur Seeseite beschrieben wurden.

Planung beim Eintreffen der Landtransportmittel

Ist ein Containerterminal für den Umschlag auf LKW ausgelegt, so ist in Bezug auf das Eintreffen der LKW im wesentlichen über zwei Fragestellungen zu entscheiden (vgl. hierzu und im Folgenden Steenken et al., 2004, S. 31): Die Reihenfolgeplanung und das Routing. An dieser Stelle werden allerdings nur betriebliche Entscheidungsprobleme aus der Perspektive des Containerterminals berücksichtigt. Das Routing der LKW im Hinterland zum Beispiel ist eine Aufgabe der Speditionen und wird nicht vom Containerterminal beeinflusst.

- Reihenfolgeplanung der LKW an der Landseite: In der Regel warten am Übergang zwischen dem Containerterminal und dem Hinterland zu einem Zeitpunkt mehrere LKW darauf, mitgeführte Container zu entladen bzw. einen Container aus dem Zwischenlager für den Inlandstransport aufzunehmen. Bei der

Reihenfolgeplanung geht es darum, eine Reihenfolge für die wartenden LKW zu bestimmen, mit der die LKW möglichst schnell abgefertigt werden können.

- Routing der LKW an der Landseite: Wenn die Reihenfolge der LKW geplant ist, kann mittels eines genauen Routenplans für den Innenbereich des Containerterminals und jeden der LKW die Gesamtdauer der Abfertigung weiter verkürzt werden. Dazu werden neben den anderen, sich simultan im Containerterminal befindlichen LKW auch weitere Fahrzeuge wie Stapelkräne oder interne Transportfahrzeuge mitberücksichtigt. Dabei wird nur die zurückgelegte Strecke innerhalb des Containerterminals minimiert.

Während die Ankunftszeit von LKW für ein Containerterminal praktisch nicht vorhersehbar ist, erfolgt die Ankunft von Zügen üblicherweise periodisch in einem fest vereinbarten Zeitfenster, so dass keine spezielle Reihenfolgeplanung notwendig ist.

Planung beim Be- und Entladen der Landtransportmittel

Für die Planung des Beladens der Landtransportmittel mit RMGs oder RTGs sind ähnliche Entscheidungen zu treffen, wie sie bereits im Abschnitt über das Zwischenlager beschreiben wurden (vgl. Froyland et al., 2008). Normalerweise werden die LKW oder Züge nicht direkt von den RTGs oder RMGs im Zwischenlager versorgt, sondern an der Landseite des Containerterminals. Für diesen müssen dann – analog zur Situation im Zwischenlager – über die geeignete Ablageposition für die Container, über die Zuordnung von RMGs/RTGs zu Bereichen und über die Zuweisung einzelner Jobs entschieden werden. Werden für den landseitigen Umsatz Portalhubwagen eingesetzt, entfällt die Entscheidung zur Ablageposition, weil die Portalhubwagen sowohl den Transport vom Zwischenlager zu den LKW als auch das eigentliche Beladen übernehmen. In diesem Fall ist dann nur die Zuordnung der Jobs zu den Portalhubwagen zu planen.

5.4 Zusammenfassung

In diesem Kapitel wurde ein ausführlicher Überblick über operative betriebliche Entscheidungsprobleme in Containerterminals gegeben, wobei der Fokus auf das

Zwischenlager des Containerterminals gelegt wurde. Damit wurde der Kontext dargestellt für die im Zentrum dieser Arbeit stehende Problemklasse der Stackingprobleme im Zwischenlager von Containerterminals. Drei Repräsentanten dieser Problemklasse, nämlich das CPMP, das CRP und das CRTP, werden in den nächsten drei Kapiteln genauer untersucht. Bei dieser Problemklasse handelt es sich um kombinatorische Optimierungsprobleme, deren effektive und effiziente Lösung für die Optimierung der Abläufe in Containerterminals große Bedeutung hat, da es sich beim Zwischenlager häufig um den Flaschenhals im Umschlagsprozess handelt. Daher schlagen Prozessverbesserungen an dieser Stelle in der Regel unmittelbar auf einige wesentliche Leistungskennzahlen für den gesamten Containerterminal durch, speziell auch auf die Produktivität. Da nahezu jeder Container im Zwischenlager untergebracht wird und Containerterminals jährlich mehrere Millionen Container umschlagen können, wirken sich selbst kleine Verbesserungen signifikant auf die Leistungskennzahlen aus.

Bei den drei genannten Stackingproblemen handelt es sich um NP-schwere kombinatorische Optimierungsprobleme. Die Wahl einer Heuristik gegenüber einem exakten Verfahren ist daher ein naheliegender Ansatz, der darauf abzielt, auch große Probleminstanzen in akzeptabler Zeit lösen zu können.

Wie in Abschnitt 4.4.2 gezeigt, wurden Verfahren der Baumsuche in den letzten Jahren auf kombinatorische Optimierungsprobleme, speziell auf Probleme aus der Logistik, sehr erfolgreich angewendet. Zur Lösung von operativen Entscheidungsproblemen im Zwischenlager von Containerterminals kam diese Verfahrensklasse allerdings bisher nur selten zum Einsatz (vgl. z.B. Guo und Huang, 2008; Speer et al., 2011). Der Frage, ob Verfahren der Baumsuche für die Lösung der genannten Stackingprobleme erfolgreich eingesetzt werden können, soll nun im weiteren Verlauf dieser Arbeit nachgegangen werden.

6 Heuristische Baumsuche für das Container Pre-Marshalling Problem (CPMP)

In diesem Kapitel wird das Container Pre-Marshalling Problem (CPMP) eingeführt und formal beschrieben. Auf eine Literaturübersicht, aus der der aktuelle Stand der Forschung in Bezug auf Lösungsverfahren für das CPMP ersichtlich wird, folgt die Verfahrensbeschreibung einer Baumsuche-Heuristik für das CPMP. In einem ausführlichen Evaluierungsteil wird die Leistungsfähigkeit der Baumsuche-Heuristik diskutiert. Das Kapitel schließt mit einer Zusammenfassung der Ergebnisse.

Die Ergebnisse dieses Kapitels wurden von Bortfeldt und Forster (2012) veröffentlicht. Eine Vorversion der hier beschriebenen Heuristik findet sich bei Bortfeldt (2004).

6.1 Problembeschreibung

Wie in Kapitel 3.3 erläutert, ist die Abfertigungsdauer eines Schiffes eine wesentliche Leistungskennzahl in der Containerschifffahrt. Das Beladen des Schiffes nimmt einen großen Teil der Abfertigungsdauer ein. Die Container, die auf ein Schiff geladen werden sollen, sind vor der Ankunft des Schiffes im Zwischenlager untergebracht und werden nach Ankunft des Schiffes von dort einzeln zur Beladestelle transportiert.

Beim Beladen des Schiffes muss eine gewisse Containerreihenfolge eingehalten werden, die vom Stauplan des Schiffes bestimmt wird. Der Stauplan enthält dabei normalerweise keine Vorgaben bis auf die Ebene eines einzelnen Containers,

sondern fasst mehrere Container zu jeweils einer Containergruppe zusammen. Containergruppen können durch natürliche Zahlen repräsentiert werden. Die Containergruppe eines Containers bringt die zeitliche Priorität des Containers beim Beladen zum Ausdruck. Container einer niedrigeren Gruppe müssen immer vor Containern einer höheren Gruppe verladen werden. Die zeitliche Reihenfolge von Containern derselben Gruppe spielt beim Verladen keine Rolle. Bestimmende Merkmale zur Definition von Containergruppen sind beispielsweise das Gewicht, die Größe oder der Bestimmungsort.

Um ein Schiff zu beladen, müssen die Container das Zwischenlager also in einer bestimmten Reihenfolge - aufsteigend sortiert nach ihrer Gruppe - verlassen. Befindet sich nun aber in einem Stack ein Container c_1 unter einem Container c_2 mit niedrigerer Priorität, muss c_2 zunächst auf einen anderen Stapel bewegt werden, bevor c_1 zugänglich ist, da der Kran[1] immer nur auf den höchsten Container eines Stapels zugreifen kann. Umladeoperationen dieser Art lassen den Transportfluss der Container ins Stocken geraten und es kommt zu Verzögerungen, wodurch die Dauer für die Beladung des Schiffes und damit auch die Abfertigungsdauer erhöht wird. Umladeoperationen sind also möglichst zu vermeiden.

Ein Ansatz, die oben genannten Umladeoperationen zu vermeiden, ist es, die Stapel bereits vor Einlaufen des Schiffes vorzusortieren, so dass die Container zum Zeitpunkt des Beladens ohne weitere Umladeoperationen aus dem Zwischenlager entnommen werden können. Beim CPMP geht es nun um die Frage, wie man eine gegebene Containerbucht möglichst schnell so vorsortiert, dass die Container im Anschluss ohne weitere Umladeoperationen gemäß der Priorität ihrer Gruppe aus dem Zwischenlager entnommen werden können. Dabei wird davon ausgegangen, dass eine Umladeoperation unabhängig von der Entfernung und Befüllung der beteiligten Stapel immer gleich lange dauert, so dass die Gesamtdauer der Vorsortierung direkt proportional zur Anzahl der durchgeführten Umladeoperationen ist. Desweiteren wird unterstellt, dass Umladeoperationen immer nur zwischen Stapeln aus derselben Bucht vorgenommen werden.

[1] Als Kran soll im Folgenden allgemein die Maschine bezeichnet werden, die zum Ein- und Ausladen von Containern im Zwischenlager verwendet wird. Dabei kann es sich, abhängig vom Operationsmodus des Containerterminals, um RTGs, RMGs oder Portalhubwagen handeln. Ohne Beschränkung der Allgemeinheit wird im Weiteren von RTGs ausgegangen.

6.2 Problemformalisierung

Das im vorherigen Abschnitt beschriebene CPMP wird nun formal definiert.

Eine Bucht im Zwischenlager eines Containerterminals lässt sich als ein Tupel (S,H) beschreiben. Dabei bezeichnet S ($S \in \mathbb{N}, S \geq 1$) die Anzahl der Stapel, aus der die Bucht besteht. H ($H \in \mathbb{N}, H \geq 1$) definiert die maximale Stapelhöhe, legt also fest, wie viele Container maximal in einem Stapel untergebracht werden können.

Der (Füll-)Zustand (im Folgenden auch: das Layout) einer Bucht kann durch eine Matrix L mit H Zeilen und S Spalten repräsentiert werden. Eine Zelle der Matrix wird durch ein Tupel (h,s) addressiert ($1 \leq h \leq H$, $1 \leq s \leq S$). Mit G ($G \in \mathbb{N}, G \geq 1$) wird die höchste vorkommende Containergruppe in einer Bucht bezeichnet. Jedem der Container in einer Bucht ist eine Containergruppe g ($g \in \mathbb{N}, 1 \leq g \leq G$) zugeordnet. Es gilt:

$$L(h,s) = \begin{cases} g, & \text{wenn sich im Stapel } s \text{ in der Ebene } h \\ & \text{ein Container der Containergruppe } g \text{ befindet,} \\ 0, & \text{wenn sich im Stapel } s \text{ in der Ebene } h \\ & \text{kein Container befindet.} \end{cases}$$

Ein Container kann nur entweder auf dem Boden der Bucht oder auf einem anderen Container stehen. Entsprechend muss, falls $L(h,s) = 0$ gilt, auch $L(h',s) = 0$ für alle $h' = h+1, \ldots, H$ gelten.

Der Zustand der Bucht in Abbildung 6.1 lässt sich beispielsweise durch eine Matrix L mit $H = 3$ Zeilen und $S = 4$ Spalten repräsentieren. Es gelten dann folgende Zuordnungen: $L(1,1) = 1$, $L(2,1) = 2$, $L(3,1) = 0$, $L(1,2) = 3$, $L(2,2) = 9$, $L(3,2) = 0$, $L(1,3) = 4$, $L(2,3) = 8$, $L(3,3) = 0$, $L(1,4) = 5$, $L(2,4) = 6$, $L(3,4) = 7$. Man beachte, dass hier die Zeilen von unten nach oben nummeriert werden.

			7
2	9	8	6
1	3	4	5

Abbildung 6.1: Bucht aus vier Containerstapeln

Eine Umladeoperation kann mit einem Tupel (d,r) von verschiedenen Stapeln ($1 \leq d,r \leq S, d \neq r$) beschrieben werden. Der erste Stapel (d) wird als Sender be-

zeichnet, der zweite Stapel (r) ist der Empfänger. Eine Umladeoperation bewegt den Container, der sich an der höchsten Position im Sender befindet, auf den niedrigsten freien Platz des Empfängers. Letzterer wird auch als Empfängerplatz bezeichnet. Eine Umladeoperation ist genau dann zulässig für einen durch L repräsentierten Zustand einer Bucht, wenn folgende zwei Bedingungen erfüllt sind:

1. Der Sender ist nicht leer: $L(1, d) > 0$.
2. Der Empfänger ist nicht voll: $L(H, r) = 0$.

Durch die Anwendung einer Umladeoperation (d, r) auf eine Matrix L entsteht eine neue Matrix L'. Mit $top_L(s)$ soll die Reihe bezeichnet werden, in der sich der oberste Container im Stapel s in der Matrix L befindet. Handelt es sich bei s um einen leeren Stapel, ist $top_L(s) = 0$. Für die Matrix L' gilt:

- Die Reihe $top_L(d)$ im Senderstapel d, in der sich der umzuladende Container in L befunden hat, ist in L' leer: $L'(top_L(d), d) = 0$.
- In L' befindet sich der umgeladene Container eine Reihe über der Reihe $top_L(r)$: $L'(top_L(r) + 1, r) = L(top_L(d), d)$.

An allen anderen Plätzen stimmen die Elemente der Matrizen L' und L überein, d.h. $L'(h, s) = L(h, s)$ für $h = 1, \ldots, H$, $s = 1, \ldots, S$, $(h, s) \neq (top_L(d), d)$ und $(h, s) \neq (top_L(r) + 1, r)$. Abbildung 6.2 zeigt grafisch, wie sich die Umladeoperation $(1, 3)$ auf die Bucht aus Abbildung 6.1 auswirkt.

Bucht vor Umladen | Umladeoperation | Bucht nach Umladen

			7
2	9	8	6
1	3	4	5

(1,3)

		2	7
	9	8	6
1	3	4	5

Abbildung 6.2: Anwendung einer Umladeoperation

Eine Bucht ist genau dann vorsortiert und kann somit ohne weitere Umladeoperationen geleert werden, wenn in jedem Stapel der Bucht Container mit niedrigerem Gruppenindex ausschließlich oberhalb von Containern mit höherem Gruppenindex lagern. Dabei ist zu beachten, dass der Wert eines Gruppenindex niedrig ist, wenn die betroffenen Container früh im Verladeprozess benötigt werden. Entsprechend

wird den Containern, die als erstes aus einer Bucht entnommen werden sollen, der Gruppenindex 1 zugewiesen. Abbildung 6.3 zeigt exemplarisch, wie sich die Bucht aus Abbildung 6.1 mit acht Umladeoperationen vorsortieren lässt.

Um die Vorsortierung zu formalisieren, sind die folgenden zwei Prädikate für Container hilfreich: Es gelte $L(h,s) = g, g > 0$. Der Container im Platz (h,s) der Gruppe g wird als schlecht platziert (im Sinne des CPMP) bezeichnet, wenn $h > 1$ und $L(h-1,s) < g$, oder wenn sich ein schlecht platzierter Container an einem Platz $(h',s), 1 < h' < h$ unter diesem Container befindet. Ansonsten wird der Container als gut platziert (im Sinne des CPMP) bezeichnet.

Eine Matrix L ist also genau dann vorsortiert, wenn alle Container gut platziert sind. Formal lässt sich diese Vorsortierungsbedingung folgendermaßen notieren:

$$\forall h, 2 \leq h \leq H : \forall s, 1 \leq s \leq S : L(h,s) \neq 0 \Rightarrow L(h,s) \leq L(h-1,s) \tag{6.1}$$

Eine Sequenz von p Umladeoperationen $s = (m_1, m_2, \ldots, m_p)$ $(p \in \mathbb{N},\ p \geq 1)$ ist zulässig für L, wenn die Umladeoperation m_1 für L zulässig, und jede Umladeoperation m_{i+1} für die durch Anwendung von m_i entstehende Matrix zulässig ist $(i = 1, \ldots, p-1)$.

Aufbauend auf dieser Definition lässt sich das CPMP nun folgendermaßen formulieren: Für einen durch eine Matrix L repräsentierten Buchtzustand soll eine Sequenz s, bestehend aus p Umladeoperationen, gefunden werden, die folgende drei Anforderungen erfüllt:

1. Die Sequenz s ist zulässig für die Matrix L.
2. Die durch die Anwendung der letzten Umladeoperation m_p entstehende Matrix L_p erfüllt die Vorsortierungsbedingung 6.1.
3. Für jede Sequenz von Umladeoperationen s', die die Anforderungen 1. und 2. erfüllt, gilt: $p' \geq p$, wobei p' die Anzahl der Umladeoperationen in s' bezeichnet.

Es können CPMP-Instanzen konstruiert werden, die prinzipiell nicht lösbar sind. So ist z.B. eine Bucht, in der alle Stapel bis zur maximalen Stapelhöhe mit Con-

tainern gefüllt sind und in der mindestens ein Container schlecht platziert ist, nicht lösbar, da kein freier Platz vorhanden ist, um den schlecht platzierten Container umzuladen. Im Folgenden werden nur lösbare Probleminstanzen betrachtet.

Caserta et al. (2011) konnten beweisen, dass es sich beim CPMP um ein NP-schweres Optimierungsproblem handelt. Der Beweis nutzt die Verwandschaft des CPMP mit dem Container Relocation Problem (CRP), das in Kapitel 7 eingehend thematisiert wird. Die Autoren zeigen, dass man für jede CRP-Instanz eine äquivalente CPMP-Instanz in polynomieller Laufzeit konstruieren kann, so dass eine optimale Lösung der CPMP-Instanz auch eine optimale Lösung der CRP-Instanz ist. Da es sich beim CRP aber um ein NP-schweres Optimierungsproblem handelt, muss das CPMP ebenfalls NP-schwer sein.

6.3 Bisherige Lösungsansätze

6.3.1 Lee und Hsu (2007)

Lee und Hsu (2007) interpretieren das CPMP als ein Multi-Commodity Network Flow Problem mit diskreten Raum- und Zeitpunkten. Das Multi-Commodity-Flussoptimierungsproblem ist eine Verallgemeinerung des klassischen Flussoptimierungsproblems, bei dem unterschiedliche Typen von Gütern durch das Netzwerk fließen können. Multi-Commodity-Flussoptimierungsprobleme sind für ganzzahlige Flüsse NP-schwer; kleinere Probleminstanzen können mit Hilfe von ganzzahligen linearen Optimierungsmodellen gelöst werden (vgl. Even et al., 1976).

Eine mögliche Containerposition zu einem diskreten Zeitpunkt wird im Modell von Lee und Hsu jeweils durch zwei Knoten (einen für den eingehenden Fluss, einen für den ausgehenden Fluss) repräsentiert. Um den Zustand in einer Bucht der Höhe 3 mit 3 Stapeln zu einem Zeitpunkt zu repräsentieren werden also $2 \cdot 3 \cdot 3 = 18$ Knoten benötigt. Zwischen zwei Zeitpunkten wird jeweils ein Container von einem Stapel auf einen anderen Stapel bewegt. Allgemein besteht der Flussgraph also für t Zeitpunkte aus $2 \cdot H \cdot S \cdot t$ Knoten.

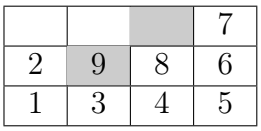
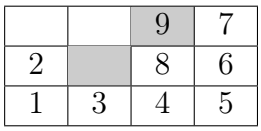
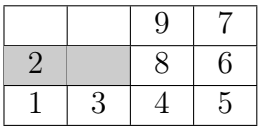
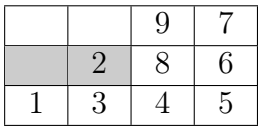
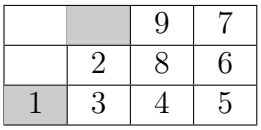
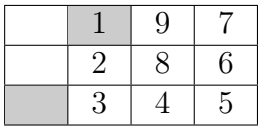
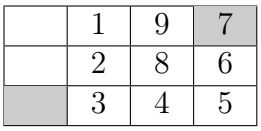
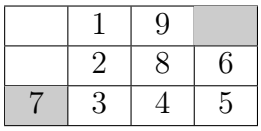
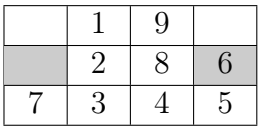
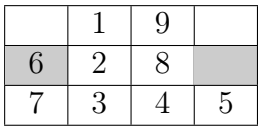
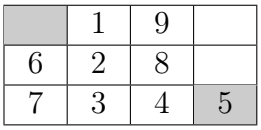
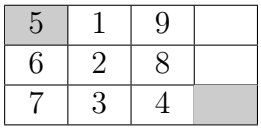
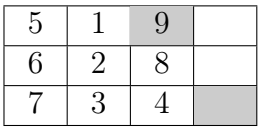
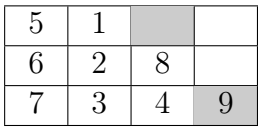
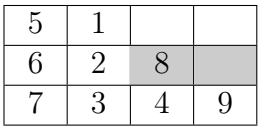
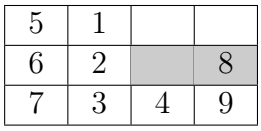
Schritt	Bucht vor Umladen	Umladeoperation	Bucht nach Umladen
1.		(2,3)	
2.		(1,2)	
3.		(1,2)	
4.		(4,1)	
5.		(4,1)	
6.		(4,1)	
7.		(3,4)	
8.		(3,4)	

Abbildung 6.3: Vorsortierung einer Matrix

Die Autoren testen die Modellierung mit einer Bucht bestehend aus 6 Stapeln, 14 Containern, 3 verschiedenen Containergruppen und einer maximalen Stapelhöhe von 4 Containern. Für diese vergleichsweise kleine Testinstanz terminiert das Flussoptimierungsprogramm auf einem PC mit 3,2 GHz CPU und 1 GB Hauptspeicher erst nach knapp 2 Stunden mit einer aus 9 Umladeoperationen bestehenden Lösungssequenz.

Da das Flussoptimierungsprogramm in dieser Form für den praktischen Einsatz nicht geeignet ist, schlagen die Autoren eine zweistufige Heuristik vor. In der ersten Stufe wird das Model durch das Entfernen einiger (zufällig ausgewählter) Kanten vereinfacht. Die Lösung dieses Models dient dann als Eingabe für die zweite Stufe. In der zweiten Stufe der Heuristik wird versucht, Zyklen, die üblicherweise in der ersten Stufe in die Lösung einfließen, zu entfernen.

6.3.2 Lee und Chao (2009)

Lee und Chao (2009) beschreiben einen Ansatz zur Lösung des CPMP, der hauptsächlich auf dem Prinzip der Nachbarschaftssuche beruht. Dabei wird iterativ versucht, eine aktuelle Lösung zu verbessern, indem zur aktuellen Lösung mehrere Nachbarschaftslösungen generiert und anhand einer Zielfunktion bewertet werden. Eine Lösung umfasst dabei eine Sequenz von Umladeoperationen sowie den daraus resultierenden Buchtzustand, welcher noch schlecht platzierte Container enthalten kann. Die Nachbarschaftslösung mit dem besten Zielfunktionswert wird dann die neue aktuelle Lösung. Als Zielfunktion verwenden die Autoren eine gewichtete Summe aus der Länge der Sequenz aus Umladeoperationen und der Anzahl der schlecht platzierten Container im resultierenden Buchtzustand.

Der Ansatz umfasst eine ganze Reihe von Heuristiken (von den Autoren subroutines genannt), die mehrfach nacheinander ausgeführt werden:

1. *Stack Emptying Subroutine:* Dabei wird ein beliebiger Stapel aus der Bucht geleert, indem die sich darin befindenden Container mit Umladeoperationen auf (beliebige) andere Stapel verteilt werden.

2. *Neighborhood Search Subroutine:* Durch randomisierte Operationen auf der aktuellen Lösung (beliebige Umladeoperation hinzufügen, beliebige Umladeoperation entfernen oder zwei beliebige Umladeoperationen in der Sequenz vertauschen) werden Nachbarschaftssequenzen generiert. Gegebenfalls entstehen dadurch ungültige Sequenzen, bei denen eine Umladeoperation von einem leeren oder zu einem vollen Stapel ausgeführt werden soll. Im Zuge einer „Rationalisierung“ werden derartige Umladeoperationen in der Sequenz identifiziert und entfernt. Auf diese Art und Weise werden solange Nachbarschaftssequenzen generiert, bis der Zielfunktionswert einer dieser Sequenzen eine Akzeptanzschranke erreicht bzw. unterschreitet. Diese Sequenz wird zur neuen aktuellen Lösung. Kann die Akzeptanzschranke auch nach einer bestimmten (im Artikel nicht näher spezifizierten) Anzahl an Versuchen nicht erreicht werden, wird die Schranke jeweils um 1 % verringert.

3. *Integer Programming Subroutine:* Mit diesem Element des Verfahrens wird versucht, die durch die vorherigen zwei Prozeduren angewachsene Länge der Lösungssequenz wieder zu reduzieren. Dazu werden für Container, die in der aktuellen Lösung mehr als einmal bewegt werden, zufällig alternative Pfade generiert, durch die der jeweilige Container am Ende auf derselben Stelle landen würde wie bei der aktuellen Lösung. Es wird ein ganzzahliges lineares Optimierungsproblem formuliert, das für jeden Container aus seinen alternativen Pfaden denjenigen auswählt, mit dem sich insgesamt die kürzeste Sequenz realisieren lässt. Dabei müssen natürlich eine Reihe von Nebenbedingungen berücksichtigt werden, insbesondere dass in der endgültigen Lösung nur Container bewegt werden können, die sich ganz oben auf dem Stapel befinden, sowie dass die maximale Stapelhöhe nicht überschritten werden darf.

4. *Mis-overlay Index Reducing Subroutine:* Hierbei handelt es sich eigentlich um zwei verschiedene Heuristiken: Erstens werden niedrige Stapel (im Aufsatz wird von einer Höhe ≤ 2 gesprochen) geleert, d.h. die Container in diesen Stapeln werden auf andere Stapel verteilt. Zweitens wird an dieser Stelle versucht, obenliegende Container, die momentan schlecht platziert sind, falls möglich durch eine Umladeoperation auf einen Stapel zu bewegen, auf dem sie gut platziert sind.

5. *Sequence Reducing Subroutine:* Mit dieser Heuristik wird versucht, eine Sorte von redundanten Umladeoperationen in der Sequenz zu eliminieren und diese damit verkürzen zu können. Die Heuristik kann nur greifen, wenn es in der Sequenz eine Umladeoperation eines Containers vom Stapel i zum Stapel j und später vom Stapel j zum Stapel k gibt. Wenn zwischen diesen Operationen in der Sequenz kein Umladeoperation eines anderen Containers auf den Stapel k existiert, kann die Umladeoperation (j, k) eliminiert werden und die Umladeoperation (i, j) in (i, k) geändert werden, also der Container einfach direkt von i nach k bewegt werden.

Nach Beendigung der *Sequence Reducing Subroutine* springt der Algorithmus wieder zur *Integer Programming Subroutine*. Dieses Vorgehen wird noch weitere zweimal wiederholt. Danach springt der Algorithmus, falls die Bucht noch nicht vorsortiert ist, wieder zur *Stack Emptying Subroutine*. Nach dreißig Iterationen wird der Algorithmus spätestens abgebrochen.

6.3.3 Caserta und Voß (2009b)

Der Lösungsansatz zum CPMP von Caserta und Voß (2009b) ist ein iteratives Verfahren, bei dem die Sequenz von Umladeoperationen mit jeder Iteration länger wird. Im wesentlichen handelt es sich um eine Nachbarschaftssuche, die, ähnlich wie bei Lee und Chao (2009), durch Heuristiken ergänzt wird. Eine Iteration des Algorithmus besteht aus der Auswahl einer zusätzlichen Umladeoperation für die aktuelle Lösung sowie der anschließenden Anwendung zweier Heuristiken. Die Autoren unterteilen den Ablauf einer Iteration in 4 Schritte:

1. *Corridor Definition/Selection*: Zunächst wird der Stapel ausgewählt, dessen oberster Container c in dieser Iteration bewegt werden soll. Die Wahrscheinlichkeit für die Wahl eines Stapels ist dabei proportional zur Anzahl der schlecht platzierten Container in diesem Stapel. Dann wird eine Anzahl von möglichen Empfängerstapeln für den gewählten Container gesucht. Dabei werden bei größeren Probleminstanzen nicht alle Stapel betrachtet, sondern nur eine fest vorgegebene Anzahl δ von Stapeln. Dieser Umstand wird als Definition eines Korridors interpretiert. Die Auswahl der Stapel, die zu diesem

Korridor gezählt werden, geschieht proportional zu einem Attraktivitätswert, der für jeden Stapel berechnet wird.

2. *Neighborhood Search:* Jede Umladeoperation des Containers c zu einem Stapel aus dem Korridor wird als möglicher Folgeschritt betrachtet. Es wird für jeden mögliche Folgeschritt die daraus resultierende Bucht berechnet.

3. *Move Evaluation and Selection:* Um die Attraktivität einer Folgebucht aus dem vorherigen Schritt bewerten zu können, wird die Anzahl der in den Folgebuchten jeweils schlecht platzierten Container als Grundlage genommen, wobei Buchten mit weniger schlecht platzierten Containern attraktiver sind als Buchten mit mehr schlecht platzierten Containern. Aus den attraktivsten 50 % dieser Buchten wird dann zufällig mit einer Wahrscheinlichkeit proportional zur Anzahl gut platzierter Container die Folgebucht gezogen. Die Umladeoperation, die zu der gezogenen Folgebucht führt, wird an die aktuelle Lösung angehängt.

4. *Local Search Improvement:* Mit zwei heuristischen Verfahren wird versucht, die neue aktuelle Lösung weiter zu verbessern:

   - *Heuristic Swap:* Wenn ein leerer Stapel in der Folgebucht existiert, wird er derart mit Containern von den anderen Stapeln gefüllt, dass ein Stapel maximaler Höhe entsteht, in dem sich ausschließlich gut platzierte Container befinden. Wenn ein Stapel mit nur einem Container gefunden wird, wird versucht, diesen Container auf einen Stack zu setzen, auf dem er gut platziert ist. Danach kann der daraus resultierende leere Stack wieder mit anderen Container gefüllt werden.

   - *Heuristic Subsequence Building:* Es werden verschiedene Umladeoperationen durchgeführt, durch die ein „enggepackter“ Stapel entstehen soll. Bei einem enggepacktem Stack ist die Differenz der Prioritäten von aufeinanderliegenden Containern ≤ 1.

Die vier Schritte werden solange wiederholt, bis eine Lösung gefunden oder eine Zeitschranke erreicht wurde.

6.3.4 Huang und Lin (2012)

Huang und Lin (2012) präsentieren ebenfalls eine Heuristik für das CPMP. Die Heuristik basiert auf mehreren Schritten, die solange wiederholt werden, bis die Bucht vorsortiert ist. Dabei wird in jeder Iteration die Sequenz der Umladeoperationen immer verlängert, eine einmal getroffene Entscheidung wird also nie revidiert.

1. Schritt: Die Stapel der Bucht werden in drei Klassen partitioniert:

   - L-Stapel: Stapel, in denen sich keine Container befinden.
   - W-Stapel: Nicht-leere Stapel, in denen sich mindestens ein schlecht platzierter Container befindet.
   - R-Stapel: Nicht-leere Stapel, in denen sich kein schlecht platzierter Container befindet.

2. Schritt: Es wird versucht, relativ hohe R-Stapel bis zur maximalen Stapelhöhe H mit Containern von W-Stapeln aufzufüllen, wobei die R-Stapel-Eigenschaft erhalten bleiben muss. Die Entscheidung, ob ein Stapel relativ hoch ist, wird mit Hilfe eines Parameters $\beta, 0 \leq \beta \leq 1$ getroffen. Ein Stapel wird genau dann als relativ hoch bezeichnet, wenn die Anzahl der Container in diesem Stapel mindestens $\beta \cdot H$ ist. Dabei werden solche Container von W-Stapeln zum Auffüllen ausgesucht, nach deren Entfernung der W-Stapel auch weiterhin ein W-Stapel bleibt. Sind mehrere Container zur Auswahl, wird derjenige Container gewählt, der minimale Prioritätendifferenz zum obersten Container im R-Stapel hat. Dies wird solange wiederholt, bis keine entsprechende Umladeoperation mehr möglich ist.

3. Schritt: Es wird versucht, die verbleibenden R-Stapel, also diejenigen, die nicht als relativ hoch bezeichnet werden, mit Containern von W-Stapeln aufzufüllen. Dabei werden die R-Stapel in einer zufälligen Reihenfolge nacheinander betrachtet. Analog zum vorherigen Schritt wird versucht, die R-Stapel um weitere Container der W-Stapel aufzustocken, wobei die R-Stapel-Eigenschaft erhalten bleiben muss. Wenn ein R-Stapel auf diese Weise nicht höher als $\beta \cdot H$ Container werden kann, werden die Umladeoperationen für diesen Stapel wieder zurückgenommen und der Stapel wird markiert.

4. Schritt: Im vorherigen Schritt markierte R-Stapel werden nach Möglichkeit abgebaut. Man beginnt beim niedrigsten dieser Stapel und betrachtet die Stapel dann aufsteigend nach ihrer Höhe. Jeder der Stapel wird gemäß bestimmter Regeln abgebaut, bis er leer ist oder bis keine Umladeoperation mehr möglich ist.

5. Schritt: In diesem Schritt werden Containern aus W-Stapeln auf leere Stapel verteilt. Dabei werden die Container derart verteilt, dass möglichst hohe R-Stapel erzeugt werden.

6. Schritt: Es wird geprüft, ob die Menge der W-Stapel leer ist, also ob das CPMP gelöst wurde. In diesem Fall terminiert der Algorithmus. Ansonsten werden die Schritte 1 bis 6 wiederholt.

Die Autoren beschreiben auch einen Lösungsansatz für eine Variante des CPMP, bei dem für jeden Container ein Zielplatz in der Bucht festgelegt wurde. Diese Variante wird in dieser Arbeit nicht weiter berücksichtigt.

6.4 Entwurf einer Heuristik für das CPMP

In diesem für die vorliegende Arbeit zentralen Abschnitt wird eine Baumsuche-Heuristik zur Lösung des CPMP beschrieben. Die Heuristik fußt auf dem Branch-and-Bound-Ansatz, der bereits im Abschnitt 4.4.1 erläutert wurde.

Wendet man einen Branch-and-Bound-Ansatz auf das CPMP an, so liegt es nahe, die Knoten des Suchbaums als Füllzustände einer Bucht des Zwischenlagers aufzufassen, repräsentiert durch die entsprechenden Matrizen. Die Wurzel des Suchbaums ist die Ausgangsmatrix L_{init}. Die direkten Nachfolgeknoten L' zu einer Matrix L werden dadurch erzeugt, dass man alle für L zulässigen Umladeoperationen konstruiert und jeweils einzeln auf L anwendet. Dieses Verzweigungsschema wendet man rekursiv auf alle neu erzeugten Matrizen an, bis man den Suchbaum soweit expandiert hat, dass sich an dessen Blättern nur noch vorsortierte Matrizen befinden.

Nun kann man durch Abzählen der Anzahl an Knoten zwischen einem Blatt und dem Wurzelknoten bestimmen, wie viele Umladeoperationen notwendig sind, um die

Bucht derart vorzusortieren. Dasjenige Blatt, das vom Wurzelknoten am kürzesten entfernt ist, entspricht der optimalen Lösung. Zusätzlich kann man mit Hilfe einer unteren Schranke an Umladeoperationen zur Lösung des CPMP für eine gegebene Matrix L Teilbäume des Suchbaums von der weiteren Betrachtung ausnehmen, wenn diese die optimale Lösung sicher nicht enthalten können.

Der soeben skizzierte Ansatz führt beim CPMP allerdings zu keinem befriedigenden Ergebnis. Die Ursache dafür wird klar, wenn man sich ansieht, wie viele Verzweigungen ausgehend von einem Knoten typischerweise vorgenommen werden müssen. Angenommen, man möchte eine Lösung des CPMP für eine Bucht bestehend aus 8 Stapeln und einer maximalen Stapelhöhe von 5 Containern finden. In allen Stapeln befinden sich jeweils 3 Container. Schon auf der Ebene direkt unter dem Wurzelknoten enthält der Suchbaum 56 Knoten, da es $8 \cdot 7 = 56$ zulässige Umladeoperationen gibt. Von jedem dieser Knoten würden sich wieder 56 Knoten verzweigen, von jedem dieser Knoten erneut und so weiter. Auf der sechsten Ebene enthält der Suchbaum bereits über 30 Milliarden Knoten. Auch wenn im späteren Verlauf des Algorithmus das Auftreten von leeren und vollen Stapeln dazu führt, dass teilweise etwas weniger als 56 Nachfolgeknoten generiert werden, ist es offensichtlich, dass die direkte Anwendung eines Branch-and-Bound-Ansatzes sehr rechen- und speicherintensiv ist.

Die nun vorgestellte Baumsuche-Heuristik leitet sich aus dem Branch-and-Bound-Ansatz ab, modifiziert diesen aber derart, dass auch für große CPMP-Testinstanzen Lösungen nach vergleichsweise kurzer Laufzeit gefunden werden. Dies geschicht allerdings auf Kosten der Optimalitätsgarantie, d.h. der Algorithmus findet in der Regel keine optimale Lösung.

6.4.1 Grundlegende Definitionen

Dieser Abschnitt enthält einige Definitionen, die für die Erläuterung der Baumsuche-Heuristik im weiteren Verlauf des Kapitels von Bedeutung sind.

Im Kapitel 6.2 wurden die Containerprädikate „schlecht platziert“ sowie „gut platziert“ eingeführt. Auf Basis dieser Prädikate lässt sich ein Klassifikationsschema für Umladeoperationen ableiten. Eine gegebene Umladeoperation m sei auf einen Buchtzustand L anwendbar. Die Umladeoperation ist vom Typ „schlecht-gut“ (kurz: SG),

wenn der durch m bewegte Container unmittelbar vor Ausführung der Umladeoperation schlecht platziert war und unmittelbar nach Beendigung der Umladeoperation gut platziert ist. Entsprechend sind die Umladeoperationstypen „gut-gut“ (GG), „schlecht-gut“ (SG) und „schlecht-schlecht“ (SS) definiert.

Mit SX sollen sowohl SG- als auch SS-Umladeoperationen bezeichnet werden. Analog fasst die Klasse GX sowohl GG- als auch GS-Umladeoperationen zusammen.

Ein Stapel wird als sauber bezeichnet, wenn er keinen schlecht platzierten Container enthält. Ansonsten wird er als unsauber bezeichnet. Entsprechend ist jeder Stapel entweder sauber oder unsauber.

Ein Platz (h, s) in einem Stapel s ist ein potenzieller Angebotsplatz für eine Gruppe g genau dann, wenn

- der Stapel s nicht leer ist,
- der höchste gut platzierte Container in s zur Gruppe g gehört und
- der Platz (h, s) über dem höchsten gut platzierten Container in s liegt.

Ein sauberer Angebotsplatz für eine Gruppe g ist ein potenzieller Angebotsplatz für g in einem sauberen Stapel. Ein leerer Stapel hat per Definition H Angebotsplätze für jede Gruppe g $(g = 1, \ldots, G)$.

Tabelle 6.1 enthält eine Reihe weiterer Definitionen, auf die im Verlauf des Kapitels zurückgegriffen wird. Alle Definitionen beziehen sich auf einen gegebenen Buchtzustand L.

6.4.2 Untere Schranke für die Anzahl an Umladeoperationen

In diesem Abschnitt wird eine untere Schranke für die Anzahl an Umladeoperationen beschrieben, die notwendig sind, um eine gegebene Bucht vorzusortieren. Zurückgreifend auf das im vorherigen Abschnitt eingeführte Klassifikationsschema für Umladeoperationen wird zunächst eine untere Schranke für die Anzahl an durchzuführenden SX-Umladeoperationen und dann eine untere Schranke für die Anzahl

Variable	Definition
n_b	Anzahl schlecht platzierter Container in einer Bucht
$n_b(s)$	Anzahl schlecht platzierter Container im Stapel s ($1 \leq s \leq S$)
$n^g(s)$	Anzahl gut platzierter Container im Stapel s mit Gruppen g' ($1 \leq g' < g \leq G$)
$n_m(seq)$	Anzahl Umladeoperationen in einer Sequenz seq von Umladeoperationen
n'_m	Untere Schranke für die Anzahl an Umladeoperationen zur Vorsortierung einer Bucht
n'_{SX}	Untere Schranke für die Anzahl an SG- und SS-Umladeoperationen zur Vorsortierung einer Bucht
n'_{GX}	Untere Schranke für die Anzahl an GG- und GS-Umladeoperationen zur Vorsortierung einer Bucht
$d(g)$	Nachfrage der Gruppe g, definiert als die Anzahl schlecht platzierter Container mit Gruppe g
$D(g)$	Kumulierte Nachfrage der Gruppe g, definiert durch: $d(g)+d(g+1)+\ldots+d(G)$
$s_p(g)$	Potenzielles Angebot für Gruppe g, definiert als die Anzahl potenzieller Angebotsplätze für g
$S_p(g)$	Kumuliertes potenzielles Angebot für Gruppe g, definiert durch: $s_p(g)+s_p(g+1)+\ldots+s_p(G)+H\cdot n_{s,empty}$, wobei $n_{s,empty}$ die Anzahl der leeren Stapel in einer Bucht bezeichnet
$D_e(g)$	Kumulierter Nachfrageüberschuss der Gruppe g, definiert durch: $D(g)-S_p(g)$
$s_c(g)$	Sauberes Angebot für Gruppe g, definiert als die Anzahl sauberer Angebotsplätze für Gruppe g
S_c	Sauberes Angebot in einer Bucht, definiert durch: $10^0 s_c(1)+10^1 s_c(2)+\ldots+10^{G-1}s_c(G)$

Tabelle 6.1: Variablen und Definitionen zur Erläuterung der Baumsuche-Heuristik für das CPMP

an durchzuführenden GX-Umladeoperationen abgeleitet. Ausgangspunkt ist immer ein Buchtzustand, der durch eine Matrix L repräsentiert wird.

Untere Schranke für die Anzahl an SX-Umladeoperationen

Die Vorsortierung ist genau dann erreicht, wenn jeder Container gut platziert ist. Es muss also mindestens eine SG-Umladeoperation für jeden Container durchgeführt werden, der anfänglich schlecht platziert ist. Entsprechend sind mindestens n_b SG-Umladeoperationen notwendig.

Sind in L alle Stapel unsauber, können SG-Umladeoperationen erst dann durchgeführt werden, wenn zumindest in einem Stapel alle schlecht platzierten Container entfernt wurden. Dies kann nur durch SS-Umladeoperationen geschehen, da ja alle potenziellen Empfängerstapel unsauber sind. Es müssen also mindestens $min\{n_b(s)|s = 1, \ldots, S\}$ SS-Umladeoperationen durchgeführt werden, bevor eine erste SG-Umladeoperation vorgenommen werden kann. Enthält L mindestens einen sauberen Stapel, ist die untere Schranke für SS-Umladeoperationen gleich null.

Somit ist $n'_{SX} = n_b + min\{n_b(s)|s = 1, \ldots, S\}$ eine untere Schranke für die Anzahl an SX-Umladeoperationen.

Untere Schranke für die Anzahl an GX-Umladeoperationen

Sei $g*$ die Gruppe in L, die den höchsten kumulierten Nachfrageüberschuss aufweist. Es gilt also $D_e(g*) \geq D_e(g)$ für $g = 1, \ldots, G$. Würden alle gut platzierten Container an ihrem Platz bleiben, würden $D_e(g*)$ Plätze fehlen, um die schlecht platzierten Container so umzusortieren, dass sie gut platziert sind. Die Stapel, in denen der höchste gut platzierte Container zu einer Gruppe $g < g*$ gehört, sollen als potenzielle GX-Stapel bezeichnet werden. Die fehlenden $D_e(g*)$ Plätze müssen durch Umladeoperationen von GX-Stapeln bereitgestellt werden, da alle anderen Stapel schon im Angebot berücksichtigt worden sind. Die Variable n^p_{GX} bezeichne die Anzahl potenzieller GX-Stapel. Damit in diesen Stapeln Container der Gruppe $g \geq g*$ untergebracht werden können, müssen dort alle gut platzierten Container entfernt werden. Das muss in mindestens $n^e_{GX} = max\{0, \lceil D_e(g*)/H \rceil\}$ der potenziellen GX-

1					3
4				3	3
2	4	3	2	4	3
2	3	1	1	1	1

Abbildung 6.4: Beispielbucht zur Berechnung einer unteren Schranke

Stapel passieren, da ein Stapel H Plätze bietet. Die n^p_{GX} potenziellen GX-Stapel seien nun aufsteigend nach der Anzahl der gut platzierten Container sortiert und gemäß dieser Sortierfolge temporär mit 1 bis n^p_{GX} indiziert. Dann gilt also die Ungleichung $n^{g*}(s) \leq n^{g*}(s+1)$ für $s = 1, \ldots, n^e_{GX} - 1$. Nach dem oben Gesagten müssen dann mindestens $n'_{GX} = \sum_{s=1}^{n^e_{GX}} n^{g*}(s)$ GX-Umladeoperationen ausgeführt werden.

Untere Schranke für die Anzahl aller Umladeoperationen

Es resultiert folgende untere Schranke n'_m für die Anzahl an Umladeoperationen zur Lösung des CPMP:

$$n'_m = n'_{SX} + n'_{GX} = n_b + min\{n_b(s)|s = 1, \ldots, S\} + \sum_{s=1}^{n^e_{GX}} n^{g*}(s) \tag{6.2}$$

Zur Veranschaulichung sei auf die Bucht mit 17 Containern in 6 Stapeln der maximalen Stapelhöhe 4 und 4 verschiedenen Containergruppen verwiesen, deren Anfangszustand in Abbildung 6.4 dargestellt ist.

Die untere Schranke für die Anzahl an SX-Umladeoperationen berechnet sich mit $n'_{SX} = n_b + min\{n_b(s)|s = 1, \ldots, S\} = 10 + 1 = 11$.

Zur Berechnung von n'_{GX} werden zunächst das Angebot und die Nachfrage für die Containergruppen berechnet (siehe Tabelle 6.1 für die Definitionen und Tabelle 6.2 für die Werte im konkreten Beispiel). In diesem Fall ist der maximale kumulative Nachfrageüberschuss 5 und gilt für die Containergruppe $g* = 3$. Wenn also alle initial gut platzierten Container an ihren Plätzen bleiben würden, würden noch 5 Plätze fehlen, um Container der Gruppen 3 und 4 gut platzieren zu können. Da

Containergruppe g	Nachfrage $d(g)$	kumulierte Nachfrage $D(g)$	potenzielles Angebot $s_p(g)$	kumuliertes potenzielles Angebot $S_p(g)$	kumulierter Angebotsüberschuss $D_e(g)$
4	3	3	0	0	3
3	5	8	3	3	**5**
2	1	9	2	5	4
1	1	10	12	17	-7

Tabelle 6.2: Angebot und Nachfrage für die Beispielbucht aus Abbildung 6.4

die maximale Stapelhöhe $H = 4$ ist, müssen gut platzierte Container in mindestens $n^e_{GX} = max\{0, \lceil D_e(3)/H \rceil\} = \lceil 5/4 \rceil = 2$ Stapeln mittels GX-Umladeoperationen bewegt werden, um die fehlenden 5 Plätze frei zu machen.

In Stapel 2 können mittels GX-Umladeoperationen keine weiteren Plätze für Container der Gruppen 3 und 4 freigemacht werden. Die Plätze über dem Container der Gruppe 3 wurden schon beim potenziellen Angebot berücksichtigt. Würde man diesen Container umladen, würde kein zusätzlicher Platz geschaffen, da der Container ja selbst wieder an einem geeigneten Platz geladen werden muss.

In den potenziellen GX-Stapeln 3 bis 6 muss jeweils mindestens ein Container umgeladen werden, um zusätzliche Plätze freizumachen, auf die Container der Gruppen 3 und 4 mittels GX-Umladeoperationen bewegt werden können. In Stapel 1 müssten beide gut platzierten Container der Gruppe 2 umgeladen werden, weshalb sich der Stapel nicht unter den ersten n^e_{GX} Stapeln befindet. Es müssen also mindestens 2 GX-Umladeoperationen durchgeführt werden: $n'_{GX} = 2$.

6.4.3 Algorithmus der Baumsuche und Verzweigungsansatz

In diesem Abschnitt werden der Algorithmus der Baumsuche und insbesondere der Verzweigungsansatz erläutert.

Beim Verzweigungsansatz werden zwei Maßnahmen getroffen, um die Größe des Suchbaums und damit die Rechenzeit des Algorithmus zu beschränken. Erstens wird

nicht nach jeweils einer Umladeoperation, sondern im Allgemeinen erst nach mehreren Umladeoperationen verzweigt. Dadurch wird die Tiefe des Suchbaums verringert. Zweitens wird nur eine feste, durch einen Parameter einstellbare Anzahl an Nachfolgeknoten generiert, wodurch die Breite des Suchbaums verringert wird. Nachfolgend wird zunächst der Verzweigungsansatz im Detail betrachtet, bevor der Grobalgorithmus der Baumsuche vorgestellt wird.

Umladeoperationsketten

In der Baumsuche-Heuristik werden direkte Nachfolgeknoten L' zu (einer Matrix) L durch Anwendung so genannter Umladeoperationsketten erzeugt. Eine Umladeoperationskette ist eine für eine Matrix L zulässige Sequenz von einzelnen Umladeoperationen. Vollständige und unvollständige Lösungen einer CPMP-Instanz werden durch Sequenzen von Umladeoperationsketten repräsentiert. Die gesamte Sequenz muss für den Ausgangszustand der Bucht L_{init} zulässig sein.

Bei der Generierung der Umladeoperationsketten wird wieder auf das vorher ausgeführte Klassifikationsschema für Umladeoperationen zurückgegriffen. Es gilt festzuhalten, dass nur SG-Umladeoperationen die Anzahl an schlecht platzierten Containern in einer Bucht verringern. Diese Anzahl kann als Abstand zu der angestrebten vollständig vorsortierten Bucht interpretiert werden; nur SG-Umladeoperationen können also diesen Abstand verringern. Entsprechend muss versucht werden, eine Lösung mit möglichst wenigen SS-, GG- und GS-Umladeoperationen zu ermitteln. Ferner sollte eine Lösung nicht mehr SG-Umladeoperationen enthalten als anfangs schlecht platzierte Container vorhanden sind. Diese Beobachtungen legen folgendes Vorgehen nahe:

- Es werden nur zwei Typen von Umladeoperationsketten betrachtet: Normale Umladeoperationsketten und Extra-Umladeoperationsketten. Normale Umladeoperationsketten bestehen ausschließlich aus SG-Umladeoperationen. Extra-Umladeoperationsketten bestehen aus SS-, GG- und GS-Umladeoperationen.
- Normale Umladeoperationsketten werden stets bevorzugt herangezogen. Konkret werden für den Fall, dass mindestens eine SG-Umladeoperation für einen gegebenen Buchtzustand anwendbar ist, ausschließlich normale Umladeope-

rationsketten generiert. Nur für den Fall, dass keine SG-Umladeoperationen möglich sind, werden Extra-Umladeoperationsketten generiert.

Normale Umladeoperationsketten. Normale Umladeoperatonsketten werden nach dem in Algorithmus 4 beschriebenen Verfahren erzeugt. Für jeden Stapel s wird maximal eine Umladeoperationskette erzeugt, die ausschließlich aus Umladeoperationen mit s als Sender besteht. Da nur SG-Umladeoperationen berücksichtigt werden, kann nur ein unsauberer Stapel als Sender s dienen. Die Generierung einer normalen Umladeoperationskette wird erst dann beendet, wenn von s aus keine weitere SG-Umladeoperation möglich ist. Es werden also immer Umladeoperationsketten maximaler Länge gebildet, da hierdurch der Abstand zu einer vorsortierten Bucht am meisten verkürzt werden kann. Zusätzlich können normale Umladeoperationsketten maximaler Länge auch dabei helfen, aus einem unsauberen Stapel s einen sauberen Stapel zu machen, und damit zusätzliche Optionen für zukünftige SG-Umladeoperationen zu schaffen.

Eine Umladeoperationskette cm wird nun schrittweise, also Umladeoperation für Umladeoperation, aufgebaut. Um die jeweils nächste Umladeoperation festzulegen, werden zunächst alle gültigen SG-Umladeoperationen von s im Nachfolgezustand L_c betrachtet. L_c repräsentiert den Buchtzustand, der sich durch Anwendung der bisher schon ausgewählten SG-Umladeoperationen in cm auf den aktuellen Buchtzustand L ergibt.

M bezeichne die Menge der gültigen SG-Umladeoperationen für den Zustand L_c. Die beste Umladeoperation $m_{best} \in M$ wird nach folgenden Filterregeln bestimmt:

1) Für jede Umladeoperation wird die Differenz δg zwischen dem Gruppenindex des obersten Containers im Senderstapel s und dem des obersten Containers im Empfängerstapel bestimmt. Für einen leeren Empfängerstapel wird der höchste Gruppenindex G benutzt. Alle Umladeoperationen aus M, für die δg nicht minimal ist, werden von weiteren Betrachtungen ausgeschlossen.

2) Aus den noch übrigen Umladeoperationen in M werden diejenigen bestimmt, deren Empfängerstapel die meisten Container beinhalten. Alle anderen Umladeoperationen werden aus M entfernt.

3) Aus den noch übrigen Umladeoperationen in M wird das erste Element als beste Umladeoperation m_{best} gewählt.

Die Filterregeln sollen sicherstellen, dass nach Anwendung von m_{best} möglichst viele weitere SG-Umladeoperationen durchgeführt werden können. Durch Filterregel 1 werden solche Umladeoperationen bevorzugt, nach deren Anwendung die Differenz der Containerprioritäten von aufeinanderliegenden Containern möglichst gering ist. Soll z.B. ein Container der Priorität 5 umgeladen werden, und es gibt dazu zwei mögliche (saubere) Empfängerstapel s_A und s_B, in denen der höchste Container die Priorität 5 bzw. 10 hat, ist es vorteilhaft, den Container auf s_A umzuladen. Würde man den Container auf s_B umladen, könnten dort im weiteren Verlauf nur noch Container bis zur Priorität 5 gut platziert werden. Lädt man den Container dagegen auf s_A um, können auf s_B weiterhin Container bis zur Priorität 10 gut platziert werden, während auf s_A auch weiterhin Container bis zur Priorität 5 gut platziert werden können.

Filterregel 2 führt dazu, dass die sauberen Containerstapel nicht alle gleichmäßig immer höher gestapelt werden, sondern dass im Gegenteil immer erst ein Containerstapel bis zur maximalen Stapelhöhe aufgefüllt wird, bevor ein anderer sauberer Containerstapel verwendet wird. Falls es notwendig sein sollte, einen sauberen Stapel im Verlauf des Lösungsverfahrens aufzulösen, ist damit die Wahrscheinlichkeit höher, dass dazu auch ein relativ niedriger Stapel vorhanden ist.

Im Anschluss wird für jede (nicht-leere) Umladeoperationskette geprüft, ob sie Teil einer neuen besten Lösung sein kann. Nur in diesem Fall wird cm der Liste der möglichen Operationsketten Cm hinzugefügt. Schließlich wird das saubere Angebot der letzten Nachfolgematrix L_c berechnet. Dieser Wert wird als sauberes Angebot für die Umladeoperationskette cm im späteren Verlauf des Verfahrens verwendet.

Eine Umladeoperationskette cm kann nur dann zu einer neuen besten Lösung führen, wenn die Ungleichung $n_m(s_c) + n_m(cm) + n'_m(L_c) < n_m(s^*)$ gilt. Dabei bezeichnet $n_m(s_c)$ bzw. $n_m(cm)$ die Anzahl an Umladeoperationen in der aktuellen (Teil-)Lösung s_c bzw. in cm. Weiterhin bezeichnet $n'_m(L_c)$ die untere Schranke für die Anzahl an Umladeoperationen, um die Bucht vom Buchtzustand L_c aus vorzusortieren. Schließlich steht $n_m(s^*)$ für die Anzahl an Umladeoperationen in der bisher besten bekannten Lösung s^*. Ganz zu Beginn der Baumsuche ist noch keine

einzige Lösung s^* bekannt. Daher wird $n_m(s^*)$ zu Beginn auf das gerundete Produkt $p_{ub} \cdot n'_m(L_{init})$ gesetzt. Der Faktor p_{ub}, $p_{ub} > 1.0$, ist ein Parameter und $n'_m(L_{init})$ die untere Schranke für die Anzahl an Umladeoperationen für den initialen Buchtzustand. Durch Verwendung einges geeigneten Faktors p_{ub} wird erreicht, dass sich die Baumsuche auf Lösungen mit wenigen Umladeoperationen fokussiert und der Suchaufwand verkürzt wird.

Im letzten Schritt werden die generierten Umladeoperationsketten in Cm absteigend nach der Anzahl ihrer Umladeoperationen und, mit niedrigerer Priorität, absteigend nach ihren sauberen Angebotswerten sortiert. Das zweite Sortierkriterium hilft dabei, bei gleich langen Umladeoperationsketten diejenigen zu bevorzugen, nach deren Anwendung potenziell mehr zukünftige SG-Umladeoperationen möglich sind. Schließlich werden (maximal) die besten $nSucc$ Umladeoperationen ausgewählt und von der Baumsuche-Heuristik zur Verzweigung verwendet; $nSucc$ ($nSucc > 1$) ist ein weiterer Parameter der Heuristik und gibt also die maximale Anzahl an Nachfolgerknoten pro Verzweigung an.

Extra-Umladeoperationsketten. Extra-Umladeoperationsketten werden in einem strukturell ähnlichen Verfahren konstruiert wie normale Umladeoperationsketten (siehe Algorithmus 5). Neben der Tatsache, dass in Extra-Umladeoperationsketten nur GG-, SS- und GS-Umladeoperationen verwendet werden (kurz: Nicht-SG-Umladeoperationen), gibt es drei wesentliche Unterschiede zum Generierungsverfahren für normale Umladeoperationsketten, auf die im Folgenden eingegangen wird.

Erstens werden teilweise andere Filterregeln angewendet, um die nächste Umladeoperation für eine Umladeoperationskette auszuwählen. Dabei bezeichne M nun eine Liste mit allen für den aktuellen Buchtzustand L_c gültigen Nicht-SG-Umladeoperationen.

1) Wenn es mindestens eine GG-Umladeoperation gibt, werden alle SS- und GS-Umladeoperationen aus M verworfen. Ansonsten wird geprüft, ob es mindestens eine Umladeoperation gibt, deren Empfängerstapel bereits vor der Umladeoperation ein unsauberer Stapel ist. Ist dies der Fall, werden alle Umladeoperationen mit sauberem Empfängerstapel verworfen.

Algorithmus 4 Erzeugung von normalen Umladeoperationsketten: determineNormalCompoundmoves

{Eingabe: aktuelle (Teil-)Lösung s_c, aktuelle Matrix L}
{Ausgabe: Liste Cm mit nCm normalen Umladeoperationsketten}

{Initialisiere Liste der Umladeoperationsketten}
1: $Cm \leftarrow \emptyset$
2: $nCm \leftarrow 0$

{Iteriere durch alle Stapel und baue Umladeoperationsketten auf}
3: **for all** $s \in \{1, \ldots, S\}$ **do**
4: Aktuelle Umladeoperationskette $cm \leftarrow \emptyset$
5: Aktuelle Nachfolgematrix $L_c \leftarrow L$
6: **repeat**
7: Generiere alle möglichen SG-Umladeoperationen von s in L_c
8: **if** Mindestens eine SG-Umladeoperation wurde gefunden **then**
9: Wähle die beste Umladeoperation m_{best} aus
{Erweitere die aktuelle Umladeoperationskette cm um m_{best}}
10: $cm \leftarrow cm \cup \{m_{best}\}$
{Wende m_{best} auf L_c an}
11: $L_c \leftarrow L_c \oplus m_{best}$
12: **end if**
13: **until** Umladeoperation m_{best} wurde nicht gefunden
14: **if** $cm \neq \emptyset$ und cm kann noch zu einer neuen besten Lösung führen **then**
15: Berechne sauberes Angebot $S_c(L_c)$ für cm
16: $Cm \leftarrow Cm \cup \{cm\}$
17: $nCm \leftarrow nCm + 1$
18: **end if**
19: **end for**

20: Sortiere die Umladeoperationsketten in Cm:
- absteigend nach Anzahl der Umladeoperationen und
- absteigend nach deren sauberem Angebot

{Beschränke die Länge der Liste auf maximal $nSucc$ Elemente}
21: $nCm \leftarrow min(nCm, nSucc)$
22: **end.**

2) Mit δg sei jeweils die Differenz zwischen dem Gruppenindex des obersten Containers im Senderstapel und des obersten Containers im Empfängerstapel für alle verbleibenden Umladeoperationen in M bezeichnet. Für einen leeren Empfängerstapel wird ein Gruppenindex von 0 verwendet. Wenn es mindestens eine Umladeoperation mit $\delta g \leq 0$ gibt, werden alle noch in M vorhandenen Umladeoperationen mit einem positiven Wert δg verworfen.

3) Aus den noch übrigen Umladeoperationen in M werden diejenigen bestimmt, deren Empfängerstapel die meisten Container beinhalten. Alle anderen Umladeoperationen werden aus M entfernt.

4) Aus den noch übrigen Umladeoperationen in M wird das erste Element als beste Umladeoperation m_{best} gewählt.

Durch die Regel 1 werden GG-Umladeoperationen bevorzugt, um neue Möglichkeiten für zukünftige SG-Umladeoperationen zu schaffen, sowie Umladeoperationen vermieden, mit denen saubere Stapel zu unsauberen Stapeln werden. Mit den anderen Regeln wird versucht, die negativen Effekte von SS- und GS-Umladeoperationen zu begrenzen. Durch Regel 2 werden die Container bevorzugt auf Container mit niedrigerer Priorität umgeladen, so dass Container mit höherer Priorität nicht blockiert werden. Mit Regel 3 wird sichergestellt, dass nicht alle Stapel gleichmäßig anwachsen, sondern dass es tendenziell hohe und niedrige Stapel gibt. Muss dann im weiteren Verlauf des Verfahrens ein Stapel aufgelöst werden, damit ein leerer Stapel erzeugt werden kann, kann dazu ein vergleichsweise niedriger Stapel genutzt werden.

Ein zweiter wesentlicher Unterschied zur Generierung von Extra- im Vergleich zu normalen Umladeoperationsketten ist der folgende: bei Extra-Umladeoperationensketten wird jede gültige Sequenz von Nicht-SG-Umladeoperationen (mit gleichem Sender s) als eine vollständige Extra-Umladeoperationskette betrachtet. Schließlich ist man ja bestrebt, so wenig Nicht-SG-Umladeoperationen wie möglich durchzuführen. Allerdings werden nur Umladeoperationsketten mit einem positiven sauberen Angebot akzeptiert, denn der einzige produktive Beitrag von Extra-Umladeoperationen besteht im Ermöglichen von zukünftigen SG-Umladeoperationen.

Schließlich werden - anders als bei normalen Umladeoperationsketten - die Extra-Umladeoperationsketten aufsteigend nach ihrer Länge sortiert und, mit niedrigerer

Priorität, absteigend nach ihren sauberen Angebotswerten. Darüber hinaus wird sichergestellt, dass die Umladeoperationskette mit maximalem sauberem Angebot immer unter den ersten $nSucc$ Umladeoperationsketten in der Liste Cm ist.

Grobalgorithmus der Baumsuche

Der Grobalgorithmus der Baumsuche-Heuristik wird durch die rekursive Prozedur *performCompoundMoves* realisiert (siehe Algorithmus 6). Vor dem ersten Aufruf werden folgende Variablen initialisiert: Aktuelle (Teil-)Lösung $s_c \leftarrow \emptyset$, beste gefundene Lösung $s^* \leftarrow \emptyset$, aktuelle Matrix $L \leftarrow L_{init}$. Wenn die Suche beendet ist, werden die beste gefundene Lösung s^* und die aus der Anwendung von s^* resultierende Matrix L zurückgegeben.

In der Prozedur *performCompoundMoves* wird zunächst geprüft, ob die Suche abgebrochen werden kann, d.h. ob entweder eine optimale Lösung gefunden oder das Zeitlimit überschritten wurde. Ferner wird die aktuelle Instanz der Prozedur beendet, wenn ein Blatt im Suchbaum erreicht wurde. Dann wird ggf. die beste bisher bekannte Lösung s^* aktualisiert.

Wurde die aktuelle Instanz der rekursiven Prozedur nicht abgebrochen, werden im nächsten Schritt maximal $nSucc$ Umladeoperationsketten erzeugt und dann alternativ angewendet. Eine Umladeoperationskette wird angewendet, indem erstens alle Umladeoperationen in der Umladeoperationskette an s angehängt werden, wodurch die neue (Teil-)Lösung s' entsteht. Zweitens werden die Umladeoperationen in der Umladeoperationskette auf die aktuelle Matrix L angewendet, um dadurch die Nachfolgematrix L' zu generieren. Die Prozedur *performCompoundMoves* wird dann für jede Umladeoperationskette erneut aufgerufen, mit s' und L' als Parameter.

Der Parameter $nSucc$ wird für eine Ausführung der Baumsuche auf einen konstanten ganzzahligen Wert oberhalb von 1 gesetzt ($nSucc > 1$). Durch die Beschränkung der Anzahl der Nachfolger eines Knoten auf maximal $nSucc$ wird die Breite des Suchbaumes beschränkt und die Baumsuche als heuristisches Verfahren ohne Optimalitätsgarantie ausgestaltet.

Algorithmus 5 Erzeugung von Extra-Umladeoperationsketten: determineExtraCompoundmoves

{Eingabe: aktuelle (Teil-)Lösung s_c, aktuelle Matrix L}
{Ausgabe: Liste Cm mit nCm Extra-Umladeoperationsketten}

{Initialisiere Liste der Umladeoperationsketten}
$Cm \leftarrow \emptyset$
$nCm \leftarrow 0$

{Durch alle Stapel iterieren und Umladeoperationsketten aufbauen}
for all $s \in S$ **do**
  Aktuelle Umladeoperationskette $cm \leftarrow \emptyset$
  Aktuelle Nachfolgematrix $L_c \leftarrow L$
  repeat
    Generiere alle möglichen Nicht-SG-Umladeoperationen von s in L_c
    if Mindestens eine Nicht-SG-Umladeoperation wurde gefunden **then**
      Wähle die beste Umladeoperation m_{best} aus
      {Erweitere die aktuelle Umladeoperationskette cm um m_{best}}
      $cm \leftarrow cm \cup \{m_{best}\}$
      {Wende m_{best} auf L_c an}
      $L_c \leftarrow L_c \oplus m_{best}$
      if cm kann zu einer neuen besten Lösung führen **then**
        Bestimme das saubere Angebot der Nachfolgematrix $S_c(L_c)$
        if $S_c(L_c) \geq 1$ **then**
          $Cm \leftarrow Cm \cup \{cm\}$
          $nCm \leftarrow nCm + 1$
        end if
      end if
    end if
  until Umladeoperation m wurde nicht gefunden oder cm kann nicht zu einer neuen besten Lösung führen
end for

Sortiere die Umladeoperationsketten in Cm:
- aufsteigend nach Anzahl der Umladeoperationen und
- absteigend nach deren sauberem Angebot

{Stelle sicher, dass cm mit maximalem sauberem Angebot berücksichtigt wird}
$cm_{mcs} \leftarrow$ Umladeoperationskette mit maximalem sauberem Angebot
if cm_{mcs} nicht unter den ersten $nSucc$ Elementen in Cm **then**
  Platziere cm_{mcs} an Stelle $nSucc$ in Cm
end if

{Beschränke die Länge der Liste auf maximal $nSucc$ Elemente}
$nCm \leftarrow min(nCm, nSucc)$
end.

Algorithmus 6 Baumsuche-Heuristik für das CPMP: performCompoundMoves

{Eingabe: aktuelle (Teil-)Lösung s_c, aktuelle Matrix L}
{Ausgabe: beste Lösung s^*}

{Breche die gesamte Suche ab bei opt. Lösung oder Erreichen des Zeitlimits}
1: **if** $s^* \neq \emptyset$ und $n_m(s^*) = n'_m(L_{init})$ oder Zeitlimit ist abgelaufen **then**
2: return
3: **end if**

{Breche die Suche auf dem aktuellen Suchpfad ab, wenn eine Lösung gefunden wurde. Aktualisiere die beste Lösung s^* wenn erforderlich.}
4: **if** L ist eine vorsortierte Matrix **then**
5: **if** $s^* = \emptyset$ oder $n_m(s_c) < n_m(s^*)$ **then**
6: $s^* \leftarrow s_c$
7: **end if**
8: return
9: **end if**

{Generiere nCm Umladeoperationsketten Cm für die Verzweigung}
10: $n_{SG} \leftarrow$ Anzahl an möglichen SG-Umladeoperationen in L
11: **if** $n_{SG} > 0$ **then**
12: determineNormalCompoundmoves
13: **else**
14: determineExtraCompoundmoves
15: **end if**

{Verzweige anhand der Umladeoperationsketten}
16: **for** $i \leftarrow 1$ **to** nCm **do**
17: $s' \leftarrow s_c \cup Cm(i)$
{Bestimme die Nachfolgematrix L' durch Anwendung von $Cm(i)$ auf L}
18: $L' \leftarrow L \oplus Cm(i)$
19: performCompoundMoves(s',L')
20: **end for**
21: **end.**

6.5 Evaluierung

In diesem Abschnitt wird die Leistungsfähigkeit der Baumsuche-Heuristik für das CPMP untersucht. Dabei wird zunächst ein Vergleich mit den in Kapitel 6.3 vorgestellten Lösungsansätzen für das CPMP vorgenommen. Danach werden die Ergebnisse von Performanzmessungen mit zufallsbasierten Probleminstanzen diskutiert. Abschließend werden einzelne wichtige Komponenten der Baumsuche-Heuristik separat evaluiert.

Für die Evaluierung wurde die im vorherigen Abschnitt beschriebene Baumsuche-Heuristik in der Programmiersprache C implementiert und mit XCode unter Mac OS X übersetzt. Alle Tests wurden auf einem Intel Core 2 Duo - Prozessor (P7350, 16 GFLOPS) mit 2 GHz Takfrequenz und 2 GB Hauptspeicher durchgeführt. Für alle Tests wurden durchgehend folgende Algorithmus-Parameterwerte festgelegt: $nSucc = 5$, $p_{ub} = 1,75$ (siehe jeweils Seite 91). Damit liegt das Zeitlimit am unteren Ende dessen, was in der Literatur für die Verwendung beim CPMP üblich ist. Alle Rechenzeiten in diesem Kapitel sind in Sekunden angegeben. Da es sich bei der Baumsuche-Heuristik um ein deterministisches Verfahren handelt, wurde jede Testinstanz nur einmal gelöst.

Zur Bestimmung der Leistungsfähigkeit der Verfahren werden folgende Kennzahlen betrachtet:

1. n^*_{rel} bzw. n_{rel}: Anzahl der Umladeoperationen in den Lösungen der Baumsuche-Heuristik bzw. eines Vergleichsverfahrens.

2. ct^* bzw. ct: Rechenzeit (Algorithmus-Laufzeit) der Baumsuche-Heuristik bzw. eines Vergleichsverfahrens in Sekunden.

3. LB_{diff}: (Absolute) Differenz zwischen n^*_{rel} und der zu Beginn des Kapitels definierten unteren Schranke für das CPMP. Ein kleiner LB_{diff}-Wert bedeutet, dass das Ergebnis der Baumsuche-Heuristik nahe am Optimalwert liegt.

4. $\frac{n^*_{rel}}{n_{rel}}$: Verhältnis zwischen der Anzahl an Umladeoperationen in den Lösungen der Baumsuche-Heuristik und denen der eines Vergleichsverfahrens. Werte kleiner als 1 bedeuten, dass die Baumsuche-Heuristik bessere Lösungen finden konnte als das Vergleichsverfahren. Für Werte größer als 1 gilt das Gegenteil.

			3	1	
3	2	1	1	3	2
1	1	3	2	1	1

Abbildung 6.5: CPMP-Testinstanz lh1 von Lee und Hsu (2007)

		4				5		6			
2		2		3		1	1	5			3
2	5	6		4	1	5	2	3	6	6	5
4	1	3	5	6	6	2	6	1	3	4	4
4	2	6	3	2	1	6	1	5	2	1	6

Abbildung 6.6: CPMP-Testinstanz lh2 von Lee und Hsu (2007)

Im Folgenden wird der Begriff Testklasse verwendet, um eine Menge von Testinstanzen zusammenfassend zu bezeichnen, die gleiche Werte für bestimmte, eine Bucht und deren Zustand charakterisierende Faktoren (z.B. Stapelanzahl und Containeranzahl) besitzen. Wird dann davon gesprochen, dass eine Testklasse gelöst wurde, ist damit gemeint, dass alle Testinstanzen der Testklasse gelöst wurden. Bei Testklassen entsprechen die Kennzahlen n^*_{rel}, n_{rel}, ct^*, ct und LB_{diff} jeweils den Mittelwerten der Kennzahlen für die einzelnen Testinstanzen der Testklasse. Der Wert für $\frac{n^*_{rel}}{n_{rel}}$ bezieht sich in diesem Fall ebenfalls auf die durch Mittelwertsbildung gewonnenen Werte für n^*_{rel} und n_{rel}. Testklassen werden in Großbuchstaben notiert, während einzelne Testinstanzen in Kleinbuchstaben notiert werden.

6.5.1 Vergleich mit bisher publizierten Lösungsverfahren

Vergleich mit Lee und Hsu (2007)

Lee und Hsu (2007) evaluieren ihre Heuristik mit einer kleinen Probleminstanz (vgl. Abbildung 6.5) sowie einer größeren Probleminstanz (vgl. Abbildung 6.6). Wesentliche Daten beider Instanzen werden in Tabelle 6.3 zusammengefasst. Die Berechnungen wurden auf einem PC mit 3,2 GHz Taktfrequenz und 1024 MB Hauptspeicher durchgeführt.

Für die kleine Probleminstanz lh1 findet die Heuristik von Lee und Hsu eine Lösung mit 10 Umladeoperationen in weniger als einer Sekunde. Für die große Pro-

			Container		
Testinstanz	Anzahl Stapel	Stapelhöhe	Prioritäten	Anzahl	schlecht platziert
lh1	6	4	3	14	6
lh2	12	5	6	45	25

Tabelle 6.3: Eigenschaften der CPMP-Testinstanzen von Lee und Hsu (2007)

	Lee und Hsu (2007)		Baumsuche-Heuristik			
Testinstanz	n_{rel}	ct	n^*_{rel}	ct^*	LB_{diff}	$\frac{n^*_{rel}}{n_{rel}}$
lh1	10,00	<1	9,00	<1	0,00	0,90
lh2	47,00	318	30,00	<1	2,00	0,64
Mittelwert						0,77

Tabelle 6.4: Leistungsvergleich zwischen dem Verfahren von Lee und Hsu (2007) und der Baumsuche-Heuristik für das CPMP

bleminstanz lh2 benötigt das Verfahren mit 318 Sekunden wesentlich länger. Die gefundene Lösung besteht aus 47 Umladeoperationen.

Die in dieser Arbeit vorgestellte Baumsuche-Heuristik findet für die kleine Probleminstanz eine Lösung mit nur 9 Umladeoperationen und für die große Probleminstanz eine Lösung mit nur 30 Umladeoperationen (vgl. Tabelle 6.4). Damit kann das Ergebnis der Heuristik von Lee und Hsu (2007) in beiden Fällen übertroffen werden. Die kleine Probleminstanz konnte sogar optimal gelöst werden. Ein Vergleich mit der unteren Schranke für die große Probleminstanz zeigt, dass die gefundene Lösung nur maximal zwei Umladeoperationen von einer optimalen Lösung entfernt ist. Darüber hinaus terminiert die Baumsuche-Heuristik für beide Testinstanzen in weniger als 1 Sekunde.

Vergleich mit Lee und Chao (2009)

Lee und Chao (2009) testen ihren Algorithmus für das CPMP mit 12 verschiedenen CPMP-Instanzen auf einem Pentium IV mit 2,4 GHz Taktfrequenz und 1024 MB Hauptspeicher. Die erste Instanz (lc1) besteht aus 10 Stapeln mit einer maxima-

1								4	
3		7			0	6		6	5
3	0	6		5	5	7		9	5
3	6	5	2	8	0	5		6	1
5	8	1	5	8	2	8	2	1	4

Abbildung 6.7: CPMP-Testinstanz lc1 von Lee und Chao (2009)[2]

len Stapelhöhe von 5 (vgl. Abbildung 6.7). Es befinden sich 35 Container mit 10 verschiedenen Prioritäten in der Bucht. 13 Container sind initial schlecht platziert.

Der Algorithmus von Lee and Chao findet eine Lösung mit 31 Umladeoperationen nach 5 Sekunden. Die Baumsuche-Heuristik findet eine Lösung mit 17 Umladeoperationen nach 1 Sekunde. Die untere Schranke für diese Probleminstanz beträgt 15.

Eine zweite, etwas größere Probleminstanz besteht aus 12 Stapeln mit einer maximalen Stapelhöhe von 6, 50 Containern und 10 verschiedenen Prioritäten. Der Algorithmus von Lee und Chao (2009) findet dafür nach 675 Sekunden eine Lösung mit 61 Umladeoperationen. Da die genaue Buchtkonfiguration nicht angegeben wird und die Autoren keine Angabe zur Anzahl der anfänglich schlecht platzierten Container machen[3], wurden zwei verschiedene Testklassen mit je mehreren Instanzen generiert und berechnet. Die generierten Testinstanzen haben in Bezug auf die Faktoren Stapelanzahl, maximale Stapelhöhe, Containeranzahl und Anzahl von Containerprioritäten die gleichen Werte wie die Testinstanzen von Lee und Chao (2009). Für die (ungewisse) Anzahl an zu Beginn schlecht platzierten Containern wurden Annahmen getroffen. Im ersten Fall wurde angenommen, dass 19 Container anfänglich schlecht platziert sind. Damit entspricht der relative Anteil der schlecht platzierten Container dem Anteil in lc1. Im zweiten Fall wurde mit 35 eine wesentlich höhere Anzahl an schlecht platzierten Containern angenommen. Für beide Fälle wurden jeweils 10 Testinstanzen zufallsbasiert generiert. Der erste Fall wird mit LC2a, der zweite Fall mit LC2b bezeichnet.

Abschließend präsentieren die Autoren Berechnungsergebnisse für 10 Testinstanzen (LC3), die fast gleiche Charakteristika wie die Instanz lc2 aufweisen, allerdings

[2]Während üblicherweise die Nummerierung der Containergruppen bei 1 beginnt, nummerieren die Autoren die Containergruppen bei 0 beginnend.

[3]Mehrere Anfragen des Verfassers der vorliegenden Arbeit zur Überlassung der Testinstanzen sowie zur Konfiguration der Testinstanzen blieben unbeantwortet.

befinden sich jeweils 4 Container mehr in der Bucht. Da wieder keine Angaben zu den anfänglich schlecht platzierten Containern gemacht wurden, werden analog zum Vorgehen bei lc2 zwei Testklassen betrachtet. Beide bestehen wieder aus 10 zufallsbasierten Testinstanzen, wobei bei LC3a anfänglich 21 Container schlecht platziert sind und bei LC3b 36. Tabelle 6.5 zeigt eine Übersicht der verwendeten Testinstanzen.

Um zumindest indikative Aussagen treffen zu können, wurden die von Lee und Chao (2009) publizierten Ergebnisse für die Testinstanz lc2 und die Testklasse LC3 jeweils mit den Mittelwerten der Baumsuche-Heuristik für die 10 Testinstanzen der Testklassen LC2a und LC2b bzw. LC3a und LC3b verglichen.

Die Ergebnisse des Leistungsvergleichs sind der Tabelle 6.6 zu entnehmen. Für die Testklassen sind Durchschnittswerte angegeben. Der Wert für $\frac{n^*_{rel}}{n_{rel}}$ in der Zeile für die Testklasse LC2a bzw. LC2b ergibt sich aus der Division des Mittelwertes für n^*_{rel} über alle Testinstanzen von LC2a bzw. LC2b durch den Wert von n_{rel} für die Testinstanz lc2. Der Wert für $\frac{n^*_{rel}}{n_{rel}}$ in der Zeile für die Testklasse LC3a bzw. LC3b ergibt sich aus der Division des Mittelwertes für n^*_{rel} über alle Testinstanzen von LC3a bzw. LC3b durch den Wert von n_{rel} für die Testklasse LC3.

Verglichen mit dem Algorithmus von Lee und Chao (2009) konnte die Baumsuche-Heuristik deutlich bessere Lösungen finden. Dies gilt unter der Annahme, dass die mit der Baumsuche-Heuristik gelösten Testklassen LC2a, LC2b, LC3a und LC3b nicht leichter zu lösen sind als die von Lee und Chao (2009) verwendete Testinstanz lc2 bzw. die Testklasse LC3. Dies ist bei der Interpretation der Werte für $\frac{n^*_{rel}}{n_{rel}}$ für die genannten Testklassen sowie entsprechend für den Mittelwert zu berücksichtigen. Die Rechenzeit lag in allen Fällen unter 5 Sekunden. Die Anzahl der Umladeoperationen in den gefundenen Lösungen liegen nur unwesentlich über der unteren Schranke für die Probleminstanzen, es wurden also nah-optimale Lösungen gefunden.

Vergleich mit Caserta und Voß (2009b)

Caserta und Voß (2009b) präsentieren ebenfalls eine Lösung für die Testinstanz lc1 aus Abbildung 6.7. Die Lösung besteht aus 19 Umladeoperationen. Die Baumsuche-

				Container		
Testklasse/ -instanz	Anzahl Instanzen	Anzahl Stapel	Stapel- höhe	Prioritäten	Anzahl	schlecht platziert
lc1		10	5	10	35	13
lc2		12	6	10	50	?
LC2a	10	12	6	10	50	19
LC2b	10	12	6	10	50	39
LC3	10	12	6	10	54	?
LC3a	10	12	6	10	54	21
LC3b	10	12	6	10	54	36

Tabelle 6.5: Eigenschaften der CPMP-Testinstanzen von Lee und Chao (2009)

	Lee und Chao (2009)		Baumsuche-Heuristik			
Testklasse/-instanz	n_{rel}	ct	n^*_{rel}	ct^*	LB_{diff}	$\frac{n^*_{rel}}{n_{rel}}$
lc1	31,00	5	17,00	<1	1,00	0,55
lc2	61,00	675				
LC2a			22,60	1	1,50	0,37
LC2b			38,40	2	1,10	0,63
LC3	75,80	2106				
LC3a			23,70	1	1,90	0,31
LC3b			42,70	4	2,80	0,56
Mittelwert						0,48

Tabelle 6.6: Leistungsvergleich zwischen dem Verfahren von Lee und Chao (2009) und der Baumsuche-Heuristik für das CPMP

				Container		
Testklasse	Instanzen	Stapel	Stapel-höhe	Prioritäten	Anzahl	schlecht platziert
CV1	10	3	9	9	9	5,4
CV2	10	4	16	16	16	10,6
CV3	10	5	25	25	25	17,5
CV4	10	6	36	36	36	26,5

Tabelle 6.7: Eigenschaften der CPMP-Testinstanzen von Caserta und Voß (2009b)

Heuristik konnte die Testinstanz lc1 mit 17 Umladeoperationen lösen (siehe Tabelle 6.6).

Ferner testen die Autoren die Leistungsfähigkeit ihres Ansatzes mit vier verschiedenen Testklassen CV1 bis CV4 (siehe Tabelle 6.7). Jede Testklasse besteht aus 10 Testinstanzen mit gleicher Anzahl an Stapeln S und Containern. Die Bucht ist zu Beginn derart mit Containern gefüllt, dass sich in jedem Stapel so viele Container befinden wie Stapel in der Bucht existieren. Jeder Container hat eine Containerpriorität exklusiv, so dass jede Priorität nur genau einmal vergeben ist. Es wird davon ausgegangen, dass die Stapel nicht in der Höhe begrenzt sind. Dazu ist es offensichtlich ausreichend, die maximale Stapelhöhe auf S^2 festzulegen.

Die Ergebnisse für die Testklassen CV1 bis CV4 sind der Tabelle 6.8 zu entnehmen. Die Autoren testeten ihren Algorithmus auf einer Linux Workstation mit Pentium IV-Prozessor und 512 MB Hauptspeicher. Es wurde ein Zeitlimit von 20 Sekunden definiert. Der Ansatz von Caserta und Voß (2009b) beinhaltet Zufallsentscheidungen. Daher wurde jede Testinstanz zehnmal berechnet und für jede Instanz wurde dann der Mittelwert der zehn Testläufe angegeben.

Für die Testklassen CV1, CV2 und CV3 konnte die Baumsuche-Heuristik signifikant bessere Lösungen in weniger als 10 Sekunden Berechnungszeit liefern. Für CV4 konnte die Baumsuche-Heuristik ebenfalls deutlich bessere Lösungen berechnen. Die Differenz zur unteren Schranke ist gering für die Testklassen CV1 und CV2 und bleibt auch für die anderen Testklassen unter 33 % der Anzahl an Umladeoperationen.

	Caserta und Voß (2009b)		Baumsuche-Heuristik			
Testklasse	n_{rel}	ct	n^*_{rel}	ct^*	LB_{diff}	$\frac{n^*_{rel}}{n_{rel}}$
CV1	21,30	20	10,50	<1	2,60	0,49
CV2	28,27	20	19,10	<1	5,50	0,68
CV3	49,61	20	30,40	9	9,10	0,61
CV4	50,14	20	44,40	20	16,30	0,89
Mittelwert						0,67

Tabelle 6.8: Leistungsvergleich zwischen dem Verfahren von Caserta und Voß (2009b) und der Baumsuche-Heuristik für das CPMP

	Huang und Lin (2012)		Baumsuche-Heuristik			
Testinstanz	n_{rel}	ct	n^*_{rel}	ct^*	LB_{diff}	$\frac{n^*_{rel}}{n_{rel}}$
lh1	9,00	<3	9,00	<1	1,00	1,00
lh2	35,00	<3	30,00	<1	2,00	0,86
Mittelwert						0,93

Tabelle 6.9: Leistungsvergleich zwischen dem Verfahren von Huang und Lin (2012) und der Baumsuche-Heuristik für das CPMP

Vergleich mit Huang und Lin (2012)

Huang und Lin (2012) evaluieren die Leistungsfähigkeit ihrer Heuristik an den zwei von Lee und Hsu (2007) beschriebenen Testinstanzen.

Für die kleine Testinstanz findet die Heuristik von Huang und Lin (2012) eine Lösung mit 9 Umladeoperationen. Für die große Probleminstanz besteht die beste mit der Heuristik von Huang und Lin (2012) gefundene Lösung aus 35 Umladeoperationen. Die Lösungen werden in beiden Fällen in weniger als 3 Sekunden gefunden, die genaue Laufzeit wird allerdings nicht angegeben. Für die Berechnung kam ein Pentium PC mit 3,4 GHz Taktfrequenz und 1,5 GB Hauptspeicher zum Einsatz.

Wie bereits im Vergleich mit den Ergebnissen von Lee und Hsu gezeigt, findet die in dieser Arbeit vorgestellte Baumsuche-Heuristik für die kleine Testinstanz ebenfalls eine mit 9 Umladeoperationen optimale Lösung. Für die große Probleminstanz

findet sie mit 30 Umladeoperationen eine um 5 Umladeoperationen kürzere Lösungsseqzuenz als die Heuristik von Huang und Lin (2012). Die Laufzeit des Algorithmus liegt für beide Instanzen unter einer Sekunde (vgl. Tabelle 6.9).

6.5.2 Evaluierung mit neuen Testinstanzen

In diesem Abschnitt werden Ergebnisse von Tests mit neu erstellten, zufallsbasierten Testinstanzen vorgestellt, um die Leistungsfähigkeit der Baumsuche-Heuristik weitergehend zu untersuchen.

Bei der Generierung der Testinstanzen wurden folgende fünf Faktoren variiert:

- Stapelanzahl: Die Buchten bestehen aus entweder 16 oder 20 Stapeln. Buchten in Containerterminals, die aus 16 Stapeln bestehen, werden bereits als sehr groß eingestuft (vgl. Lee und Hsu, 2007).
- Maximale Stapelhöhe: Die Stapel haben eine maximale Stapelhöhe von 5 oder 8 Containerhöhen.
- Containerprioritäten: Die Anzahl an unterschiedlichen Containerprioritäten wurde so gewählt, dass sie entweder 20 % oder 40 % der Anzahl an Containern in der Bucht entspricht.
- Containeranzahl: Die Containeranzahl wurde so gewählt, dass entweder 60 % oder 80 % aller verfügbaren Plätze in der Bucht mit Containern belegt sind.
- Anzahl der zu Beginn schlecht platzierten Containern: Es sind entweder 60 % oder 75 % der Container zu Beginn schlecht platziert.

Für jeden der Faktoren gibt es also zwei Ausprägungen: eine mit niedrigerem und eine mit höherem Wert.

Gemessen an den Anforderungen in Containerterminals handelt es sich hier um große Testinstanzen, da nur wenige, hautpsächlich mit RMGs bewirtschaftete Containerterminals annähernd derartige Buchtbreiten und -höhen vorweisen können (vgl. Abschnitt 3.2.2).

Insgesamt wurden 32 verschiedene Testklassen definiert. Eine Testklasse repräsentiert dabei genau eine der 2^5 möglichen Kombinationen aus den zwei Ausprägungen der fünf oben genannten Faktoren. Für jede Testklasse wurden 20 Testinstanzen zufallsbasiert generiert (siehe Tabelle 6.10). Die 640 CPMP Testinstanzen wurden anschließend mit der Baumsuche-Heuristik bei einem Zeitlimit von 60 Sekunden vorsortiert.

Die Ergebnisse sind Tabelle 6.11 zu entnehmen. Für jede Testklasse ist dort die durchschnittliche Anzahl an Umladeoperationen n^*_{rel}, die durchschnittliche Laufzeit des Algorithmus ct^* in Sekunden und die durchschnittliche absolute Differenz zur unteren Schranke LB_{diff} angegeben.

Tabelle 6.12 zeigt eine andere Sicht auf die Ergebnisse. Hier lässt sich der Einfluss der bei der Generierung der Testinstanzen variierten Faktoren auf die Baumsuche-Heuristik hinsichtlich der durchschnittlichen Laufzeit des Algorithmus ct^* sowie der durchschnittlichen absoluten Differenz zur unteren Schranke LB_{diff} ablesen. Werte in einer Spalte mit der Überschrift „niedriger“ beziehen sich auf die Testklassen, bei denen der jeweilige Faktor den niedrigeren Wert besitzt, Werte in einer Spalte mit der Überschrift „höher“ beziehen sich entsprechend auf Testklassen, bei denen der jeweilige Faktor den höheren Wert besitzt.

Bei gestiegener Stapelanzahl erhöht sich sowohl die Laufzeit der Baumsuche-Heuristik (um 53 % von ca. 20 auf 31 Sekunden) als auch die mittlere absolute Differenz zur unteren Schranke (um 29 % von ca. 10 auf ca. 13 Umladeoperationen).

Die maximale Stapelhöhe hat einen sehr starken Einfluss auf die Laufzeit und die Differenz zur unteren Schranke. Bei der niedrigeren Stapelhöhe von 5 Containerhöhen findet die Baumsuche-Heuristik im Mittel nah-optimale Lösungen in ca. 8 Sekunden, die absolute Differenz zur unteren Schranke liegt unter einer Umladeoperation. Bei der höheren Stapelhöhe von 8 dagegen erhöht sich die Laufzeit im Durchschnitt um 462 % von unter 8 Sekunden auf ca. 44 Sekunden pro Testinstanz. Die durchschnittliche absolute Abweichung zur unteren Schranke erhöht sich um den Faktor 27 auf knapp 23.

Die Containeranzahl hat einen ebenfalls starken Einfluss auf Laufzeit und Differenz zur unteren Schranke, wenn auch nicht im gleichen Maße wie die maximale Stapelhöhe. Bei den Testinstanzen mit geringerer Anzahl von Containern in der

Bucht liegt die durchschnittliche Laufzeit bei ca. 14 Sekunden, bei den Instanzen mit höherer Anzahl von Containern erhöht sie sich um 165 % auf ca. 37 Sekunden. Die absolute Differenz zur unteren Schranke wächst dabei fast um den Faktor 13, von unter 2 auf knapp 22 Umladeoperationen.

Der Faktor „Containerprioritäten“ hat einen geringeren Einfluss auf Laufzeit und Differenz zur unteren Schranke. Die Laufzeit erhöht sich bei den Testinstanzen mit einer höheren Anzahl an Containerprioritäten verglichen mit den Testinstanzen mit einer niedrigeren Anzahl von Containerprioritäten um 17 %, von ca. 24 auf ca. 28 Sekunden. Die mittlere absolute Abweichung zur unteren Schranke steigt um 30 %, von ca. 10 auf ca. 13 Umladeoperationen.

Bei den Testinstanzen, bei denen eine höhere Anzahl von Containern zu Beginn schlecht platziert sind, ist die Laufzeit der Baumsuche-Heuristik im Mittel um 10 % geringer als bei den Testinstanzen mit niedrigerer Anzahl an zu Beginn schlecht platzierten Containern. Die durchschnittliche absolute Differenz zur unteren Schranke liegt allerdings um 6 % bzw. etwas weniger als eine Umladeoperation höher. Die Ursache für die Verringerung der Laufzeit könnte darin liegen, dass die Anzahl an zu Beginn schlecht platzierten Containern ein wichtiges Element in der Berechnung der unteren Schranke darstellt. Ist die Anzahl an zu Beginn schlecht platzierten Containern höher, verbessert sich potenziell die Güte der unteren Schranke. Damit können häufiger Äste des Suchbaums abgeschnitten werden, so dass der Algorithmus früher terminiert.

Es lässt sich abschließend festhalten, dass die Baumsuche-Heuristik für die Testinstanzen mit einer geringeren maximalen Stapelhöhe und einer geringeren Anzahl von Containern in der Bucht sehr gute Ergebnisse liefert. Im Mittel liegt hier die maximale absolute Differenz zu einer optimalen Lösung bei weniger als 2 Umladeoperationen, und der Algorithmus terminiert im Durchschnitt nach weniger als 15 Sekunden. Sind die Werte der zwei genannten Faktoren erhöht, führt dies zu einer signifikant höheren Laufzeit und Differenz zur unteren Schranke. Die anderen untersuchten Faktoren haben einen deutlich geringeren Einfluss.

				Container		
Testklasse	Instanzen	Stapel	Stapel-höhe	Anzahl	Prioritäten	schlecht platziert
BF1	20	16	5	48	10	29
BF2	20	16	5	48	10	36
BF3	20	16	5	48	20	29
BF4	20	16	5	48	20	36
BF5	20	16	5	64	13	39
BF6	20	16	5	64	13	48
BF7	20	16	5	64	26	39
BF8	20	16	5	64	26	48
BF9	20	16	8	77	16	47
BF10	20	16	8	77	16	58
BF11	20	16	8	77	31	47
BF12	20	16	8	77	31	58
BF13	20	16	8	103	21	62
BF14	20	16	8	103	21	78
BF15	20	16	8	103	42	62
BF16	20	16	8	103	42	78
BF17	20	20	5	60	12	36
BF18	20	20	5	60	12	45
BF19	20	20	5	60	24	36
BF20	20	20	5	60	24	45
BF21	20	20	5	80	16	48
BF22	20	20	5	80	16	60
BF23	20	20	5	80	32	48
BF24	20	20	5	80	32	60
BF25	20	20	8	96	20	58
BF26	20	20	8	96	20	72
BF27	20	20	8	96	39	58
BF28	20	20	8	96	39	72
BF29	20	20	8	128	26	77
BF30	20	20	8	128	26	96
BF31	20	20	8	128	52	77
BF32	20	20	8	128	52	96

Tabelle 6.10: Eigenschaften der neuen Testinstanzen für das CPMP

Testklasse	n^*_{rel}	ct^*	LB_{diff}
BF1	29,15	<1	0,05
BF2	36,10	<1	0,10
BF3	29,40	<1	0,30
BF4	36,15	<1	0,15
BF5	41,70	4	1,20
BF6	48,60	5	1,45
BF7	43,25	10	1,95
BF8	51,90	11	2,85
BF9	52,05	11	2,75
BF10	60,35	10	1,95
BF11	53,65	17	4,45
BF12	61,45	13	3,25
BF13	96,05	60	30,05
BF14	111,55	60	30,03
BF15	106,85	60	41,10
BF16	124,00	60	42,25
BF17	36,60	<1	0,05
BF18	45,15	<1	0,15
BF19	36,55	<1	0,10
BF20	45,00	<1	0,00
BF21	51,65	21	1,00
BF22	61,40	14	0,75
BF23	51,75	26	1,40
BF24	62,10	25	1,45
BF25	62,90	44	2,55
BF26	74,10	25	2,00
BF27	64,35	55	4,00
BF28	3,95	42	3,95
BF29	122,00	60	40,60
BF30	147,90	60	48,16
BF31	131,20	60	50,10
BF32	153,75	60	54,55

Tabelle 6.11: Berechnungsergebnisse der Baumsuche-Heuristik für das CPMP für die neuen Testinstanzen

	ct^*		LB_{diff}	
Faktor	Niedriger	Höher	Niedriger	Höher
Stapelanzahl	20,31	31,00	10,24	13,18
Maximale Stapelhöhe	7,75	43,56	0,81	22,61
Containeranzahl	14,06	37,25	1,61	21,81
Anzahl Containerprioritäten	23,63	27,69	10,18	13,24
Anzahl schlecht platzierter Container	27,00	24,31	11,35	12,07

Tabelle 6.12: Einfluss unterschiedlicher Faktoren der neuen Testinstanzen auf die Baumsuche-Heuristik für das CPMP

6.5.3 Evaluierung einzelner Komponenten der Baumsuche-Heuristik

In diesem Abschnitt werden einzelne Komponenten der Baumsuche-Heuristik evaluiert.

Zunächst wird die in der Baumsuche-Heuristik verwendete untere Schranke isoliert betrachtet. Dass die Anzahl schlecht platzierter Container eine untere Schranke für die Anzahl an Umladeoperationen bildet, ist selbstverständlich. Daher stellt sich die Frage, wie stark die untere Schranke durch die Einbeziehung der SS- und GX-Umladeoperationen verbessert wird.

Um sie zu beantworten wurden drei Gruppen von Testklassen betrachtet. Für alle Testinstanzen der in einer Gruppe inkludierten Testklassen wurden die minimale Anzahl an SG-Umladeoperationen n'_{SG} und die in der Baumsuche-Heuristik verwendete untere Schranke n'_m berechnet. In der Tabelle 6.13 werden für jede dieser Gruppen die Anzahl der Instanzen, die inkludierten Testklassen sowie die mittlere Verbesserung der unteren Schranke durch den Einbezug der SS- und GX-Umladeoperationen angegeben. Die Verbesserung wurde für jede CPMP-Instanz p mittels $((n'_m(p) - n'_{SG}(p))/n'_{SG}) \cdot 100$ berechnet.

Für die Testklassen von Caserta und Voß (2009b) wurde eine sehr deutliche Verbesserung der unteren Schranke erzielt, da die Stapel initial alle gleichmäßig gefüllt und keine leeren Stapel vorhanden sind, so dass vergleichsweise viele SS- und GX-Umladeoperationen notwendig sind. Bei der Testinstanz lc1 und den Testklassen

Instanzenmenge	Anzahl Instanzen	Inkludierte Testklassen	Verbesserung der unt. Schranke in %
LC	41	lc1, LC2a, LC2b, LC3a, LC3b	8,78
CV	40	CV1 - CV4	27,40
BF	640	BF1 - BF32	2,76

Tabelle 6.13: Verbesserung der unteren Schranke durch die Berücksichtigung von SS- und GX-Umladeoperationen über 721 CPMP-Instanzen

von LC2a bis LC3b konnten auch noch beachtliche Verbesserungen von fast 9 % erreicht werden. Für die neuen Testklassen BF1-BF32 konnten dagegen im Mittel nur geringe Verbesserungen erzielt werden. Der Verbesserungseffekt für die Testklassen BF1-BF32 ist etwas stärker für Testklassen mit einer geringeren Anzahl an initial schlecht platzierten Containern sowie generell für Testklassen mit mehr Containern. Alles in allem ist durch die Berücksichtigung von SS- und GX-Umladeoperationen eine Verbesserung der unteren Schranke festzustellen.

Nach der isolierten Betrachtung der unteren Schranke soll nun im weiteren der Einfluss einzelner Komponenten der Heuristik auf die Lösungsqualität und die Rechenzeit der Baumsuche-Heuristik untersucht werden.

Dazu wurde die vorgestellte Baumsuche-Heuristik für das CPMP (im weiteren als $V0$ bezeichnet) variiert und mit vier Varianten $V1$-$V4$ verglichen. In jeder der Varianten ist eine Komponente der Baumsuche-Heuristik geändert:

- $V1$: In dieser Variante wird nur die einfache untere Schranke n'_{SG} verwendet anstelle der unteren Schranke n'_m.
- $V2$: In dieser Variante werden nur einelementige Umladeoperationsketten verwendet. Es wird also nach jeder einzelnen Umladeoperation verzweigt.
- $V3$: In dieser Variante wird die Liste der Umladeoperationsketten Cm nicht sortiert, nachdem sie vollständig aufgebaut wurde.
- $V4$: In dieser Variante werden keine Filterregeln angewendet, um die geeignetste nächste Umladeoperation für eine Umladeoperationskette auszuwählen. Es wird stattdessen immer die erste mögliche Umladeoperation ausgewählt.

Variante der Baumsuche-heuristik	Gelöste Instanzen in %	Durchschn. Anzahl Umladeoperationen in gefundenen Lösungen gegenüber $V0$ in %	Durchsch. Rechenzeit für gefundene Lösungen gegenüber $V0$ in %
$V0$	100,00	100,00	100,00
$V1$, $p_{ub} = 1,75$	81,42	102,01	205,86
$V1$, $p_{ub} = 5,0$	100,00	102,23	188,78
$V2$	68,14	100,88	151,81
$V3$	76,11	106,30	163,20
$V4$	77,88	113,43	214,13

Tabelle 6.14: Berechnungsergebnisse mit unterschiedlichen Varianten der Baumsuche-Heuristik über 113 CPMP-Instanzen

Die normale Baumsuche-Heuristik $V0$ sowie die beschriebenen Varianten $V1$ bis $V4$ wurden auf eine spezielle Menge von Testinstanzen, die sich aus der Testinstanz lc1 und Testinstanzen aus den Testklassen LC2a, LC2b, LC3a, LC3b und CV1 bis CV4 zusammensetzt, angewendet. Zusätzlich wurde jeweils eine Instanz aus den 32 neuen Testklassen berücksichtigt, so dass pro Variante insgesamt 113 CPMP-Instanzen berechnet wurden. Die Parameter wurden wie bereits in den vorherigen Abschnitten beschrieben gewählt, d.h. die LC- und CV-Instanzen wurden mit einem Zeitlimit von 20 Sekunden berechnet, während für die neuen Testinstanzen ein Zeitlimit von 60 Sekunden gesetzt wurde. Schließlich wurde die Variante $V1$ mit zwei unterschiedlichen Parameterwerten für p_{ub} berechnet ($p_{ub} = 1,75$ und $p_{ub} = 5,0$, siehe Seite 91), da für den niedrigeren Parameterwert $p_{ub} = 1,75$ in Kombination mit der wesentlich schwächeren unteren Schranke n'_{SG} sehr viele Instanzen ungelöst blieben.

Die Ergebnisse der Berechnungen sind Tabelle 6.14 zu entnehmen. Für jede Variante ist zunächst das Verhältnis der gelösten Instanzen zu den insgesamt zu berechnenden 113 Instanzen angegeben. Die in den Spalten zwei und drei angegebenen Werte beziehen sich nur auf die Instanzen, für die eine Lösung von der jeweiligen Variante gefunden wurde. In der dritten Spalte wird die durchschnittliche relative Anzahl an zusätzlichen Umladeoperationen (im Vergleich zur vollständigen Baumsuche-Heuristik $V0$) angegeben. In der vierten Spalte ist die durchschnittliche Rechenzeit zu sehen, relativ zur durchschnittlichen Rechenzeit der vollständigen Baumsuche-Heuristik $V0$.

Die Ergebnisse lassen sich wie folgt zusammenfassen:

- Nur mit den Varianten $V0$ und $V1$ (mit dem Parameterwert $P_{ub} = 5,0$) konnten alle Testinstanzen gelöst werden. Für die Varianten $V2$ bis $V4$ blieben zwischen 22 % und 32 % der Testinstanzen ungelöst.

- Berücksichtigt man nur die erfolgreich gelösten Testinstanzen, zeigt sich für alle Varianten $V1$ bis $V4$ eine Verschlechterung der Lösungsqualität, d.h. die Lösungen enthalten im Mittel mehr Umladeoperationen als bei $V0$. Die Varianten $V3$ (keine Sortierung der Umladeoperationsketten) und $V4$ (keine Filterregeln) benötigen im Mittel 6 % bzw. 13 % mehr Umladeoperationen als $V0$.

- Ebenfalls nur unter Berücksichtigung der gelösten Instanzen lässt sich für alle Varianten $V1$ bis $V4$ ein drastischer Anstieg der Rechenzeit beobachten. Der Anstieg liegt dabei zwischen 50 % und 114 %. Es zeigt sich im Besonderen, dass die Berücksichtigung von SS- und GX-Umladeoperationen bei der Berechnung der unteren Schranke die Rechenzeit wesentlich verringert (vgl. Ergebnisse für Variante V1).

Zusammenfassend lassen die Untersuchungsergebnisse den Schluss zu, dass die beschriebenen Komponenten jeweils einen wichtigen Beitrag zur Effektivität und Effizienz der Baumsuche-Heuristik leisten.

6.6 Zusammenfassung

In diesem Kapitel wurde das Container Pre-Marshalling Problem (CPMP), ein NP-schweres kombinatorisches Optimierungsproblem, das im Zwischenlager von Containerterminals von Bedeutung ist, informell und formal beschrieben. Danach wurden die bisher veröffentlichten Ansätze zur Lösung des CPMP vorgestellt.

Im Anschluss wurde eine Baumsuche-Heuristik für das CPMP beschrieben, die beim Verzweigen sowie bei der Berechnung der unteren Schranke auf eine Klassifikation der Umladeoperationen zurückgreift. Die Leistungsfähigkeit der Heuristik wurde durch den Vergleich mit den bisher veröffentlichten Lösungsverfahren nach-

gewiesen. Daneben zeigte die Evaluierung mit zufallsbasierten Testklassen, dass die Baumsuche-Heuristik auch für große Testinstanzen gute Ergebnisse liefern kann. Durch die Untersuchung einzelner Komponenten der Baumsuche-Heuristik, insbesondere der unteren Schranke, konnte die Relevanz der einzelnen Komponenten für den Erfolg der Baumsuche-Heuristik gezeigt werden.

7 Heuristische Baumsuche für das Container Relocation Problem (CRP)

In diesem Kapitel wird das Container Relocation Problem (CRP) erläutert und formal definiert. Darauf folgt eine Übersicht über bisher veröffentlichte Lösungsansätze für das CRP. Dann wird eine Baumsuche-Heuristik zur Lösung des CRP beschrieben und in einem anschließenden Evaluierungsabschnitt eingehend auf ihre Leistungsfähigkeit untersucht.

Die Ergebnisse dieses Kapitels wurden von Forster und Bortfeldt (2012b) veröffentlicht.

7.1 Problembeschreibung

Die Ausgangssituation des Container Relocation Problems (CRP) ähnelt der des CPMP. Wieder soll ein Schiff mit Containern beladen werden. Die Container befinden sich alle in einer Bucht des Zwischenlagers. Das Beladen des Schiffes und damit die Entnahme der Container aus der Bucht muss in einer gewissen Reihenfolge geschehen, die in den Containergruppen bzw. den Prioritäten der Container gegeben ist.

Während beim CPMP davon ausgegangen wird, dass das zu beladende Schiff noch nicht im Containerterminal eingetroffen ist, ist das CRP auf die Situation zugeschnitten, dass das Schiff bereits angelegt hat und unmittelbar zu beladen ist. Dadurch ergibt sich eine neue Option für die Stapelkräne, nämlich einen Container,

der in einem Stapel oben liegt und aktuell zur Containergruppe mit der höchsten Priorität gehört, aus der Bucht zu entnehmen. Eine solche Kranoperation wird im Folgenden als Entnahmeoperation bezeichnet. Der Container wird vom Stapelkran dann auf ein internes Transportmittel (in der Regel einen Terminaltruck) platziert, der den Container zur Containerbrücke befördert. Die Containerbrücke lädt den Container schließlich auf das Schiff.

Beim CRP wird nun nach einer möglichst kurzen Sequenz von Kranoperationen (Entnahme- und Umladeoperationen) gesucht, die eine gegebene Bucht, bestehend aus Stapeln von Containern, leert. Wie beim CPMP wird beim CRP davon ausgegangen, dass eine Kranoperation unabhängig von ihrem Typ und den involvierten Stapeln immer gleich lange dauert, und so wird anstatt der eigentlichen Kranzeit die Anzahl der durchgeführten Kranoperationen minimiert. Das CRP beschränkt sich wie das CPMP auf eine einzige Bucht im Zwischenlager.

7.2 Problemformalisierung

Das im vorherigen Abschnitt beschriebene CRP wird nun formal definiert.

Eine Bucht im Zwischenlager eines Containerterminals lässt sich als ein Tupel (S,H) beschreiben. Dabei bezeichnet S ($S \in \mathbb{N}, S \geq 1$) die Anzahl der Stapel, aus der die Bucht besteht. H ($H \in \mathbb{N}, H \geq 1$) definiert die maximale Stapelhöhe, legt also fest, wie viele Container maximal in einem Stapel untergebracht werden können.

Der (Füll-)Zustand (im Folgenden auch: das Layout) einer Bucht kann durch eine Matrix L mit H Zeilen und S Spalten repräsentiert werden. Eine Zelle der Matrix wird durch ein Tupel (h,s) addressiert ($1 \leq h \leq H$, $1 \leq s \leq S$). Mit G ($G \in \mathbb{N}, G \geq 1$) wird die höchste vorkommende Containergruppe in einer Bucht bezeichnet. Jedem der Container in einer Bucht ist eine Containergruppe g ($g \in \mathbb{N}, 1 \leq g \leq G$) zugeordnet. Es gilt:

$$L(h,s) = \begin{cases} g, & \text{wenn sich im Stapel } s \text{ in der Ebene } h \\ & \text{ein Container der Containergruppe } g \text{ befindet,} \\ 0, & \text{wenn sich im Stapel } s \text{ in der Ebene } h \\ & \text{kein Container befindet.} \end{cases}$$

Ein Container kann nur entweder auf dem Boden der Bucht oder auf einem anderen Container stehen. Entsprechend muss, falls $L(h, s) = 0$ gilt, auch $L(h', s) = 0$ für alle $h' = h + 1, \ldots, H$ gelten.

Eine Umladeoperation kann mit einem Tupel (d, r) von verschiedenen Stapeln $(1 \leq d, r \leq S, d \neq r)$ beschrieben werden. Der erste Stapel (d) wird als Sender bezeichnet, der zweite Stapel (r) ist der Empfänger. Eine Umladeoperation bewegt den Container, der sich an der höchsten Position im Sender befindet, auf den niedrigsten freien Platz des Empfängers. Letzterer wird auch als Empfängerplatz bezeichnet. Eine Umladeoperation ist genau dann zulässig für einen durch L repräsentierten Zustand einer Bucht, wenn folgende zwei Bedingungen erfüllt sind:

1. Der Sender ist nicht leer: $L(1, d) > 0$.
2. Der Empfänger ist nicht voll: $L(H, r) = 0$.

Durch die Anwendung einer Umladeoperation (d, r) auf eine Matrix L entsteht eine neue Matrix L'. Mit $top_L(s)$ soll die Reihe bezeichnet werden, in der sich der oberste Container im Stapel s in der Matrix L befindet. Handelt es sich bei s um einen leeren Stapel, ist $top_L(s) = 0$. Für die Matrix L' gilt:

- Die Reihe $top_L(d)$ im Senderstapel d, in der sich der umzuladende Container in L befunden hat, ist in L' leer: $L'(top_L(d), d) = 0$.
- In L' befindet sich der umgeladene Container eine Reihe über der Reihe $top_L(r)$: $L'(top_L(r) + 1, r) = L(top_L(d), d)$.

An allen anderen Plätzen stimmen die Elemente der Matrizen L' und L überein, d.h. $L'(h, s) = L(h, s)$ für $h = 1, \ldots, H$, $s = 1, \ldots, S$, $(h, s) \neq (top_L(d), d)$ und $(h, s) \neq (top_L(r) + 1, r)$.

Neben Umladeoperationen gibt es beim CRP mit den Entnahmeoperationen einen zweiten Typ von Kranoperationen. Eine Entnahmeoperation entfernt einen Container aus der Bucht. Eine Entnahmeoperation von einem Stapel s wird mit $\bar{s}$ bezeichnet. Sie ist für den Zustand einer Bucht L und einen Stapel s genau dann zulässig, wenn folgende Bedingungen erfüllt sind:

- Der Stapel s ist nicht leer: $L(1, s) > 0$

- Es gibt keinen Container in L, der eine höhere Priorität bzw. einen niedrigeren Gruppenindex hat als der oberste Container im Stapel s. Es muss also gelten: $\forall s', 1 \leq s' \leq S : \forall h', 1 \leq h' \leq H : L(h', s') \neq 0 \Rightarrow L(h', s') \geq L(top_L(s), s)$

Die Anwendung einer Entnahmeoperation ($\bar{s}$) auf L führt zur Nachfolgematrix L'. In der Matrix L' ist die Reihe $top_L(d)$ im Senderstapel d, in der sich der umzuladende Container in L befunden hat, nicht belegt: $L'(top_L(d), d) = 0$. Ansonsten sind die Werte von L' identisch mit den Werten von L.

Beim CRP ist der gewünschte Zielzustand erreicht, wenn alle Container mit (gültigen) Entnahmeoperationen aus der Bucht entfernt worden sind. Dann sind alle Stapel der Bucht leer, und für die entsprechende Matrix L gilt:

$$\forall h, 1 \leq h \leq H : \forall s, 1 \leq s \leq S : L(h, s) = 0 \tag{7.1}$$

Eine Sequenz von p Kranoperationen $s = (m_1, m_2, \ldots, m_p)$ ($p \in \mathbb{N}$, $p \geq 1$) ist zulässig für L, wenn die Kranoperation m_1 für L zulässig, und jede Kranoperation m_{i+1} für die durch Anwendung von m_i entstehende Matrix zulässig ist ($i = 1, \ldots, p-1$).

Das CRP lässt sich schließlich folgendermaßen formulieren: Für einen durch eine Matrix L repräsentierten Zustand einer Bucht soll eine Sequenz s, bestehend aus p Kranoperationen $m_1, \ldots, m_p$, gefunden werden, die folgende drei Anforderungen erfüllt:

1. Die Sequenz s ist zulässig für die Matrix L.
2. Die durch die Anwendung der letzten Kranoperation m_p entstehende Matrix L_p ist leer, erfüllt also die Bedingung 7.1.
3. Für jede Sequenz von Kranoperationen s', die die Anforderungen 1. und 2. erfüllt, gilt: $p' \geq p$, wobei p' die Anzahl der Kranoperationen in s' bezeichnet.

Es können CRP-Instanzen konstruiert werden, die prinzipiell nicht lösbar sind. So ist z.B. eine Bucht, in der alle Stapel bis zur maximalen Stapelhöhe mit Containern gefüllt sind und bei der die höchsten Container in den Stapeln jeweils schlecht platziert sind, nicht lösbar, da kein freier Platz vorhanden ist, um die schlecht platzier-

ten Container umzuladen. Im Folgenden wird immer von lösbaren Probleminstanzen ausgegangen.

Caserta et al. (2012) konnten beweisen, dass es sich beim CRP mit endlicher maximaler Stapelhöhe $H < \infty$ und endlicher Stapelanzahl $S < N < \infty$, wobei N die Anzahl an Containern in der Bucht bezeichnet, um ein NP-schweres Optimierungsproblem handelt. Der Beweis erfolgt durch Reduktion der Entscheidungsproblemvariante des CRP auf die Entscheidungsproblemvariante des Mutual Exclusion Scheduling Problems, für das Jansen (1998) die NP-Vollständigkeit gezeigt hat. Ferner konnten Caserta et al. (2012) zeigen, dass sich alle Instanzen des CRP, bei denen die Bucht aus mindestens so vielen Stapeln besteht, wie sich Container in der Bucht befinden, mit polynomieller Laufzeit optimal lösen lassen. Dies ist unmittelbar einsichtig, da dann für jeden Container, der umgeladen werden muss, ein leerer Stapel vorhanden ist.

7.3 Bisherige Lösungsansätze

7.3.1 Kim und Hong (2006)

Kim und Hong (2006) präsentieren einen Branch-and-Bound-Algorithmus für das CRP. Als untere Schranke nutzt der Algorithmus die Anzahl der aktuell in der Bucht schlecht platzierten Container. Bei der Verzweigung verfolgt der Algorithmus eine Strategie der Tiefensuche. Von den Knoten des Suchbaums gleicher Tiefe wird immer derjenige mit niedrigster unterer Schranke bevorzugt. Um die Komplexität des Verfahrens zu reduzieren, werden nur Umladeoperationen von Stapeln betrachtet, in denen sich ein Container mit aktuell höchster Priorität (unterhalb des höchsten Containers) befindet. Nach der Evaluierung dieses Ansatzes mit einer Reihe von Testinstanzen kommen die Autoren zu dem Schluss, dass sich der (exakte) Branch-and-Bound-Algorithmus nur für relativ kleine Instanzen nutzen lässt.

Die Autoren beschreiben dementsprechend zusätzlich eine Greedy-Heuristik. Dazu definieren sie zunächst eine Schätzfunktion, mit der die Anzahl an noch durchzuführenden Kranoperationen für einen Buchtzustand L abgeschätzt werden soll. Diese berücksichtigt die Anzahl an schlecht platzierten Containern und die Anzahl

an freien Plätzen in jedem Stapel. Beginnend mit der Ausgangsbucht bestimmt die Heuristik alle gültigen Kranoperationen und die entsprechenden Folgebuchten, die sich durch Anwendung der Kranoperationen jeweils ergeben. Dann wird mit Hilfe der Schätzfunktion für jede Folgebucht bestimmt, wie viele Kranoperationen vermutlich noch durchgeführt werden müssen. Schließlich wird diejenige Kranoperation gewählt, die zu einer Folgebucht mit niedrigstem Schätzfunktionswert unter allen Folgebuchten führt. Diese Schritte werden solange wiederholt, bis die Bucht leer ist.

Die Autoren evaluieren beide vorgeschlagenen Verfahren mit verschiedenen zufällig erzeugten Testinstanzen. Dabei zeigt sich, dass das Branch-and-Bound-Verfahren für Buchten, bei denen das Produkt aus der Stapelanzahl und der maximalen Stapelhöhe größer als 20 ist, unpraktikabel wird, also z.B. für eine Bucht aus 5 Stapeln mit Stapelhöhe 5. Das Greedy-Verfahren dagegen kann auch größere Probleminstanzen mit Laufzeiten im Sekundenbereich lösen.

7.3.2 Caserta et al. (2009b)

Caserta et al. (2009b) untersuchen einen auf dynamischer Programmierung (DP) basierenden Lösungsansatz für das CRP. Um das DP-Verfahren auch auf größere Probleminstanzen anwenden zu können, schlagen die Autoren vor, die in einer Bucht gültigen Kranoperationen auf einen zweidimensionalen Korridor zu beschränken, der durch die Algorithmusparameter δ und λ definiert wird. Angenommen, der nächste aus der Bucht zu entfernende Container befindet sich im Stapel s auf Höhe h, dann müssen alle Container die sich in s in einem Platz über h befinden notwendigerweise auf andere Stapel umgeladen werden. Anstatt nun alle (nicht-vollen) Stapel als potenzielle Empfängerstapel für diese Container zu betrachten, werden bei dem beschriebenen Verfahren nur Stapel berücksichtigt, die nicht weiter als δ Stapel vom Senderstapel entfernt sind und die momentan nicht mehr als λ Container beinhalten. Auf diese Weise wird der Lösungsraum eingeschränkt. Die Autoren evaluieren ihren Ansatz mit zufallsbasierten Testinstanzen und vergleichen die Ergebnisse mit denen von Kim und Hong (2006). Es zeigt sich, dass der Korridor-Ansatz wesentlich bessere Ergebnisse liefert, und das sowohl für kleine und mittlere als auch für große Testinstanzen.

7.3.3 Caserta et al. (2009a)

Caserta et al. (2009a) formulieren zunächst eine Greedy-Heuristik für das CRP (Min-Max-Heuristik genannt), die auf folgender Idee beruht: Angenommen, keine Containerpriorität ist mehr als einem Container zugewiesen. Dann gibt es - so lange die Bucht nicht leer ist - zu jedem Zeitpunkt genau einen Container c_0 mit höchster Priorität. Befindet sich dieser Container auf dem obersten Platz in einem Stapel, wird er aus der Bucht entnommen. Ansonsten gibt es m Container c_j $(1 \leq j \leq m)$ über c_0, die umgeladen werden müssen, beginnend mit dem obersten Container im Stapel. Für einen Container c_j der Priorität g, der umgeladen werden soll, wird der Empfängerstapel durch Anwendung folgender Entscheidungsregeln bestimmt. Mit $min(s)$ sei die höchste Priorität (niedrigste Containergruppe) im Stapel s bezeichnet. Wenn es mindestens einen Stapel s gibt, der nicht voll ist und für den gilt, dass $g < min(s)$, dann wird derjenige Stapel unter diesen Stapeln ausgewählt, für den $min(s)$ den niedrigsten Wert annimmt. Es werden also Empfängerstapel bevorzugt, in denen alle Container niedrigere Priorität haben als der umzuladende Container. Gibt es keinen solchen Stapel, wird ein leerer Stapel als Empfängerstapel ausgewählt. Existiert kein leerer Stapel, wird derjenige Stapel s gewählt, der nicht voll ist und für den $min(s)$ maximal ist. Dies wird solange wiederholt, bis alle m Container umgeladen worden sind und c_0 entfernt werden kann. Das Verfahren wird dann erneut auf die aktuelle Bucht angewendet, bis die Bucht schließlich leer ist.

Die Autoren betten diese Greedy-Heuristik nun in ein randomisiertes Lösungsverfahren ein. Das Lösungsverfahren erzeugt die Lösung schrittweise. In jedem Schritt wird zunächst geprüft, ob eine Entnahmeoperation möglich ist. Ist dies der Fall, wird diese Entnahmeoperation durchgeführt. Ist keine Entnahmeoperation möglich, wird zunächst der Stapel identifiziert, der den Container mit aktuell höchster Priorität beinhaltet. Dann werden alle möglichen Umladeoperationen des obersten Containers in diesem Stapel generiert, also alle Umladeoperationen mit dem besagten Stapel als Senderstapel und allen anderen, nicht vollen Stapeln als Empfängerstapel. Für jede der Umladeoperationen wird die Nachfolgematrix, die sich aus der Anwendung der jeweiligen Umladeoperation ergibt, berechnet und die hierdurch gegebene CRP-Instanz wird dann mit Hilfe der beschriebenen Min-Max-Heuristik testweise gelöst. Schließlich wird die tatsächlich in diesem Verfahrensschritt durchzuführende Umladeoperation zufällig aus allen getesteten Umladeoperationen ausgewählt. Jede getes-

tete Umladeoperation besitzt dabei eine Auswahlwahrscheinlichkeit, die umgekehrt proportional zur Länge der dazugehörigen Greedy-Lösung ist. Das ganze randomisierte Verfahren wird wiederum mehrfach angewendet, so lange bis ein Zeitlimit erreicht wurde. Es werden zur Laufzeit des Algorithmus also in der Regel mehrere Lösungen generiert, und der Algorithmus liefert bei Erreichen der Zeitschranke die Lösung mit den wenigsten Kranoperationen zurück.

Die Autoren zeigen, wie sich ein Buchtzustand als Matrix von Binärwerten repräsentieren lässt und argumentieren, dass ihr Verfahren sehr stark von dieser Repräsentierungsform profitiert, da mit dieser die Greedy-Lösung besonders effizient berechnet werden kann.

7.3.4 Caserta und Voß (2009a,c)

Caserta und Voß (2009a,c) beschreiben in zwei weiteren Artikeln Varianten des Lösungsansatzes aus Caserta et al. (2009a). Die Lösung wird auch hier wieder mit Hilfe eines randomisierten Verfahrens schrittweise aufgebaut. Wenn eine Umladeoperation notwendig ist, wird diese aus einer Teilmenge aller möglichen Umladeoperationen zufällig gewählt. Die Zusammenstellung dieser Teilmenge erfolgt bereits zufallsbasiert, wobei die Wahrscheinlichkeit für die Wahl einer Umladeoperation von einem Attraktivitätswert abhängt, der für jeden Empfängerstapel einzeln berechnet wird. Die Auswahl der Umladeoperation für den aktuellen Verfahrensschritt aus dieser Teilmenge erfolgt erneut zufallsbasiert. Die Wahrscheinlichkeit, dass eine bestimmte Umladeoperation gewählt wird, ist dabei umgekehrt proportional zur Anzahl der schlecht platzierten Container nach deren Anwendung.

Zur Evaluierung ihres Verfahrens lösen die Autoren verschiedene Testinstanzen, jeweils mit unterschiedlichen Größen für die Teilmengen, aus denen die Umladeoperationen gezogen werden. Für jede Testklasse mit je mehreren ähnlichen Testinstanzen geben die Autoren dann die durchschnittliche Anzahl an Kranoperationen in der jeweils besten Lösung an sowie die verwendeten Parameterwerte für das Verfahren.

7.3.5 Caserta et al. (2012)

Auch der Artikel von Caserta et al. (2012) beschäftigt sich mit dem CRP. Die Autoren präsentieren zunächst eine Formulierung des CRP als binäres lineares Optimierungsproblem (BRP-I). Das Modell benötigt für eine Bucht aus W Stapeln, maximaler Stapelhöhe H und N Containern $2 \cdot W \cdot H \cdot N \cdot T + (W \cdot H)^2 \cdot N \cdot T + N \cdot T$ binäre Entscheidungsvariablen. Die Variable T repräsentiert dabei die Anzahl an diskreten Zeiteinheiten, die das Modell berücksichtigen soll. In jeder Zeiteinheit wird ein Container umgeladen, so dass T mit einer oberen Schranke für die Anzahl benötigter Kranoperationen in der zu lösenden Testinstanz initialisiert werden kann.

Die Autoren geben eine Formel zur Berechnung einer oberen Schranke der benötigten Kranoperation für das CRP an und weisen darauf hin, dass auch jedes Ergebnis einer Heuristik für das CRP als obere Schranke für das Modell dienen kann. Die Autoren kommen dennoch zu dem Schluss, dass ihr Modell in dieser Form für in der Praxis auftretende Problemdimensionen nicht schnell genug gelöst werden kann.

Um ein schneller lösbares Modell zu entwickeln, betrachten die Autoren eine eingeschränkte Variante des CRP, bei der Umladeoperationen immer nur von einem Stapel erfolgen können, in dem sich ein Container mit aktuell höchster Priorität befindet. Die Autoren zeigen, dass auch diese Variante des CRP NP-schwer ist. Sie beschreiben ein binäres lineares Optimierungsmodell (BRP-II) für das CRP. Der wesentliche Unterschied zu BRP-I ist nun, dass eine Zeiteinheit nicht mehr eine einzige Umladeoperation betrifft, sondern alle Umladeoperationen zum Freilegen des Containers mit der aktuell höchsten Priorität sowie die Entnahmeoperation für diesen Container. Es werden also in der Regel pro Zeiteinheit mehrere Container bewegt. Die Anzahl an benötigen binären Variablen im Modell ist mit $(W \cdot H \cdot N)^2 + 2 \cdot W \cdot H \cdot N^2$ gegenüber BRP-I verringert. Somit lassen sich etwas größere Instanzen des CRP lösen. Ein wesentlicher Nachteil des BRP-II-Modells ist - neben der eingangs erwähnten Einschränkung der Problemstellung - allerdings, dass nur die Anzahl an Kranoperationen in einer optimalen Kranoperationssequenz ermittelt werden kann, nicht aber die optimale Sequenz als solche.

Die Evaluierung auf einer Linux Workstation mit Pentium IV-Prozessor und 512 MB Hauptspeicher zeigt, dass bereits für Testinstanzen mit ca. 20 Containern, gleichmäßig verteilt auf 5 Stapel, Laufzeiten von knapp 45 Minuten in Kauf genommen

werden müssen, um Optimallösungen für das BRP-II zu finden. Testinstanzen mit mehr als 24 Containern, gleichmäßig verteilt auf 6 Stapeln, konnten nicht mehr innerhalb eines Tages gelöst werden. Zum Vergleich wird die in Caserta und Voß (2009c) vorgeschlagene Min-Max-Heuristik herangezogen. Diese löst alle gerechneten Testklassen in weniger als 1 Sekunde und weist nur geringe Abweichungen im einstelligen Prozentbereich zu den Optimallösungen des BRP-II auf (für die Testinstanzen, für die das BRP-II eine Lösung finden konnte). Die Modellformulierung BRP-I wird experimentell nicht weiter untersucht.

7.4 Entwurf einer Heuristik für das CRP

Im Kapitel 6 dieser Arbeit wurde gezeigt, wie ein erfolgreiches heuristisches Baumsucheverfahren zur Lösung des CPMP entwickelt werden kann. Wegen der strukturellen Ähnlichkeit der Probleme ist es ein naheliegender Ansatz, eine problemspezifische Baumsuche-Heuristik für das CRP mit der gleichen Methodik zu entwerfen: Zunächst soll ein Klassifikationsschema für die Kranoperationen im CRP entwickelt werden, auf dessen Grundlage dann sowohl eine untere Schranke für die Anzahl an Kranoperationen abgeleitet als auch die Baumsuche-Heuristik entwickelt wird.

7.4.1 Grundlegende Definitionen

Das Klassifikationsschema für die Kranoperationen beim CRP unterscheidet Entnahmeoperationen und Umladeoperationen.

Umladeoperationen werden mit Hilfe der Containerprädikate „schlecht platziert“ und „gut platziert“ weiter differenziert. Ein Container wird als „schlecht platziert“ bezeichnet, wenn er notwendigerweise mindestens einmal umgeladen werden muss, bevor die Bucht geleert werden kann. Dies trifft auf einen Container zu, wenn dieser verhindert, dass ein anderer Container aus der Bucht entfernt werden kann. D.h. der Container befindet sich im Stapel über mindestens einem anderen Container mit höherer Priorität. Formal lässt sich das Prädikat wie folgt formulieren: Es gelte $L(h,s) = g, g > 0$. Der Container in (h,s) der Gruppe g wird als „schlecht platziert“ (im Sinne des CRP) bezeichnet, wenn $h > 1$ und für mindestens ein h' gilt:

$L(h', s) < g, 1 \leq h' < h$. Ansonsten wird der Container als „gut platziert“ (im Sinne des CRP) bezeichnet.

Auf Basis dieser Prädikate lässt sich ein Klassifikationsschema für Umladeoperationen ableiten. Eine gegebene Umladeoperation m sei auf einen Buchtzustand L anwendbar. Die Umladeoperation ist vom Typ „schlecht-gut“ (kurz: SG), wenn der durch m bewegte Container unmittelbar vor Ausführung der Umladeoperation schlecht platziert war und unmittelbar nach Beendigung der Umladeoperation gut platziert ist. Entsprechend sind die Umladeoperationstypen „gut-gut“ (GG), „gut-schlecht“ (GS) und „schlecht-schlecht“ (SS) definiert.

Mit SX sollen sowohl SS- als auch SG-Umladeoperationen bezeichnet werden. Analog fasst die Klasse GX sowohl GG- als auch GS-Umladeoperationen zusammen.

Das Klassifikationsschema für die SX-Umladeoperationen wird weiter verfeinert. Für diese Umladeoperationstypen soll nun unterschieden werden, ob sich im Senderstapel der Umladeoperation ein Container mit aktuell höchster Priorität befindet oder nicht. Der erste Fall wird mit dem Suffix HP (höchste Priorität) angezeigt, der zweite mit dem Suffix AP (andere Priorität).

Jede Umladeoperation lässt sich anhand dieses Klassifikationsschemas eindeutig einer der folgenden Klassen zuordnen, abhängig vom Prädikat des umzuladenden Containers direkt vor und direkt nach der Ausführung der Umladeoperation sowie von der Zusammensetzung des Senderstapels bei SX-Umladeoperationen:

- SG-HP: „Schlecht platziert“ zu „Gut platziert“, ein Container mit höchster Priorität im Senderstapel vorhanden
- SG-AP: „Schlecht platziert“ zu „Gut platziert“, kein Container mit höchster Priorität im Senderstapel vorhanden
- SS-HP: „Schlecht platziert“ zu „Schlecht platziert“, ein Container mit höchster Priorität im Senderstapel vorhanden
- SS-AP: „Schlecht platziert“ zu „Schlecht platziert“, kein Container mit höchster Priorität im Senderstapel vorhanden
- GG: „Gut platziert“ zu „Gut platziert“

- GS: „Gut platziert“ zu „Schlecht platziert“

Als „produktive Kranoperationen“ werden Umladeoperationen der Klassen SG-HP, SG-AP, SS-HP, GG sowie Entnahmeoperationen bezeichnet. Produktive Kranoperationen sind im Allgemeinen hilfreicher für die Erreichung des Zielzustands einer leeren Bucht als die anderen Typen von Kranoperationen. Entnahmeoperationen sowie SG-Umladeoperationen sind letztendlich alternativlos, da jeder Container aus der Bucht entfernt werden muss und nur gut platzierte Container entfernt werden können. Schlecht platzierte Container müssen also notwendigerweise durch SG-Umladeoperationen umgeladen werden. Sind weder Entnahme- noch SG-Umladeoperationen möglich, sind SS-HP-Umladeoperationen den SS-AP-Umladeoperationen vorzuziehen, da durch erstere zumindest eine künftige Entnahmeoperation vorbereitet wird. Ebenso sind GG-Umladeoperationen den GS-Umladeoperationen vorzuziehen, weil durch letztere in der Zukunft zwingend eine zusätzliche SG-Umladeoperation notwendig wird, was bei ersteren nicht der Fall ist. In der Beschreibung des Baumsuche-Verfahrens in Abschnitt 7.4.4 werden produktive Kranoperationen detaillierter beleuchtet.

Ein Stapel wird als sauber bezeichnet, wenn er keinen schlecht platzierten Container enthält. Ansonsten wird er als unsauber bezeichnet. Entsprechend ist jeder Stapel entweder sauber oder unsauber.

Ein Platz (h, s) in einem Stapel s ist ein potenzieller Angebotsplatz für eine Gruppe g genau dann, wenn

- der Stapel s nicht leer ist,
- der höchste gut platzierte Container in s zur Gruppe g gehört, und
- der Platz (h, s) über dem höchsten gut platzierten Container in s liegt.

Ein sauberer Angebotsplatz für eine Gruppe g ist ein potenzieller Angebotsplatz für g in einem sauberen Stapel. Ein leerer Stapel hat per Definition H Angebotsplätze für jede Gruppe $g (g = 1, \ldots, G)$.

Tabelle 7.1 enthält eine Reihe weiterer Definitionen, auf die im Verlauf des Kapitels zurückgegriffen wird. Alle Definitionen beziehen sich auf einen gegebenen Buchtzustand L.

Variable	Definition
n	Anzahl Container in einer Bucht
n_b	Anzahl schlecht platzierter Container in einer Bucht
$n_b(s)$	Anzahl schlecht platzierter Container im Stapel s $(1 \leq s \leq S)$
$n_m(seq)$	Anzahl Kranoperationen in einer Sequenz seq von Kranoperationen
n'_m	Untere Schranke für die Anzahl an Kranoperationen zur Leerung einer Bucht
n'_{SG}	Untere Schranke für die Anzahl an SG-Umladeoperationen zur Leerung einer Bucht
$n'_{nicht-SG}$	Untere Schranke für die Anzahl an Nicht-SG Umladeoperationen (also SS, GG- und GS-Umladeoperationen) zur Leerung einer Bucht
$d(g)$	Nachfrage der Gruppe g, definiert als die Anzahl schlecht platzierter Container mit Gruppe g $(1 \leq g \leq G)$
$D(g)$	Kumulierte Nachfrage der Gruppe g, definiert durch: $d(g)+d(g+1)+\ldots+d(G)$
$s_p(g)$	Potenzielles Angebot für Gruppe g, definiert als die Anzahl potenzieller Angebotsplätze für g
$S_p(g)$	Kumuliertes potenzielles Angebot für Gruppe g, definiert durch: $s_p(g)+s_p(g+1)+\ldots+s_p(G)+H \cdot n_{s,empty}$, wobei $n_{s,empty}$ die Anzahl der leeren Stapel in einer Bucht bezeichnet
$D_e(g)$	Kumulierter Nachfrageüberschuss der Gruppe g, definiert durch: $D(g)-S_p(g)$
$s_c(g)$	Sauberes Angebot für Gruppe g, definiert als die Anzahl sauberer Angebotsplätze für Gruppe g
S_c	Sauberes Angebot in einer Bucht, definiert durch: $10^0 s_c(1)+10^1 s_c(2)+\ldots+10^{G-1}s_c(G)$

Tabelle 7.1: Variablen und Definitionen zur Erläuterung der Baumsuche-Heuristik für das CRP

7.4.2 Untere Schranke

Die Einordnung der Umladeoperationen in ein Klassifikationsschema und die anschließende Suche nach unteren Schranken für jede dieser Klassen hat beim CPMP zu einer sehr guten unteren Schranke geführt. Daher soll nun geprüft werden, ob und wie weit sich dieser Ansatz auf das CRP übertragen lässt.

Untere Schranke für die Anzahl an SX-Umladeoperationen

Ein schlecht platzierter Container verhindert, dass ein anderer Container mit höherer Priorität aus der Bucht entfernt werden kann. Da zur Lösung des CRP alle Container entfernt werden müssen, muss auch für jeden schlecht platzierten Container mindestens eine SG-Umladeoperation ausgeführt werden. Es sind also mindestens n_b SG-Umladeoperationen notwendig.

Der Ansatz zur Berechnung der unteren Schranke für die Anzahl an SS-Umladeoperationen beim CPMP lässt sich allerdings nicht entsprechend auf das CRP übertragen. Der Ansatz des CPMP nutzt die Tatsache aus, dass eine Umladeoperation eines schlecht platzierten Containers auf einen unsauberen Stapel beim CPMP immer eine SS-Umladeoperation ist. Beim CRP kann es sich in diesem Fall aber auch um eine SG-Umladeoperation handeln, wie im folgenden Beispiel ersichtlich wird (siehe Abbildung 7.1).

3	6
2	5
1	4

Abbildung 7.1: Bucht aus unsauberen Stapeln

Die Container 2, 3, 5 und 6 sind schlecht platziert, beide Stapel sind demnach unsauber. Durch die Umladeoperationen (1,2), (1,2) kann der Container 1 freigelegt werden. Obwohl die Container 3 und 2 dabei auf einen unsauberen Stapel umgeladen werden, sind sie dennoch nicht schlecht platziert. Es handelt sich im Sinne des CRP also nicht um SS-, sondern um SG-Umladeoperationen. Obwohl beide Stapel unsau-

ber sind, kann die Bucht ohne SS-Umladeoperationen geleert werden, und zwar mit der Sequenz (1,2), (1,2), $\bar{1}$, $\bar{2}$, (2,1), (2,1), $\bar{2}$, $\bar{1}$, $\bar{1}$.

Untere Schranke für die Anzahl an GX-Umladeoperationen

Die Berechnung einer unteren Schranke für die Anzahl an GX-Umladeoperationen beim CPMP beruht auf folgender Idee: Unter der Annahme, dass die zu Beginn gut platzierten Container nicht bewegt werden müssen, wird ggf. gezeigt, dass nicht genug geeignete Plätze in allen Stapeln vorhanden sind, in denen die schlecht platzierten Container gut platziert werden können. So kann man die Anzahl an zu Beginn gut platzierten Containern bestimmen, die in einer Lösung mindestens einmal bewegt werden müssen. Dieser Ansatz lässt sich allerdings nicht vom CPMP auf das CRP übertragen. Dies liegt vornehmlich daran, dass beim CRP die Anzahl an Containern in der Bucht nicht konstant bleibt, sondern nach und nach abnimmt, bis die Bucht am Ende leer ist. Somit entstehen laufend neue freie Plätze.

Es bleibt festzuhalten, dass trotz der Gemeinsamkeiten der beiden Optimierungsprobleme nur der Ansatz zur Bestimmung einer unteren Schranke für die Anzahl an SG-Umladeoperationen vom CPMP auf das CRP übertragen werden kann.

Für das CRP lässt sich allerdings trotzdem noch eine (wenn auch geringe) Verbesserung der unteren Schranke erreichen. Angenommen, eine Bucht erfüllt die folgenden drei Bedingungen:

1) Es kann momentan keine Entnahmeoperation vorgenommen werden, da alle Container mit höchster Priorität von Containern mit niedrigerer Priorität „geblockt“ sind.

2) Es gibt keine leeren Stapel in der Bucht.

3) Bezeichne g_1 die niedrigste Containergruppe (höchste Priorität) aller Container, die sich in der höchsten Position in ihrem jeweiligen Stapel befinden, und g_2 das Maximum der niedrigsten vorkommenden Containergruppen in den einzelnen Stapeln. Es gelte $g_1 > g_2$.

Dann muss, über die Anzahl n'_{SG} hinaus, noch mindestens eine weitere Umladeoperation durchgeführt werden, um die Bucht zu leeren. Da in der Bucht wegen Bedingung 1 keine Entnahmeoperation möglich ist, muss eine Umladeoperation durchgeführt werden. Eine SG-Umladeoperation ist nicht möglich, weil wegen Bedingung 2 kein leerer Stapel existiert und es wegen Bedingung 3 keinen nicht-leeren Stapel gibt, auf dem einer der aktuell zugänglichen Container nach einer Umladeoperation gut platziert sein würde. Folglich muss also eine Nicht-SG-Umladeoperation durchgeführt werden. Im Folgenden soll $n'_{nicht-SG}$ eine ganzzahlige Variable sein, die für eine Bucht den Wert 1 annimmt, wenn die Bucht die Bedingungen 1 bis 3 erfüllt. Ansonsten hat die Variable den Wert 0.

Die soeben abgeleitete Erhöhung der Anzahl notwendiger Umladeoperationen soll exemplarisch anhand der Bucht aus Abbildung 7.2 veranschaulicht werden. Hier gilt $g_1 = min\{4, 6\} = 4$, $g_2 = max\{1, 3\} = 3$ und entsprechend $g_1 > g_2$. Es muss also eine Nicht-SG Umladeoperation durchgeführt werden, im Beispiel wird man die SS-Umladeoperation (1,2) wählen, bei der der schlecht platzierte Container der Gruppe 4 von Stapel 1 auf Stapel 2 umgeladen wird. Er ist dort erneut schlecht platziert.

4	6
2	5
1	3

Abbildung 7.2: Bucht mit notwendiger SS-Umladeroperation

Bisher wurde nur von den Umladeoperationen gesprochen. Eine untere Schranke für die Anzahl an Entnahmeoperationen ist offensichtlich $n'_{ent} = n$, da in einer Lösung für das CRP jeder Container aus der Bucht entfernt werden muss und jeder Container nur genau einmal entfernt werden kann.

Fasst man die obigen Überlegungen zusammen, so ergibt sich folgende untere Schranke n'_m der Anzahl an Kranoperationen für das Leeren einer Bucht:

$$n'_m = n'_{ent} + n'_{SG} + n'_{nicht-SG} = n + n_b + n'_{nicht-SG} \tag{7.2}$$

7.4.3 Greedy-Verfahren

Die frühzeitige Bereitstellung einer ersten, bereits möglichst guten zulässigen Lösung kann ein Baumsuche-Verfahren wesentlich beschleunigen. Die Anzahl der Kranoperationen in dieser „Startlösung“ dient dann als obere Schranke, mit der sich früher und / oder im größeren Umfang Teile des Suchbaums von weiteren Untersuchungen ausschließen lassen; nämlich solche, in denen der Zielfunktionswert der Startlösung nicht erreicht werden kann.

Für die Baumsuche-Heuristik zur Lösung des CRP dient dazu ein Greedy-Verfahren (vgl. Algorithmus 7).

Das Greedy-Verfahren startet mit der Ausgangsbucht und einer leeren Lösungssequenz. In jedem Schritt des Algorithmus wird eine Kranoperation ausgewählt und der Lösungssequenz hinzugefügt, und zwar nach folgender Ordnung:

1. Entnahmeoperation
2. SG-Umladeoperation auf einen nicht-leeren Stapel
3. SG-Umladeoperation auf einen leeren Stapel
4. SS-HP-Umladeoperation

Ist z.B. eine Entnahmeoperation möglich, wird diese durchgeführt und keine Kranoperation mit niedrigerer Ordnung ins Auge gefasst. Solange die Bucht nicht leer ist, gibt es immer eine Kranoperation aus zumindest einer der vier Prioritätsgruppen: Ist keine Entnahmeoperation möglich (1.), ist die Bucht entweder leer oder alle Container mit aktuell niedrigster Containergruppe sind blockiert. Wenn die Bucht nicht leer ist, gibt es also mindestens einen Container, der einen anderen Container mit niedrigster Containergruppe blockiert und sich auf der höchsten Position in seinem Stapel befindet. Wenn keine SG-Umladeoperationen (2./3.) möglich ist, können die durchführbaren Umladeoperationen nur vom Typ GG, GS oder SS sein. Da alle blockierenden Container definitionsgemäß schlecht platziert sind, muss eine Umladeoperation eines blockierenden Containers in dieser Situation vom Typ SS sein. Nun gibt es mindestens einen Container an höchster Position in seinem Stapel, der außerdem einen Container der niedrigsten Gruppe (höchsten Priorität) blockiert.

Also gibt es, wenn keine Kranoperation der Typen 1.-3. möglich ist, mindestens eine SS-HP-Umladeoperation (4.).

Falls mehrere Entnahmeoperationen möglich sind, werden alle diese Entnahmeoperationen durchgeführt, ohne dass zwischen ihnen eine spezielle Reihenfolge eingehalten werden muss. Für die Auswahl einer Umladeoperation aus mehreren möglichen Umladeoperationen des gleichen Typs werden jedoch verschiedene heuristische Regeln angewendet. Wenn mehr als eine SG-Umladeoperation auf einen nicht-leeren Stapel möglich ist, wird diejenige Umladeoperation gewählt, bei der die Differenz der Containergruppen zwischen dem bewegten Container und dem obersten Container auf dem Empfängerstapel minimal ist. Die Idee hinter dieser Regel ist, möglichst Stapel zu erzeugen, bei denen aufeinanderliegende Container jeweils eine möglichst geringe Containergruppendifferenz aufweisen. Dadurch sollen in der Zukunft möglichst viele weitere SG-Umladeoperationen ermöglicht werden. Können z.B. sowohl ein Container mit Priorität 10 als auch ein Container mit Priorität 15 auf einen Stapel mit einem Container der Priorität 20 an oberster Stelle umgeladen werden, ist es besser, zuerst den Container mit Priorität 15 umzuladen, weil dann im nächsten Schritt der Container 10 mittels SG-Umladeoperation auf den Container mit der Priorität 15 geladen werden kann. Wählt man zuerst den Container mit Priorität 10, entfällt diese Möglichkeit für den Container mit Priorität 15. Ähnlich zu der Regel für SG-Umladeoperationen auf nicht-leere Stapel werden bei Umladeoperationen auf leere Stapel Container mit möglichst niedriger Containerpriorität (d.h. hoher Gruppennummer) bevorzugt. Ist dagegen mehr als eine SS-HP-Umladeoperation möglich, wird als Empfängerstapel derjenige Stapel bevorzugt, dessen minimale Containerpriorität maximal ist in Bezug auf alle möglichen Empfängerstapel. Gibt es mehrere Senderstapel wird derjenige Stapel bevorzugt, dessen oberster Container die höchste Containergruppe aufweist. Durch diese Regeln soll verhindert werden, dass der bewegte Container sehr früh erneut bewegt werden muss, da es sich in diesem Fall häufig erneut um eine SS-Umladeoperation handelt. Sobald eine Umladeoperation gewählt wurde, wird diese an die Lösungssequenz angehängt und auf die Bucht angewendet. Die Baumsuche-Heuristik endet, sobald ein gegebenes Zeitlimit erreicht wird. Letzteres stellt sicher, dass der Algorithmus auch für unlösbare Testinstanzen terminiert.

Das hier beschriebene Greedy-Verfahren kann als eine Generalisierung der Min-Max-Heuristik von Caserta et al. (2009a) verstanden werden, bei denen die betrachteten SG-Umladeoperationen nicht nur auf Stapel beschränkt sind, in denen sich ein Container der aktuell niedrigsten Containergruppe befindet.

Die Lösung des Greedy-Verfahrens dient im Folgenden ausschließlich dazu, eine bereits recht gute erste Lösung zu finden, deren Länge eine obere Schranke für die Baumsuche-Heuristik darstellt. Wie eingangs erwähnt, soll damit sichergestellt werden, dass der in der Baumsuche durchsuchte Lösungsraum von Beginn an wirksam eingeschränkt wird.

7.4.4 Algorithmus der Baumsuche und Verzweigungsansatz

Im vorliegenden Abschnitt werden der (gesamte) Algorithmus der Baumsuche und der Verzweigungsansatz beschrieben. Bei der Definition des Verzweigungsansatzes werden zwei wesentliche Maßnahmen zur Beschränkung der Suchbaumgröße eingesetzt: Verzweigung nach Kranoperationsketten anstatt nach einzelnen Kranoperationen sowie Beschränkung der maximalen Anzahl an Nachfolgeknoten im Suchbaum. Im Folgenden wird zuerst der Verzweigungsansatz detailliert dargestellt, bevor der Grobalgorithmus der Baumsuche spezifiziert wird.

Kranoperationsketten

In der Baumsuche-Heuristik stellen die Knoten des Baums Füllzustände oder Layouts einer Bucht des Zwischenlagers dar, repräsentiert als zweidimensionale Matrizen. Der Wurzelknoten entspricht dem Ausgangszustand der Bucht und jedes Blatt entspricht einer leeren Bucht. Ein direkter Nachfolger L' eines Knotens L wird mittels einer Kranoperationskette erzeugt, also einer Sequenz von Kranoperationen, die für L zulässig ist. Vollständige und unvollständige Lösungen für eine CRP-Instanz werden wiederum durch Sequenzen von Kranoperationsketten repräsentiert. Die Anzahl an Kranoperationen in einer Sequenz s von Kranoperationsketten wird mit $n_m(s)$ bezeichnet.

Algorithmus 7 Bestimmung einer Greedy-Lösung für das CRP: findInitialSolution

{Eingabe: initiale Matrix L}
{Ausgabe: Liste von Kranoperationen s}

{Initialisiere Lösung als leere Liste von Kranoperationen}
1: $s \leftarrow \emptyset$

2: **while** L ist keine leere Matrix und Zeitlimit ist nicht abgelaufen **do**
3: $Rm \leftarrow$ Menge aller gültigen Entnahmeoperationen für L
4: **if** $Rm \neq \emptyset$ **then**
{Führe alle Entnahmeoperationen aus}
5: **for all** $rm \in Rm$ **do**
6: $s \leftarrow s \cup \{rm\}$; $L \leftarrow L \oplus rm$
7: **end for**
8: **else**
9: $Sg \leftarrow$ Menge aller SG-Umladeoperationen auf nicht-leeren Stapel in L
10: **if** $Sg \neq \emptyset$ **then**
11: Bestimme $sg_{best} \in Sg$, so dass die Differenz der Containergruppen zwischen dem umzuladenden Container und dem obersten Container im Empfängerstapel minimal ist
12: $s \leftarrow s \cup \{sg_{best}\}$; $L \leftarrow L \oplus sg_{best}$
13: **else**
14: $Sge \leftarrow$ Menge aller SG-Umladeoperationen auf leeren Stapel in L
15: **if** $Sge \neq \emptyset$ **then**
16: Bestimme $sge_{best} \in Sge$, so dass die Containergruppe des umzuladenden Containers maximal ist
17: $s \leftarrow s \cup \{sge_{best}\}$; $L \leftarrow L \oplus sge_{best}$
18: **else**
19: $Sshp \leftarrow$ Menge aller SS-Umladeoperationen, bei denen sich mindestens ein Container der aktuell höchsten Containerpriorität im Senderstapel befindet.
20: Bestimme $sshp_{best} \in Sshp$, bei der der Empfängerstapel der Stapel mit maximaler minimaler Containerpriorität unter allen möglichen Empfängerstapeln ist und bei der der Senderstapel den Container mit maximaler Containerpriorität an oberster Stelle in $Sshp$ enthält
21: $s \leftarrow s \cup \{sshp_{best}\}$; $L \leftarrow L \oplus sshp_{best}$
22: **end if**
23: **end if**
24: **end if**
25: **end while**
26: **end.**

Kranoperationsketten werden mittels der rekursiven Prozedur *determineCompoundMoves* (siehe Algorithmus 8) jeweils aus einzelnen Kranoperationen aufgebaut. Beim Aufbau der Kranoperationsketten beschränkt sich die Prozedur auf die so genannten produktiven Kranoperationen, die durch die Prozedur *determineProductiveMoves* bestimmt und weiter unten eingehend erläutert werden. Ein erwähnenswerter Unterschied zu anderen, bisher veröffentlichten Lösungsverfahren für das CRP ist es, dass die Baumsuche-Heuristik beim Aufbau von Kranoperationsketten auch Umladeoperationen von Stapeln aus berücksichtigt, in denen sich kein Container mit aktuell höchster Priorität befindet. Die bisherigen Verfahren beschränken sich auf Stapel, die Container mit aktuell höchster Priorität enthalten.

Die Prozedur *determineCompoundMoves* verarbeitet die aktuelle Matrix L und die aktuelle Kranoperationskette $cmWork$ als Eingabeparameter. Der zusätzliche Eingabeparameter $cmStopValue$ wird verwendet, um die Länge der Kranoperationskette dynamisch zu beschränken. Die Kranoperationskette wird nämlich nur dann verlängert, wenn $cmStopValue$ kleiner als der Wert des globalen Algorithmusparameters $cmStopThreshold$ ist. Für jeden rekursiven Aufruf der Prozedur wird der neue $cmStopValue$-Wert durch die Multiplikation des aktuellen $cmStopValue$-Wertes mit der Anzahl an produktiven Kranoperationen in der aktuellen Matrix bestimmt. Dadurch werden längere Kranoperationsketten gebildet, wenn es nur wenige produktive Kranoperationen gibt und kürzere, wenn es viele produktive Kranoperationen gibt.

Der Eingabeparameter $maxMoves$ gibt die maximale Anzahl an zusätzlichen Kranoperationen an, die noch angewendet werden dürfen, um die aktuelle Matrix zu leeren ohne dass ingesamt mehr Kranoperationen benötigt werden als in der bisher besten bekannten Lösung.

Wenn *determineCompoundMoves* von der Baumsuche-Heuristik *performCompoundMoves* aufgerufen wird, wird $cmWork$ mit einer leeren Liste initialisiert, $cmStopValue$ hat den Wert 1 und $maxMoves$ wird auf $n_m(s^*) - 1$ gesetzt, wodurch sichergestellt werden soll, dass eine Lösungssequenz aus Kranoperationen mindestens eine Kranoperation kürzer ist als die durch die Greedy-Heuristik gefundene Lösung s^*. Der Eingabe-/Ausgabe-Parameter Cm dient dazu, die Kranoperationsketten zu sammeln, die durch die mehrfachen Aufrufe der Prozedur *determineCompoundMoves* entstehen.

Algorithmus 8 Erzeugung von Kranoperationsketten beim CRP: determineCompoundMoves

{Eingabe: Aktuelle Matrix L, aktuelle Kranoperationskette $cmWork$, maximale Anzahl an Kranoperationen um L zu leeren $maxMoves$, Zähler zur Längenbeschränkung der Kranoperationskette $cmStopValue$}
{Ein-/Ausgabe: Liste Cm der generierten Kranoperationsketten}

{Bestimme die produktiven Kranoperationen in L}
1: determineProductiveMoves(L, Pm)
{Aktuelles $cmWork$ speichern}
2: $cmWorkSave \leftarrow cmWork$
3: **for all** produktive Kranoperation $pm \in Pm$ **do**
{Wende pm auf L an}
4: $L' \leftarrow L \oplus pm$
{Prüfe ob vollständige Lösung erreicht wurde}
5: **if** L' ist eine leere Matrix **then**
6: Aktualisiere beste Lösung s^* falls nötig
7: **continue**
8: **end if**
{Prüfe ob erweiterte Kranoperationskette überflüssig ist}
9: **if** $n'_m(L') > maxMoves$ **then**
10: **continue**
11: **end if**
{Hänge aktuelle produktive Kranoperation an ursprüngliche Kranoperationskette an}
12: $cmWork \leftarrow cmWorkSave \cup \{pm\}$
13: **if** $cmStopValue < cmStopThreshold$ **then**
{Kranoperationskette erweitern}
14: determineCompoundMoves(L',$cmWork$,$maxMoves - 1$, $cmStopValue \cdot |Pm|$, Cm)
15: **else**
{Kranoperationskette nicht erweitern, zur Liste der Umladeoperationsketten hinzufügen}
16: $Cm \leftarrow Cm \cup \{cmWork\}$
17: **end if**
18: **end for**
19: **end.**

Zunächst wird durch den Aufruf der Prozedur *determineProductiveMoves* die Menge an produktiven Kranoperationen Pm für L bestimmt. Dann wird die die durch den globalen Parameter $cmStopThreshold$ definierte Grenze noch nicht erreicht wurde, wird die Kranoperationskette mit einem rekursiven Aufruf von *determineCompoundMoves* weiter verlängert. Ansonsten wird die Kranoperationskette nicht weiter verlängert und der Ergebnismenge Cm hinzugefügt.

Wenn die Anwendung einer produktiven Kranoperation zu einer leeren Bucht führt, handelt es sich bei der Konkatenation der Teillösung s, der aktuellen Kranoperationskette $cmWork$ sowie der produktiven Kranoperation pm um eine neue vollständige Lösung. In diesem Fall wird die bisher beste bekannte Lösung s^*, falls notwendig, aktualisiert. Ansonsten wird die produktive Kranoperation ignoriert. Eine produktive Kranoperation pm wird ebenfalls übergangen, wenn die Erweiterung der aktuellen Kranoperationskette $cmWork$ um pm zu einer Kranoperationskette führt, mit der keine neue beste Lösung erreicht werden kann. Das ist der Fall, wenn die untere Schranke $n'_m(L')$ größer als $maxMoves$ ist.

Bestimmung produktiver Kranoperationen

Bei der Erzeugung der Kranoperationsketten in der Prozedur *determineCompoundMoves* werden nur bestimmte Klassen von Kranoperationen berücksichtigt, nämlich die so genannten produktiven Kranoperationenklassen SG-HP, SG-AP, SS-HP, GG und die Klasse der Entnahmeoperationen.

Die Menge der produktiven Kranoperationen Pm für einen Buchtzustand L wird in der Prozedur *determineProductiveMoves* (siehe Algorithmus 9) in maximal drei Schritten bestimmt:

- Wenn Entnahmeoperationen in L möglich sind, wird die Menge der produktiven Kranoperationen in L als die Menge aller Entnahmeoperationen, die in L zulässig sind, definiert. Da die Entnahmeoperationen notwendigerweise durchgeführt werden müssen, um eine Bucht zu entleeren, und jede Entnahme eines Containers das weitere Lösen des Problems vereinfacht, werden Entnahmeoperationen immer bevorzugt.

Algorithmus 9 Bestimmung der produktiven Kranoperationen beim CRP: determineProductiveMoves

{Eingabe: Aktuelles Layout L}
{Ausgabe: Menge Pm der produktiven Kranoperationen in L}

1: $Pm \leftarrow$ Menge aller Entnahmeoperationen in L
2: **if** $Pm = \emptyset$ **then**
3: $Pm \leftarrow$ Menge aller SG-Umladeoperationen in L
4: **if** $Pm = \emptyset$ **then**
5: $Sshp \leftarrow$ Menge aller SS-HP-Umladeroperationen in L
6: Sortiere $Sshp$ aufsteigend nach der Containerprioritätendifferenz zwischen dem bewegten Container und dem Container mit höchster Priorität im Zielstapel
7: Beschränke $Sshp$ auf maximal $maxSshp$ Kranoperationen
8: $Gg \leftarrow$ Menge aller GG-Umladeroperationen in L
9: Sortiere Gg aufsteigend nach dem Wert $(diffZuvor - diffDanach)$, wobei $diffZuvor$ die Differenz der Gruppen des bewegten Containers m und des Containers direkt unter m im Senderstapel, und $diffDanach$ die Differenz der Gruppen von m und dem (bisher) obersten Container im Empfängerstapel bezeichnet.
10: Beschränke Gg auf maximal $maxGg$ Kranoperationen
11: $Pm \leftarrow Sshp \cup Gg$
12: **end if**
13: **end if**
14: **end.**

- Falls keine Entnahmeoperationen, dafür aber SG-Umladeoperationen möglich sind, wird Pm als die Menge der im Buchtzustand L zulässigen SG-Umladeoperationen bestimmt. Da SG-Umladeoperationen dazu dienen, Entnahmeoperationen im weiteren Verlauf zu ermöglichen, werden sie gegenüber anderen Klassen von Umladeoperationen bevorzugt.

- Wenn weder Entnahmeoperationen noch SG-Umladeoperationen durchgeführt werden können, wird die Menge Pm aus SS-HP- und GG-Umladeoperationen gebildet. SS-HP-Umladeoperationen sind meistens unvermeidbar, um blockierte Container mit aktuell höchster Priorität freizulegen. Die Sortierung der SS-HP-Umladeoperationen anhand der Containerprioritätendifferenz zwischen dem bewegten Container und dem Container mit minimaler Containerpriorität im Senderstapel hilft dabei, mehrere SS-Umladeoperationen desselben Containers zu vermeiden. GG-Umladeoperationen können zukünftige SG-Umladeoperationen ermöglichen. Die Sortierung der GG-Umladeoperationen erfolgt so, dass solche Umladeoperationen am Anfang der Liste stehen, bei denen die Prioritätendifferenz zwischen dem umzuladenden Container und dem sich direkt darunter befindlichen Container im Empfängerstapel möglichst groß ist und die Prioritätendifferenz zwischen dem dann umgeladenen Container und dem (vormals) obersten Container im Empfängerstapel möglichst gering ist. Dadurch werden Buchtzustände mit höherem sauberen Angebot bevorzugt, was wiederum SG-Umladeoperationen in der Zukunft wahrscheinlicher macht. Um den Suchaufwand zu beschränken ist die maximale Anzahl an berücksichtigten SS-HP-Umladeoperationen und GG-Umladeoperationen durch die globalen Algorithmusparameter $maxSshp$ bzw. $maxGg$ limitiert.

Grobalgorithmus der Baumsuche

Der Grobalgorithmus der Baumsuche wird durch die rekursive Prozedur *performCompoundMoves* realisiert (vgl. Algorithmus 10). Die Prozedur wird initial mit einer leeren aktuellen Lösung $s_c \leftarrow \emptyset$ und der Ausgangsmatrix als aktuelle Matrix $L \leftarrow L_{init}$ aufgerufen. Die globale Variable s^*, die die beste bisher bekannte Lösung speichert, ist mit dem Ergebnis der Greedy-Heuristik s_{Greedy} vorbelegt.

Beim Durchlaufen der Prozedur wird zunächst geprüft, ob die gesamte Suche abgebrochen werden kann, d.h., ob eine Optimallösung gefunden wurde oder das vorgegebene Zeitlimit erreicht wurde. Danach wird geprüft, ob es sich bei L um eine leere Bucht handelt. Ist dies der Fall, wird der aktuelle Prozeduraufruf beendet. Falls die aktuelle Lösung s_c weniger Kranoperationen als die bisher beste bekannte Lösung s^* enthält, wird sie zur neuen besten bekannten Lösung.

Wird die Suche nicht abgebrochen, wird - wie bereits beschrieben - eine sortierte Liste von in L ausführbaren Kranoperationsketten generiert. Die Elemente der Liste werden gemäß dreier Sortierkriterien so geordnet, dass sich die vielversprechendsten Kranoperationsketten am Anfang der Liste befinden sollten:

- Aufsteigend nach der zu erwartenden Anzahl an Kranoperationen, um die aktuelle Matrix L zu leeren (optimistische Anzahl der Kranoperationen): Die zu erwartende Anzahl an Kranoperationen wird als Summe zweier Anteile berechnet. Der erste Anteil ist die Anzahl an Kranoperationen in der Kranoperationskette. Der zweite Anteil wird als untere Schranke der Anzahl erforderlicher Kranoperationen (bis zur Lösung) angewendet auf den Buchtzustand bestimmt, der sich durch die Anwendung der Kranoperationskette auf die aktuelle Matrix L ergibt. Offensichtlich sind Kranoperationsketten zu bevorzugen, bei denen die Chance (im zuvor erläuterten Sinne), eine leere Matrix mit möglichst wenigen Kranoperationen zu erreichen, relativ hoch ist.

- Absteigend nach der Anzahl an Kranoperationen in der Kranoperationskette: Haben zwei Kranoperationsketten für das erste Kriterium gleich große Werte, wird diejenige Kranoperationskette bevorzugt, die aus mehr Kranoperationen besteht. Bei dieser Kranoperationskette ist der Beitrag der unteren Schranke für die Gesamtanzahl an zu erwartenden Kranoperationen bis zu einer leeren Matrix geringer. Damit ist die Wahrscheinlichkeit höher, dass die so berechnete Anzahl an Kranoperationen annähernd erreicht werden kann.

- Absteigend nach dem sauberen Angebot der Matrix, die durch Anwendung der Kranoperationskette auf L entsteht: Wenn eine Matrix einen hohen Wert für das saubere Angebot hat, ist es wahrscheinlich, dass zukünftig viele SG-Umladeoperationen durchgeführt werden können, was generell zu kürzeren Lösungssequenzen führen sollte als im gegenteiligen Fall.

Wenn die Liste der Kranoperationsketten sortiert ist und mehr als $nSucc$ Kranoperationsketten enthält, werden nur die ersten $nSucc$ Kranoperationsketten weiter betrachtet, um die Größe des Suchbaumes und damit die Algorithmus-Laufzeit zu beschränken.

7.5 Evaluierung

In diesem Abschnitt wird die Leistungsfähigkeit der Baumsuche-Heuristik für das CRP untersucht. Zunächst erfolgt ein Vergleich mit den von anderen Forschern entworfenen und in Kapitel 7.3 vorgestellten Lösungsverfahren für das CRP. Dann werden die Ergebnisse von Performanzmessungen mit zufallsbasierten Probleminstanzen beschrieben. Am Ende des Abschnitts werden einzelne wichtige Komponenten der Baumsuche-Heuristik gesondert untersucht.

Für die Evaluierung wurde die im vorherigen Abschnitt beschriebene Baumsuche-Heuristik in der Programmiersprache Java implementiert. Alle Tests wurden auf einem Intel Core 2 Duo - Prozessor (P7350, 16 GFLOPS) mit 2 GHz Takfrequenz und 2 GB Hauptspeicher durchgeführt. Für alle Tests wurden durchgehend folgende Algorithmus-Parameterwerte festgelegt: $cmStopThreshold = 150$ (siehe Seite 137), $maxSshp = 3$, $maxGg = 2$ (siehe jeweils Seite 139), $nSucc = 10$ (siehe Seite 141). Das Zeitlimit wurde auf 60 Sekunden gesetzt. Alle Rechenzeiten in diesem Kapitel sind in Sekunden angegeben. Weil alle gültigen Lösungen für eine CRP-Instanz dieselbe Anzahl an Entnahmeoperationen enthalten, ist es für den Vergleich ausreichend, die Anzahl an Umladeoperationen n_{rel} zu betrachten. Da es sich bei der Baumsuche-Heuristik um ein deterministisches Verfahren handelt, wurde jede Testinstanz nur einmal gelöst.

Zur Bestimmung der Leistungsfähigkeit der Verfahren werden folgende Kennzahlen betrachtet:

- n^*_{rel} bzw. n_{rel}: Anzahl der Umladeoperationen in den Lösungen der Baumsuche-Heuristik bzw. eines Vergleichsverfahrens.
- ct^* bzw. ct: Rechenzeit (Algorithmus-Laufzeit) der Baumsuche-Heuristik bzw. eines Vergleichsverfahrens in Sekunden.

Algorithmus 10 Baumsuche-Heuristik für das CRP: performCompundMoves

{Eingabe: aktuelle (Teil-)Lösung s_c, aktuelle Matrix L}
{Ausgabe: beste Lösung s^*}

{Breche die Suche ab bei optimaler Lösung oder Erreichen des Zeitlimits}
1: **if** $s^* \neq \emptyset$ und $n_m(s^*) = n_m^{'}(L_{init})$ oder Zeitlimit ist abgelaufen **then**
2: return
3: **end if**
{Breche die aktuelle Prozedur-Instanz ab, wenn eine Lösung gefunden wurde. Aktualisiere die beste Lösung s^* wenn erforderlich.}
4: **if** L ist eine leere Matrix **then**
5: **if** $n_m(s_c) < n_m(s^*)$ **then**
6: $s^* \leftarrow s_c$
7: **end if**
8: **return**
9: **end if**
{Generiere die Kranoperationsketten Cm für die Verzweigung}
10: $Cm \leftarrow \emptyset$
11: $cmWork \leftarrow \emptyset$
12: determineCompoundMoves(L,$cmWork$,$n_m(s^*) - n_m(s) - 1$, 1, Cm)
13: Sortiere die Umladeoperationsketten in Cm:
- aufsteigend nach der optimistischen Anzahl der Kranoperationen
- absteigend nach der Anzahl an Kranoperationen in der Kranoperationskette
- absteigend nach dem sauberem Angebot der Matrix, die durch Anwendung der Kranoperationskette auf L entsteht

{Beschränke die Länge der Liste auf maximal $nSucc$ Elemente}
14: $nCm \leftarrow min(nCm, nSucc)$
{Verzweige anhand der generierten Kranoperationsketten}
15: **for** $i \leftarrow 1$ **to** nCm **do**
16: $s' \leftarrow s_c \cup \{Cm(i)\}$
{Bestimme die Nachfolgematrix L' durch Anwendung von $Cm(i)$ auf L}
17: $L' \leftarrow L \oplus Cm(i)$
18: performCompoundMoves(s',L')
19: **end for**
20: **end.**

- $\frac{n^*_{rel}}{n_{rel}}$: Verhältnis zwischen der Anzahl an Umladeoperationen in den Lösungen der Baumsuche-Heuristik und denen eines Vergleichsverfahrens. Werte kleiner als 1 bedeuten, dass die Baumsuche-Heuristik bessere Lösungen finden konnte als das Vergleichsverfahren. Für Werte größer als 1 gilt das Gegenteil.

Bei Testklassen entsprechen die Kennzahlen n^*_{rel}, n_{rel}, ct^*, ct und LB_{diff} jeweils den Mittelwerten der Kennzahlen für die einzelnen Testinstanzen der Testklasse. Der Wert für $\frac{n^*_{rel}}{n_{rel}}$ bezieht sich in diesem Fall ebenfalls auf die durch Mittelwertsbildung gewonnenen Werte für n^*_{rel} und n_{rel}. Testklassen werden in Großbuchstaben notiert, während einzelne Testinstanzen in Kleinbuchstaben notiert werden.

7.5.1 Vergleich mit bisher publizierten Lösungsverfahren

Caserta et al. (2009b) haben 21 Testklassen für das CRP bereitgestellt[1] (vgl. Tabelle 7.2). Jede Testklasse besteht aus 40 Testinstanzen mit einer konstanten Anzahl an Stapeln und Containern. Jeder Container hat seine eigene Containerpriorität und die Stapelhöhe ist nicht beschränkt.

Zu Beginn sind alle Stapel gleichmäßig mit Containern gefüllt, d.h. die Buchten haben eine rechteckige Form. Die Testklassen sind nach dem Schema [CVS | *Container pro Stapel* | *Anzahl Stapel*] benannt. Zum Beispiel bezeichnet CVS 3-4 eine Menge von 40 Testinstanzen bestehend aus Buchten mit 4 Stapeln, in denen jeweils 3 Container platziert sind.

Da für alle bisher veröffentlichten Verfahren zur Lösung des CRP Ergebnisse mit diesen Testklassen vorliegen, eignen sie sich besonders, um die Leistungsfähigkeit der Baumsuche-Heuristik im Vergleich zu anderen Verfahren zu evaluieren.

Vergleich mit Kim und Hong (2006)

Im direkten Vergleich mit dem Ansatz von Kim und Hong (2006) zeigt sich, dass die Lösungsqualität der Baumsuche-Heuristik über alle Testklassen hinweg wesentlich höher liegt (vgl. Tabelle 7.3). Im Durchschnitt wiesen die mit der Baumsuche-

[1] http://iwi.econ.uni-hamburg.de/IWIWeb/GetDocument.aspx?documentid=1341

				Container		
Testklasse	Instanzen	Stapel	Stapel-höhe	Prioritäten	Anzahl	schlecht platziert
CVS 3-3	40	3	-	9	9	3,70
CVS 3-4	40	4	-	12	12	4,78
CVS 3-5	40	5	-	15	15	5,78
CVS 3-6	40	6	-	18	18	7,20
CVS 3-7	40	7	-	21	21	8,18
CVS 3-8	40	8	-	24	24	9,30
CVS 4-4	40	4	-	16	16	7,30
CVS 4-5	40	5	-	20	20	10,18
CVS 4-6	40	6	-	24	24	11,08
CVS 4-7	40	7	-	28	28	13,40
CVS 5-4	40	4	-	20	20	10,60
CVS 5-5	40	5	-	25	25	13,15
CVS 5-6	40	6	-	30	30	16,80
CVS 5-7	40	7	-	35	35	19,13
CVS 5-8	40	8	-	40	40	22,20
CVS 5-9	40	9	-	40	45	25,10
CVS 5-10	40	10	-	40	50	27,73
CVS 6-6	40	6	-	36	36	21,45
CVS 6-10	40	10	-	60	60	36,08
CVS 10-6	40	6	-	60	60	42,25
CVS 10-10	40	10	-	100	100	70,53

Tabelle 7.2: Eigenschaften der CRP-Testinstanzen von Caserta et al. (2009b)

	Kim und Hong (2006)		Baumsuche-Heuristik		
Testfall	n_{rel}	ct	n^*_{rel}	ct^*	$\frac{n^*_{rel}}{n_{rel}}$
CVS 3-3	7,10	<1	4,95	<1	0,70
CVS 3-4	10,70	<1	6,05	<1	0,57
CVS 3-5	14,50	<1	6,85	<1	0,47
CVS 3-6	18,10	<1	8,28	<1	0,46
CVS 3-7	20,10	<1	9,20	<1	0,46
CVS 3-8	26,00	<1	10,45	<1	0,40
CVS 4-4	16,00	<1	9,90	<1	0,62
CVS 4-5	23,40	<1	12,62	<1	0,54
CVS 4-6	26,20	<1	13,70	<1	0,52
CVS 4-7	32,20	<1	15,78	<1	0,49
CVS 5-4	23,70	<1	15,00	<1	0,63
CVS 5-5	37,50	<1	18,62	<1	0,50
CVS 5-6	45,50	<1	21,82	<1	0,48
CVS 5-7	52,30	<1	23,58	<1	0,45
CVS 5-8	61,80	<1	26,72	<1	0,43
CVS 5-9	72,40	<1	29,68	<1	0,41
CVS 5-10	80,90	<1	31,85	<1	0,39
CVS 6-6	37,30	<1	29,45	<1	0,79
CVS 6-10	75,10	<1	43,60	2	0,58
CVS 10-6	141,60	<1	75,65	46	0,53
CVS 10-10	178,60	<1	116,9	60	0,65
Mittelwert					0,53

Tabelle 7.3: Leistungsvergleich zwischen dem Verfahren von Kim und Hong (2006) und der Baumsuche-Heuristik für das CRP

Heuristik gefundenen Lösungen um 47 % weniger Umladeoperationen auf, d.h. die Anzahl an benötigten Umladeoperationen wurde fast halbiert. Das Verfahren von Kim und Hong benötigt auch bei den großen Probleminstanzen CVS 6-6 bis CVS 10-10 durchweg Laufzeiten unter 1 Sekunde (auf einem Pentium III mit 800 GHz Taktfrequenz und 128 MB Speicher). Die Baumsuche-Heuristik bleibt aber mit gut 5 Sekunden durchschnittlicher Laufzeit und einer maximalen Laufzeit von einer Minute ebenfalls in einem für den Praxiseinsatz sicherlich geeigneten Bereich.

Vergleich mit Caserta et al. (2009b)

Der in Caserta et al. (2009b) beschriebene Algorithmus (von den Autoren ausgeführt auf einem Pentium 4-PC mit 512 MB RAM - Taktfrequenz nicht angegeben) findet für zwei Testklassen (CVS 3-6 und CVS 3-7) bessere Ergebnisse, in einem Fall (CVS 4-4) exakt gleich gute Ergebnisse, und in den anderen 18 Fällen schlechtere Ergebnisse als die Baumsuche-Heuristik (vgl. Tabelle 7.4). Im Mittel über alle Testklassen enthalten die mit der Baumsuche-Heuristik gefundenen Lösungen ca. 10 % weniger Umladeoperationen. Allerdings nimmt die Laufzeit der Baumsuche-Heuristik bei sehr großen Testklassen (CVS 6-10, CVS 10-6 und CVS 10-10) stärker zu als die des Verfahrens von Caserta et al. (2009b).

Es soll an dieser Stelle noch angemerkt werden, dass bei den Resultaten von Caserta et al. (2009b) für jede Testklasse die jeweils optimalen Parameterwerte für den Korridor verwendet wurden, die durch Berechnung aller Testinstanzen mit unterschiedlichen Parameterwerten ermittelt wurden. Zu kleine Korridore führen dazu, dass potenziell gute Lösungen nicht gefunden werden können, während große Korridore den Suchraum schnell explodieren lassen. In der Praxis wäre es für ein konkretes Problem aber nicht unbedingt ex ante klar, welche Korridorwerte nun zu einem guten Ergebnis führen. Zusätzlich erweitert der Algorithmus für den Fall, dass keine Kranoperation im vorgegebenen Korridor möglich ist, den Korridor zufällig vertikal oder horizontal, so dass es leicht zu ungültigen Lösungen kommen kann, wenn tatsächlich eine Höhenbeschränkung der Stapel vorliegt.

Auch bei der Baumsuche-Heuristik sind einige Algorithmus-Parameterwerte vor dem Start festzulegen, so z.B. die maximale Anzahl an Nachfolgeknoten $nSucc$. Diese Parameterwerte bleiben aber unverändert über alle Testklassen. Es wurde also keine Parameteroptimierung bezogen auf die einzelnen Testklassen vorgenommen, da eine solche eigentlich auf die Laufzeit des Algorithmus angerechnet werden müsste.

Vergleich mit Caserta et al. (2009a)

Caserta et al. (2009a) geben in ihrem Artikel nur Ergebnisse für 4 der 21 CVS-Testklassen an. Der Verfasser der vorliegenden Arbeit hat auf Basis des öffentlich zugänglichen Quellcodes des Lösungsverfahrens Ergebnisse für alle Testklassen be-

	Caserta et al. (2009b)		Baumsuche-Heuristik		
Testfall	n_{rel}	ct	n^*_{rel}	ct^*	$\frac{n^*_{rel}}{n_{rel}}$
CVS 3-3	5,40	<1	4,95	<1	0,92
CVS 3-4	6,50	<1	6,05	<1	0,93
CVS 3-5	7,30	<1	6,85	<1	0,94
CVS 3-6	7,90	<1	8,28	<1	1,05
CVS 3-7	8,60	<1	9,20	<1	1,07
CVS 3-8	10,50	<1	10,45	<1	1,00
CVS 4-4	9,90	<1	9,90	<1	1,00
CVS 4-5	16,50	<1	12,63	<1	0,77
CVS 4-6	19,80	<1	13,70	<1	0,69
CVS 4-7	21,50	<1	15,78	<1	0,73
CVS 5-4	16,60	<1	15,00	<1	0,90
CVS 5-5	18,80	<1	18,63	<1	0,99
CVS 5-6	22,10	<1	21,83	<1	0,99
CVS 5-7	25,80	<1	23,58	<1	0,91
CVS 5-8	30,10	<1	26,73	<1	0,89
CVS 5-9	33,10	<1	29,68	<1	0,90
CVS 5-10	36,40	2	31,85	<1	0,88
CVS 6-6	32,40	2	29,45	<1	0,91
CVS 6-10	49,50	2	43,60	2	0,88
CVS 10-6	102,00	5	75,65	46	0,74
CVS 10-10	128,30	6	116,90	60	0,91
Mittelwert					0,90

Tabelle 7.4: Leistungsvergleich zwischen dem Verfahren von Caserta et al. (2009b) und der Baumsuche-Heuristik für das CRP

rechnet. Die Laufzeit des Algorithmus wurde auf 60 Sekunden begrenzt. Die Tests wurden auf einem Core 2 Duo - Prozessor (P7350, 16 GFLOPS) mit 2 GHz Takfrequenz und 2 GB Hauptspeicher Arbeitsspeicher durchgeführt.

Im direkten Vergleich mit dem Algorithmus aus Caserta et al. (2009a) zeigt sich in Bezug auf die Lösungsqualität, dass die Baumsuche-Heuristik für jede Testklasse kürzere Lösungssequenzen findet (vgl. Tabelle 7.5). Im Mittel über alle Testklassen liegt die Verbesserung der Lösungsqualität gegenüber Caserta et al. (2009a) bei 5 %.

Dabei gilt es zu beachten, dass der Algorithmus aus Caserta et al. (2009a) auf einer Binärmatrix-Darstellung der Bucht basiert, mit der nur Buchten abgebildet werden können, in denen jede Containerpriorität genau einmal vorkommt. Dies ist in der Realität nicht immer der Fall. Die Baumsuche-Heuristik kommt ohne Randomisierungselemente aus und kann auch auf Probleminstanzen angewendet werden, in denen Containerprioritäten mehrfach vorkommen.

Vergleich mit Caserta und Voß (2009a,c)

Der Vergleich mit dem Ansatz von Caserta und Voß (2009a) und Caserta und Voß (2009c) beschränkt sich auf die vier größten Testklassen, da nur für diese Ergebnisse in den Publikationen angegeben werden und der Quellcode des Verfahrens nicht öffentlich zugänglich ist. Die Autoren testeten ihren Algorithmus auf einem Pentium 4-PC mit 512 MB RAM (Taktfrequenz nicht angegeben).

Die Baumsuche-Heuristik findet für drei der vier Testklassen bessere Lösungen. Im Mittel wird über diese Testklassen eine Verbesserung der Lösungsgüte um ca. 4 % erreicht. Für die größte Testklasse CVS 10-10 sind die Ergebnisse allerdings um ca. 11 % schlechter (vgl. Tabelle 7.6). Im Mittel über die vier vorliegenden Testklassen sind die beiden Verfahren hinsichtlich der Lösungsgüte ebenbürtig.

Vergleich mit Caserta et al. (2012)

Wie im Kapitel 7.3 erläutert, beschreiben Caserta et al. (2012) drei weitere Lösungsansätze für das CRP. Zwei davon, nämlich der auf dem CRP-Modell BRP-II

	Caserta et al. (2009a)		Baumsuche-Heuristik		
Testfall	n_{rel}	ct	n^*_{rel}	ct^*	$\frac{n^*_{rel}}{n_{rel}}$
CVS 3-3	5,05	60	4,95	<1	0,98
CVS 3-4	6,20	60	6,05	<1	0,98
CVS 3-5	7,13	60	6,85	<1	0,96
CVS 3-6	8,58	60	8,28	<1	0,96
CVS 3-7	9,35	60	9,20	<1	0,98
CVS 3-8	10,83	60	10,45	<1	0,96
CVS 4-4	10,30	60	9,90	<1	0,96
CVS 4-5	13,43	60	12,63	<1	0,94
CVS 4-6	14,35	60	13,70	<1	0,95
CVS 4-7	16,63	60	15,78	<1	0,95
CVS 5-4	15,73	60	15,00	<1	0,95
CVS 5-5	19,75	60	18,63	<1	0,94
CVS 5-6	23,05	60	21,83	<1	0,95
CVS 5-7	24,90	60	23,58	<1	0,95
CVS 5-8	29,05	60	26,73	<1	0,92
CVS 5-9	32,00	60	29,68	<1	0,93
CVS 5-10	34,78	60	31,85	<1	0,92
CVS 6-6	32,60	60	29,45	<1	0,90
CVS 6-10	47,75	60	43,60	2	0,91
CVS 10-6	83,58	60	75,65	46	0,91
CVS 10-10	121,33	60	116,90	60	0,96
Mittelwert					0,95

Tabelle 7.5: Leistungsvergleich zwischen dem Verfahren von Caserta et al. (2009a) und der Baumsuche-Heuristik für das CRP

	Caserta und Voß (2009a,c)		Baumsuche-Heuristik		
Testfall	n_{rel}	ct	n^*_{rel}	ct^*	$\frac{n^*_{rel}}{n_{rel}}$
CVS 6-6	30,85	60	29,45	<1	0,95
CVS 6-10	46,17	60	43,60	2	0,94
CVS 10-6	76,55	60	75,65	46	0,99
CVS 10-10	105,50	60	116,90	60	1,11
Mittelwert					1,00

Tabelle 7.6: Leistungsvergleich zwischen dem Verfahren von Caserta und Voß (2009a,c) und der Baumsuche-Heuristik für das CRP

basierende Ansatz sowie die Min-Max-Heuristik werden im Evaluierungsteil des Aufsatzes untersucht. Die Autoren verwendeten einen Pentium 4-PC mit 512 MB RAM (Taktfrequenz nicht angegeben). Es kommen wieder die CVS-Testklassen zum Einsatz, allerdings beschränken sich die Autoren auf die Testklassen CVS 3-3 bis CVS 3-8, CVS 4-4 bis CVS 4-6 und CVS 10-10. Zusätzlich wurden die Ansätze jeweils nur auf die ersten fünf Instanzen einer Testklasse angewendet. Um Missverständnisse zu vermeiden, wird hier der Name der Testklasse mit einem * geschrieben, wenn nur die ersten fünf Instanzen gemeint sind.

Im Vergleich zum BRP-II erreicht die Baumsuche-Heuristik im Allgemeinen eine bessere Lösungsqualität, im Mittel sind die Lösungssequenzen gut 3 % kürzer (vgl. Tabelle 7.7). Das scheint zunächst ein Widerspruch zu der Tatsache zu sein, dass BRP-II nur optimale Lösungen für das CRP findet. Die Diskrepanz ergibt sich daraus, dass BRP-II - wie auch alle anderen bisher veröffentlichten Ansätze - Umladeoperationen immer auf Stapel begrenzt, in denen sich ein Container mit aktuell höchster Priorität befindet. Dadurch lässt sich der Suchraum massiv einschränken. Die Baumsuche-Heuristik dagegen betrachtet generell alle gültigen Umladeoperationen. Somit besteht das Potenzial, Lösungen zu finden, die bei den anderen Ansätzen nicht erreichbar sind.

Wie zu erwarten, zeigt BRP-II ein nachteiliges Laufzeitverhalten. BRP-II benötigt bei Buchten ab 20 Containern bereits mehr als 10 Stunden und ist damit für den Praxiseinsatz in Containerterminals nicht geeignet.

Die Min-Max-Heuristik dagegen berechnet alle Ergebnisse - auch für die großen Probleminstanzen in CVS 10-10* - in weniger als 1 Sekunde (vgl. Tabelle 7.8). Die Baumsuche-Heuristik findet die Ergebnisse für 9 der 10 Testklassen im selben Zeitraum, benötigt aber für CVS 10-10* mit 60 Sekunden wesentlich länger. Auch wenn die Autoren keine Angabe zur Taktfrequenz des von ihnen verwendeten Prozessors machen, ist aufgrund des Prozessortyps davon auszugehen, dass es sich um ein weniger leistungsfähiges Modell handelt als jenes, das zur Evaluierung der Baumsuche-Heuristik verwendet wurde.

Die Lösungsqualität der Baumsuche-Heuristik ist bei 7 von 10 Testklassen höher und nur bei einer Testklasse geringer. Im Mittel findet die Baumsuche-Heuristik knapp 7 % kürzere Lösungssequenzen.

	BRP-II		Baumsuche-Heuristik		
Testfall	n_{rel}	ct	n^*_{rel}	ct^*	$\frac{n^*_{rel}}{n_{rel}}$
CVS 3-3*	3,60	3	3,60	<1	1,00
CVS 3-4*	5,20	8	5,20	<1	1,00
CVS 3-5*	7,20	125	7,20	<1	1,00
CVS 3-6*	8,20	102	7,60	<1	0,93
CVS 3-7*	9,20	989	9,00	<1	0,98
CVS 3-8*	9,80	1046	9,60	<1	0,98
CVS 4-4*	10,40	116	9,20	<1	0,88
CVS 4-5*	11,40	2482	12,60	<1	1,11
CVS 4-6*	15,40	86 000	13,00	<1	0,84
Mittelwert					0,97

Tabelle 7.7: Leistungsvergleich zwischen dem BRP-II-Verfahren von Caserta et al. (2012) und der Baumsuche-Heuristik für das CRP

	Min-Max-Heuristik		Baumsuche-Heuristik		
Testfall	n_{rel}	ct	n^*_{rel}	ct^*	$\frac{n^*_{rel}}{n_{rel}}$
CVS 3-3*	3,60	<1	3,60	<1	1,00
CVS 3-4*	5,20	<1	5,20	<1	1,00
CVS 3-5*	7,60	<1	7,20	<1	0,95
CVS 3-6*	8,20	<1	7,60	<1	0,93
CVS 3-7*	9,20	<1	9,00	<1	0,98
CVS 3-8*	10,00	<1	9,60	<1	0,96
CVS 4-4*	10,60	<1	9,20	<1	0,87
CVS 4-5*	11,80	<1	12,60	<1	1,07
CVS 4-6*	16,00	<1	13,00	<1	0,81
CSV 10-10*	147,40	<1	112,60	60	0,76
Mittelwert					0,93

Tabelle 7.8: Leistungsvergleich zwischen der Min-Max-Heuristik von Caserta et al. (2012) und der Baumsuche-Heuristik für das CRP

Zusammenfassung der Testergebnisse

Zusammenfassend lässt sich aus dem Vergleich mit allen bisherigen veröffentlichten Ansätzen zur Lösung des CRP schließen, dass die in dieser Arbeit entwickelte Baumsuche-Heuristik ein sehr erfolgreiches Lösungsverfahren darstellt: In Bezug auf die Lösungsqualität konnten die bisherigen Ansätze im Mittel über alle untersuchten Testklassen ausnahmslos übertroffen werden. Die Ergebnisse des Ansatzes aus Caserta und Voß (2009c) kamen dabei denen der Baumsuche-Heuristik noch am nächsten, allerdings konnte die Baumsuche-Heuristik auch hier für 3 von 4 Testklassen bessere Lösungen finden. Die Baumsuche bietet darüber hinaus eine für einen Praxiseinsatz wichtige Kombination aus hoher Lösungsqualität und kurzer Laufzeit.

Mit Blick auf den Praxiseinsatz in Containerterminals soll an dieser Stelle ein weiterer Aspekt beleuchtet werden. Wie erwähnt, wurde bei allen in diesem Kapitel berechneten Testklassen davon ausgegangen, dass für die Stapel keine Höhenbeschränkung gilt. Dies deckt sich allerdings nicht mit der Realität in Containerterminals, da die Kranhöhe und weitere Faktoren die maximale Stapelhöhe beschränken. Um die Auswirkungen einer Höhenbeschränkung auf die Leistungsfähigkeit der Baumsuche-Heuristik zu evaluieren, wurden die CVS-Testklassen mit der Baumsuche-Heuristik erneut berechnet, diesmal unter der Annahme, dass Container in einer Bucht mit n Containern pro Stapel nicht höher als $n+2$ Container gestapelt werden können. Z.B. wurde für den Testklasse CVS 3-4 die maximale Stapelhöhe auf 5 beschränkt. Die Ergebnisse sind Tabelle 7.9 zu entnehmen. Es zeigt sich, dass die Qualität der Lösungen durch die Berücksichtigung der Höhenrestriktion im Mittel über alle Testklassen nur um 2 % verschlechtert wird.

7.5.2 Evaluierung mit neuen Testinstanzen

In diesem Abschnitt werden Ergebnisse von Berechnungen mit den in Kapitel 6.5.2 eingeführten zufallsbasierten Testinstanzen präsentiert, die sich auch zur Evaluierung von CRP-Verfahren eignen. Tabelle 7.10 fasst die wesentlichen Parameter der Testklassen im Kontext des CRP zusammen. In der Spalte mit dem Titel „Schlecht platziert“ wird der Mittelwert der initial im Sinne des CRP schlecht platzierten Container über alle Testinstanzen einer Testklasse angegeben. Die Werte für das

	H unbeschränkt	H beschränkt	
Testklasse	n^1_{rel}	n^2_{rel}	$\frac{n^1_{rel}}{n^2_{rel}}$
CVS 3-3	4,95	4,98	0,99
CVS 3-4	6,05	6,05	1,00
CVS 3-5	6,85	6,85	1,00
CVS 3-6	8,28	8,28	1,00
CVS 3-7	9,20	9,23	1,00
CVS 3-8	10,45	10,50	1,00
CVS 4-4	9,90	9,93	1,00
CVS 4-5	12,62	12,65	1,00
CVS 4-6	13,70	13,70	1,00
CVS 4-7	15,78	15,78	1,00
CVS 5-4	15,00	15,52	0,97
CVS 5-5	18,62	18,80	0,99
CVS 5-6	21,82	22,08	0,99
CVS 5-7	23,58	23,58	1,00
CVS 5-8	26,72	27,03	0,99
CVS 5-9	29,68	30,05	0,99
CVS 5-10	31,85	32,25	0,99
CVS 6-6	29,45	31,13	0,95
CVS 6-10	43,60	44,48	0,98
CVS 10-6	75,65	83,03	0,91
CVS 10-10	116,90	125,38	0,93
Mittelwert			0,98

Tabelle 7.9: Vergleich der Ergebnisse der Baumsuche-Heuristik für das CRP mit und ohne Höhenbeschränkung

CRP unterscheiden sich von den Werten für das CPMP, weil das Prädikat „schlecht platziert“ für die zwei Probleme unterschiedlich definiert ist.

Alle 640 Testinstanzen wurden mit der in diesem Kapitel vorgestellten Baumsuche-Heuristik für das CRP mit einem Zeitlimit von 60 Sekunden gelöst. Die Ergebnisse sind Tabelle 7.11 zu entnehmen. Für jede Testklasse ist dort die durchschnittliche Anzahl an Umladeoperationen n^*_{rel}, die durchschnittliche Laufzeit des Algorithmus ct^* in Sekunden und die durchschnittliche absolute Differenz zur unteren Schranke LB_{diff} angegeben.

Tabelle 7.12 zeigt eine andere Sicht auf die Ergebnisse. Hier lässt sich der Einfluss der bei der Generierung der Testinstanzen variierten Faktoren auf die Baumsuche-Heuristik hinsichtlich der durchschnittlichen Laufzeit des Algorithmus ct^* sowie der durchschnittlichen absoluten Differenz zur unteren Schranke LB_{diff} ablesen. Werte in einer Spalte mit der Überschrift „niedriger“ beziehen sich auf die Testklassen, bei denen der jeweilige Faktor den niedrigeren Wert besitzt, Werte in einer Spalte mit der Überschrift „höher“ beziehen sich entsprechend auf Testklassen, bei denen der jeweilige Faktor den höheren Wert besitzt.

Die Erhöhung der Stapelanzahl führt im Durchschnitt zu einer um 57 % höheren Laufzeit (von ca. 12 auf ca. 19 Sekunden). Die mittlere absolute Differenz zur unteren Schranke erhöht sich dabei mit 9 % bzw. ca. 0,2 Umladeoperationen im wesentlich geringerem Maße.

Der Faktor „maximale Stapelhöhe“ stellt sich als dominierender Faktor mit Einfluss auf die Laufzeit und die absolute Differenz zur unteren Schranke heraus. Für Testinstanzen, bei denen dieser Wert niedriger ist (maximale Stapelhöhe 5), findet die Baumsuche-Heuristik Lösungen, die im Durchschnitt maximal um nur 0,14 Umladeoperationen von einer Optimallösung abweichen. Bei Testinstanzen mit höheren Werten für die maximale Stapelhöhe (maximale Stapelhöhe 8) steigt die mittlere Laufzeit um den Faktor 18, von unter 2 Sekunden auf ca. 30 Sekunden. Die durchschnittliche absolute Differenz zur unteren Schranke ist fast um den Faktor 33 erhöht und steigt von 0,14 auf knapp 4,5 Umladeoperationen. Für diese Testfälle erreicht die Baumsuche-Heuristik auch mehrfach das Zeitlimit von 60 Sekunden.

Die Containeranzahl beeinflusst die Laufzeit und die absolute Differenz zur unteren Schranke ebenfalls maßgeblich. Für die Testinstanzen mit höheren Werten für

				Container		
Testklasse	Instanzen	Stapel	Stapel-höhe	Anzahl	Prioritäten	schlecht platziert
BF1	20	16	5	48	10	21,80
BF2	20	16	5	48	10	25,80
BF3	20	16	5	48	20	22,55
BF4	20	16	5	48	20	26,80
BF5	20	16	5	64	13	29,65
BF6	20	16	5	64	13	35,80
BF7	20	16	5	64	26	30,10
BF8	20	16	5	64	26	36,80
BF9	20	16	8	77	16	38,00
BF10	20	16	8	77	16	45,05
BF11	20	16	8	77	31	38,20
BF12	20	16	8	77	31	45,90
BF13	20	16	8	103	21	50,40
BF14	20	16	8	103	21	62,15
BF15	20	16	8	103	42	51,15
BF16	20	16	8	103	42	63,05
BF17	20	20	5	60	12	27,25
BF18	20	20	5	60	12	31,70
BF19	20	20	5	60	24	27,55
BF20	20	20	5	60	24	34,35
BF21	20	20	5	80	16	37,25
BF22	20	20	5	80	16	44,55
BF23	20	20	5	80	32	37,30
BF24	20	20	5	80	32	43,95
BF25	20	20	8	96	20	46,15
BF26	20	20	8	96	20	55,65
BF27	20	20	8	96	39	46,70
BF28	20	20	8	96	39	57,00
BF29	20	20	8	128	26	63,10
BF30	20	20	8	128	26	76,15
BF31	20	20	8	128	52	64,25
BF32	20	20	8	128	52	77,40

Tabelle 7.10: Eigenschaften der neuen Testinstanzen für das CRP

diesen Faktor ist die Laufzeit der Baumsuche-Heuristik mit ca. 26 Sekunden knapp 324 % erhöht gegenüber den ca. 6 Sekunden, die der Algorithmus im Mittel für das Lösen der Testinstanzen mit niedrigeren Werten für diesen Faktor benötigt. Die durchschnittliche Differenz zur unteren Schranke ist stark erhöht, sie steigt um 743 % von ca. 0,5 auf knapp 4,5 Umladeoperationen.

Verglichen mit den zwei zuletzt genannten Faktoren hat der Faktor „Containerprioritäten“ wieder einen geringeren Einfluss auf Laufzeit und Differenz zur unteren Schranke. Die Laufzeit erhöht sich bei den Testinstanzen mit einer höheren Anzahl von Containerprioritäten gegenüber den Testinstanzen mit einer niedrigeren Anzahl von Containerprioritäten um 32 %, von ca. 14 auf ca. 18 Sekunden. Die mittlere absolute Abweichung zur unteren Schranke steigt nur um 23 %, von ca. 2,2 auf ca. 2,8 Umladeoperationen.

Die Erhöhung der zu Beginn schlecht platzierten Container hat dagegen einen leicht positiven Einfluss auf die Laufzeit (Verringerung um ca. 2 % von ca. 16,1 Sekunden auf 15,7 Sekunden) und einen deutlicheren positiven Einfluss auf die mittlere absolute Differenz zur unteren Schranke. Diese reduziert sich um 29 % bzw. um fast eine Umladeoperation. Die Ursache könnte darin liegen, dass die Anzahl an zu Beginn schlecht platzierten Containern ein wichtiges Element in der Berechnung der unteren Schranke für das CRP ist. Ist die Anzahl an zu Beginn schlecht platzierten Containern höher, verbessert sich potenziell die Güte der unteren Schranke. Damit können tendenziell häufiger Äste des Suchbaums abgeschnitten werden, so dass der Algorithmus früher terminiert. Die Verringerung der mittleren absoluten Differenz zur unteren Schranke könnte dem Effekt geschuldet sein, dass eine höhere untere Schranke ceteris paribus automatisch auch eine Verringerung der mittleren absoluten Abweichung zu dieser Schranke ergibt. Man muss zwar annehmen, dass die Wahl der zu präferierenden Kranoperationen speziell zu Beginn des Lösungsverfahrens bei einer höheren Anzahl an initial schlecht platzierten Containern erschwert ist, da z.B. die Wahrscheinlichkeit, Entnahmeoperationen oder SG-Umladeoperationen durchführen zu können, geringer ist. Allerdings ist dieser Effekt beim CRP durch die laufende Entnahme von Containern aus der Bucht und die damit einhergehende Entstehung von neuen freien Plätzen für produktive Umladeoperationen wohl nicht so wesentlich.

Zusammenfassend lässt sich festhalten, dass in Situationen, in denen 60 % der Plätze mit Containern gefüllt sind, auch für große Buchten nah-optimale Lösungen gefunden werden konnten. Dabei wird die Nähe zur Optimalität anhand der Differenz von n_{rel} zur unteren Schranke bestimmt. Insgesamt konnten 56 % der Probleminstanzen nachweislich optimal gelöst werden, da für diese keine Differenz zur unteren Schranke bestand. Wenn die Buchten zu 80 % mit Containern gefüllt sind, ist die mittlere absolute Abweichung zur unteren Schranke gegenüber Buchten, die zu 60 % mit Containern gefüllt sind, zwar relativ gesehen deutlich höher. Betrachtet man aber die absoluten Zahlen, liegen die Ergebnisse bei einer Stapelhöhe von 5 immer noch sehr nahe an den Werten für die untere Schranke. Nur bei Buchten mit einer Stapelhöhe von 8 weicht die Anzahl an Umladeoperationen in der Lösung deutlich von der unteren Schranke ab. Wie schon bei der Baumsuche-Heuristik für das CPMP sind auch bei der Baumsuche-Heuristik für das CRP die Faktoren maximale Stapelhöhe und Containeranzahl (relativ zur Buchtdimension) die bestimmenden Einflussgrößen für die Laufzeit und die Differenz zur unteren Schranke.

7.5.3 Evaluierung einzelner Komponenten der Baumsuche-Heuristik

In diesem Abschnitt werden einzelne Komponenten der Baumsuche-Heuristik evaluiert, um deren Einfluss auf die Lösungsqualität und die Rechenzeit der Baumsuche-Heuristik zu bestimmen.

Analog zum Vorgehen in Kapitel 6.5.3 wird die vorgestellte Baumsuche-Heuristik für das CRP (bezeichnet als $V0$) variiert und mit mit vier Varianten $V1$-$V4$ verglichen. Dabei wurden für die Varianten folgende Modifikationen gegenüber $V0$ vorgenommen:

- $V1$: In dieser Variante wird anstelle der unteren Schranke n'_m nur die einfache untere Schranke n'_{SG} und n als Anzahl der Entnahmeoperationen verwendet. Es wird also auf $n'_{nicht-SG}$ verzichtet.
- $V2$: In dieser Variante werden nur einelementige Kranoperationsketten verwendet. Es wird also nach jeder einzelnen Kranoperation verzweigt.

Testklasse	n^*_{rel}	ct^*	LB_{diff}
BF1	21,80	<1	0,00
BF2	25,80	<1	0,00
BF3	22,55	<1	0,00
BF4	26,80	<1	0,00
BF5	29,90	<1	0,25
BF6	35,80	<1	0,00
BF7	30,90	<1	0,80
BF8	36,80	<1	0,00
BF9	39,45	5	1,45
BF10	45,55	7	0,50
BF11	40,60	12	2,45
BF12	46,00	5	0,10
BF13	58,30	23	7,90
BF14	68,30	40	6,15
BF15	60,70	44	9,55
BF16	72,25	54	9,20
BF17	27,25	<1	0,00
BF18	31,70	<1	0,00
BF19	27,60	<1	0,05
BF20	34,35	<1	0,00
BF21	37,70	5	0,45
BF22	44,55	<1	0,00
BF23	38,05	6	0,75
BF24	43,95	<1	0,00
BF25	47,45	12	1,30
BF26	55,85	3	0,20
BF27	48,75	27	2,05
BF28	57,40	18	0,40
BF29	73,05	60	9,95
BF30	83,90	57	7,75
BF31	74,95	57	10,07
BF32	86,25	59	8,85

Tabelle 7.11: Berechnungsergebnisse der Baumsuche-Heuristik für das CRP für die neuen Testinstanzen

	ct^*		LB_{diff}	
Faktor	Niedriger	Höher	Niedriger	Höher
Stapelanzahl	12,38	19,38	2,40	2,61
Maximale Stapelhöhe	1,56	30,19	0,14	4,87
Containeranzahl	6,06	25,69	0,53	4,48
Anzahl Containerprioritäten	13,69	18,06	2,24	2,77
Anzahl schlecht platzierter Container	16,06	15,69	2,94	2,07

Tabelle 7.12: Einfluss unterschiedlicher Faktoren der neuen Testinstanzen auf die Baumsuche-Heuristik für das CRP

Variante der Baumsuche-heuristik	Gelöste Instanzen in %	Durchschn. Anzahl Umladeoperationen in gefundenen Lösungen gegenüber $V0$ in %	Durchsch. Rechenzeit für gefundene Lösungen gegenüber $V0$ in %
$V0$	100,00	100,00	100,00
$V1$	100,00	100,10	100,30
$V2$	100,00	102,00	259,60
$V3$	100,00	107,40	107,30
$V4$	100,00	113,80	>1000,00

Tabelle 7.13: Berechnungsergebnisse mit unterschiedlichen Varianten der Baumsuche-Heuristik über 21 CRP-Testklassen

- $V3$: In dieser Variante wird die Liste der Kranoperationsketten Cm nicht sortiert, nachdem sie vollständig aufgebaut wurde.
- $V4$: In dieser Variante wird bei der Wahl der Kranoperationen, die für die Generierung von Kranoperationsketten verwendet werden, auf ein spezielles Auswahlschema verzichtet. Es wird zufällig eine gültige Kranoperation ausgewählt.

Für alle Varianten wurden alle 21 CVS-Testklassen berechnet, jeweils mit einem Zeitlimit von 60 Sekunden. Die Ergebnisse der Berechnungen sind Tabelle 7.13 zu entnehmen und lassen sich wie folgt zusammenfassen:

- In allen Varianten konnte jede Testinstanz gelöst werden, was nicht verwundert, da die mit der Greedy-Heuristik gefundene Lösung als Verfahrenslösung dient, falls die Baumsuche-Heuristik keine bessere Lösung findet.

- Die Berücksichtigung von $n'_{nicht-SG}$ ($V1$) hat einen messbaren, aber äußerst geringen positiven Einfluss auf die Lösungsqualität und die Rechenzeit.
- Für alle Varianten $V1$ bis $V4$ zeigt sich eine Verschlechterung der Lösungsqualität, d.h. die Lösungen enthalten im Mittel mehr Umladeoperationen als bei $V0$. Bei den Varianten $V3$ (keine Sortierung der Umladeoperationsketten) und $V4$ (keine Auswahlregeln) ist die Verschlechterung mit 7 % bzw. 13 % besonders signifikant.
- Für alle Varianten $V1$ bis $V4$ ist eine Erhöhung der Rechenzeit zu beobachten. Besonders stark fällt diese bei den Varianten $V2$ und $V4$ aus.

Insgesamt hat der Komponententest gezeigt, dass die beschriebenen Komponenten einen positiven und meist wesentlichen Beitrag zur Effektivität und Effizienz der Baumsuche-Heuristik für das CRP leisten.

7.6 Zusammenfassung

In diesem Abschnitt wurde das Container Relocation Problem (CRP), ein dem CPMP ähnliches, NP-schweres kombinatorisches Optimierungsproblem informell und formal beschrieben. Danach wurden bisher vorgestellte Lösungsansätze für das CRP eingehend erläutert.

Daraufhin wurde eine Baumsuche-Heuristik für das CRP vorgeschlagen, die mit der unvollständigen Baumsuche das gleiche Grundgerüst verwendet wie das im vorherigen Kapitel vorgestellte Verfahren für das CPMP. Dieses Gerüst musste allerdings grundlegend anders ausgestaltet werden, um ein kompetitives Verfahren für das CRP zu entwickeln. So war es notwendig, ein gegenüber dem CPMP erweitertes und verfeinertes Klassifikationsschema für Kranoperationen heranzuziehen. Die Vorgehensweise zur Generierung von Kranoperationsketten der beiden Verfahren ist ebenfalls essentiell verschieden: Beim CPMP werden entweder nur Sequenzen aus SG-Umladeoperationen oder aus Nicht-SG-Umladeoperationen betrachtet. Beim CRP werden gemischte Sequenzen aus einer zuvor definierten Untermenge der möglichen Kranoperationen gebildet (produktive Kranoperationen), bei deren Auswahl

auf verschiedene, problemspezifische heuristische Sortierungsregeln zurückgegriffen wird.

Die untere Schranke für das CPMP ist wesentlich stärker als die untere Schranke für das CRP. Die Tatsache, dass die Logik zur Berechnung einer unteren Schranke für das CPMP nicht auf das CRP übertragbar ist, hat gezeigt, dass die beiden Probleme, trotz ihrer auf den ersten Blick großen Ähnlichkeit, fundamentale Unterschiede aufweisen. Um eine erfolgreiche Baumsuche-Heuristik für das CRP zu entwerfen, war es im Gegensatz zum Ansatz für das CPMP auch notwendig, eine Greedy-Heuristik zu konstruieren und in die Baumsuche zu integrieren.

In einer umfangreichen Evaluierung konnte gezeigt werden, dass die Baumsuche-Heuristik für das CRP die bisher veröffentlichten Ansätze in Bezug auf die Lösungsqualität fast immer übertrifft und auch im Hinblick auf die Rechenzeit eine gute Performanz zeigt. Die Evaluierung mit neu eingeführten zufallsbasierten Testinstanzen zeigte, dass mit der Baumsuche-Heuristik auch extrem umfangreiche Probleminstanzen gelöst werden können, bei überwiegend sehr hoher Ergebnisqualität. Die Untersuchung einzelner Komponenten der Baumsuche-Heuristik half dabei, den Beitrag der Komponenten zur Gesamtleistung der Baumsuche-Heuristik zu charakterisieren.

8 Heuristische Baumsuche für das Container Retrieval Problem (CRTP)

In diesem Kapitel wird das Container Retrieval Problem (CRTP) behandelt. Auf die Problembeschreibung und -definition folgt die Erläuterung eines von Lee und Lee (2010) veröffentlichten Lösungsansatzes für das Problem. Daraufhin wird eine Baumsuche-Heuristik zur Lösung des CRTP beschrieben. Schließlich wird die Leistungsfähigkeit dieses Ansatzes durch einen Vergleich mit dem Verfahren von Lee und Lee (2010) evaluiert. Die Ergebnisse dieses Kapitels wurden von Forster und Bortfeldt (2012a) veröffentlicht.

8.1 Problembeschreibung

Beim Container Retrieval Problem ist eine Menge von Containern aus dem Zwischenlager eines Containerterminals in einer vorgegebenen Reihenfolge zu entnehmen. Jedem Container ist eine Containergruppe bzw. Priorität zugewiesen, die die Entnahmereihenfolge der Container festlegt. Die Container sind, wie in Containerterminals üblich, in Stapeln untergebracht, und die Kräne, die für den Betrieb des Zwischenlagers eingesetzt werden, können immer nur den obersten Container eines Stapels bewegen.

Im Gegensatz zum im vorherigen Kapitel betrachteten CRP befinden sich die zu entfernenden Container beim CRTP nicht zwangsläufig alle in derselben Bucht,

sondern liegen in unterschiedlichen Buchten eines Blocks des Zwischenlagers. Es wird also ein ganzer Containerblock betrachtet.

Es wird ferner davon ausgegangen, dass die Container von einem RTG oder RMG bewegt werden, der für die Ausführung einer Kranoperation (Entnahme oder Umladen eines Containers) jeweils eine gewisse, hauptsächlich von der Anzahl an zu überbrückenden Stapeln und Buchten abhängige Zeit benötigt. Im Gegensatz zum CPMP und zum CRP wird also beim CRTP nicht vereinfachend davon ausgegangen, dass Kranoperationen grundsätzlich gleich lange Ausführungsdauern besitzen.

Die Aufgabenstellung beim CRTP ist es nun, für einen gegebenen Containerblock eine Sequenz von Kranoperationen zu finden, durch die der Block möglichst schnell und mit möglichst wenigen Kranoperationen vollständig und unter Berücksichtigung der Containerprioritäten geleert wird. Das CRTP wurde erstmals von Lee und Lee (2010) beschrieben.

8.2 Problemformalisierung

Das im vorherigen Abschnitt erläuterte CRTP wird nun formal definiert.

Ein Block BL im Zwischenlager eines Containerterminals lässt sich als ein Tripel (B,S,H) beschreiben. B ($B \in \mathbb{N}, B \geq 1$) steht für die Anzahl an Buchten, aus denen der Block besteht, S ($S \in \mathbb{N}, S \geq 1$) bezeichnet die Anzahl der Stapel, die sich in einer Bucht jeweils befinden und H ($H \in \mathbb{N}, H \geq 1$) legt die maximale Stapelhöhe der Stapel fest.

Der (Füll-)Zustand LBL (im Folgenden auch: das Layout) eines Blocks BL kann durch B Matrizen L_b ($1 \leq b \leq B$) mit H Zeilen und S Spalten repräsentiert werden. Eine Zelle einer Matrix wird durch ein Tupel (h,s) addressiert ($1 \leq h \leq H$, $1 \leq s \leq S$). Mit G ($G \in \mathbb{N}, G \geq 1$) wird die höchste vorkommende Containergruppe im gesamten Block bezeichnet. Jedem der Container im Block ist eine Containergruppe g ($g \in \mathbb{N}, 1 \leq g \leq G$) zugeordnet. Es gilt:

$$L_b(h,s) = \begin{cases} g, & \text{wenn sich in der Bucht } b \text{ im Stapel } s \text{ in der Ebene } h \\ & \text{ein Container der Containergruppe } g \text{ befindet,} \\ 0, & \text{wenn sich in der Bucht } b \text{ im Stapel } s \text{ in der Ebene } h \\ & \text{kein Container befindet.} \end{cases}$$

Ein Container kann nur entweder auf dem Boden der Bucht oder auf einem anderen Container stehen. Entsprechend muss, falls $L_b(h,s) = 0$ gilt, auch $L_b(h',s) = 0$ für alle $h' = h+1,\ldots,H$ gelten.

Ein Stapel des Blocks kann eindeutig durch ein Tupel $st = (b,s)$ ($1 \leq b \leq B$, $1 \leq s \leq S$) referenziert werden, wobei b die Bucht bezeichnet und s den Stapel innerhalb der Bucht. Eine Umladeoperation kann mit einem Tupel (st_d, st_r) von verschiedenen Stapeln beschrieben werden. Der erste Stapel $st_d = (b_d, s_d)$ wird als Sender bezeichnet, der zweite Stapel $st_r = (b_r, s_r)$ ist der Empfänger. Eine Umladeoperation bewegt den Container, der sich an der höchsten Position im Sender befindet, auf den niedrigsten freien Platz des Empfängers. Letzterer wird auch als Empfängerplatz bezeichnet. Bezeichnet man mit $L_d = L_{b_d}$ das Buchtlayout des Senders und mit $L_r = L_{b_r}$ das Buchtlayout des Empfängers, ist eine Umladeoperation genau dann zulässig für ein gegebenes Blocklayout LBL, wenn folgende zwei Bedingungen erfüllt sind:

1. Der Sender ist nicht leer: $L_d(1,s_d) > 0$.

2. Der Empfänger ist nicht voll: $L_r(H,s_r) = 0$.

Durch die Anwendung einer Umladeoperation (st_d, st_r) auf ein Blocklayout LBL werden die Matrizen L_d und L_r modifiziert und es entsteht ein neues Blocklayout LBL'. In diesem haben die Matrizen L', die nicht von der Umladeoperation betroffen sind, die gleichen Werte wie in LBL, d.h. $L'_b(h,s) = L_b(h,s)$ für $1 \leq b \leq B$, $b \neq d$, $b \neq r$, $1 \leq s \leq S$, $1 \leq h \leq H$.

Die von der Umladeoperation betroffenen Matrizen L'_d und L'_r in LBL' sind gegenüber L_d und L_r aus LBL modifiziert. Mit $top_L(s)$ soll die Reihe bezeichnet werden, in der sich der oberste Container im Stapel s in einer Matrix L befindet. Handelt es sich bei s um einen leeren Stapel, ist $top_L(s) = 0$. Es gilt damit:

- Die Reihe $top_{L_d}(s_d)$ im Senderstapel st_d, in der sich der umzuladende Container in L_d befunden hat, ist in L'_d leer: $L'_d(top_{L_d}(s_d), s_d) = 0$.

- In L'_r befindet sich der umgeladene Container eine Reihe über der Reihe $top_{L_r}(s_r)$: $L'_r(top_{L_r}(s_r) + 1, s_r) = L_d(top_{L_d}(s_d), s_d)$.

An allen anderen Plätzen stimmen die Elemente der Matrizen L' und L für beide Buchten weiterhin überein, d.h. $L'_d(h, s) = L_d(h, s)$ für $1 \leq h \leq H$, $1 \leq s \leq S$, $(h, s) \neq (top_{L_d}(s_d), s_d)$ und $L'_r(h, s) = L_r(h, s)$ für $1 \leq h \leq H$, $1 \leq s \leq S$, $(h, s) \neq (top_{L_r}(s_r) + 1, s_r)$.

Eine Entnahmeoperation entfernt einen Container aus der Bucht. Eine Entnahmeoperation von einem Stapel s_e in der Bucht b_e wird mit dem Tupel (b_e, s_e) bezeichnet. Mit $L_e = L_{b_e}$ sei das Buchtlayout der Bucht bezeichnet, aus der die Entnahmeoperation erfolgt. Eine Entnahmeoperation ist für ein Blocklayout LBL genau dann zulässig, wenn folgende Bedingungen erfüllt sind:

- Der Stapel (b_e, s_e) ist nicht leer: $L_e(1, s_e) > 0$

- Es gibt keinen Container in LBL, der eine höhere Priorität bzw. einen niedrigeren Gruppenindex hat als der oberste Container im Stapel s_e in der Bucht b_e. Es muss also gelten: $\forall b, 1 \leq b \leq B : \forall s, 1 \leq s \leq S : \forall h, 1 \leq h \leq H : L_b(h, s) \neq 0 \Rightarrow L_b(h, s) \geq L_e(top_{L_e}(s_e), s_e)$

Die Anwendung einer Entnahmeoperation (b_e, s_e) auf L_e in einem Blocklayout LBL wird die Matrix L_e modifiziert und es entsteht ein neues Blocklayout LBL'. In LBL' haben die Matrizen L', die nicht von der Entnahmeoperation betroffen sind, die gleichen Werte wie in LBL, d.h. $L'_b(h, s) = L_b(h, s)$ für $1 \leq b \leq B$, $b \neq b_e, 1 \leq s \leq S, 1 \leq h \leq H$. In der Matrix L'_e ist die Reihe $top_{L_e}(s_e)$ im Stapel s_e, in der sich der umzuladende Container in L_e befunden hat, nicht belegt: $L'_e(top_{L_e}(s_e), s_e) = 0$. Ansonsten sind die Werte von L'_e identisch mit den Werten von L_e, d.h. $L'_e(h, s) = L_e(h, s)$ für $1 \leq h \leq H$, $1 \leq s \leq S$, $(h, s) \neq (top_{L_e}(s_e), s_e)$.

Eine Sequenz von p Kranoperationen $s = (m_1, m_2, \ldots, m_p)$ $(p \in \mathbb{N}, p \geq 1)$ ist zulässig für ein Blocklayout LBL, wenn die Kranoperation m_1 für das Blocklayout LBL zulässig ist, und jede Kranoperation m_{i+1} für das durch Anwendung von m_i $(i = 1, \ldots, p - 1)$ entstehende Blocklayout LBL' zulässig ist.

Beim CRTP ist der gewünschte Zielzustand erreicht, wenn alle Container mit (gültigen) Entnahmeoperationen aus dem Block entfernt worden sind. Dann sind alle Stapel des Blocks leer und es gilt:

$$\forall b, 1 \leq b \leq B : \forall h, 1 \leq h \leq H : \forall s, 1 \leq s \leq S : L_b(h, s) = 0 \tag{8.1}$$

Die Kranoperationen werden beim CRTP durch RTGs oder RMGs ausgeführt. Für die Berechnung der Ausführungszeit für eine Kranoperation werden vier Faktoren berücksichtigt:

1. Die absolute Differenz n_b zwischen dem Index der Zielbucht und dem Index der Startbucht zur Ausführung der Kranoperation. Die Fahrtzeit in Bezug auf die Buchten wird mit $n_b \cdot t_b$ berechnet, wobei t_b eine Konstante für die Fahrtzeit des Krans in Bezug auf Buchtüberquerungen darstellt.

2. Die Anzahl an Stapeln n_s, die der Kran von seiner Ausgangsposition aus passieren muss, um den Container aufzunehmen und zum Zielstapel zu bringen. Die Fahrtzeit in Bezug auf die Stapel wird mit $n_s \cdot t_s$ berechnet, wobei t_s eine Konstante für die Fahrtzeit zur Überbrückung eines Stapels darstellt.

3. Die Zeit t_{ad}, die der Kran zum Beschleunigen und Abbremsen benötigt. Dieser Faktor kommt nur dann zum Tragen, wenn das Gerüst des Krans bewegt werden muss, d.h. der Kran von einer Bucht zu einer anderen Bucht fährt.

4. Die Zeit t_{pp}, die der Spreader zum Aufnehmen bzw. Abladen eines Containers benötigt.

Die Krandauer für die Ausführung einer Kranoperation, bei der der Kran n_b Buchten und n_s Stapel überquert, lässt sich anhand der Formel 8.2 bestimmen:

$$t = n_b \cdot t_b + n_s \cdot t_s + ad \cdot t_{ad} + t_{pp} \quad (n_b \geq 1 \Rightarrow ad = 1, n_b = 0 \Rightarrow ad = 0) \tag{8.2}$$

Die Krandauer T_{ms} für eine Sequenz ms von p Kranoperationen ergibt sich aus der Summe der Krandauern $t_i (1 \leq i \leq p)$ der einzelnen Kranoperationen.

Um die Krandauer für eine Entnahmeoperation zu bestimmen, wird angenommen, dass es einen speziell ausgezeichneten Stapel im Block gibt, der als Übergabestation fungiert. Die Krandauer für eine Entnahmeoperation entspricht dann der Krandauer einer Umladeoperation auf diesen speziellen Stapel.

Das CRTP lässt sich schließlich folgendermaßen formulieren: Für den Anfangszustand eines durch B Matrizen $L_1, \ldots, L_B$ repräsentierten Blocklayouts LBL soll eine Sequenz ms, bestehend aus p Kranoperationen $m_1, \ldots, m_p$, gefunden werden, die folgende drei Anforderungen erfüllt:

1. Die Sequenz ms ist zulässig für den Ausgangszustandes des Blocks.

2. Der Block BL ist nach Anwendung der letzten Kranoperation m_p leer, der Zustand LBL' erfüllt also die Bedingung 8.1.

3. Für jede Sequenz von Kranoperationen ms' bestehend aus p' Kranoperationen, die die Anforderungen 1. und 2. erfüllt, gilt: $w_T \cdot T_{ms'} + p' \geq w_T \cdot T_{ms} + p$. Dabei ist $w_T \in \mathbb{R}, w \geq 0$ ein Faktor, mit dem die relative Bedeutung der Krandauer im Verhältnis zur Anzahl der Kranoperationen festgelegt wird. Je größer w_T gewählt wird, desto stärker ist der Einfluss der Krandauer auf die Zielfunktion und desto geringer ist der Einfluss der Anzahl an Kranoperationen auf die Zielfunktion.

Es können CRTP-Instanzen konstruiert werden, die prinzipiell nicht lösbar sind. So ist z.B. ein Block, in dem alle Stapel bis zur maximalen Stapelhöhe mit Containern gefüllt sind und bei dem die höchsten Container in den Stapeln jeweils schlecht platziert sind, nicht lösbar, da kein freier Platz vorhanden ist, um die schlecht platzierten Container umzuladen. Im Folgenden wird stets die Lösbarkeit der Probleminstanzen unterstellt.

Die Komplexität des CRTP wurde bisher in der Literatur noch nicht diskutiert. Ausgehend von der Erkenntnis, dass es sich beim verwandten CRP um ein NP-schweres Optimierungsproblem handelt, lässt sich allerdings leicht nachvollziehen, dass auch das CRTP ein NP-schweres Optimierungsproblem ist.

Man setze den Parameter w_T zur Gewichtung der Krandauer in der Zielfunktion auf 0. In diesem Fall geht nur die Anzahl der Kranoperationen in die Zielfunktion

für das CRTP ein, und somit entspricht die Zielfunktion der des CRP. Jede Bucht einer CRP-Instanz kann als einelementiger Block betrachtet werden und somit von einem Verfahren für das CRTP gelöst werden. Da es sich beim CRP um ein NP-schweres Optimierungsproblem handelt, muss es sich demnach auch beim CRTP um ein (mindestens) NP-schweres Optimierungsproblem handeln.

8.3 Bisherige Lösungsansätze

Bisher wurde nur ein weiterer Lösungsansatz für das CRTP veröffentlicht. Lee und Lee (2010) präsentieren eine Heuristik für das CRTP, die auf drei nacheinander zu durchlaufenden Phasen basiert. In der Phase 1 (*initial phase*) wird für die gegebene Probleminstanz eine erste Lösung s generiert. In Phase 2 (*movement reduction phase*) wird versucht, die Anzahl der Kranoperationen in dieser Lösung zu verringern, während die Zielsetzung der Phase 3 (*time reduction phase*) schließlich die Verringerung der Krandauer der Lösung ist.

Die Generierung einer initialen Lösung in Phase 1 erfolgt nach einem einfachen Greedy-Schema, bei dem allerdings davon ausgegangen wird, dass jede Containerpriorität nur maximal einmal vergeben ist: Wenn der Container mit aktuell höchster Priorität im Block zugänglich ist, wird dieser entfernt. Ist der Container nicht zugänglich, befinden sich $n, n \in \mathbb{N}, n \geq 1$ Container über ihm im Stapel. Diese n Container werden nun nacheinander jeweils auf den nächsten freien Platz in einem direkten Nachbarstapel umgeladen (die konkrete Auswahlregel wird nicht benannt). Somit wird der Container mit höchster Priorität zugänglich und kann entfernt werden. Dieses Verfahren wird solange wiederholt, bis der gesamte Block geleert ist. Ergebnis der Phase 1 ist eine gültige Lösung s.

In der zweiten Phase wird versucht, die Anzahl an Kranoperation in der Lösung s zu verringern, indem für Container, die in s mehr als einmal bewegt werden, alternative (kürzere) Pfade gesucht werden, durch die die Container am Ende trotzdem im selben Stapel landen wie bei der Anwendung von s. Dazu werden randomisiert alternative Pfaden für jeden der Container generiert und mittels eines geeigneten gemischt-ganzzahligen linearen Optimierungsprogrammes dann die optimale Kombination (mit möglichst wenig Kranoperationen) aus allen diesen alternativen Pfaden

berechnet, wobei die Zulässigkeit der Lösung erhalten bleibt. Dieser Schritt enspricht der *Integer programming subroutine* des Lösungsverfahrens von Lee und Chao (2009) für das CPMP (siehe Kapitel 6.3). Das Ergebnis dieser Phase ist die modifizierte Lösung s'.

In Phase 3 werden nun, ausgehend von s', wieder alternative Pfade für die einzelnen Container erzeugt. Erneut wird ein gemischt-ganzzahliges lineares Optimierungsprogramm zur Selektion einer optimalen Kombination aus den verschiedenen Pfaden der Container formuliert, allerdings wird nun auch die Krandauer in der Zielfunktion berücksichtigt. Da dadurch die Rechenzeit für das Programm signifikant erhöht wird, können in dieser Phase in der Regel nicht für alle Container alternative Pfade evaluiert werden. Aus dem Artikel geht allerdings nicht hervor, nach welchen Entscheidungsregeln die Auswahl der zu berücksichtigenden Container erfolgt.

Abschließend soll herausgestellt werden, dass das Verfahren trotz Einsatzes eines gemischt-ganzzahligen linearen Optimierungsprogrammes eine Heuristik ist, d.h. in der Regel keine optimale Lösung findet. Die Optimalität kann nur bezüglich der Kombinationen aus den verschiedenen, randomisiert generierten Alternativpfaden für die Container in Phase 2 und 3 erreicht werden. Diese Optimalität ist also nur in Bezug auf die generierten Alternativpfade zu verstehen, nicht auf alle möglichen Kranoperationssequenzen für die gegebene Probleminstanz.

8.4 Entwurf einer Heuristik für das CRTP

In diesem Abschnitt wird gezeigt, wie eine Baumsuche-Heuristik zur Lösung des CRTP eingesetzt werden kann.

In Kapitel 7.4 wurde eine Baumsuche-Heuristik für das CRP beschrieben, die sich im Vergleich zu den bisher veröffentlichten alternativen Lösungsverfahren als sehr kompetitiv herausgestellt hat. Beim CRP wird allerdings nur eine einzige Bucht betrachtet, während beim CRTP ein ganzer Containerblock berücksichtigt werden muss. Entsprechend stellt sich die Frage, wie das Verfahren adaptiert werden kann, um es für das CRTP einsetzen zu können.

Die Grundidee dazu ist, eine CRTP-Probleminstanz in zwei sequentiellen Schritten zu lösen. Im ersten Schritt wird für jede der B Buchten des Containerblocks eine dazugehörige CRP-Probleminstanz erzeugt. Diese B CRP-Instanzen werden dann mit der Baumsuche-Heuristik für das CRP gelöst, so dass am Ende des ersten Schritts B Sequenzen $s_1, \ldots, s_B$ von Kranoperationen vorliegen, mit der jeweils eine der Buchten des Blocks unter Berücksichtigung der Containerprioritäten *innerhalb der Bucht* geleert werden könnten. Es ist erwähnenswert, dass die Lösung der einzelnen CRP-Probleminstanzen parallel durchgeführt werden kann, da bei diesem Ansatz keinerlei Abhängigkeiten zwischen den Instanzen bestehen.

Im zweiten Verfahrensschritt müssen die B Sequenzen von Kranoperationen derart in einer neuen Sequenz s^* zusammengeführt werden, dass eine gültige Lösung für die CRTP-Instanz entsteht. Bei diesen Schritt muss also sichergestellt werden, dass eine Entnahmeoperation eines Containers aus einer Bucht b nur dann erfolgt, wenn dieser die höchste Priorität in allen B Buchten hat, nicht nur die höchste Priorität in b.

Dies wird durch Algorithmus 11 sichergestellt. Der Algorithmus benötigt als Eingabeparameter den initialen Zustand des Containerblocks BL sowie die B Lösungen $s_1, \ldots, s_B$ der CRP-Baumsuche-Heuristik für die einzelnen Buchten des Blocks. Die Lösung s^* für das CRTP ist initial leer und wird sukzessive durch Anhängen von Kranoperationen aus $s_1, \ldots, s_B$ aufgebaut. Solange der Block nicht leer ist, wird zunächst die niedrigste Containergruppe (höchste Containerpriorität) *lgbl* unter allen Containern im Block ermittelt. Dann wird eine Bucht b_i gesucht, in der sich ein Container c der Gruppe *lgbl* befindet. Nun werden alle Kranoperationen in s_i ausgeführt bis einschließlich der Entnahmeoperation von c. Letztere ist automatisch auch die erste Entnahmeoperation, die sich noch in der Sequenz s_i befindet (siehe Algorithmus Zeilen 13-15). Dies ist durch die Zulässigkeit der CRP-Lösungssequenzen gewährleistet. Alle ausgeführten Kranoperationen werden dann aus s_i entfernt und ans Ende von s^* angehängt. Dies wird solange wiederholt, bis der komplette Containerblock BL geleert ist.

Die dadurch entstehende Sequenz von Kranoperationen s^* ist eine gültige Lösung für das CRTP, d.h. es handelt sich um eine zulässige Sequenz von Kranoperationen, nach deren Ausführung der Block geleert ist (Bedingungen 1 und 2 aus Abschnitt 8.2 werden erfüllt).

Algorithmus 11 Erzeugung einer CRTP-Lösung auf Basis eines CRP-Verfahrens

{Eingabe: Ausgangszustand LBL des Blocks BL in Form von B Matrizen $L_1, \ldots, L_B$, Lösungen $s_1, \ldots, s_B$ der CRP-Baumsuche-Heuristik für diese Matrizen}

{Ausgabe: Lösung s^* für das CRTP}

```
1:  s* ← ∅
2:  while mindestens eine Matrix in LBL ist nicht leer do
3:    lgbl ← niedrigste Containergruppe in LBL
4:    for i ← 1 to B do
5:      lgb ← niedrigste Containergruppe in L_i
6:      if L_i ist keine leere Matrix und lgbl = lgb then
7:        removed ← false
8:        while removed = false do
9:          move ← erstes Element in s_i
            {Wende move auf das Blocklayout LBL an}
10:         LBL ← LBL ⊕ move
11:         Entferne move aus s_i
            {Erweitere s* um move}
12:         s* ← s* ∪ {move}
13:         if move ist eine Entnahmeoperation then
14:           removed ← true
15:         end if
16:       end while
17:     end if
18:   end for
19: end while
20: end.
```

8.5 Evaluierung

In diesem Abschnitt erfolgt ein Vergleich der Leistungsfähigkeit der Baumsuche-Heuristik für das CRTP und dem Verfahren von Lee und Lee (2010).

Lee und Lee (2010) evaluieren ihren Ansatz mit zwei unterschiedlichen Typen von Testfällen (siehe Tabelle 8.1). Die Autoren beschreiben 10 Testklassen des Typs R, die jeweils aus 5 Testinstanzen bestehen, sowie 10 Testklassen des Typs U, die jeweils aus 2 Testinstanzen bestehen. Alle Testinstanzen bestehen aus Buchten mit 16 Stapeln. Die Testklassen unterscheiden sich hinsichtlich der Anzahl an Buchten und Containern sowie der maximalen Stapelhöhe. Der Unterschied zwischen den Testklassen des Typs R und des Typs U liegt darin, dass beim Typ R die Containerprioritäten zufällig vergeben wurden, während beim Typ U die Prioritäten derart verteilt sind, dass alle Container, die nicht auf dem Boden eines Stapels stehen, initial schlecht platziert sind.

Zur Bestimmung der Krandauer wurden folgende Parameter verwendet: $t_b = 3,5s$, $t_s = 1,2s$, $t_{ad} = 40s$, $t_{pp} = 30s$ (siehe Seite 167). In den Tabellen sind die zwei Dimensionen der Zielfunktionen, die Krandauer sowie die Anzahl an Kranoperationen, separat ausgewiesen. Lee und Lee (2010) verwenden für die Tests einen Core 2 Duo - Prozessor (E8500 CPU, 23 GFLOPS) mit 3,42 GHz Taktfrequenz und 3,46 GB Hauptspeicher.

Für die Evaluierung wurde die im vorherigen Abschnitt beschriebene Baumsuche-Heuristik für das CRTP in der Programmiersprache Java implementiert. Alle Tests mit der Baumsuche-Heuristik wurden auf einem Intel Core 2 Duo - Prozessor (P7350, 16 GFLOPS) mit 2 GHz Takfrequenz und 2 GB Hauptspeicher durchgeführt. Die Baumsuche-Heuristik für das CRP, auf die im ersten Schritt des Lösungsverfahrens zurückgegriffen wird, wurde mit den in Abschnitt 7.5 dokumentierten Parametern aufgerufen. Der Gewichtsfaktor w_T kommt nicht zum Tragen, da für die Evaluierung die Krandauer und die Anzahl der Kranoperationen analog zu Lee und Lee (2010) separat angegeben werden.

Die Ergebnisse sind Tabelle 8.2 zu entnehmen. Dabei gelten folgende Definitionen für die verwendeten Variablen:

Testklasse	Instanzen	Buchten	Stapel	Stapelhöhe	Container	
					Prioritäten	Anzahl
(R/U)011606-0070	5/2	1	16	6	70	70
(R/U)011608-0090	5/2	1	16	8	90	90
(R/U)021606-0140	5/2	2	16	6	140	140
(R/U)021608-0190	5/2	2	16	8	190	190
(R/U)041606-0280	5/2	4	16	6	280	280
(R/U)041608-0380	5/2	4	16	8	380	380
(R/U)061606-0430	5/2	6	16	6	430	430
(R/U)081606-0570	5/2	8	16	6	570	570
(R/U)061608-0570	5/2	6	16	8	570	570
(R/U)101606-0720	5/2	10	16	6	720	720

Tabelle 8.1: Eigenschaften der CRTP-Testinstanzen der Typen R/U von Lee und Lee (2010)

- cmt^* bzw. cmt: Krandauer der Lösungen der Baumsuche-Heuristik bzw. des Verfahrens von Lee und Lee (2010).
- n^*_{rel} bzw. n_{rel}: Anzahl der Umladeoperationen in den Lösungen der Baumsuche-Heuristik bzw. des Verfahrens von Lee und Lee (2010). Da jede Lösung des CRTP gleich viele Entnahmeoperationen enthält, ist es für einen Vergleich ausreichend, die Umladeoperationen zu betrachten.
- ct^* bzw. ct: Rechenzeit (Algorithmus-Laufzeit) der Baumsuche-Heuristik bzw. des Verfahrens von Lee und Lee (2010) in Sekunden.
- $\frac{cmt^*}{cmt}$: Verhältnis zwischen der Krandauer der Lösungen der Baumsuche-Heuristik und des Verfahrens von Lee und Lee (2010). Werte kleiner als 1 bedeuten, dass die Baumsuche-Heuristik bessere Lösungen finden konnte als das Verfahren von Lee und Lee (2010). Für Werte größer als 1 gilt das Gegenteil.
- $\frac{n^*_{rel}}{n_{rel}}$: Verhältnis zwischen der Anzahl an Umladeoperationen in den Lösungen der Baumsuche-Heuristik und denen des Verfahrens von Lee und Lee (2010). Werte kleiner als 1 bedeuten, dass die Baumsuche-Heuristik bessere Lösungen finden konnte als das Verfahren von Lee und Lee (2010). Für Werte größer als 1 gilt das Gegenteil.

Es zeigt sich, dass die Baumsuche-Heuristik für alle Testklassen durchgängig bessere Lösungen finden konnte als das Verfahren von Lee und Lee (2010), sowohl in Bezug auf die Krandauer (im Durchschnitt ca. 16 % kürzer) als auch auf in Bezug auf die Anzahl an Umladeoperationen (im Durchschnitt ca. 8 % weniger).

Die Verbesserung in Bezug auf die Krandauer ist im Mittel bei den Testinstanzen des Typs R und den Testinstanzen des Typs U gleich hoch. In Bezug auf die Anzahl an Umladeoperationen konnte die Baumsuche-Heuristik für Testinstanzen des Typs R die Ergebnisse von Lee und Lee (2010) im Schnitt um 10 % verbessern, während die Verbesserung für Testinstanzen des Typs U durchschnittlich knapp 6 % beträgt. Die Laufzeit für Testinstanzen des Typs R ist mit im Mittel ca. 10 Sekunden um den Faktor 2 höher als die Laufzeit für Testinstanzen des Typs U, die im Durchschnitt bei ca. 5 Sekunden liegt.

In allen Fällen kann mit der Baumsuche-Heuristik die Rechenzeit im Vergleich zu Lee und Lee (2010) drastisch verkürzt werden, trotz der Tatsache, dass Lee und Lee (2010) einen leistungsstärkeren Prozessor zur Ermittlung ihrer Ergebnisse verwendet haben. Im Schnitt benötigte die Baumsuche-Heuristik etwas weniger als 8 Sekunden zur Lösung einer Testinstanz, während der Algorithmus von Lee und Lee (2010) dazu rund 5,4 Stunden benötigt. Es lässt sich vermuten, dass dies im wesentlichen darauf zurückzuführen ist, dass der Ansatz von Lee und Lee (2010) auf der Lösung von relativ rechenintensiven gemischt-ganzzahligen linearen Optimierungsprogrammen beruht, während die Baumsuche-Heuristik ohne solche Komponenten auskommt.

8.6 Zusammenfassung

In diesem Kapitel wurde das Container Retrieval Problem (CRTP) eingeführt und formal definiert. Durch Bezugnahme auf das CRP konnte gezeigt werden, dass es sich auch beim CRTP um ein NP-schweres kombinatorisches Optimierungsproblem handelt. Im Anschluss wurde das Lösungsverfahren von Lee und Lee (2010) für das CRTP vorgestellt.

Dann wurde gezeigt, wie sich, basierend auf der Baumsuche-Heuristik für das CRP, eine Baumsuche-Heuristik für das CRTP entwickeln lässt. Kernidee des An-

satzes ist es, in einem ersten Schritt die Baumsuche-Heuristik für das CRP einzeln auf die Buchten des Containerblocks anzuwenden. Im zweiten Schritt wird aus den Kranoperationen in den Lösungen für das CRP eine gemeinsame Sequenz von Kranoperationen generiert, die den Bedingungen für eine Lösung des CRTP hinsichtlich der Entnahmereihenfolge der Container genügt. Das vorgestellte Verfahren kann prinzipiell auch mit jedem anderen Lösungsverfahren für das CRP gekoppelt werden.

Der Leistungsvergleich mit dem Verfahren von Lee und Lee (2010) ergab, dass die Baumsuche-Heuristik hinsichtlich der Lösungsqualität (Krandauer und Anzahl Kranoperationen) und der Rechenzeit den Ansatz von Lee und Lee (2010) dominiert.

	Lee und Lee (2010)			Baumsuche-Heuristik				
Testklasse	cmt	n_{rel}	ct	cmt^*	n^*_{rel}	ct^*	$\frac{cmt^*}{cmt}$	$\frac{n^*_{rel}}{n_{rel}}$
R011606-0070	11 446	125,40	8204	9534	108,00	8	0,83	0,86
R011608-0090	16 943	191,40	9597	12 502	148,40	10	0,74	0,78
R021606-0140	21 950	230,20	9608	19 413	215,60	9	0,88	0,94
R021608-0190	34 146	367,80	12 824	27 367	324,60	10	0,80	0,88
R041606-0280	45 664	454,20	12 427	41 103	439,80	9	0,90	0,97
R041608-0380	73 979	769,20	13 353	58 853	657,80	11	0,80	0,86
R061606-0430	74 885	709,80	15 016	67 417	684,20	10	0,90	0,96
R061608-0570	122 613	1242,00	17 105	92 363	988,20	12	0,75	0,80
R081606-0570	104 955	945,40	17 105	93 557	910,20	11	0,89	0,96
R101606-0720	136 716	1169,20	20 009	122 451	1134,00	12	0,90	0,97
U011606-0070	11 539	127,50	21 580	10 077	125,00	<1	0,87	0,98
U011608-0090	15 976	177,50	21 574	13 245	167,00	10	0,83	0,94
U021606-0140	23 982	257,00	21 563	20 902	251,50	3	0,87	0,98
U021608-0190	36 453	400,50	21 555	28 677	353,50	5	0,79	0,88
U041606-0280	50 042	507,50	21 541	44 406	501,00	6	0,89	0,99
U041608-0380	76 112	800,00	21 528	60 499	706,00	5	0,79	0,88
U061606-0430	81 894	793,00	21 516	70 906	772,50	6	0,87	0,97
U061608-0570	122 542	1234,00	21 515	95 397	1059,50	5	0,78	0,86
U081606-0570	113 705	1044,00	21 510	97 740	1019,50	4	0,86	0,98
U101606-0720	153 101	1345,00	21 520	128 219	1291,00	7	0,84	0,96
Mittelwert							0,84	0,92

Tabelle 8.2: Leistungsvergleich zwischen dem Verfahren von Lee und Lee (2010) und der Baumsuche-Heuristik für das CRTP

9 Zusammenfassung und Ausblick

9.1 Zusammenfassung

Im Zentrum dieser Arbeit standen die drei Stackingprobleme Container Pre-Marshalling Problem (CPMP), Container Relocation Problem (CRP) und Container Retrieval Problem (CRTP), die in Zwischenlagern von Containerterminals auftreten.

An Containerterminals erfolgt der Containerumschlag von Schiffen auf Landtransportmittel und umgekehrt. In Kapitel 2 wurde dargelegt, dass der Warentransport in Containern ein Eckpfeiler des Weltwirtschaftssystems ist und dessen außerordentlich hohe ökonomische Bedeutung herausgestellt. In Kapitel 3 wurden dann Containerterminals als neuralgische Punkte des globalen Containertransports detailliert beschrieben. Dabei wurden auch verschiedene Leistungskennzahlen vorgestellt.

Einige wichtige Grundbegriffe aus dem Themenbereich der Optimierungs- und Entscheidungsprobleme wurden in Kapitel 4 eingeführt. Speziell wurde auf den Begriff des kombinatorischen Optimierungsproblems eingegangen, da es sich bei allen drei im späteren Verlauf der Arbeit untersuchten Problemen um (NP-schwere) kombinatorische Optimierungsprobleme handelt. In diesem Kapitel wurden auch verschiedene exakte und heuristische Verfahrensklassen zur Lösung von kombinatorischen Optimierungsproblemen vorgestellt. Schließlich wurde noch thematisiert, wie ein Baumsucheverfahren zu einem heuristisches Verfahren ausgestaltet werden kann und argumentiert, warum ein Baumsucheverfahren ein erfolgsversprechender Ansatz zur Lösung der erwähnten Stackingprobleme sein kann.

In Kapitel 5 wurde ein umfangreicher Überblick über betriebliche Entscheidungsprobleme in Containerterminals und Ansätze zu deren Lösung gegeben, wobei der Fokus auf den operativen betrieblichen Entscheidungsproblemen im Zwischenlager

der Containerterminals lag. Für diese Entscheidungsprobleme wurde ein Klassifizierungsschema mit den vier Gruppen „Bestimmung der Blöcke für Container", „Bestimmung der Position der Container im Block (Stackingprobleme)", „Zuordnung von Stapelkränen zu Blöcken“ und „Zuordnung von Stapelkränen zu Jobs in den Blöcken“ vorgeschlagen.

In den drei folgenden Kapiteln wurde jeweils eines der eingangs erwähnten Stackingprobleme intensiv untersucht. Zunächst erfolgte eine Beschreibung des Problems und eine formelle Darstellung. Bei allen drei Problemen ist es das Ziel, mit möglichst wenig (Zeit-)Aufwand einen gegebenen Ausgangszustand des Zwischenlagers mit Kranoperationen in einen angestrebten Zielzustand zu überführen. Dabei geht man davon aus, dass für die Container im Zwischenlager eine gewisse Entnahmereihenfolge verlangt wird, die sich in der Containergruppe eines Container ausdrückt. Je niedriger die Gruppe, desto eher muss der Container entladen werden. Grundsätzlich dürfen Container mit höherer Gruppe erst aus dem Zwischenlager entnommen werden, wenn alle Container mit niedrigerer Gruppe bereits das Zwischenlager verlassen haben.

Beim CPMP müssen die Container so innerhalb einer Bucht umgeladen werden, dass sich am Ende kein Container unter einem Container mit höherer Containergruppe befindet. Dieser Zustand erlaubt es, dass im Anschluss alle Container aus dem Zwischenlager entnommen werden können, ohne dass eine Umladeoperation nötig ist. Beim CRP ist dagegen das Ziel, eine Bucht von Containern vollständig zu leeren, wobei immer nur Container mit aktuell niedrigster Containergruppe unter allen sich in der Bucht befindlichen Containern entnommen werden können. In beiden Fällen gilt es, den jeweiligen Zielzustand mit möglichst wenig Kranoperationen zu erreichen. Das CRTP stellt eine Verallgemeinerung des CRP dar, bei der nicht nur eine einzige Bucht, sondern ein ganzer Containerblock im Zwischenlager geleert werden soll. Darüber hinaus ist neben der Anzahl an Kranoperationen auch die Krandauer, definiert als die Zeit, die der Kran zur Durchführung der Kranoperationen benötigt, zu berücksichtigen.

Auf die Problembeschreibung folgte jeweils eine Übersicht der bisher publizierten Ansätze zur Lösung des Problems, in der die Grundidee wie auch die wesentlichen Komponenten der verschiedenen Ansätze erläutert wurden.

Das Kernstück der Kapitel war die Darstellung einer Baumsuche-Heuristik zur Lösung des jeweiligen Problems. Dabei wurde der grundlegende Ansatz einer Baumsuche-Heuristik in jedem Kapitel problemspezifisch ausgestaltet, um zu einem kompetitiven Verfahren zu gelangen.

Die Baumsuche-Heuristik für das CPMP in Kapitel 6 basiert auf dem Ansatz, ausschließlich entweder SG- (schlecht-gut) oder Nicht-SG-Kranoperationsketten anzuwenden und kann darüber hinaus auf eine relativ starke untere Schranke zurückgreifen. Der Vergleich mit den bisher veröffentlichten Ansätzen sowie die Berechnungsergebnisse mit neuen, komplexen Testinstanzen zeigte, dass die Baumsuche-Heuristik einen überaus erfolgreichen Ansatz zur Lösung des CPMP darstellt.

Die Baumsuche-Heuristik für das CRP in Kapitel 7 setzt eine Greedy-Heuristik ein, um gleich zu Beginn des Lösungsverfahrens mit einer oberen Schranke arbeiten zu können. Im Gegensatz zum Ansatz beim CPMP verwendet die Baumsuche-Heuristik für das CRP gemischte Kranoperationsketten, die aus einer Untermenge aller möglichen Kranoperationen ausgewählt werden. Der Vergleich mit den existierenden Lösungsverfahren zeigte, dass die Baumsuche-Heuristik sowohl in Hinblick auf die Lösungsqualität als auch auf die Rechenzeit ein äußerst leistungsstarkes Verfahren ist. Berechnungen mit neuen, komplexen Testinstanzen legten ebenfalls den Schluss nahe, dass es sich bei der Baumsuche-Heuristik um ein vielversprechendes Lösungsverfahren für das CRP handelt.

Die Baumsuche-Heuristik für das CRTP in Kapitel 8 greift auf die Baumsuche-Heuristik für das CRP zurück, um Lösungen für die einzelnen Buchten des Containerblocks zu finden, und konstruiert aus den einzelnen Lösungen schließlich eine geeignete Lösung für das CRTP. Gegenüber dem einzigen bisher veröffentlichten Verfahren von Lee und Lee (2010) konnte mit der Baumsuche-Heuristik eine signifikante Verbesserung der Lösungsqualität erreicht werden, gepaart mit einer um mehrere Größenordnungen niedrigeren Berechnungsdauer.

9.2 Ausblick

Die Ergebnisse der Arbeit bieten eine ganze Reihe von interessanten Anknüpfungspunkten, die sich grob in zwei Gruppen einteilen lassen.

Bei der ersten Gruppe geht es darum, die Problemstellungen und die Baumsuche-Heuristik als Lösungsverfahren noch stärker an die Realität in Containerterminals anzupassen. Beispielsweise berücksichtigen weder das CRP noch das CRTP das Einbringen von Containern in das Zwischenlager. In der Praxis passiert aber genau dies häufig parallel zur Entnahme von Containern aus demselben Bereich des Zwischenlagers. Eine Baumsuche-Heuristik zur Lösung dieses erweiterten Problems müsste entsprechend eine neuen Typ von Kranoperation in das Schema zur Generierung von Kranoperationsketten mit einbeziehen.

Auch das CPMP ließe sich durch Modifikation der Annahmen weiter an die Realität anpassen. So finden sich in der Praxis neben der Reihenfolge der geplanten Entnahme oft weitere Bedingungen, die bei der Stapelung berücksichtigt werden müssen. Beispielsweise kann es angestrebt sein, dass sehr schwere Container nur bis zu einer maximalen Höhe gestapelt werden dürfen, die niedriger als die allgemeine maximale Stapelhöhe ist. Es kann auch notwendig sein, bestimmte Container nur in bestimmten Stapeln einer Bucht zu erlauben - z.B. Container mit Kühlaggregaten nur an den Außenseiten der Bucht, weil es nur dort geeignete Stromanschlüsse gibt. Dies kann soweit führen, dass man die Bucht in einen Buchtzustand bringen möchte, bei dem für jeden Container festgelegt ist, an welchem Platz (Stapel und Höhe) er sich am Ende zu befinden hat. Die Ergebnisse der Arbeit legen nahe, dass die heuristische Baumsuche auch für diese Problemvarianten ein vielversprechender Ansatz sein könnte.

Bei der zweiten Gruppe wird versucht, den Lösungsansatz in einem anderen Kontext anzuwenden. So ist der Einsatz der Baumsuche-Heuristik potenziell überall dort naheliegend, wo Objekte mit einer Stapelsemantik gelagert sind, z.B. in Lagerhäusern, beim Einlagern und Auslagern von Objekten in LKW oder Schiffen, oder bei der Unterbringung von Zügen in Depots (vgl. Caserta et al., 2011). Auch hier ist zu erwarten, dass die Anwendung einer an die in dieser Arbeit vorgestellten Verfahren angelehnten Baumsuche-Heuristik zu guten Ergebnissen führt.

Literaturverzeichnis

Akeb, H., Hifi, M. und Lazure, D., 2013: A heuristic based algorithm for the 2D circular strip packing problem. In: Fidanova, S. (Herausgeber), Recent advances in computational optimization, 73–92. Springer International Publishing, Cham.

Alba, E., Blum, C., Isasi, P., León, C. und Gómez, J. A. (Herausgeber), 2009: Optimization techniques for solving complex problems. John Wiley & Sons, New York.

Araya, I. und Riff, M. C., 2014: A beam search approach to the container loading problem. *Computers & Operations Research*, 43:100–107.

Baldacci, R., Christofides, N. und Mingozzi, A., 2008: An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115, 2:351–385.

Bazzazi, M., Safaei, N. und Javadian, N., 2009: A genetic algorithm to solve the storage space allocation problem in a container terminal. *Computers & Industrial Engineering*, 56, 1:44–52.

Beuthe, M., 2007: Intermodal freight transport in Europe. In: Leinbach, T. R. und Capineri, C. (Herausgeber), Globalized freight transport: intermodality, e-commerce, logistics and sustainability, 54–102. Edward Elgar Publishing, Cheltenham.

Bichler, K., Krohn, R. und Philippi, P. (Herausgeber), 2011: Gabler Kompaktlexikon Logistik. Gabler, Wiesbaden, 2. Auflage.

Bierwirth, C. und Meisel, F., 2010: A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202, 3:615–627.

Blum, C., 2005: Beam-ACO: hybridizing ant colony optimization with beam search: an application to open shop scheduling. *Computers & Operations Research*, 32, 6:1565–1591.

Blum, N., 2001: Theoretische Informatik: Eine anwendungsorientierte Einführung. Oldenbourg Wissenschaftsverlag, München, 2. Auflage.

Bohrer, P., 2005: Crane scheduling in container terminals. Diplomarbeit, Technische Universität Kaiserslautern, Fakultät für Mathematik.

Bortfeldt, A., 1995: Informierte Graphensuchverfahren und genetische Algorithmen zur Lösung von Containerbeladeproblemen. Dissertation, Freie Universität Berlin, Fakultät für Wirtschaftswissenschaften.

Bortfeldt, A., 2004: A heuristic for the container pre-marshalling problem. In: Bertram, V. und Armada, M. (Herausgeber), Proceedings of the third international conference on computer and IT applications in the maritime industries, 419–429.

Bortfeldt, A. und Forster, F., 2012: A tree search procedure for the container premarshalling problem. *European Journal of Operational Research*, 217, 3:531–540.

Brinkmann, B., 2005: Seehäfen – Planung und Entwurf. Springer, Berlin.

Bundeszentrale für politische Bildung, 2009: Globalisierung: Zahlen und Fakten. http://www.bpb.de/files/8ME2U0.pdf. Letzter Zugriff am 08.02.2014.

Cao, Z., Lee, D.-H. und Meng, Q., 2008: Deployment strategies of double-rail-mounted gantry crane systems for loading outbound containers in container terminals. *International Journal of Production Economics*, 115, 1:221–228.

Carlo, H. J. und Vis, I. F., 2008: Routing new types of stacking crane configurations at container terminals. In: Ellis, K. P., Meller, R. D., Ogle, M. K., Peters, B. A., Taylor, G. D. und Usher, J. S. (Herausgeber), Progress in material handling research, 55–70. Material Handling Institute, Charlotte.

Caserta, M., Schwarze, S. und Voß, S., 2009a: A new binary description of the blocks relocation problem and benefits in a look ahead heuristic. In: Evolutionary computation in combinatorial optimization proceedings 2009, 37–48.

Caserta, M., Schwarze, S. und Voß, S., 2011: Container rehandling at maritime container terminals. In: Böse, J. W. (Herausgeber), Handbook of terminal planning, 247–269. Springer, Berlin.

Caserta, M., Schwarze, S. und Voß, S., 2012: A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research*, 219, 1:96–104.

Caserta, M. und Voß, S., 2009a: A cooperative strategy for guiding the corridor method. In: Krasnogor, N., Melián-Batista, B., Moreno-Pérez, J. A., Moreno-Vega, J. M. und Pelta, D. A. (Herausgeber), Nature inspired cooperative strategies for optimization proceedings 2008, 273–286. Springer, Berlin.

Caserta, M. und Voß, S., 2009b: A corridor method-based algorithm for the premarshalling problem. In: Giacobini, M., Brabazon, A., Cagnoni, S., Caro, G. A., Ekárt, A., Esparcia-Alcázar, A. I., Farooq, M., Fink, A. und Machado, P. (Herausgeber), Applications of evolutionary computing proceedings 2009, 788–797. Springer, Berlin.

Caserta, M. und Voß, S., 2009c: Corridor selection and fine tuning for the corridor method. In: Stützle, T. (Herausgeber), Learning and intelligent optimization proceedings 2009, 163–175. Springer, Berlin.

Caserta, M., Voß, S. und Sniedovich, M., 2009b: Applying the corridor method to a blocks relocation problem. *OR Spectrum*, 33, 4:915–929.

Choe, R., Park, T., Oh, M. S., Kang, J. und Ryu, K. R., 2009: Generating a rehandling-free intra-block remarshaling plan for an automated container yard. *Journal of Intelligent Manufacturing*, 22, 4:201–217.

Colbourn, C. J. und Colbourn, M. J. (Herausgeber), 1985: Algorithms in combinatorial design theory. Elsevier Science, New York.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R. und Schrijver, A., 1998: Combinatorial optimization. John Wiley & Sons, New York.

Cudahy, B. J., 2006: Box boats: how container ships changed the world. Fordham University Press, New York.

Diaby, M., 2007: The traveling salesman problem: a linear programming formulation. *World Scientific and Engineering Academy and Society (WSEAS) Transactions on Mathematics*, 6, 6:745–754.

Domschke, W. und Scholl, A., 2006: Heuristische Verfahren. Technischer Bericht, Friedrich-Schiller-Universität Jena, Fakultät für Wirtschaftswissenschaften.

Edelkamp, S. und Schrödl, S., 2012: Heuristic search: theory and applications. Morgan Kaufmann, San Francisco.

Even, S., Itai, A. und Shamir, A., 1976: On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5, 4:691–703.

Falkenauer, E., 1997: Genetic algorithms and grouping problems. John Wiley & Sons, New York.

Fanslau, T. und Bortfeldt, A., 2010: A tree search algorithm for solving the container loading problem. *INFORMS Journal on Computing*, 22, 2:222–235.

Forster, F. und Bortfeldt, A., 2012a: A tree search heuristic for the container retrieval problem. In: Klatte, D., Lüthi, H.-J. und Schmedders, K. (Herausgeber), Operations research proceedings 2011, 257–262. Springer, Berlin.

Forster, F. und Bortfeldt, A., 2012b: A tree search procedure for the container relocation problem. *Computers & Operations Research*, 39, 2:299–309.

Froyland, G., Koch, T., Megow, N., Duane, E. und Wren, H., 2008: Optimizing the landside operation of a container terminal. *OR Spectrum*, 30, 1:53–75.

Gendreau, M. und Potvin, J.-Y., 2010: Tabu Search. In: Gendreau, M. und Potvin, J.-Y. (Herausgeber), Handbook of metaheuristics, 41–60. Springer, Berlin, 2. Auflage.

Gleißner, H. und Femerling, J. C., 2007: Logistik: Grundlagen – Übungen – Fallbeispiele. Gabler, Wiesbaden.

Goldreich, O., 2010: P, NP, and NP-completeness. Cambridge University Press, New York.

Göpfert, I. und Braun, D., 2008: Weltumspannende Güterflüsse und Logistikleistungen sowie Rahmenbedingungen einer globalen Logistik. In: Göpfert, I. und Braun, D. (Herausgeber), Internationale Logistik, 1–23. Gabler, Wiesbaden.

Gudehus, T., 2010: Logistik: Grundlagen – Strategien – Anwendungen. Springer, Berlin, 4. Auflage.

Günther, H.-O. und Kim, K. H., 2006: Container terminals and terminal operations. *OR Spectrum*, 28, 4:437–445.

Günther, H.-O. und Tempelmeier, H., 2012: Produktion und Logistik. Springer, Berlin, 9. Auflage.

Guo, X. und Huang, S. Y., 2008: Performing A* search for yard crane dispatching in container terminals. In: Tools with artificial intelligence proceedings 2008, 263–267. IEEE.

Han, Y., Lee, L. H., Chew, E. P. und Tan, K. C., 2008: A yard storage strategy for minimizing traffic congestion in a marine container transshipment hub. *OR Spectrum*, 30, 4:697–720.

Hansen, P. und Mladenovic, N., 2001: Variable neighborhood search: principles and applications. *European Journal of Operational Research*, 130, 3:449–467.

Hapag-Lloyd, 2010: Container specification. http://www.hapag-lloyd.com/downloads/press_and_media/publications/Brochure_Container_Specification_en.pdf. Letzter Zugriff am 08.02.2014.

Hart, P. E., Nilsson, N. J. und Raphael, B., 1968: A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, 4, 2:100–107.

Heinen, E., 1985: Industriebetriebslehre. Entscheidungen im Industriebetrieb. Gabler, Wiesbaden, 8. Auflage.

Huang, S.-H. und Lin, T.-H., 2012: Heuristic algorithms for container premarshalling problems. *Computers & Industrial Engineering*, 62, 1:13–20.

Hummels, D., 2007: Transportation costs and international trade in the second era of globalization. *Journal of Economic Perspectives*, 21, 3:131–154.

Imai, A., Nishimura, E. und Papadimitriou, S., 2001: The dynamic berth allocation problem for a container port. *Transportation Research Part B*, 37, 5:437–457.

Imai, A., Sun, X., Nishimura, E. und Papadimitriou, S., 2005: Berth allocation in a container port: using a continuous location space approach. *Transportation Research Part B*, 39, 3:199–221.

Jansen, K., 1998: The mutual exclusion scheduling problem for permutation and comparability graphs. In: Morvan, M., Meinel, C. und Krob, D. (Herausgeber), Symposium on theoretical aspects of computer science proceedings 1998, 287–297. Springer, Berlin.

Jin, J. G., Cao, J. X., Chen, J. H. und Lee, D.-H., 2011: A service-oriented model for the yard management problem in container terminals. In: Böse, J. W., Hu, H., Jahn, C., Shi, X., Stahlbock, R. und Voß, S. (Herausgeber), Computational logistics proceedings 2011, 233–242. Springer, Berlin.

Kim, K. H. und Bae, J. W., 1998: Re-marshaling export containers in port container terminals. *Computers & Industrial Engineering*, 35, 3-4:655–658.

Kim, K. H. und Hong, G.-P., 2006: A heuristic rule for relocating blocks. *Computers & Operations Research*, 33, 4:940–954.

Kim, K. H., Park, Y. M. und Jin, M. J., 2007: An optimal layout of container yards. *OR Spectrum*, 30, 4:675–695.

Korte, B. und Vygen, J., 2007: Combinatorial optimization: theory and algorithms. Springer, Berlin, 3. Auflage.

Le-Griffin, H. D. und Murphy, M., 2006: Container terminal productivity: experiences at the ports of Los Angeles and Long Beach. Technischer Bericht, METRANS Transportation Center, Los Angeles.

Lee, Y. und Chao, S.-L., 2009: A neighborhood search heuristic for pre-marshalling export containers. *European Journal of Operational Research*, 196, 2:468–574.

Lee, Y. und Hsu, N.-Y., 2007: An optimization model for the container pre-marshalling problem. *Computers & Operations Research*, 34, 11:3295–3313.

Lee, Y. und Lee, Y.-J., 2010: A heuristic for retrieving containers from a yard. *Computers & Operations Research*, 37, 6:1139–1147.

Lemper, B., 2009: Globalisation and container shipping demand - consequences of the financial crisis. In: Lemper, B. und Zachcial, M. (Herausgeber), Trends in container shipping / maritime logistics, 13–24. Peter Lang, Frankfurt am Main.

Leu, Y.-Y., Huang, P. und Russell, R., 1997: Using beam search techniques for sequencing mixed-model assembly lines. *Annals of Operations Research*, 70:379–397.

Levinson, M., 2006: The box: how the shipping container made the world smaller and the world economy bigger. Princeton University Press, Princeton.

Li, W., Goh, M., Wu, Y., Petering, M. E. H. und de Souza, R., 2012: A continuous time model for multiple yard crane scheduling with last minute job arrivals. *International Journal of Production Economics*, 136, 2:332–343.

Li, W., Wu, Y., Petering, M. E. H., Goh, M. und de Souza, R., 2009: Discrete time model and algorithms for container yard crane scheduling. *European Journal of Operational Research*, 198, 1:165–172.

Lim, S.-M., 1994: Economies of container ship size: a new evaluation. *Maritime Policy & Management*, 21, 2:149–160.

Linn, R., Liu, J. Y., Wan, Y. W., Zhang, C. und Murty, K. G., 2003: Rubber tired gantry crane deployment for container yard operation. *Computers & Industrial Engineering*, 45, 3:429–442.

Liu, Y., Kang, H.-g. und Zhou, P.-f., 2010: Fuzzy optimization of storage space allocation in a container terminal. *Journal of Shanghai Jiaotong University*, 15, 6:730–735.

Lowe, D., 2005: Intermodal freigh transport. Elsevier Butterwort-Heinemann, Oxford.

Mak, K. L. und Sun, D., 2009: Scheduling yard cranes in a container terminal using a new genetic approach. *Engineering Letters*, 17, 4:274–280.

Meersmans, P. J. M. und Dekker, R., 2001: Operations research supports container handling. Technischer Bericht, Erasmus Universität Rotterdam, Econometric Institute.

Meisel, F., 2009: Seaside operations planning in container terminals. Springer, Berlin.

Mensen, H., 2013: Handbuch der Luftfahrt. Springer, Berlin, 2. Auflage.

Mitchell, J. E., 2002: Branch-and-cut algorithms for combinatorial optimization problems. In: Pardalos, P. M. und Resende, M. G. C. (Herausgeber), Handbook of applied optimization, 65–77. Oxford University Press, Oxford.

Moccia, L., Cordeau, J.-F., Gaudioso, M. und Laporte, G., 2006: A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics*, 53, 1:45–59.

Murty, K. G., 2007: Yard crane pools and optimum layouts for storage yards of container terminals. *Journal of Industrial and Systems Engineering*, 1, 3:190–199.

Ng, W. C., 2005: Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research*, 164, 1:64–78.

Nikolaev, A. G. und Jacobson, S. H., 2010: Simulated annealing. In: Grendreau, M. und Potvin, J.-Y. (Herausgeber), Handbook of metaheuristics, 1–40. Springer, Berlin, 2. Auflage.

Obermaier, R., Müller, F. R. und Braun, H., 2007: Der Container als Artefakt eines Transportparadigmas: Akteure und Diffusionsphasen. In: Otto, A. und Obermaier, R. (Herausgeber), Logistikmanagement – Analyse, Bewertung und Gestaltung logistischer Systeme. Gabler Edition Wissenschaft, Wiesbaden.

Osman, I. H. und Kelly, J. P. (Herausgeber), 1996: Meta-heuristics: Theory and applications. Kluwer Academic Publishers, Dordrecht.

Papadimitriou, C. H. und Steiglitz, K., 1998: Combinatorial optimization: algorithms and complexity. Dover Publications, Mineola.

Petering, M. E. H., Wu, Y., Li, W., Goh, M. und de Souza, R., 2009: Development and simulation analysis of real-time yard crane control systems for seaport container transshipment terminals. *OR Spectrum*, 31, 4:801–835.

Pomberger, G. und Dobler, H., 2008: Algorithmen und Datenstrukturen: Eine systematische Einführung in die Programmierung. Pearson Education, Upper Saddle River.

Rackwitz, H., 1969: Der Container-Verkehr zur See – Eine Analyse organisatorischer und wirtschaftlicher Probleme unter besonderer Berücksichtigung der Bundesrepublik Deutschland. Magica, Meilen.

Rakrouki, M. A., Ladhari, T. und T'kindt, V., 2012: Coupling genetic local search and recovering beamsearch algorithms for minimizing the total completion time in the single machine scheduling problem subject to release dates. *Computers & Operations Research*, 39, 6:1257–1264.

Reeves, C. R., 2010: Genetic algorithms. In: Gendreau, M. und Potvin, J.-Y. (Herausgeber), Handbook of metaheuristics, 109–140. Springer, 2. Auflage.

Resende, M. G. und Riberio, C. C., 2010: Greedy randomized adaptive search procedure: advances, hybridizations, and applications. In: Gendreau, M. und Potvin, J.-Y. (Herausgeber), Handbook of metaheuristics, 283–320. Springer, 2. Auflage.

Sibbesen, L. K., 2008: Mathematical models and heuristic solutions for container positioning problems in port terminals. Dissertation, Technische Universität von Dänemark, Fakultät für Management Engineering.

Speer, U., John, G. und Fischer, K., 2011: Scheduling yard cranes considering crane interference. In: Böse, J. W., Hu, H., Jahn, C., Shi, X., Stahlbock, R. und Voß, S. (Herausgeber), Computational logistics proceedings 2011, 321–340. Springer, Berlin.

Stahlbock, R. und Voß, S., 2008: Operations research at container terminals: a literature update. *OR Spectrum*, 30, 1:1–52.

Steenken, D., Voß, S. und Stahlbock, R., 2004: Container terminal operation and operations research - a classification and literature review. *OR Spectrum*, 26, 1:3–49.

Stopford, M., 2009: Maritime economics. Routledge, London, 3. Auflage.

Streim, H., 1975: Heuristische Lösungsverfahren. Versuch einer Begriffsklärung. *Zeitschrift für Operation Research*, 19, 5:143–162.

The World Bank, 2014: Container port traffic. http://data.worldbank.org/indicator/IS.SHP.GOOD.TU. Letzter Zugriff am 08.02.2014.

UNCTAD, 2006: Review of maritime transport 2006. http://unctad.org/en/docs/rmt2006_en.pdf. Letzter Zugriff am 27.02.2014.

UNCTAD, 2008: Review of maritime transport 2008. http://unctad.org/en/docs/rmt2008_en.pdf. Letzter Zugriff am 08.02.2014.

UNCTAD, 2010: Review of maritime transport 2010. http://unctad.org/en/docs/rmt2010_en.pdf. Letzter Zugriff am 08.02.2014.

UNCTAD, 2012: Review of maritime transport 2012. http://unctad.org/en/PublicationsLibrary/rmt2012_en.pdf. Letzter Zugriff am 08.02.2014.

Vacca, I., Bierlaire, M. und Salani, M., 2007: Optimization at container terminals: status, trends and perspectives. In: Proceedings of the swiss transport research conference, 1–21.

Vahrenkamp, R., 2011: Der Aufstieg der Logistik in der Massenkonsumgesellschaft. Campus, Frankfurt am Main.

Vis, I. F. A. und Carlo, H. J., 2010: Sequencing two cooperating automated stacking cranes in a container terminal. *Journal of Transportation Science*, 44, 2:169–182.

Vis, I. F. A. und Roodbergen, K. J., 2009: Scheduling of container storage and retrieval. *Operations Research*, 57, 2:456–467.

Wäscher, G., 1998: Logistik. In: Berndt, R., Altobelli, C. F. und Schuster, P. (Herausgeber), Springers Handbuch der Betriebswirtschaftslehre (Band 1), 421–468. Springer, Berlin.

Wegener, I., 2005: Complexity theory: exploring the limits of efficient algorithms. Springer, Berlin.

Wolsey, L. A., 1998: Integer programming. Wiley-Interscience, New York.

Won, S. H., Zhang, X. und Kim, K. H., 2011: Workload-based yard-planning system in container terminals. *Journal of Intelligent Manufacturing*, 23, 6:2193–2206.

World Trade Organization, 2011: WTO international trade and market access data. http://www.wto.org/english/res_e/statis_e/statis_e.htm. Letzter Zugriff am 08.02.2014.

Yeo, G.-T., Roe, M. und Dinwoodie, J., 2008: Evaluating the competitiveness of container ports in Korea and China. *Transportation Research Part A: Policy and Practice*, 42, 6:910–921.

Zhang, C., Liu, J. Y., Wan, Y. W., Murty, K. G. und Linn, R., 2003: Storage space allocation in container terminals. *Transportation Research Part B: Methodological*, 37, 10:883–903.

Zhang, C., Wan, Y. W., Liu, J. Y. und Linn, R., 2002: Dynamic crane deployment in container storage yards. *Transportation Research Part B: Methodological*, 36, 6:537–555.

Zhang, H. und Kim, K. H., 2009: Maximizing the number of dual-cycle operations of quay cranes in container terminals. *Computers & Industrial Engineering*, 56, 3:979–992.

Zhu, W. und Lim, A., 2012: A new iterative-doubling greedy–lookahead algorithm for the single container loading problem. *European Journal of Operational Research*, 222, 3:408–417.