Containerumlade- und Stapelprobleme
Version 1.2

Peer Giemsch[1]
Institut für Anwendungen des Operations Research

14. Juli 2004

[1]Tel. +49-(0)721-6098-4189 - peer.giemsch@andor.wiwi.uni-karlsruhe.de

Inhaltsverzeichnis

Kapitel 1

Einfache Stapel

Es sei $G = (V, A)$ ein (gerichteter) Graph mit V Menge der Ecken, A Menge der Kanten. Es ist $A \subset V \times V$.
Ein gerichteter Graph $G = (V, A)$ heißt genau dann

Pfad der Länge $n-1, n \geq 1$, wenn $V = \{v_1, \ldots, v_n\}$ und $A = \{(v_i, v_{i+1}) : i \in \{1, \ldots, n-1\}\}$ ist.

Kreis der Länge $n, n \geq 3$, wenn $V = \{v_1, \ldots, v_n\}$ und $A = \{(v_i, v_{i+1}) : i \in \{1, \ldots, n-1\}\} \cup \{(v_1, v_n)\}$ ist.

Man sagt, daß ein Graph kreisfrei ist, wenn er keinen Kreis als Untergraph enthält. Man nennt einen Graph $G = (V, A)$ schlingenfrei, wenn $(v_i, v_i) \notin A$ für alle $v_i \in V$. Ein Graph heißt endlich, wenn $|V| < \infty$.

Definition 1 ([14]) *Ein Graph $G = (V, A)$ heißt genau dann ein (verzweigter) **Stapel**, wenn G endlich, schlingenfrei, gerichtet und kreisfrei ist.*

Definition 2 *Ein Stapel $G = (V, A)$ mit $V = \{v_1, \ldots, v_n\}$ heißt genau dann **einfach**, wenn $A = \{(v_i, v_{i+1}) : i \in \{1, \ldots, n-1\}\}$ ist.*

Bemerkung: Ein einfacher Stapel ist ein Pfad.

Beispiel: **Türme von Hanoi**
Gegeben sei ein „Turm“ bestehend aus n aufsteigend sortierter Scheiben paarweise verschiedender Größe. Der Turm soll auf eine anderen Stapelplatz umgesetzt werden. Dabei sind die folgenden Regeln einzuhalten:

- Es darf immer nur eine Scheibe bewegt werden.
- Eine größere Scheibe darf auf keiner kleineren Scheibe liegen - die sog. göttliche Regel („divine rule“).
- Es gibt nur einen Hilfsstapelplatz.

Gesucht ist eine minimale Anzahl von „Zügen“ oder Bewegungen der Scheiben.
Offensichtlich ist der Turm zu Beginn ein einfacher Stapel und jede Konfiguration nach einem zulässigen Zug ist ein 3-Tupel von einfachen Stapeln.

Definition 3 *Ein Knoten v_i eines Stapels $G = (V, A)$ heisst **blockiert**, wenn es ein $v_j \in V$ gibt mit $(v_j, v_i) \in A$ gibt. v_j blockiert in diesem Fall v_i.*

Definition 4 *Es sei $G = (V, A)$ ein Stapel und $\omega : V \to \mathbb{Z}$. Ein Knoten v heisst **überstaut**, wenn v blockiert ist und auf einem Pfad $\{(v_1, v_2), \ldots, (v_i, v)\} \subset A$ existiert ein Knoten v_j mit $\omega(v_j) > \omega(v)$.*

Bemerkung: ω ist die Ausladeabbildung, d.h. sie ordnet jedem Knoten einen Zeitpunkt zu, zudem es aus dem Stapel entfernt werden soll.

Definition 5 *Ein Stapel heißt **geordnet**, wenn es keine überstauten Knoten gibt.*

Beispiel: Sei $G = (V, A)$ ein einfacher Stapel mit Bezeichnungen wie in Definition 2. Sei nun zum Beispiel

- $\omega : v_i \mapsto i$, dann ist G geordnet.
- $\omega : v_i \mapsto n + 1 - i$, dann ist G nicht geordnet.
- $\omega : v_i \mapsto 5$, dann ist G geordnet.

Definition 6 *Die **symmetrische Differenz** $A \triangle B$ zweier Mengen $A, B \subset \Omega$ ist $A \triangle B := \{x \in \Omega : x \in A \,\bar{\vee}\, x \in B\}$.*

Eine „Umstapelung“ stellt einen Übergang von einem Stapel zu einem anderen dar. Dabei sollen alle Stapelelemente erhalten bleiben und nur deren Stellung zueinander ändert sich. Dies wollen wir nun formal beschreiben in der

Definition 7 *Ein Stapel $G_2 = (V, A_2)$ geht durch eine **Umstapelung** aus einem anderen Stapel $G_1 = (V, A_1)$ hervor, wenn die folgenden Eigenschaften auf G_1 und G_2 zutreffen:*

1. *Es existiert ein $v :\in V$, das weder in G_1 noch in G_2 blockiert ist.*
2. *Es existiert ein $u \in V$ mit $(v, u) \in A_1$ und $(v, u) \notin A_2$ bzw. $(v, u) \notin A_1$ und $(v, u) \in A_2$.*
3. *Es ist $(A_1 \triangle A_2) \backslash \{(v, u)\} \subset \{(v, x) : x \in V \backslash \{u\}\}$.*

Die 1. Bedingung gibt an, dass es einen Knoten gibt der an der Spitze der Stapel steht und somit umgesetzt werden darf. 2. gibt an, daß mindestens ein Knoten versetzt wird und 3., dass sich die Stellung der nicht bewegten Stapelelemente zueinander unverändert bleibt. Ein Beispiel ist in Abb.1.1 bzw. Abb.1.2 dargestellt.

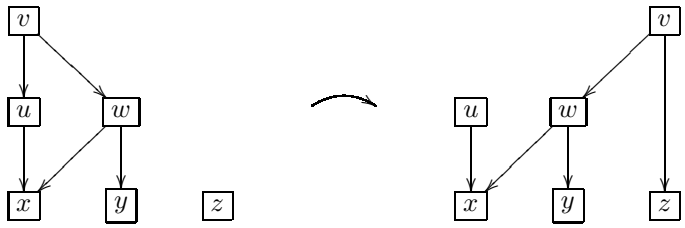


Abbildung 1.1: Umstapelung

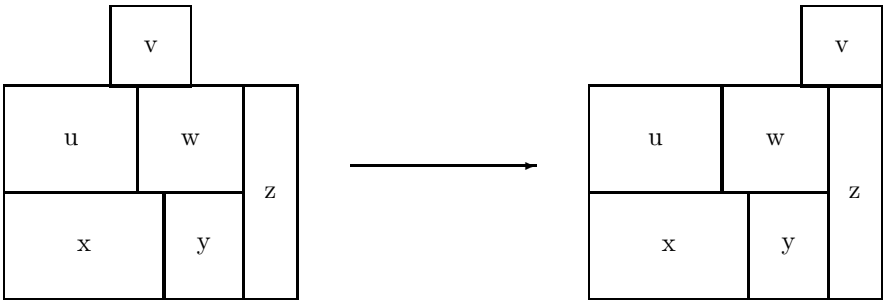


Abbildung 1.2: Interpretation der Umstapelung

Definition 8 *Ein Stapel $G_2 = (V_2, A_2)$ geht aus einem Stapel $G_1 = (V_1, A_1)$ durch **Einstapelung** eines Knotens v hervor, wenn gilt:*

1. *G_1 ist ein induzierter Subgraph von G_2,*
2. *$V_2 = V_1 \cup \{v\}$*
3. *$A_2 \subset (A_1 \cup \{(v, u) : u \in V_1\})$.*

Sprechweise: v wird in G_1 eingestapelt.
Wenn alle $u \in V_2$ mit $(v, u) \in A_2$ in G_1 nicht blockiert waren, dann sagt man: „v wurde obenauf eingestapelt“.

Bemerkung 1 Der eingestapelte Knoten v kann in G_2 auch isoliert liegen.

Definition 9 *Ein Stapel $G_2 = (V_2, A_2)$ geht aus einem Stapel $G_1 = (V_1, A_1)$ durch **Ausstapelung** eines Knotens v hervor, wenn gilt:*

1. *v ist in G_1 nicht blockiert*
2. *$V_2 = V_1 \backslash \{v\}$*
3. *$A_2 = A_1 \backslash \{(v, u) : u \in V_1\}$.*

Sprechweise: v wird aus G_1 ausgestapelt.

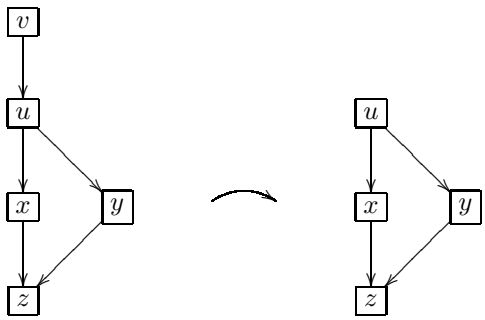


Abbildung 1.3: Eine Ausstapelung

Gegeben sei eine Menge $V = \{v_{(1)}, \ldots, v_{(n)}\}$ von Knoten und eine Ausladeabbildung $\omega : V \to \mathbb{Z}$.

Gesucht: Eine Folge von einfachen Stapeln $(G_i)_{i=0,\ldots,n}$ mit $G_0 = (\emptyset, \emptyset)$ und G_{i+1} entsteht aus G_i durch -nach möglicher Umnummerierung - sukzessive Einstapelung der Knoten aus V. Der Stapel G_n soll geordnet sein. Dies wird durch folgenden Algorithmus gelöst:

Prozedur SORT_V {
$i = 1, \mathcal{V}_() = \{v_{(1)}, \ldots, v_{(n)}\}, \mathcal{V} = \emptyset$
Solange $(\mathcal{V}_() \neq \emptyset)${
$v_{(j)} :\in (\max \omega)\mathcal{V}_()$
$v_i := v_{(j)}, \mathcal{V}_() = \mathcal{V}_()\backslash\{v_{(j)}\}, \mathcal{V} = \mathcal{V} \cup \{v_j\}$
} }
Prozedur EINSTAPEL{
$G_0 = (\emptyset, \emptyset), G_1 = (\{v_1\}, \emptyset), i = 1$
Solange $(\mathcal{V} \neq \emptyset)${
$G_{i+1} = (V_i \cup \{v_{i+1}\}, A_i \cup \{(v_{i+1}, v_i)\})$
$\mathcal{V} = \mathcal{V}\backslash\{v_{i+1}\}$
}
}

Bemerkung: Die obige Fragestellung lässt sich folgendermaßen interpretieren:
Ein Fahrzeug wird an einem Depot für eine feststehende Rundreise beladen, und die zu ladenden Kisten lassen sich jeweils nur abladen, wenn sie von der Ladeluke des Fahrzeugs aus direkt entnommen werden können. Die Abmessungen der Kisten und des Laderaums seien derart, dass jeweils nur eine Kiste direkt entnommen werden kann und zwar die „hinterste“ Kiste. Wie soll nun das Fahrzeug beladen werden?

Bemerkung: Der Algorithmus erzeugt offensichtlich einen Ladeplan nach dem Prinzig der umgekehrten Reihenfolge (Last in - First out).

Übungsaufgabe 1 Geben Sie einen Algorithmus zur Lösung des klassischen „Türme von Hanoi“-Problems an. Einen solchen finden Sie in vielen einführenden Büchern der Informatik im Abschnitt über Rekursion. Verwenden Sie einen geeigneten Pseudocode und Schreibweisen aus der Vorlesung.

Lösung 1 Die rekursive Lösung[1]:

1. Schaffe den Turm aus $n-1$ Steinen vom Startplatz zum Hilfsplatz (dafür ist der ursprüngliche Zielplatz der aktuelle Hilfplatz).
2. Lege den größten Stein vom Startplatz auf den Zielplatz.
3. Schaffe den Turm aus $n-1$ Steinen vom Hilfsplatz zum Zielplatz (dafür ist der ursprüngliche Startplatz der neue Hilfsplatz).

Prozedur HANOI(n, Start, Ziel, Hilf){
Falls $n > 1$
Dann HANOI($n-1$, Start, Hilf, Ziel);
Stein von Start nach Ziel;
HANOI($n-1$, Hilf, Ziel, Start)
Sonst Stein von Start nach Ziel
}

[1] http://wwwicg.informatik.uni-rostock.de/Lehre/Informatik1/WS2003-04/Skript/Inf6neu.pdf

Definition 10 *Ein Tupel (V, α, ω) mit V Menge von Knoten und zwei Abbildungen $\omega, \alpha : V \to \mathbb{Z}$ heißt **Verschiffungsmenge** wenn $\alpha(v) < \omega(v)$ für alle $v \in V$.*
*$\alpha(v)$ heißt dann der **Einladepunkt** von v und $\omega(v)$ der **Ausladepunkt** von v.*

Bestimmte Teilmengen von Verschiffungsmengen werden ausgezeichnet:

Definition 11 *Sei (V, α, ω) eine Verschiffungsmenge. Eine **Gruppe** $V_{(i,j)}$ von Knoten bezeichne eine Teilmenge von V mit gleichem Einladepunkt i und Ausladepunkt j.*
*Als **Einlademenge** $V_{\alpha=t}$ am Punkt t bezeichne man eine Teilmenge von V mit gleichem Einladepunkt und als **Ausladem enge** $V_{\omega=t}$ am Punkt t bezeichne man eine Teilmenge von V mit gleichem Ausladepunkt. Es sei $V_t := \{v \in V : \omega(v) \geq t, \alpha(v) \leq t\}$.*

Definition 12 (OSOP) *Gegeben sei eine Verschiffungsmenge (V, α, ω). Es sei $\bar{\alpha} = \min\{\alpha(v) : v \in V\}$ und $\bar{\omega} := \max\{\omega(v) : v \in V\}$ Man bezeichnet eine Folge $(G_{t_i} = (V_{t_i}, A_{t_i}))$ mit $t = \bar{\alpha}, \ldots, \bar{\omega}$ und $i = 1, \ldots, N_t$ von einfachen Stapeln als eine **eindimensionale Umstapelfolge mit einem Zugriffspunkt**, wenn*

1. *$G_{\bar{\alpha}_1} = \emptyset = G_{\bar{\omega}_{N_{\bar{\omega}}}}$,*
2. *G_{j+1} entsteht aus G_j entweder durch Einstapelung oder Ausstapelung von Knoten aus V,*
3. *$V_{t_i} \subset V_t := \{v \in V : \omega(v) \geq t, \alpha(v) \leq t\}$ und*
4. *$V_{t_{|I_t|}} = \{v \in V : \omega(v) > t, \alpha(v) \leq t\}$,*
5. *jedes Element aus V wird mindestens einmal eingestapelt und mindestens einmal ausgestapelt.*

*Gesucht ist nun eine Umstapelfolge **minimaler Länge**, d.h. kleinstem $|I| = \sum_{t=\bar{\alpha}}^{\bar{\omega}} |N_t|$.*

Beispiel: Gegeben sei die Verschiffungsmenge (V, α, ω) mit $V = \{u, v, w\}$ und $\alpha(u) = 1, \alpha(v) = \alpha(w) = 2$ und $\omega(u) = 4, \omega(v) = 5, \omega(w) = 3$. Eine eindimensionale Umstapelfolge mit einem Zugriffspunkt ist in der Abbildung unten zu sehen. Ist dies eine Umstapelfolge minimaler Länge?

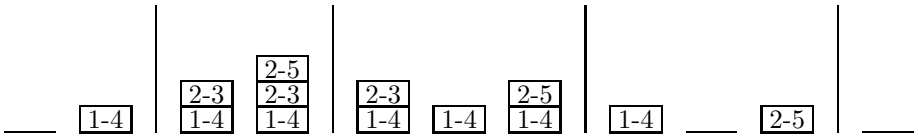


Abbildung 1.4: Eine Umstapelfolge

Es ist $|I| = 2 + 2 + 3 + 3 + 1 = 11$. Wie man leicht sieht ist 9 das Minimum.

Hinweis: Die Bedingung $t = \bar{\alpha}, \ldots, \bar{\omega}$ ist geeignet zu interpretieren. Es werden alle Ausladeund Einladezeitpunkte der Verschiffungsmenge ihrer Ordnung nach als Werte von t angenommen.

Bemerkung: (OSOP) kann als ein Modell für ein Containerschiff verwendet werden , das eine Reise von Hafen 0 bis zum Hafen $N + 1$ macht und die zu transportierenden Container in einem einzigen Turm aufeinander gestellt werden. In Hafen 0 ist das Schiff zu Beginn leer und in Hafen $N + 1$ wird es komplett gelöscht. An jedem Hafen dazwischen werden Container

ab- und aufgeladen gemäß ihrer Zielhäfen und der Zugriffsmöglichkeiten auf überstaute Container.

Eine erste Heuristik zur Erzeugung einer Umstapelfolge sieht nun beispielsweise so aus: Gegeben sei eine bzgl. α und ω lexikographisch sortierte Verschiffungsmenge V. Die Einladezeitpunkte bzw. Ausladezeitpunkte seien geeignet nummeriert von 0 bis N bzw. 1 bis $N+1$.

Die Heuristik:

Prozedur UMSTAPELFOLGE_1 {
$G_0 = (\emptyset, \emptyset), t = 0, i = 0,$
Solange($V \neq \emptyset$) {
Solange ($V_{\alpha=t} \neq \emptyset$) {
$v \in (\max \omega) V_{\alpha=t}$;
G_{i+1} := Einstapelung von v in G_i;
$V_{\alpha=t} = V_{\alpha=t} \backslash \{v\}, V = V - \{v\}, i = i+1$
}
$t = t + 1$;
Solange($V_{\omega=t} \neq \emptyset$) {
$v \in G_i$ nicht blockiert;
G_{i+1} := Ausstapelung von v aus G_i;
$i = i + 1$;
Wenn($\omega(v) = t$){
$V_{\omega=t} = V_{\omega=t} \backslash \{v\}, V = V \backslash \{v\}\}$
Sonst { $\alpha(v) := t; V = V \cup \{v\}\}$
}
}

Erzeugt die Heuristik eine Umstapelfolge minimaler Länge? Gibt es untere und obere Schranken für die Länge der erzeugten Folge? Welche Komplexität hat das Verfahren?

Hier ein Beispiel:
Die Verschiffungsmenge sei gegeben durch $V = \{(1-3), (2-5), (2-5)', (2-6)\}$, wobei ein Knoten benannt ist nach dem Muster „Einladezeitpunkt-Ausladezeitpunkt“. Der Algorithmus UMSTAPELFOLGE_1 liefert das Bild 1.5.

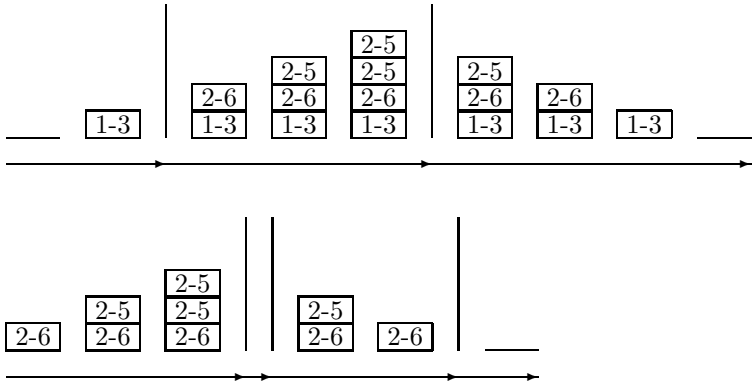


Abbildung 1.5: Output des Algorithmus

Wie sieht eine Umstapelfolge minimaler Länge aus? In Abb.1.6 ist eine zu sehen.

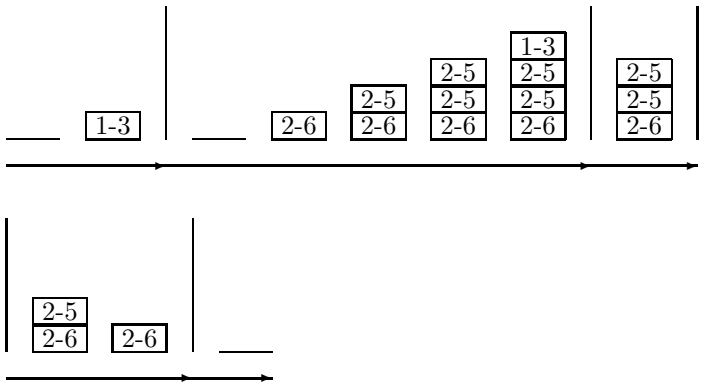


Abbildung 1.6: Minimale Umstapelfolge

Definition 13 *Gegeben sei eine Umstapelfolge* $(G_{t_i} = (V_{t_i}, A_{t_i}))$ *der Verschiffungsmenge* (V, α, ω). *Man nennt die Stapel* $G_{t_{N_t}}$ *den* ***Stauplan*** *am Punkt* t. *Die Folge* $(G_{t_{N_t}})_{t=\bar{\alpha},\ldots,\bar{\omega}}$ *heißt der* ***Stauplan*** *der Verschiffungsmenge.*

O.B.d.A. sei im folgenden $\bar{\alpha} = 0, \bar{\omega} = m + 1$ und $t = 1, 2, \ldots, m, m + 1$. Durch geeignete Umnummerierung kann dies immer erreicht werden.
Weiterhin sagt man: „Knoten $v_1, \ldots, v_n$ bilden einen Pfad“ in einem einfachen Stapel $G = (V, A)$, wenn $\{(v_i, v_{i+1})\} \subset A$ für $i = 1, \ldots, n - 1$.

Fakt 1 *Eine untere Schranke für die Länge einer (OSOP) ist* $2|V| + 1$.

Dies ist eine triviale Feststellung. Eine Stapelfolge, in der jeder Knoten genau einmal eingestapelt und genau einmal ausgestapelt wird, hat die Länge $2|V| + 1$.
Eine obere Schranke gibt es für eine (OSOP) nicht, da jeder Knoten beliebig oft ein- und ausgestapelt werden kann. Interessant ist es aber, wenn jeder Knoten an jedem Punkt höchstens einmal eingestapelt und einmal ausgestapelt werden soll. Das ist mit Sicherheit auch ein sinnvolle Annahme.

Fakt 2 *Eine obere Schranke für die Länge einer (OSOP) bei der jeder Knoten an einem Punkt höchstens einmal eingestapelt und einmal ausgestapelt wird ist* $2|V|(m - 1) + 1$.

Dies ist eine sehr grobe obere Schranke, da angenommen wird, dass jeder Knoten an jedem Punkt zwischen 1 und m bewegt wird.

Fakt 3 *Im Stauplan der Verschiffungsmenge* (V, α, ω) *einer minimalen Umstapelfolge bilden die Knoten* $V_{(i,j)}$ *einen Pfad.*

Das heißt die Knoten aus $V_{(i,j)}$ liegen aufeinander und es kann die Umstapelfolge nicht verkürzen, wenn andere Knoten zwischen ihnen eingestapelt werden.

Wie könnte ein Beweis für Fakt 3 aussehen? Die Aussage meint, dass alle Knoten aus $V_{(i,j)}$ in einer minimalen (OSOP) gleich behandelt werden. Was wäre nun, wenn ein Knoten von $V_{(i,j)}$ nicht im Pfad eingestapelt wird? Wird die Folge dadurch länger?

Zeigen Sie den Fakt 3, d.h. dass alle Knoten aus $V_{(i,j)}$ in einer minimalen (OSOP) gleich behandelt werden.

Wie man in obigem Beispiel gesehen hat, erzeugt UMSTAPELFOLGE_1 keine minimale Umstapelfolge. Man kann sehen, daß es zweckmäßig sein kann an Einladepunkten bis zu einer gewissen „Tiefe“ auszustapeln um dann einen geordneten Stapel zu bilden. UMSTAPELFOLGE_1 kann als die Strategie angesehen werden, an keinem Einladepunkt umzusortieren, sondern nur blockierende Knoten der überstauten Knoten auszuladen und dann zusammen mit den einzustapelnden Knoten geordnet wieder einzustapeln. Um ein Maß für Umstauer zu haben nun die

Definition 14 *Ein Stauplan an Punkt t heisst $k-$**geordnet**, wenn kein Knoten v mit Ausladepunkt $\omega(v) \leq k$ überstaut ist.*

Um einen $k-$geordneten Stauplan $G_{t_{N_t}}$ an einem Punkt t aus einem Stapel G_{t_i} zu erhalten, müssen alle in G_{t_i} überstauten Knoten v mit $\omega(v) = k$ aus G_{t_i} ausgestapelt werden. Die überstauenden Knoten werden dabei auch ausgeladen. Die Menge der nun ausgestapelten Knoten und $\bigcup_{i=t+1}^{m+1} V_{(i,j)}$ werden nun nach absteigendem Ausladepunkt sortiert eingestapelt. Dieses $k-$Ordnen verlängert die Stapelteilfolge am Punkt t, verkürzt aber auch die Stapelteilfolge an zukünftigen Punkten.
Es gilt nun der folgende

Fakt 4 (Aslidis 1990) *Die minimale eindimensionale Umstapelfolge mit einem Zugriffspunkt kann gefunden werden durch einen Vektor $(P(1), \ldots, P(m)) \in \{1, \ldots, m\}^m$ mit $P(i) \geq i$ und $P(i) = k$ gibt an, daß am Punkt i ein $k-$geordneter Stauplan vorliegt. Dieser Vektor heisse der optimale Ordnungsvektor.*

Was ist mit „gefunden werden“ gemeint? Wie ist der Vektor $(P(1), \ldots, P(m))$ anzuwenden? Eine Umstapelfolge wird wie folgt erzeugt: Erreicht die Stapelfolge den nächsten Punkt werden zunächst alle Knoten, die dort ihren Ausladepunkt haben, zusammen mit den sie blockierenden Knoten ausgeladen Die $k-$geordneten Staupläne werden danach durch Ausstapelung und Einstaplungen gebildet.

Korollar 1 *Es gilt: $\forall(i,j) : i < j$ und $P(i) \geq j \Rightarrow P(i) \geq P(j)$*

Wie berechnet man nun aber die $P(i)$ für ein (OSOP)? Dazu wird ein rekursiver Algorithmus verwendet, der zunächst Teilprobleme des (OSOP) löst. Es sei $P(i,j)$ das (OSOP) mit der Verschiffungsmenge, die dardurch entsteht, dass alle Knoten mit Einladepunkt $\leq i$ und Ausladepunkt $\geq j$ den Einladepunkt i und den Ausladepunkt j erhalten, und die anderen Knoten unverändert bleiben . Das ist ein Subproblem des ursprünglichen (OSOP). Seien $V(i,j)$ die Zahl der zusätzlich ein- und ausgestapelten Knoten in $P(i,j)$. Diese Zahl ensteht durch übestauende Knoten und das k-Ordnen der Stapel. Ist $V(i,j) = s$, so verlängert sich die Umstapelfolge um $2s$.
$r(i,k,j)$ sei die Zahl der aus- und wieder eingestapelten Knoten (Umordnungskosten) an Punkt k für Knoten der Ausladepunkte $k+1, \ldots, j-1$ in $P(i,j)$ unter der Annahme, dass keine Umordnungen - außer den notwendigen - für Knoten mit Ausladepunkt $i+1, \ldots, k-1$ vorgenommen wurden und der Stauplan bei k dann $(j-1)$-geordnet ist. Dann gilt:

Satz 1 (Aslidis 1990) *Die minimalen Kosten der Überstauungen und der optimale $k-$ Ordnungsvektor einer (OSOP) kann in $\mathcal{O}(n^3)$ durch Lösen der folgenden Rekursion gefunden werden:*

$$V(i,j) = \min_{i+1 \leq k \leq j-1} \{V(i,k) + r(i,k,j) + V(k,j)\} \qquad i = 0,1,\ldots,m+1 \tag{1.1}$$

$$V(i,i+1) = V(i,i+2) = 0 \qquad i = 0,1,\ldots,m+1 \tag{1.2}$$

Wenn nun $|V_{(i,j)}| \neq 0$ für alle (i,j), dann berechnet sich $r(i,k,j)$ (siehe Abb.1.7) so:

$$r(i,k,j) = \sum_{q=i+1}^{k-1} \sum_{l=k+1}^{m+1} |V_{(q,l)}| + \sum_{p=0}^{i} \sum_{n=k+1}^{j-1} |V_{(p,n)}|$$

Gibt es jedoch auch Ein- und Ausladepaare ohne zugehörige Knoten, so müssen einige

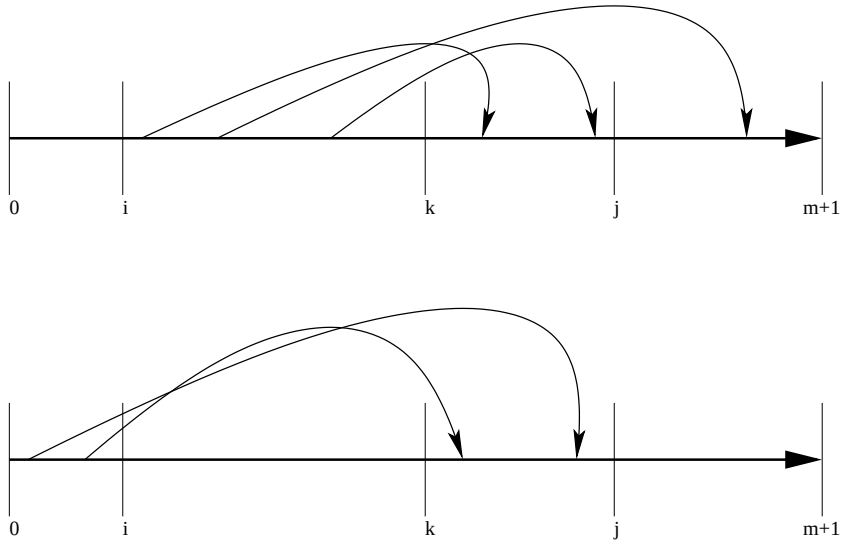


Abbildung 1.7: Die zwei Summenteile von $r(i,k,j)$

redundante Umordnungskosten abgezogen weren, die zuviel gezählt wurden. Für weitere Hinweise siehe Abschnitt 3.7 in [1] bzw. in Übungsaufgabe 11.
Wie bekommt man nun die $P(i)$ aus den berechneten Werten? Man speichert für ein $V(i,j)$ das entsprechende optimale k, also $h(i,j) = k$. Man kriegt den optimalen Ordnungsvektor so:

Sei $S = \{V(0,M+1)\}$
  Solange $(S \neq \emptyset)\{$
    Wähle $V(i,j) \in S$
    $S := S \backslash \{(V(i,j)\}$
    $S := S \cup \{V(i,h(i,j)), V(h(i,j),j)\}$
    $P(h(i,j)) = j-1$
    $\}$

Ein numerisches Beispiel:

Es sei

$$C = (V_{(i,j)})_{i=0,\dots,4}^{j=1,\dots,5} = \begin{pmatrix} 5 & & & & \\ 6 & 7 & & & \\ 2 & 3 & 4 & & \\ 7 & 1 & 2 & 6 & \\ 3 & 6 & 11 & 2 & 4 \end{pmatrix}$$

Es ist

$$\begin{aligned} h(0,2) = 1, h(1,3) = 2, h(2,4) = 3, h(3,5) = 4 \qquad &\text{Warum?} \\ 0 = V(0,1) = V(0,2) = V(1,2) = V(1,3) = V(2,3) & \\ 0 = V(2,4) = V(3,4) = V(3,5) = V(4,5) & \end{aligned}$$

$$\begin{aligned} r(0,1,2) &= 0 & r(0,1,3) &= 6 \\ r(0,2,3) &= 3+1+6 = 10 & r(0,1,4) &= 6+2 = 8 \\ r(0,2,4) &= 3+1+6+2 = 12 & r(0,3,4) &= 1+6+2+11 = 20 \\ r(0,1,5) &= 6+2+7 = 15 & r(0,2,5) &= 3+1+6+2+7 = 19 \\ r(0,3,5) &= 1+6+2+11+7 = 27 & r(0,4,5) &= 6+11+2 = 19 \\ r(1,2,3) &= 0 & r(1,2,4) &= 2+3 = 5 \\ r(1,3,4) &= 2+11 = 13 & r(1,2,5) &= 2+3+7+1 = 13 \\ r(1,3,5) &= 2+11+7+1 = 21 & r(1,4,5) &= 11+12 = 13 \\ r(2,3,4) &= 0 & r(2,3,5) &= 7+1+2 = 10 \\ r(2,4,5) &= 2 & r(3,4,5) &= 0 \end{aligned}$$

$$\begin{aligned} V(0,3) &= \min \begin{cases} V(0,1) + r(0,1,3) + V(1,3) = 6 \\ V(0,2) + r(0,2,3) + V(2,3) = 10 \end{cases} \\ &h(0,3) = 1, V(0,3) = 6 \\ V(1,4) &= \min \begin{cases} V(1,2) + r(1,2,4) + V(2,4) = 5 \\ V(1,3) + r(1,3,4) + V(3,4) = 13 \end{cases} \\ &h(1,4) = 2, V(1,4) = 5 \end{aligned}$$

$$\begin{aligned} V(2,5) &= \min \begin{cases} V(2,3) + r(2,3,5) + V(3,5) = 10 \\ V(2,4) + r(2,4,5) + V(4,5) = 2 \end{cases} \\ &h(2,5) = 4, V(2,5) = 2 \\ V(0,4) &= \min \begin{cases} V(0,1) + r(0,1,4) + V(1,4) = 13 \\ V(0,2) + r(0,2,4) + V(2,4) = 12 \\ V(0,3) + r(0,3,4) + V(3,4) = 26 \end{cases} \\ &h(0,4) = 1, V(0,4) = 12 \end{aligned}$$

$$V(1,5) = \min \begin{cases} V(1,2) + r(1,2,5) + V(2,5) = 15 \\ V(1,3) + r(1,3,5) + V(3,5) = 21 \\ V(1,4) + r(1,4,5) + V(4,5) = 18 \end{cases}$$

$$h(1,5) = 2, V(1,5) = 15$$

$$V(0,5) = \min \begin{cases} V(0,1) + r(0,1,5) + V(1,5) = 30 \\ V(0,2) + r(1,2,5) + V(2,5) = 21 \\ V(0,3) + r(0,3,5) + V(3,5) = 33 \\ V(0,4) + r(0,4,5) + V(4,5) = 31 \end{cases}$$

$$h(0,5) = 2, V(0,5) = 21 = \text{optimaler Wert}$$

Der optimale Ordnungsvektor $(P(i))$ lautet nun:

$$P(h(0,5)) = 5 - 1, \Rightarrow P(2) = 4$$
$$P(h(0,2)) = 2 - 1, \Rightarrow P(1) = 1$$
$$P(h(2,5)) = 5 - 1, \Rightarrow P(4) = 4$$
$$P(h(2,4)) = 4 - 1, \Rightarrow P(3) = 3$$

Übungsaufgabe 2 Erweitern Sie das klassische „Türme von Hanoi“-Problem durch weitere Hilfsstapel. Wann wird das erweiterte Problem trivial lösbar? Erörtern Sie Möglichkeiten der Erweiterung des Algorithmus aus Aufgabe 1 zur Lösung des „Türme von Hanoi“-Problems mit zwei Hilfsplätzen. Lösen diese das erweiterte Problem optimal? Lässt sich das beweisen?

Lösung 2 Das Problem wird sicherlich trivial, wenn es bei n Scheiben $n-1$ Hilfstapelplätze gibt. Dann braucht es höchstens $2 \cdot (n-1) + 1$ zulässige Züge.
Für den Fall von zwei Hilfsstapelplätzen sieht eine Erweiterung des Algorithmus zum Beispiel so aus:

1. Vergesse einen Hilfsstapelplatz. Rufe HANOI(n, Start, Ziel, Hilf_1) auf.

Das geht natürlich besser.

Verfahren POS (Presumed Optimal Scolution):
Wähle ein $l \in \{0, \ldots, n-1\}$.

1. Bewege die $n - l - 1$ kleinsten Scheiben auf den Hilfsstapel Hilf_1 unter Verwendung aller vier Plätze (und lasse die $l + 1$ größten Scheiben unberührt).
2. Bewege die $l + 1$ größten Scheiben zum Zielplatz unter Verwendung der Plätze Start, Ziel, Hilf_2 (nach der Prozedur HANOI).
3. Bewege die Scheiben von Hilf_1 nach Ziel unter Verwendung aller vier Plätze (und lasse die $l + 1$ größten Scheiben wieder unberührt).

Entscheident ist hier die Wahl des l. Man *vermutet*, daß für $l = \lfloor \sqrt{2n} - \frac{1}{2} \rfloor$ die minimale Zahl von Zügen $\nu(n) := (n - 1 - \frac{l(l-1)}{2})2^l + 1$ benötigt wird.

Allgemein (für p Plätze) kann man es so machen:
Staple eine optimale Zahl von n_1-Scheiben auf einen Hilfsplatz P_i. Die verbleibenden $n - n_1$ Scheiben werden auf dem Zielplatz gestapelt unter Benutzung von $p - 1$ Plätzen (da P_i ja

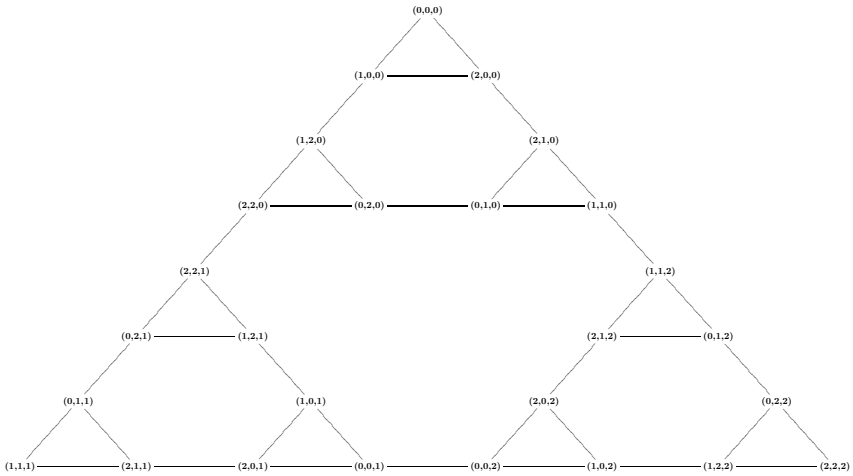


Abbildung 1.8: Der Graph $S(3,3)$

mit kleineren Scheiben beladen ist und deshalb nicht als Hilfplatz benutzt werden kann). Rufe diese Strategie rekursiv auf um die auftretenden Subprobleme zu lösen.

Übungsaufgabe 5 Geben Sie eine Bedingung für Verschiffungsmengen an, so daß die Prozedur UMSTAPELFOLGE_1 immer eine Umstapelfolge minimaler Länge erzeugt.

Lösung 5 Es sei V eine Verschiffungsmenge. Wenn $\nexists u, v \in V : \alpha(u) < \alpha(v) < \omega(u) < \omega(v)$, so erzeugt UMSTAPELFOLGE_1 eine Umstapelfolge minimaler Länge, nämlich der Länge $2|V| + 1$. In dem Fall kann dann zu keinen Überstaungen kommen. Schwieriger wird es beim Betrachten anderer Verschiffungsmengen. Malen sie sich Beispiele auf, bei denen zwar Überstauungen vorhanden sind, die sich aber durch $k-$Ordnen die Umstapelfolge nicht verkürzen. Ein einfaches Beispiel ist $V = \{u, v\}$ mit $\alpha(u) = 1, \omega(u) = 3$ und $\alpha(v) = 2, \omega(v) = 4$. Die Umstapelfolge hat die minimale Länge von 7.

Übungsaufgabe 10 Sei $S(3,3) := (V, A)$ ein Graph dessen Ecken die möglichen Konfigurationen eines „Türme von Hanoi"-Problems mit drei Stangen und drei Scheiben darstellen und zwei Ecken sind durch eine Kante verbunden, wenn sie durch einen zulässigen Zug auseinander hervorgehen. Viele Ecken hat der Graph $S(3,3)$? Wie lässt sich in dem Graph die Lösung des „Türme von Hanoi"-Problems auffinden? Malen Sie den Graph auf.

Lösung 10 Es gibt verschiedene Möglichkeiten die Konfiguration einer Stange anzugeben. Durch die göttliche Regel lassen sich die zulässigen Konfiguration sehr bequem angeben:
Es sei $V = \{(s_1, s_2, s_3) \in \{0,1,2\} \times \{0,1,2\} \times \{0,1,2\}\}$, wobei die $i-$te Komponente angibt, auf welcher Stange die $i-$te Scheibe liegt.
Dann gibt es z.Bsp. eine Kante zwischen $(0,0,0)$ und $(2,0,0)$ aber keine zwischen $(0,0,0)$ und $(0,0,2)$. G hat 27 Ecken (ist da eine Regel ableitbar für mehr als drei Scheiben?) und 39 Kanten.
Eine Lösung des „Türme von Hanoi"-Problems lässt sich nun formulieren als der kürzeste Weg zwischen $(0,0,0)$ und $(2,2,2)$ in $S(3,3)$. Dieser Weg hat eine Länge von sieben Kanten. Es lässt sich durch Auffinden kürzster Wege nun leicht die minimal nötige Zuganzahl zwischen zwei beliebigen Konfiurationen angeben. Der Graph ist in Abb.1.8 zu sehen.

$$V_{P(i,j)} = \begin{pmatrix}
t_{i,i+1} & & & & & & & & \\
t_{i,i+2} & t_{i+1,i+1} & & & & & & & \\
\vdots & \vdots & & & & & & & \\
t_{i,k} & t_{i+1,k} & t_{i+2,k} & \cdots & t_{k-1,k} & & & & \\
t_{i,k+1} & t_{i+1,k+1} & t_{i+2,k+1} & \cdots & t_{k-1,k+1} & t_{k,k+1} & & & \\
\vdots & \vdots & \vdots & \cdots & \vdots & \vdots & & & \\
t_{i,j-2} & t_{i+1,j-2} & t_{i+2,j-2} & \cdots & t_{k-1,j-2} & t_{k,j-2} & \cdots & & \\
t_{i,j-1} & t_{i+1,j-1} & t_{i+2,j-1} & \cdots & t_{k-1,j-1} & t_{k,j-1} & \cdots & t_{j-2,j-1} & \\
t_{i,j} & t_{i+1,j} & t_{i+2,j} & \cdots & t_{k-1,j} & t_{k,j} & \cdots & t_{j-2,j} & t_{j-1,j}
\end{pmatrix}$$

Abbildung 1.9: Verschiffunsgmenge für $P(i,j)$

Übungsaufgabe 11 Wenn nicht alle $|V_{(i,j)}| \neq 0$ sind, so ist evtl. $r(i,k,j)$ grösser als die tatsächlichen Umordnungskosten. Wie muss $r(i,k,j)$ modifiziert werden, damit dies nicht zutrifft?

Lösung 11 Es sei w_{min} das kleinste w, das

$$\sum_{p=i+1}^{k} \sum_{i=w+1}^{m+1} V_{(m,i)} = 0 \quad w = k+1, \ldots, j-1$$

erfüllt (wenn es ein solches zwischen $k+1$ und $j-1$ gibt) bzw. $w_{\min} = j$ sonst.
Es sei $w_{\max}$ das größte w, das

$$\sum_{p=0}^{w} \sum_{n=k+1}^{j-1} V_{(p,n)} = 0 \quad w = i, i+1, \ldots, k-1$$

erfüllt (wenn es ein solches zwischen i und $k-1$ gibt) bzw. $w_{\max} = i-1$ sonst.
Dies sind die „Zeilen- bzw. Spaltenkorrekturen“.
Dann gilt

$$r(i,k,j) = \sum_{q=i+1}^{k-1} \sum_{l=k+1}^{m+1} |V_{(q,l)}| + \sum_{p=0}^{i} \sum_{n=k+1}^{w_{min}-1} |V_{(p,n)}| - \sum_{q=i+1}^{w_{max}+1} \sum_{l=j}^{m+1} |V_{(q,l)}|$$

Es sei $t_{rs} := |V_{(r,s)}|$ in $P(i,j)$. Dann wird obige Formel verständlicher durch die Abbildung 1.9.

Definition 15 *Gegeben sei eine Verschiffungsmenge V mit den üblichen Bezeichnungen. Unter einem einer einfachen R-Umstapelfolge mit einem Zugriffspunkt zu V versteht man ein (OSOP) dessen Staupläne höchsten Pfade der Länge $R \in \mathbb{N}$ sind. Gibt es für ein gegebenes R keine R-(OSOP), dann heißt V eine* ***Überladung****.*

Wie kann man nun bei gegebenen $R \in \mathbb{N}$ feststellen, ob eine Verschiffungsmenge eine Überladung ist?

Seien die Aus- und Einladepunkte $i = 0, \ldots, m+1$.

$$S(0) = R - \sum_{j=1}^{m+1} |V_{(0,j)}| \tag{1.3}$$

$$S(i) = S(i-1) + \sum_{k=0}^{i-1} |V_{(k,i)}| - \sum_{j=i+1}^{m+1} |V_{(i,j)}| \qquad i = 1, \ldots, m \tag{1.4}$$

V ist eine Überladung, wenn $S(i) < 0$ für ein $i = 0, \ldots, m$.

Ein (OSOP) beginnt immer mit einem leeren Stapel. Es wäre aber auch denkbar, daß der erste einfache Stapel G_{0_1} nicht-leer ist und aus Knoten besteht mit Einladepunkt 0 und beliebigen Ausladepunkten. Wie ist die die Rekursion aus Satz 1 zu modifizieren?

Kapitel 2

Stauraumplanung

In diesem Abschnitt soll gezeigt werden, wie Stapel zur Modellierung der Stauraumplanung („stowage planning“) verwendet werden können. Stauraumplanung tritt beim Beladen von Schiffen, insbesondere von Vollcontainerschiffen, auf. Diese Planung hat eine Vielzahl von Nebenbedingungen zu berücksichtigen.

2.1 Das uCSP

In einfachsten Modell betrachten wir ein Containerschiff als einen rechteckigen Laderaum, der mit gleich hohen Stapeln von Containern angefüllt werden kann. Keiner dieser Stapel sei ausgezeichnet und somit bleibt seine Lage bzgl. anderer Stapel unberücksichtigt. Damit wird der Laderaum zu einem zweidimensionalen Array. Es werden C Stapel angenommen, die jeweils eine Höhe von R haben können. Technische Restriktionen wie ein bzgl. des Gewichtes ausgeglichener Laderaum bleiben zunächst unberücksichtigt.

Definition 16 *Ein (Primitiv-)Containerschiff oder auch Laderaum ist gegeben durch eine Anzahl C von Spalten (columns) und eine Anzahl R Reihen (rows) mit $C, R \in \mathbb{N}$. Seine Route sei gegeben durch eine Reihenfolge von (Hafen-)Punkten $0, \ldots, m+1$.*

Definition 17 *Eine Transportmatrix $T = (t_{ij})$ eines (Primitiv-)Containerschiffs ist gegeben durch $t_{ij} = |V_{(i,j)}|$, wobei $V_{(i,j)}$ Knoten einer Verschiffungsmenge V sind.*

Der Laderaum kann nun interpretiert werden als ein Stapel aus maximal C einfachen Stapeln mit maximaler Länge R. Um eine geeignete Erweiterung unserer Begriffe zu finden, muss nun Einiges festgehalten werden:

- Der Begriff der Umstapelfolge muss nun neu definiert werden.
- Es gibt nun für jeden Teilstapel einen Zugriffspunkt.
- Die Zuordnung der Elemente der Verschiffungsmenge zu den Teilstapeln ist nicht eindeutig.
- Es sollen möglichst viele Dinge aus Kapitel 1 Anwendung finden.

Definition 18 (MSCP) *Gegeben sei eine Verschiffungsmenge (V, α, ω). Es sei $\bar{\alpha} = \min\{\alpha(v) : v \in V\}$ und $\bar{\omega} := \max\{\omega(v) : v \in V\}$. Man bezeichnet eine Folge $(G_{t_i} = (V_{t_i}, A_{t_i}))$ mit $t = \bar{\alpha}, \ldots, \bar{\omega}$ und $i = 1, \ldots, N_t$ von Stapeln als eine **mehrdimensionale Umstapelfolge mit C Zugriffspunkten**, wenn*

1. *$G_{\bar{\alpha}_1} = \emptyset = G_{\bar{\omega}_{N_{\bar{\omega}}}}$,*
2. *G_{t_i} besteht aus maximal C Komponenten,*
3. *jede Komponente von G_{t_i} ist ein einfacher Stapel,*
4. *G_{j+1} entsteht aus G_j entweder durch Einstapelung oder Ausstapelung von Knoten aus V,*
5. *$V_{t_i} \subset V_t := \{v \in V : \omega(v) \geq t, \alpha(v) \leq t\}$*
6. *$V_{t_{|I_t|}} = \{v \in V : \omega(v) > t, \alpha(v) \leq t\}$ und*
7. *jedes Element aus V wird mindestens einmal eingestapelt und mindestens einmal ausgestapelt.*

*Gesucht ist nun eine Umstapelfolge **minimaler Länge**, d.h. kleinstem $|I| = \sum_{t=\bar{\alpha}}^{\bar{\omega}} |N_t|$.*

Die Folgenglieder $G_{t_{N_t}}$ heißt Stauplan am Punkt t und die Vereinigung der Staupläne heisst der Stauplan von V.

Man kann die folgenden Fälle unterscheiden: Ist die Länge der einfachen Stapel beschränkt durch ein R oder unbeschränkt? Ist die Zahl der Ein- und Ausladepunkte kleiner als C oder größer als C?

Sind die einfachen Stapel unbeschränkt und ist $C \geq m + 1$, so ist eine minimale MSCP leicht anzugeben. Frage: Wie sieht sie dann aus? Die erste Komponente besteht dann nur aus Knoten v mit $\omega(v) = 1$, die zweite Komponente nur aus Knoten $v$mit $\omega(v) = 2$ usw. Da es genügend mögliche Komponeten gibt und in einer Komponente alle Knoten denselben Ausladepunkt haben, gibt es keine überstauten Knoten. Also ist die Umstapelfolge von minimaler Länge.

Den Fall mit Komponenten unbeschränkter Länge und $C < m+1$ nennen wir das $u-CSP$, das unrestringierte Stauraumproblem für (Primitiv-)Containerschiffe (unrestricted containership stowage problem). Ist die Suche nach einer minimalen MSCP in diesem Fall dort ebenso polynomial lösbar wie das Auffinden einer minimalen OSOP? Das Auffinden einer minimalen OSOP ist ein Spezialfall des Auffindens einer minimalen MSCP und auf den ersten Blick erscheint es ja bei mit mehreren Stapeln einfacher, da die Stapelelemente nun geschickter auf die einzelnen Komponenten „verteilt“ werden können. So wie im Fall des „Türme von Hanoi“-Problems mit einem Hilfsstapelplatz viel mehr Züge nötig sind als bei einer Variante mit mehr Hilfsstapelplätzen. Das eigentlich Neue ist nun die Möglichkeit ein Stapelelement an einem Punkt echt umzustapeln, d.h. aus seiner Komponenten zu stapeln und in eine andere Komponente einzustapeln. Dadurch wird der Raum der möglichen Folgen für eine Verschiffungsmenge sehr viel grösser und das kann man auch beweisen. (Was soll das heissen?)

Fakt 5 *Es gibt eine minimale MSCP eines uCSP in dem die Knoten $V_{(i,j)}$ in den Staupländen an den Punkten zwischen i und j einen Pfad bilden.*

Das ist analog zu Fakt 3. Ist das Containerschiff jedoch restringiert, so gilt das nicht. Wir wollen nun zeigen, dass die Suche nach einer minimalen MSCP eines uCSP $\mathcal{NP}$-vollständig ist.

Definition 19 *Ein (endlicher) Graph $G = (V, A)$ heißt $C-$färbbar, wenn es eine Zerlegung der Knoten V in C Mengen gibt derart, daß adjazente Knoten nicht in derselben Teilmenge liegen. Das Problem die kleinste Anzahl C von benötigten Teilmengen (Farben) für einen gegebenen Graphen zu finden, heißt das Färbungsproblem. Das C-Färbungsproblem ist die Frage: Ist G mit C Farben färbbar?*

Seien $[i, j], [k, l]$ zwei reelle Intervalle. Man sagt, daß $[i, j]$ und $[k, l]$ überlappen, wenn $i < k < j < l$

Definition 20 *Ein Intervallüberlappungsgraph (overlap graph) ist ein einfacher Graph, dessen Ecken zu einem realen Intervall korrespondieren und zwei Ecken sind durch eine Kante verbunden, wenn die zugehörigen Intervalle sich überlappen.*

Fakt 6 (Unger[30]) *Sei $C \in \mathbb{N}$ mit $C \geq 4$. Das C-Färbungsproblem ist $\mathcal{NP}$-vollständig für Intervallüberlappungsgraphen.*

Wir versuchen nun eine polynomiale Transformation anzugeben, die die Suche einer minimalen MSCP eines $u - CSP$ (ganz) ohne Überstauungen in ein C-Färbungsproblem eines Intervallüberlappungsgraphen überführt. Dadurch wäre die $\mathcal{NP}$-Vollständigkeit des Problems MSCP im Falle eines $u - CSP$ bewiesen.
Sei $T = (T_{ij})$ eine Transportmatrix eines $u - CSP$ und $G = (V, A)$ ein Graph mit $V := \{v_{ij} : T_{ij} \neq 0\}$ und $A \subset V \times V$ mit $A := \{(v_{ij}, v_{kl}) : i < k < j < l\}$. G ist ein Intervallüberlappungsgraph und Kanten geben mögliche Quellen für Überstauungen an. Die Frage, ob es nun eine MSCP ohne überstaute Knoten gibt, ist nun gleichbedeutend mit der Suche nach einer C-Färbung in G. Jedes Intervallpaar, das sich überschneidet, muss paarweise in verschiedene Komponenten. Das fassen wir nun zusammen in dem

Fakt 7 (Avriel et al. [3]) *Gegeben sei eine Transportmatrix eines $u-CSP$. Der zugehörige Intervallgraph ist C-färbbar genau dann, wenn C Komponenten (Spalten) ausreichen für eine minimale MSCP ohne überstaute Knoten.*

Zusammen mit Fakt 6 ergibt sich nun der

Satz 2 (Avriel et al [3]) *Sei C die Zahl der Spalten eines $u-CSP$ und T eine Transportmatrix. Das Problem, die Frage zu beantworten, ob es eine minimale MSCP ohne überstaute Knoten gibt, ist $\mathcal{NP}$-vollständig für $C \geq 4$.*

Wenn $R < \infty$, also im kapazitierten Fall, den wir $r - CSP$ nennen wollen, ist die Komplexität für ein festes R unbekannt. Dort geht natürlich auch die Eigenschaft aus Fakt 5 verloren, weil dann nicht alle Stapelelemente aus einem $V_{(i,j)}$ in eine Komponente passen müssen. Aber selbst wenn sie alle in einer Komponente liegen könnten, kann es notwendig sein, sie auf verschiedene Komponenten zu verteilen. Deshalb ist der „Dekompensationsmethode“ aus [2] tatsächlich nur ein heuristisches Verfahren[1]. Die „Dekompensationsmethode“ verteilt die zu transportierenden Container derart auf die Stapel, so dass gleiche Gruppen

[1] Bernd Stauss 7.11.2003 (persönliche Mitteilung)

immer zusammenbleiben wenn möglich. Dadurch enstehen volle Stapel, die keine Überstauer enthalten. Jedoch bleibt eine gewissen Menge an Containern übrig, die weiter verstaut werden müssen.
Hier ein Beispiel:

Sei ein Containerschiff gegeben durch $C = 2, R = 2$ mit den Häfen $1, \ldots, 5$. Die Transportmatrix sei $T_{15} = 2, T_{24} = 1, T_{3,5} = 1$ und $T_{ij} = 0$ für sonstige Matrixelemente. Wenn die Stapelelemente, die von Hafen 1 nach Hafen 5 müssen in derselben Komponente liegen, so gibt es ein überstautes Stapelelement. Liegen sie jedoch in verschiedenen Komponenten, so gibt es keine Überstauungen.
Siehe Abbildung unten.

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

1.Variante (Stauplan an 3)

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

2.Variante (Stauplan an 3)

Ist $R = 2$, so kann man eine heuristisches Verfahren angeben, das einen Stauplan mit höchstens $\lfloor \frac{m-1}{2} \rfloor$ überstauten Stapelelementen erzeugt. Sei T eine zulässige Transportmatrix, d.h. an jedem Hafen gibt es zulässige Staupläne. Fülle jeweils Spalten mit Container der gleichen Gruppe auf. Dadurch entstehen erstmal keine überstauten Knoten. Die resultierende Matrix enthält nur noch Nullen und Einsen: $T^{neu} = T \bmod 2$. Am ersten Hafen: Fülle die verbliebenen Spalten jeweils mit Stapelelementen auf, zuerst nach dem weitest entfernten Hafen, dann nach dem zweit-weitesten usw. Dadurch bleibt höchstens eine Spalte übrig mit genau einem Stapelelement. Dann geht es zum zweiten Hafen: Entlade, was entladen werden muss. Fülle wie im ersten Hafen auf. Ein Stapelelement bleibt dann höchstens übrig. Das stelle in die Spalte mit dem einen Stapelelement aus dem vorherigen Hafen (wenn das sein muss → tritt nur auf wenn das Schiff am Kapazitätslimit ist). Dies ist dann eine Quelle von überstauten Stapelelementen. Dieser Fall tritt höchstens $\lfloor \frac{m-1}{2} \rfloor$ mal (warum?) auf. Iteriere nun. Leider ergibt dieses Verfahren keinen optimalen Stauplan wie man an dem obigen Gegenbeispiel sieht.

Sei C^* die kleinste Anzahl von Spalten, so dass für eine Transportmatrix T ein Stauplan ohne überstaute Knoten existiert. Nun wollen wir obere und untere Schranken bestimmen.

Satz 3 (Avriel et al [3]) *Sei T eine $(n-1) \times (n-1)$-Transportmatrix T eines uCSP mit $T_{ij} > 0$ für alle $i \geq j$, d.h. alle Matrixeinträge in der unteren Dreiecksmatrix sind $\neq 0$. Dann ist $C^* = \lceil \frac{n}{2} \rceil$ und es gibt einen einfachen Algorithmus mit linearer Laufzeit um einen Stauplan ohne überstaute Knoten mit C^* Spalten zu erhalten.*

Wie sieht dieser Algorithmus nun aus? Für jeden Intervall $[i, j]$ sei $A_{ij} = i + j$. Da für zwei Intervalle $[i, j], [k, l]$, die sich überlappen, gilt:

$$2 \leq |A_{ij} - A_{kl}| \leq n - 2$$

können gewisse Stapelelmente auf jeden Fall in dieselbe Spalte gestapelt werden. Hierzu ein Bild:

Farbe	1	2	3	$\cdots$	$\frac{n}{2}-1$	$\frac{n}{2}$
A_{ij}	3	5	7	$\cdots$	$n-1$	$n+1$
	4	6	8	$\cdots$	$n-1$	$n+2$
	$n+3$	$n+5$	$n+7$	$\cdots$	$2n-1$	
	$n+4$	$n+6$	$n+8$	$\cdots$		

Wie sieht es nun aus, wenn die Matrix nicht voll besetzt ist. Dann kommt man evtl. mit weniger Farben, d.h. weniger Säulen aus. Es gilt das folgende

Korollar 2 *Sei T eine $(n-1)\times(n-1)$-Transportmatrix eines uCSP und ω die Kardinalität einer maximalen Clique*[2] *im zugehörigen Intervallgraph. Dann ist*

$$\omega \leq C^* \leq \min\{2\omega, \left\lceil \frac{n}{2} \right\rceil\}$$

und es gibt einen polynomialen Algorithmus zur Erzeugung eines Stauplans mit $\min\{2\omega, \lceil \frac{n}{2} \rceil\}$ *Säulen.*

Definition 21 *Eine Clique eines Graphen ist eine maximaler vollständiger Subgraph.*

Für allgemeine Graphen ist das Auffinden von maximalen Cliquen $\mathcal{NP}-$vollständig. Nach [30] jedoch ist es möglich einen Intervallüberlappungsgraphen mit $2k$ Farben zu färben, wenn er eine maximale Clique der Größe k enthält. Eine maximale Clique in einem Intervallüberlappungsgraphen kann nach [23] in $\mathcal{O}(n^2)$ gefunden werden. (Wie?)

2.2 Das rCSP

Wir gehen nun davon aus, daß ein Primitivcontainerschiff gegeben ist. Dazu eine Transportmatrix T und die Häfen von 1 bis N nummeriert sind. Wir nennen ein einen Knoten v_{rc} in einem Stauplan einen Stellplatz (r, c), wenn der Knoten in der c-ten Komponente an der r-ten Position im Pfad liegt.

Es sei $x_{ijv}(r, c) = 1$ wenn ein Container in Hafen i mit Zielhafen j in Stellplatz (r, c) geladen wird und dieser Container in Hafen v entladen wird und 0 sonst.
Weiterhin sei $y_i(r, c) = 1$, wenn bei Abfahrt von Hafen i Stellplatz (r, c) von einem Container belegt ist und 0 otherwise.
Die Formulierung des rCSP als binäres ganzzahliges Programm (siehe in [2]) sieht dann so aus:

$$Min \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{v=i+1}^{j-1} x_{ijv}(r, c)$$

unter den Nebenbedingungen

Verschiffungsmenge $(i = 1, \ldots, N-1, j = i+1, \ldots, N)$:

$$\sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{v=i+1}^{j} x_{ijv}(r, c) - \sum_{k=1}^{i-1} \sum_{r=1}^{R} \sum_{c=1}^{C} x_{kji}(r, c) = T_{ij} \tag{2.1}$$

[2] http://mathworld.wolfram.com/Clique.html

Stellplatz ($i = 1, \ldots, N-1, r = 1, \ldots, R, c = 1, \ldots, C$):

$$\sum_{k=1}^{i} \sum_{j=i+1}^{N} \sum_{v=i+1}^{j} x_{kjv}(r,c) = y_i(r,c) \tag{2.2}$$

Obenauf ($i = 1, \ldots, N-1, r = 1, \ldots, R-1, c = 1, \ldots, C$):

$$y_i(r,c) - y_i(r+1,c) \geq 0 \tag{2.3}$$

Überstauung ($j = 2, \ldots, N, r = 1, \ldots R-1, c = 1, \ldots, C$):

$$\sum_{i=1}^{j-1} \sum_{p=j}^{N} x_{ipj}(r,c) + \sum_{i=1}^{j-1} \sum_{p=j+1}^{N} \sum_{v=j+1}^{p} x_{ipv}(r+1,c) \leq 1 \tag{2.4}$$

$$x_{ijv}(r,c) \in \{0,1\} \qquad y_i(r,c) \in \{0,1\}$$

Die Nebenbedingungen haben die folgende Bedeutungen:

- (2.1) gibt an, dass alle Container eingeladen werden.
- (2.2) erzwingt, dass in einem Stauplan an einem Hafen nur ein Container an einem Stellplatz steht.
- (2.3) erzwingt, dass die Container einer Spalte aufeinander stehen.
- (2.4) definiert Umstapelungen.

Die Zielfunktion zählt die Umstapelungen, die durch überstaute Container hervorgerufen werden.
Dieses binäre lineare Programm ist allerdings sehr groß. Für den Fall von $R = 2$ und $C = 5$ mit $N = 5$ gibt es 240 Variablen und 106 Nebenbedingen. Wieviel sind es im allgemeinen Fall? Wie verhält es sich mit der Lösbarkeit solcher binärer Programme? Welche Software kann dazu verwendet werden?

Fakt 8 *Der optimale Zielfunktionswert der Relaxierung des obigen binären ganzzahligen Programm ist* 0.

Was heisst Relaxierung? Was bedeutet der der Fakt 8 für die Lösbarkeit des binären ganzzahligen Programms?
Die Zulässigkeit einer Transportmatrix muss natürlich auch hier als erster Schritt überprüft werden. Es sei $X = R * C$. Dann sei

$$S(0) = X - \sum_{j=2}^{N} T_{1j} \tag{2.5}$$

$$S(i) = S(i-1) + \sum_{k=1}^{i-1} T_{ki} - \sum_{j=i+1}^{N} T_{ij} \qquad i = 1, \ldots, N-1 \tag{2.6}$$

Die Transportmatrix ist zulässig, wenn $S(i) \geq 0$ für $i = 1, \ldots, N-1$.

Da das uCSP $\mathcal{NP}$-vollständig ist, ist es so, dass das rCSP im allgemeinen Fall auch $\mathcal{NP}$-vollständig ist. Für ein fixes r ist die Komplexität jedoch eine offene Frage.

Wir versuchen nun einen Stauplan für das rCSP aufbauen. Änlich wie bei der Suche einer minimalen OSOP gibt es Regeln um einen guten, jedoch nicht unbedingt optimalen Stauplan zu erstellen. So wie beim OSOP eine Tiefe des Ordnens berechnet wird, berechnet man nun eine Tiefe für eine Spalte.

Bemerkung 2 Unter wünschenswerten Umstauungen verstehen wir solche, die an einem Hafen durchgeführt werden, ohne dass sie an dem Hafen notwendig wären.

Die kurzsichtige Umstauregel („myoptic shifting rule“) entsteht nun so: Es sei ein Schiff in Hafen i und alle Container mit Ziel i, sowie die sie blockierenden Container, seien entladen. Dann kann man für **eine** Spalte entscheiden:

1. Lade Container mit Einladehafen i und die blockierenden Container vom Dock in Spalte ein.
2. $m-$Ordne die Spalte und lade Container ein.

Man geht dabei davon aus, daß alle Container an Dock in diese eine Spalte passen. Wie lässt sich nun entscheiden, ob wir Regel 1 oder Regel 2 anwenden? In der kurzsichtigen Umstauregel gehen wir weiterhin davon aus, daß zwischen dem Hafen $i+1$ und m keine Container mehr in die Spalte geladen werden (Bem: Das kann man natürlich aufgrund von Satz 1 allgemeiner machen). Wie lautet eine Formel dafür, zu entscheiden, ob und wie Regel 2 angewandt werden soll? Das steht z. Bsp. in [4]:

Sei G^i_m die kleinste Zahl (wünschenswerter) Umstauungen um eine Spalte bis m zu Ordnen am Hafen i und $Q^i_m := G^i_m - G^i_{m-1}$. Es ist $G^i_N \geq G^i_{N-1} \geq \ldots \geq G^i_i = 0$. Es sei weiterhin M_i die Gesamtzahl aller Container an Hafen i, die auf das Schiff müssen, nachdem alle Container mit Ziel i und die sie überstauenden Container abgeladen wurden. P^i_j sei die Gesamtzahl der $j-$Container in Hafen i und in der Spalte.

Wenn wir nun an Hafen i bis $m-1$ ordnen und $Q^i_m > 0$ ist, dann sind die Kosten des m-Ordnens Q^i_m. Wenn wir nicht m-Ordnen sind die zusätzlichen Kosten an Hafen m

$$M_i + G^i_m - \sum_{j=i+1}^{m} P^i_j$$

Das kann man nun folgendermaßen umformen

$$M_i + G^i_m - \sum_{j=i+1}^{m} P^i_j = M_i + G^i_{m-1} + Q^i_m - \sum_{j=i+1}^{m} P^i_j$$

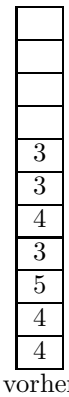
Also, wenn $Q^i_m > 0$ und

$$Q^i_m < M_i + G^i_{m-1} + Q^i_m - \sum_{j=i+1}^{m} P^i_j$$

sollte man umordnen Es ergibt sich nun also die folgende **Regel**:
m-Ordne $(m = i+1, \ldots, N)$, wenn

$$Q^i_m > 0 \quad \text{und} \quad \sum_{j=i+1}^{m} P^i_j < M_i + G^i_{m-1} \tag{2.7}$$

Ein Beispiel:



vorher

An Dock seien Container mit Zielen 5,3,5 und 4, also $M_2 = 4$. Es ergibt sich nun

$$\begin{array}{llll} P_3^2 = 4 & P_4^2 = 3 & P_5^2 = 4 & \\ G_2^2 = 0 & G_3^2 = 4 & G_4^2 = 6 & G_5^2 = 6 \\ Q_3^2 = 4 & Q_4^2 = 2 & Q_5^2 = 0 & \end{array}$$

Da $P_3^2 \geq M_2 + G_2^2$ und $P_3^2 + P_4^2 < M_2 + G_3^2$, sollte man an Hafen 2 bis zum Hafen 4 Ordnen. Da $Q_5^2 = 0$, ist es natürlich auch geordnet bis Hafen 5.

nachher

Kommt es dazu, dass Überstauungen unvermeidlich sind, so muss entschieden werden, in welche Spalte dieser überstauende Container eingeladen werden soll.
Hier zwei Regeln - die Funktionsregel und die Zwangsregel:

Die **Funktionsregel**

Sei F_c die Zahl noch freier Stauplätze in einer Spalte c und $Y_{rc} \neq 0$ die Nummer des Zielhafens des Containers in Stellplatz (r, c). Für jede Spalte c mit noch freien Stellplätzen sei

$$\hat{c} := \sum_{\substack{r \\ (r,c) \text{ belegt}}} \frac{1}{Y_{rc}} + \log(F_c + 1) \tag{2.8}$$

Die Funktionsregel wählt nun die Spalte für den Container aus, für die $\hat{c}$ den größten Wert hat.

Die **Zwangsregel**

Sei ein Container (i, j) unterzubringen. Berechne für jede Spalte c die Zahl $\check{c}$ der jeweils auftretenden Überstauungen, wenn Container (i, j) der Spalte c zugeordnet wird und keine weitere Container dieser Spalte zugeordnet werden bis zum Ende der Reise. Die Zwangsregel ordnet den Container der Spalte c mit minimalem $\check{c}$ zu.

2.2.1 Die hängende Heuristik

In [4] wird nun die hängende Heuristik („suspensory heuristic procedure“) angegeben. Diese kann man in folgende grobe Züge unterteilen:

- Im ersten Hafen gibt es einen Stauplan mit geordneten Stapeln
- In allen weiteren Häfen werden nun alle Container des entsprechenden Zielhafens, sowie ihre blockierenden Container ausgeladen.
- Dann werden alle Container den Stellplätzen zugeordnet nach Regeln:
  - Ordne sie Spalten zu ohne Überstauungen zu produzieren mit Blick auf weitere Häfen
  - Bei unvermeidbaren Überstauungen wende die Funktionsregel oder die Zwangsregel an, um eine geeignete Spalte zu finden

  - Dann verwende die kurzsichtige Umstauregel für die einzelnen Spalten

- Während der Prozedur werden Container an einem Hafen evtl. mehrmals zugeordnet und wieder zurückgesetzt, wobei die realen Aus- und Einladevorgänge natürlich nur einmal durchgeführt werden.

Es sei J_i = Menge der Container an Hafen i. Nun der Algorithmus: Gegeben sei R, C und eine Transportmatrix T mit Häfen 1 bis N, die für den Laderaum mit R Reihen C Spalten zulässig sei.

Input: R, C, T
Output: Stauplan und Anzahl der Umstapelungen
Start: Anzahl Umstapelungen = 0
Stauraumplanung am Hafen 1:

H (Hängen):
- Container in einer steigenden Folge zuordnen
- „Aufhängen“ der j-Container in einer leeren Spalte
- Falls Spalte voll nächste leere Spalte
- Alle Container zugeordnet → F.5

F (Füllen): $l = 1$
F.1 Falls Spalte l voll → F.3
F.2 Sei $k_1 = \max\{d : d \in J_1\}$. Zuordnen der k_1-Container zum Boden der Spalte l. Falls notwendig, aktualisieren von J_1.
F.3 Wenn $l < C$, dann $l = l + 1$ und → F.1
F.4 Zuordnen der übrigen Container ohne Umstapelungen
F.5 „Fallenlassen“ aller aufgehängten Container. Setze $i = 2$.

Stauraumplanung am Hafen i:

U (Ausladen der i-Container):
- Ausladen aller i-Container und alle sie blockierenden Container
- Anzahl der Umstapelungen aktualisieren
- T aktualisieren
- Setze $j = i + 1$

A (Zuordnen (Assign) der j-Container):
A.1 - Wenn alle j-Container zugeordnet → A.4.2
- Zulässige Spalte mit einem j-Container in der obersten Reihe und ohne z-Container, $z < j$ in der Spalte →
Zuordnen der nicht zugewiesenen Container in die Spalte.
A.2 - $k_j = \max\{d : d \in J_1\}$ eine zulässige Spalte ohne z-Container, $z < k_j$
→ Zuordnen der nicht zugewiesenen Container in diese Spalte
- alle j-Container zugeordnet → A.4.2
A.3 Nach Durchgang der folgenden Fälle sind alle j-Container zugeordnet → A.4.2
A.3.1 Zulässige Spalte ohne aufgehängte Container:
Oberster h-Container mit $j < h < k_j$
→ hänge j-Container in dieser Spalte auf.
A.3.2 Zulässige Spalte mit aufgehängten Containern:
Nicht hängende sind y-Container mit $y > j$.
j' ist der Hafen, der in den hängenden Containern mit $j' < j$ die weiteste Entfernung hat
→ Anordnen der j-Container direkt unter den j'- Containern
A.3.3 Aufhängen der j-Container in einer leeren Spalte
A.3.4 Sonst → A.5

A.4

A.4.1 Wiederholung von A.1-A.3 bis alle j-Container zugeordnet sind

A.4.2 - $j < N$, dann $j = j + 1 \rightarrow$ A.1

- $j = N \rightarrow$ Anzahl der Umstapelungen aktualisieren
  Fallenlassen alle aufgehängten Container
  - $i < N - 1$, dann $i = i + 1 \rightarrow U$
  - Gehe zu Ende

A.5 Unassign

- Zuordnung aller h Container mit $h < j$ aufheben,
  die an Hafen i zugeordnet wurden
- Zuordnen der j-Container:
  1. ganz oben in einer zulässigen Spalte ohne h-Container, $h < j$
  2. nicht zugeordnete j-Container
  ganz unten in einer leeren Spalte
  Sind alle zugeordnet, dann $j = i + 1 \rightarrow$ A.1
  Sonst A.6

A.6 Spaltenselektion:

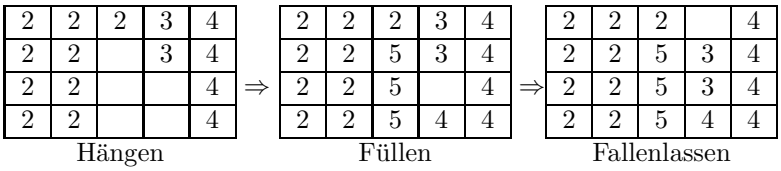
- Wähle Spalte $\hat{c}$ nach Funktions- oder Zwangsregel
- Zuordnen möglichst vieler j-Container zu $\hat{c}$
- Wiederhole A.6 bis alle j-Container zugeordnet sind
- Setze j = i + 1 $\rightarrow$ A.1

End: Ende des SH-Verfahrens

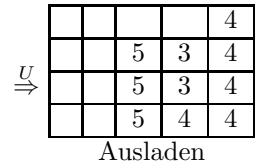
Hier nun ein Beispiel: Gegeben sei ein Containerschiff mit fünf Spalten und vier Reihen. Das Schiff fährt sechs Häfen an und die Transportmatrix ist wie folgt

Quelle ↓ Ziel →	2	3	4	5	6
1	9	2	5	3	0
2	0	4	1	3	0
3	0	0	0	3	4
4	0	0	0	2	2
5	0	0	0	0	4

Stauplanerstellung am Hafen 1



Stauplanerstellung am Hafen 2



Es gibt hier vier 3-Container, einen 4-Container und drei 5-Container. In den folgenden Stauplänen sind die temporär zugeordneten Container mit einem ' gekennzeichnet. Die einzelnen Ladestufen:

$\overset{A.1,A.3.1,A.3.3}{\Longrightarrow}$

	3’	3’	3’	4
	3’	5	3	4
		5	3	4
		5	4	4

3-Container

$\overset{A.3.2}{\Rightarrow}$

	3’	3’	3’	4
	3’	5	3	4
	4’	5	3	4
		5	4	4

4-Container

$\overset{A.3.2,A.3.3}{\Rightarrow}$

	3	3	3	4
	3	5	3	4
5	4	5	3	4
5	5	5	4	4

5-Container

Stauplanerstellung am Hafen 3

$\overset{U}{\Rightarrow}$

				4
		5		4
5	4	5		4
5	5	5	4	4

Ausladen

$\overset{A.1}{\Rightarrow}$

5’		5’		4
5’		5		4
5	4	5		4
5	5	5	4	4

5-Contanier

Es werden nun A.3, A.5 und A.6 überprüft. In Stufe A.5 werden die drei 5-Container wieder entfernt. In Stufe A.6 wird nun die Funktionsregel für die Spalten 1 bis 4 angewandt. Sie ergibt die folgenden Werte: Spalte 1: 1.499, Spalte 2: 1.549, Spalte 3: 1.293, Spalte 4: 1.636. Also wählen wir Spalte 4 für die 6-Container. Die kurzsichtige Umstauregel wird angewandt und sie gibt an, daß wir die Spalte vier 4-Ordnen. Das ergibt den folgenden Stauplan:

			4’	4
			6’	4
5	4	5	6’	4
5	5	5	6’	4

Hängen

Jetzt ist noch ein 6-Container am Kai. Die Funktionsregel ergibt Zuordnung zu Spalte 2 und die kurzsichtige Umstauregel ergibt keine Umordnung. Es ergibt sich eine Umstauung in Spalte 4 und es sieht nun so aus:

			4’	4
	6’	5	6’	4
5	4	5	6’	4
5	5	5	6’	4

$\overset{A.1}{\Rightarrow}$

5		5	4	4
5	6	5	6	4
5	4	5	6	4
5	5	5	6	4

Stauplanerstellung am Hafen 4

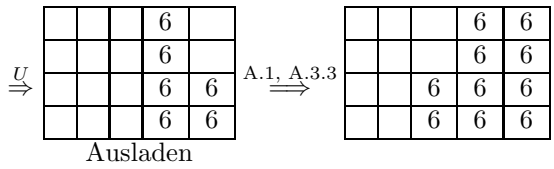
Ein zusätzlicher 6-Container wird hier abgeladen (er überstaut einen 4-Container). Hier entstehen zwei Umstauungen (shifts).

5		5		
5		5	6	
5		5	6	
5	5	5	6	

$\overset{A.1,A.3.3}{\Longrightarrow}$

5		5	6	
5	5	5	6	
5	5	5	6	6
5	5	5	6	6

Stauplanerstellung am Hafen 5



2.2.2 Mathematische Modellierung und LP-Löser

Um die optimale Lösungen zu erhalten, kann man versuchen es in AMPL® (A Modeling Language for Mathematical Programming[3]) zu modellieren. Diese Sprache erzeugt für verschiedene Löser, d.h. Optimierungssoftware, geeignete Datenformate. Man kann z.Bsp. für lineare Programme und ihre ganzzahligen Gegenstücke ILOG CPLEX®[4] daran anbinden um Lösungen zu finden. Hier nun ein Beispiel für das gemischt-ganzzahlige Programm aus 2.2:

```
set PORTS ordered;
set ROWS ordered;
set COLUMNS ordered;
param TRANSPORT {PORTS, PORTS} >= 0;
var X {i in PORTS, j in PORTS, v in PORTS,r in ROWS, c in COLUMNS:
ord(i)< ord(v) <= ord(j)} binary;
var Y {i in PORTS, r in ROWS, c in COLUMNS} binary;

minimize Shiftings:
sum {i in PORTS, j in PORTS, v in PORTS, r in ROWS, c in COLUMNS:
ord(i)< ord(v) < ord(j) } X[i,j,v,r,c];

subject to Shipquant{i in PORTS, j in PORTS:
ord(i) < card(PORTS) and ord(j)>ord(i)}:
sum {r in ROWS, c in COLUMNS, v in PORTS:
ord(i) < ord(v) <= ord(j)} X[i,j,v,r,c]
-
sum {r in ROWS, c in COLUMNS, k in PORTS: ord(k)<ord(i)
and ord(k)< ord(i)<=ord(j)}
X[k,j,i,r,c]
=
TRANSPORT[i,j];

subject to Slot{i in PORTS, r in ROWS, c in COLUMNS:
ord(i) < card(PORTS)}:
sum {k in PORTS, j in PORTS, v in PORTS:
ord(k) <= ord(i) and ord(j)>ord(i) and ord(k)<ord(v)<=ord(j)}
X[k,j,v,r,c]
=
Y[i,r,c];

subject to Ontop{i in PORTS, r in ROWS, c in COLUMNS:
```

[3]http://www.ampl.com/
[4]http://www.ilog.com/products/cplex/

```
ord(i)< card(PORTS) and ord(r)<card(ROWS)}:
Y[i,r,c]-Y[i,r+1,c]>= 0;

subject to Shift{j in PORTS, r in ROWS, c in COLUMNS:
ord(r)<card(ROWS)}:
sum{i in PORTS, p in PORTS: ord(i)<ord(j) and ord(p)>=ord(j)
and ord(i)<ord(j)<=ord(p)}X[i,p,j,r,c]
+
sum{i in PORTS, p in PORTS, v in PORTS:
ord(i)<j and ord(p)>ord(j) and ord(v)>ord(j) and ord(v)<=ord(p)
and ord(i)<ord(v)<=ord(p)}
X[i,p,v,r+1,c]<=1;
```

Ein Datensatz sieht z. Bsp dann so aus:

```
data;  ############ DATA STARTS HERE ############
set PORTS := 1 2 3 4 5 6 ;
set ROWS := 1 2 3 4;
set COLUMNS := 1 2 3 4 5;
param TRANSPORT: 1 2 3 4 5 6 :=
               1 0 9 2 5 3 0
               2 0 0 4 1 3 0
               3 0 0 0 0 3 4
               4 0 0 0 0 2 2
               5 0 0 0 0 0 4
               6 0 0 0 0 0 0 ;
```

2.2.3 Avanced whole column heuristic

Ein neuer Algorithmus wurden im Rahmen einer Diplomarbeit [13] von A. Jellinghaus entwickelt. Im Rahmen dieser Arbeit wurde versucht, eine bessere Heuristic als Alternative zur ,,Suspensory Heuristic Procedure" [4] zu entwickeln. In Anlehnung an die ,,Whole Column Heuristic" vor Avriel [2] wurde diese ,,Advanced Whole Column Heuristic" genannt, da eine grundlegende Idee übernommen wurde.

Der Algorithmus basiert auf mehreren Schritten und ein paar Prinzipien, die für alle Schritte gelten. Die Prinzipien im einzelnen:

1. Die Staupläne für alle Transportschritte werden parallel aufgebaut, die Algorithmen der einzelnen Schritte orientieren sich immer an einem oder mehreren Containern, die dann für alle Transportschritte von Quell- bis Zielhafen dem Stauplan auf einmal zugeordnet werden.
2. Container von Hafen 1 zu Hafen N können keine Umstapelungen verursachen, diese werden einfach als erstes verplant und landen dadurch unten. Alle Schritte arbeiten Container in einer lexikografischen Reihenfolge ab, erst gilt die Entfernung die ein Container zurücklegt (Container mit maximaler Entfernung zuerst), dann der Zielhafen des Containers (Container mit höchstem Zielhafen zuerst).
3. Container von Hafen i zu Hafen $i+1$ können keine Umstapelungern verursachen; diese werden einfach als letzte Container an beliebigem Ort oben auf gestapelt. Der letzte

Schritt weist diese Container zu, alle anderen ignorieren solche Container.

Es gibt verschiedene Szenarien, in denen es trivial ist, einen Stauplan ohne Umstapelungen aufzustellen. Die Heuristik versucht Lösungen für mehrere spezielle Szenarien zu verbinden und für viele Fälle ein gutes Ergebnis zu erreichen. Es ist jedoch nicht ausgeschlossen, dass die Heuristik in besonderen Fällen sehr schlechte Ergebnisse produziert.

Zunächst jedoch eine Beschreibung der Datenstrukturen, da diese Grundlage für die einzelnen Schritte sind. Übergeben wird natürlich der Transportplan. Dieser darf nicht verändert werden und wird daher kopiert. Die Kopie wird nach und nach reduziert, bis alle Zellen den Wert null enthalten, also alle Container zugeordnet sind.

Ergebnis des Algorithmus ist ein Stauplan, der zu Beginn des Algorithmus leer angelegt wird und dann nach und nach gefüllt wird.

Hinzu kommt eine Datenstruktur, die aggregierte Daten über den Stauplan enthält: Betrachtet man den Inhalt einer Säule über den gesamten Transportzeitraum, so ändert sich der Inhalt oft nicht. Die aggregierte Datenstruktur benutzt einen Zeitraum i bis j als Schlüssel und speichert darin jeweils eine Liste aller Säulen, die in genau diesem Zeitraum ihren Inhalt nicht ändern, allerdings nur solcher Säulen, die in dem jeweiligen Zeitraum nicht voll sind.

Zu Beginn enthält die aggregierte Datenstruktur daher nur Information zum Schlüssel $1 : N$ und in dieser Liste sind alle Säulen vermerkt.

2.2.3.1 Erster Schritt

Alle Säulen des Laderaums werden mit einer einheitlichen Höhe R modelliert. Enthalten alle Zellen der Transportmatrix ein Vielfaches von R, so kann das Problem gekürzt formuliert werden mit einem Stauraum der neuen Höhe 1 und einer Transportmatrix, in der alle Werte durch R geteilt wurden. Dieses Problem ist weiterhin lösbar, kann aber wegen der Höhe 1 aller Säulen keine Umstapelungen verursachen.

Im ersten Schritt wird eine Liste aller Transporte erstellt, die ganze Säulen füllen können, und die Transportmatrix wird auf Werte modulo R reduziert. Diese Liste wird so gefüllt, das je ein Eintrag $i : j$ für R Container von Quellhafen i nach Zielhafen j enthalten ist.

Solange diese Liste Einträge nicht leer ist, wird wie folgt verfahren: Eine komplett leere Säule wird gewählt und möglichst nahtlos gefüllt. Komplett leere Säulen finden sich über den Schlüssel 1:N in der aggregierten Datenstruktur; es wird jeweils der erste Eintrag aus dem Stapel entnommen.

Nun wird aus der Liste der Transport mit dem spätesten Ende entnommen, im Zweifel der längste Transport, und in der Säule verplant. Für den Zeitraum vor diesem Transport wird aus der Liste der Transport mit dem spätesten Ende entnommen, der Transport muss aber komplett im noch nicht verplanten Bereich sein, und auch hier im Zweifel der längste Transport. Dies wird solange fortgesetzt, bis ab Hafen 1 der Säule Container zugeordnet sind, oder es keinen Transport gibt, der komplett in den noch nicht verplanten Bereich passt.

In diesem Verfahren werden solange weiter Säulen entnommen und Container diesen Säulen zugeordnet, bis die Liste der Transporte leer ist.

Während der gesamten Verarbeitung wird die aggregierte Datenstruktur gepflegt, in der vermerkt wird, zu welchen Transportschritten welche Säule nicht vollständig belegt sind.

Da der Zuordnungsalgorithmus dieses Schrittes kompakt arbeitet, ist sichergestellt das immer alle Container zugeordnet werden können. Weiter werden in diesem Schritt alle Container so zugeordnet, dass keine Umstapelungen anfallen.

Dieser Schritt erstellt optimale Ergebnisse, wenn alle Transporte in vielfachen von R Containern stattfinden, andernfalls kann es jedoch zu einem Ergebnis führen, in dem die weiteren Schritte Umstapelungen erzeugen müssen.

2.2.3.2 Zweiter Schritt

Der zweite Schritt versucht alle verbliebenen Container so zuzuweisen, das keine Umstapelungen notwendig sind.

Für einen Container von i nach j wird nach Säulen gesucht, die in den Transportschritten von i nach j nicht voll sind. Ausgewählt werden erst nur Säulen mit einem nicht vollen Bereich von a bis b für den mindestens gilt $a \leq i < j \leq b$.

Der erste Teilschritt wählt nur Säulen mit $j = b$, verlangt, dass mindestens ein Container bereits der Säule in den fraglichen Transportschritten zugeordnet ist, und erst Säulen mit größerem a ausgewählt sind. Es werden dann so viele Container dieser Säule für diesen Bereich zugewiesen, aber nur maximal soviele wie freie Zellen in der Säule sind, und nur soviele Container wie von i nach j noch nicht zugeordnet sind.

Die agreggierte Datenstruktur wird daraufhin aktualisiert, für i bis j wird die Säule nur vermerkt, wenn noch Zellen frei sind, für a bis i wird die Säule als nicht voll vermerkt (nur falls $a < i$).

Der zweite Teilschritt entspricht dem ersten, jedoch werden auch Säulen mit $j < b$ berücksichtigt und die Säulen dürfen für diesen Bereich komplett leer sein. Es werden erst Säulen mit kleinerem b und dann mit größerem a ausgewählt.

Die aggregierte Datenstruktur wird ebenfalls aktualisiert wie gehabt, zusätzlich wird die Säule von j bis b als nicht voll vermerkt (nur falls $j < b$).

In beiden Teilschritten kann es passieren, dass keine Säule gefunden wird, in den ein Container von i bis j zugewiesen werden könnte. In diesem Fall verbleiben die Container in der Transportmatrix und werden im dritten Schritt zugewiesen.

Dieser zweite Schritt erzeugt auch keine Umstapelungen, da alle Container nur passenden Säulen zugewiesen werden.

2.2.3.3 Dritter Schritt

Der dritte Schritt geht nach gleicher Reihenfolge alle Zellen der Transportmatrix durch und weist alle verbliebenen Container freien Säulen nach Belieben zu.

Die Container können nicht von Quellhafen i bis j in der gleichen Säule ohne Umstapelung verbleiben, sonst wären die Container schon im zweiten Schritt zugewiesen worden.

Unter gegebenem Stauplan, der nicht modifiziert wird, ausser dass ein Container oben auf hinzugefüht wird, wird nun die optimale Kombination gesucht, um einen weiteren Contaier von Quellhafen i bis Zielhafen j unterzubringen. Optimal bedeuted natürlich, möglichst wenige Umstapelungen.

Die aggregierte Datenstruktur liefert dabei Fragmente mit einem Anfang a und einem Ende b für die es mindestens eine Säule gibt, die mindestens einen weiteren Container von a nach b ohne Umstapelung transportieren kann. Möglichst wenig Fragmente sollen zur Strecke i nach j ohne Lücke kombiniert werden. Überstehen und überlappen ist zulässig.

Mit Branch und Bound wird die optimale Lösung gesucht; die Umsetzung ist rekursiv implementiert. Unter allen verfügbaren Fragmenten sucht jeder Lauf der Funktion alle raus, die mindestens i bis $i + 1$ enthalten, sortiert diese nach höchstem Ende b zuerst, verwirft alle Fragmente $a1$ nach $b1$ wenn ein Fragment $a2$ nach $b2$ existiert mit $j \leq b2 \leq b1$.

Der Funktion wird der Bereich und eine Schranke übergeben, wieviele Fragmente maximal kombiniert werden dürfen. Die Schranke dient als bisher bester Wert und wird überschrieben, sobald ein Ergebnis mit weniger Fragmenten gefunden wird. Der Reihe nach werden alle Fragmente geprüft.

Wenn durch ein Fragment der gesamte Bereich abgedeckt werden kann, so wird das Fragment in die Liste der besten Lösungen eingetragen. War der Bestwert bisher grösser als Eins, so wurde die Liste der besten Lösungen vorher gelöscht und der Bestwert auf 1 gesetzt.

Für Fragmente, die den Bereich nicht komplett abdecken, wird, wenn der aktuelle Bestwert aber über eins liegt, die Funktion rekursiv aufgerufen. Als Bereich wird der bisher nicht abgedeckte Bereich übergeben, als Schranke der bisherige Bestwert minus eins.

Die Funktion liefert alle besten Fragmentkombinationen zurück. Wurden Fragmentkombinationen gefunden, die sich aus genausovielen Fragmenten zusammen setzen, wie es die Schranke erlaubt, so wird die Liste der besten Pfade um Kombinationen aus dem verwendeten Fragment und den zurückgelieferten Pfaden erweitert.

Wurden Fragmentkombinationen gefunden, die sich aus weniger Fragmenten zusammensetzen, so wird die bisherige Bestenliste gelöscht und alle Kombinationen aus dem verwendeten Fragment und den zurückgelieferten Pfaden eingetragen. Der Bestwert wird auf die Anzahl Fragmente in diesen Kombinationen reduziert.

Der initiale Aufruf der Funktion erfolgt mit dem Bereich von i bis j und der Schranke $j - i$. Diese Schranke ermöglicht eine Kombination, die den Container nach jedem Transportschritt ausläd. Der initiale Aufruf der Funktion findet immer ein Ergebnis.

Unter allen optimalen Kombinationen wird diejenige ausgewählt, die am wenigsten übersteht, also mit $b - a$ minimal.

Mit diesem optimalen Pfad wird nun ein Container zugeordnet. Für jedes Fragment von a nach b wird eine Säule der aggregierten Datenstruktur entnommen. Der Container wird der Säule entsprechend dem ersten Fragment von i bis b zugewiesen; allen weiteren Säulen entsprechend. Die aggregierte Datenstruktur wird aktualisiert, die Säule also falls $a < i$ für a bis i eingetragen sowie für i bis b, falls nicht komplett voll. Die letzte Säule wird zusätzlich für j bis b eingetragen, falls $j < b$.

Dieses Verfahren wird für jeden Container einzeln durchgeführt. Auch wenn der einzelne Container so bei gegebener Situation optimal untergebracht wird, kann dieser Schritt natürlich für im Ganzen das Gesamtproblem evtl. keine optimalen Ergebnisse liefern.

Kapitel 3

Genetische Algorithmen

3.1 Allgemeines

In Optimierungsverfahren[1] wird immer *eine* Lösung als Ausgangspunkt eines Verbessungsverfahrens oder als Ergebnis eines Konstruktionsverfahrens betrachtet. Um eine Lösung zu verbessern gibt es nun die Möglichkeit deterministische Regeln zu verwenden (z. Bsp. lokale Suche in der Nachbarschaft), probabilistische Regeln (z. Bsp. „Simulated Annealing“) oder bei einer Suche die Suchhistorie im Blick zu halten (z. Bsp. bei Tabu Suche). Nun wollen wir eine ganz Population von Lösungen verwenden und ein neues Verfahren auf diese Population anwenden (nicht nur die parallele Ausführung schon bekannter Verfahren).
Um neue Lösungen zu finden, verwendet man nun mehr als eine Ausgangslösung. Diese Verfahren nennt man genetische Algorithmen (GA), evolutionäre Stategien oder evolutionäres Programmieren. Diese Verfahren sehen allgemein so aus:

1. Erzeuge eine Anzahl von Individuen, welche zulässige Lösungen des vorliegenden Problems sind.
2. Berechne deren Zielfunktionswert.
3. Führe einen Selektionsdruck aus, um bessere Lösungen zu fördern oder schlechte Lösungen zu eliminieren.
4. Wende einige Variationsoperatoren auf die übriggebliebenen Lösungen an um neue Lösungen zum Testen zu finden.
5. Iteriere die Schritte 2.,3.,4.

Allgemein kann über das Verfahren gesagt werden:

- Änderungen in der Zielfunktion sind leicht zu beücksichtigen
- Das Verfahren muss bei einer Änderung der Nebenbedingungen oder Daten nicht wieder ganz neu gestartet werden.
- Es ist sehr leicht zu parallelisieren.

[1] Bem: Das Folgende ist stark geprägt von [22]

- Die Grenze zwischen Phänotyp (Lösung des betrachteten Problems) und der Genotyp (Codierung der Lösung) verschwindet hier schnell.

3.2 Das TSP

Wie wird nun ein GA auf ein Problem angewendet? Um dies verständlich zu machen, gehen wir von einem wohlbekannten Problem des Operations Research aus - dem TSP („travelling salesman problem“).

Definition 22 (TSP nach [11]) *Das Problem des Handlungsreisenden (TSP) ist für ein* $0 < n \in \mathbb{N}$ *gegeben durch eine nicht negative reelle* $n \times n$*-Matrix* $D = (d_{ij})$*. Es sei* S_n *die Menge der Permutationen der Zahlen* $\{1, \ldots, n\}$*. Gesucht ist nun ein* $\pi \in S_n$ *mit minimalem Wert*

$$c\pi := d_{\pi(n),\pi(1)} + \sum_{1 \leq i < n} d_{\pi(i),\pi(i+1)}$$

Eine Tour des Handlunsgreisenden für $n = 8$ kann man darstellen als $[1, 2, 6, 7, 3, 4, 5, 8]$ - die sogenannte Pfad Darstellung - und man interpretiert dies so: Er startet an Stadt 1 und geht zur Stadt 2 usw. bis zur Stadt 8 und von dort wieder zu Stadt 1. Eine andere Tour erhält man nun beispielsweise durch Tauschen zweier Städte (Transposition). Wie kann man nun aus zwei Touren als Eltern eine Kindtour (“offspring“) bauen, die Eigenschaften der beiden Elterntouren enthält? Eine Möglichkeit wäre eine simples Crossover, d.h. von der ersten Lösung die ersten k Städte und von der zweiten die restlichen $n - k$ Städte. Das sähe dann z.Bsp. so aus:

$$[\underline{1, 2, 3, 4}, 5, 6, 7, 8] \times [2, 1, 4, 3, \underline{8, 6, 5, 7}] \Rightarrow [1, 2, 3, 4, 8, 6, 5, 7]$$

Warum geht das schief? Die rekombinierte Tour (offspring) kann einige Städte mehrfach enthalten und andere garnicht (hier passt es zufällig).
Besser, d.h. ohne nach einer Rekombination das resultierende Kind einer „Reparatur“ unterziehen zu müssen, ist das sogenannte OX-Crossover („order crossover“):

1. Wähle zwei Abschnittpunkte i, j mit $1 \leq i < j \leq n$.
2. Kopiere die Einträge zwischen i und j der ersten Lösung P_1 als Beginn der Offspring-Lösung O_1.
3. Ergänze fehlende Städte in der Reihenfolge wie sie in der zweiten Lösung P_2 auftreten.

Dadurch wird eine zulässige Lösung erzeugt, die von P_1 einen Abschnitt hat und die restlichen Städt in der Reihenfolge erhalten bleiben wie in P_2. Ein Beispiel:

$$\begin{array}{l} P_1 = [1, 3, |5, 6, 2, |4, 7, 8] \\ \\ P_2 = [3, 6, 2, 1, 5, 4, 8, 7] \end{array} \quad \longrightarrow \quad O_1 = [5, 6, 2, 3, 1, 4, 8, 7]$$

Da wird einen genetischen Algorithmus für das r-CSP suchen, wollen wir uns noch eine andere Respräsentation des TSP ansehen - die Ordinaldarstellung. Gegeben sei eine geordnete Liste C der Städte. C dient als Referenzpunkt für die Ordinaldarstellung. Die Ordinaldarstellung ist eine Liste l von n Zahlen, wobei das i-te Element von l eine Zahl zwischen 1 und $n - i + 1$ ist. Listeneinträge l_i geben welchen Stadt aus C jeweils als nächstes eingeplant wird - wobei aus C jeweils die gewählten Städte gelöscht werden.

Hier ein Beispiel: Es sei $C = [1, 2, 3, 4, 5, 6, 7, 8, 9]$ und $l = (1, 1, 2, 1, 4, 1, 3, 1, 1)$, dann ist die zugehörige Tour T=$[1, 2, 4, 3, 8, 5, 9, 6, 7]$. Der Vorteil der Ordinaldarstellung ist nun, dass das simple Crossover funktioniert: Man kann einen Teil von P_1 nehmen und den restliche dann von P_2. Dann ergeben sich für $C = [1, 2, 3, 4, 5, 6, 7, 8, 9]$:

Kodierung	Tour in Pfaddarstellung
$P_1 = (1, 1, 2, 1, \|4, 1, 3, 1, 1)$	$[1, 2, 4, 3, 8, 5, 9, 6, 7]$
$P_2 = (5, 1, 5, 5, \|5, 3, 3, 2, 1)$	$[5, 1, 7, 8, 9, 4, 6, 3, 2]$

(wobei | den Crossoverpunkt markiert) die folgenden Nachkommen:

Kodierung	Tour in Pfaddarstellung
$O_1 = (1, 1, 2, 1, 5, 3, 3, 2, 1)$	$[1, 2, 4, 3, 9, 7, 8, 6, 5]$
$O_2 = (5, 1, 5, 5, 4, 1, 3, 1, 1)$	$[5, 1, 7, 8, 6, 2, 9, 3, 4]$

Was ist hier jedoch das Problem? Wenn man sich die Nachkommen in der Pfaddarstellung ansieht, so stellt man fest, dass von einem Elternteil viel übernommen wird und vom anderen eigentlich nichts.
Neben dem Crossover kann man sich auch das Erzeugen von Nachkommen aus mehr als zwei Eltern vorstellen (gibt es sowas in der Natur?). Die oben genannten Operatoren kann man auch auf nur ein Individuum anwenden, z. Bsp. eine Mutation, d.h. eine zufällige Änderung eines Teilabschnittes und dergleichen. Derart könnte sich dann auch für jeden Nachkommen eine lokale Suche anschliessen.
Im Falle des TSP ist die Evaluation einer Tour klar (entsprechende Distanzen aufsummieren) und auch die Größe der Population etc. sind gut einstellbare Parameter. Das Kapitel 8 in [22] gibt noch eine Vielzahl weiterer Darstellungsforum und Modifikationsoperatoren an. Der große Vorteil der genetischen Algorithmen liegt in ihrer Flexibilität und Parallelisierbarkeit. Nachteile sind meines Erachtens die Vielzahl der einstellbaren Parameter und die Undurchsichtigkeit des Optimierens - warum sollte das zu guten oder besseren Lösungen führen als ein anders Verfahren?

In der Tat gilt dazu ein

Satz 4 (NFL, [12]) *Es gibt keine Lösungsstrategie die allen anderen auf allen Problemen überlegen ist.*
Oder anders ausgedrückt:
Gemittelt über alle Kostenfunktionen haben alle Suchalgorithmen dieselbe Güte.

NFL heisst „No free lunch“ und gilt als eine der zentralen Aussagen der Optimierungstheorie (wenn auch mit wenig praktischer Bedeutung). Wenden wir uns nun den Stapelproblemen und der Anwendbarkeit der genetischen Algorithmen zu.

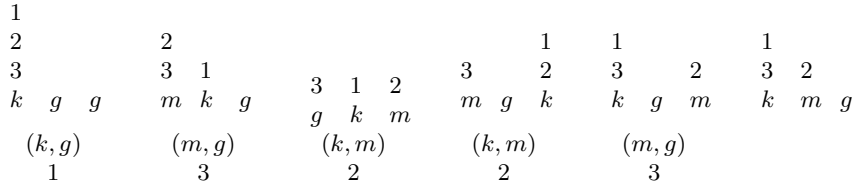
3.3 Turm von Hanoi

Da es beim rCSP und der Anwendung von genetischen Algorithmen darauf einige Fallstricke gibt, entwickeln wir nun zuerst einen genetischen Algorithmus für das TvH-Problem. Eine Möglichkeit wäre eine Codierung der Individuen als Folge der Zustände der drei Türme. Dort wäre durch Crossover schnell ein Nachkomme erzeugt, der nicht mehr durch legale Züge erreicht werden kann. Deshalb verwenden wir hier eine dynamische Repräsentation nach [20].

Die Kodierung soll relativ sein und nicht absolut. Deshalb geben wir die Zugfolge an und nicht die Zustände des Türme. Es gibt immer eine kleine Scheibe k, eine mittlere Scheibe m und grosse Scheibe g die oben-auf liegen (der leere Platz sei eine grosse Scheibe). Es gibt nun die Züge:

1. Kleine Scheibe k auf die grosse Scheibe g - Code 1
2. Kleine Scheibe k auf die mittlere Scheibe m - Code 2
3. Mittlere Scheibe m auf die grosse Scheibe g - Code 3

Mit diesen Abkürzungen lassen sich Zugfolgen nun einfach darstellen



Um die Zustände der Stangen zu erhalten, bedarf es natürlich einer Dekodierungsmethode um den Phänotyp zu erhalten. Diesen erhält man durch die Ausführung der Züge vom Anfangszustand aus. Im Gegensatz zu sonstigen genetischen Algorithmen ist die Anzahl der Züge und damit die Länge der Kodierung unbekannt bzw. wie gross müsste sie sein um eine optimale Zugfolge bzgl. der Zuganzahl zu ermöglichen? Natürlich $2^n - 1$. Problematisch ist auch die Bewertung einer Zugfolge - was ist eine gute, was eine schlechte Zugfolge? Gewöhnlich ist das kein grosses Problem bei der Formulierung eines GA. (Dazu gleich eine Übungsaufgabe: Wie kann man das TvH-Problem mit mehr Stangen auf diese Weise kodieren?)
Die Zielfunktion, d.h. das Bewerten einer Zugfolge z sollte monoton wachsend sein mit „Näherkommen“ an den optimalen Zustand. Eine Möglichkeit ist

$$F(z) = 1 + \alpha^{n_3(z)} \quad \text{mit} \quad n_3 = \text{Anzahl der Scheiben auf Stange 3}$$

Sowas kann man weiter verbessern durch

$$F(z) = \sum_{i=1}^{n} A(i,p) \cdot n \cdot 10^{n-i)}$$

wobei $A(i,p)$ ein Gewichtsmatrix mit Anzahl der Scheiben Zeilen und Anzahl der Stangen Spalten ist. Zum Beispiel ein Gewicht gemäß der Dimension der jeweiligen Scheibe. Dies sind alles wenig monotone Funktionen.

3.4 Genetische Algorithmen für das r-CSP

Nach diesem Einführungen wollen wir nun versuchen einen genetischen Algorithmus für das r-CSP zu finden.

3.4.1 Vollständige Codierung

In [29] wird die vollständige Codierung vorgeschlagen. Die Codierung ist ein Vektor v der Länge $N \cdot P$ mit N = Anzahl der Häfen und $P = R \cdot C$ die Stellplätze im Schiff. Der Vektor v

hat N Abschnitte, und in jedem Abschnitt steht der Stauplan bei Ausfahrt aus dem Hafen. Die Stellplätze sind von unten nach oben und spaltenweise geordnet.
Die Autoren integrieren vier Ziele: Minimiere überstaute Container, minimiere die longitudinalen und transversalen Momente und Trennung gewisser Container im Schiff (Gefahrgut u.a.). Die Auswahl der guten Lösungen findet nun anhand der Pareto-Optimalität statt.
Ohne nun weiter darauf einzugehen, stellen wir die folgenden Dinge fest:

- Die Codierung hat extrem lange Vektoren. Zum Beispiel bracht ein Schiff mit 1500 Stellplätzen und 15 Zielhäfen einen Vektor der Länge 22500. Da der GA viele Lösungen in einer Population hat, ist dieser sehr speicherintensiv.
- Die Evaluation der Ziele ist sehr aufwändig.
- Die Codierung berücksichtigt nicht, daß die Container konsistent zu laden sind. Es werde keine drastischen Staupланwechsel zwischen den Häfen verhindert.
- Das einfach Crossover erzeugt sehr schnell unzulässige Staupläne. Deshalb muss ein Reparaturmechanismus eingebaut werden.

Aus diesen Gründen ist die Suche nach einer Verbessung nötig. Diese vollständige Codierung entspricht ein wenig der Pfaddarstellung des TSP. Letztlich ist es aber beim rCSP so, daß es am Problem selbst liegt, daß eine Lösung durch die Aussmaße des Schiffes und die Anzahl der Häfen bestimmt bestimmt wird. Geht es nun irgendwie besser? Kann die Codierung des TvH aus 3.3 übertragen werden?

3.4.2 Genetischer Algorithmus mit kompakter Codierung

Wir versuchen nur die Änderungen eines Stauplans von Hafen zu Hafen zu verwenden. Da viele Container lange an Bord bleiben, wird der codierte Vektor kleiner und die Rechenzeit damit auch. Wir folgen hier nun dem Ansatz aus [8].

3.4.2.1 Kompakte Codierung

Definition 23 (Kompakte Codierung) *Die kompakte Codierung des rCSP mit N Häfen und einem Schiff mit Laderaum $R * C$ sei gegeben durch einen Vektor $\mathbf{v}$ mit $v_i \in \{1, \dots C\}$, der aus N Sektionen besteht. Jede Sektion wiederum besteht aus vier Abschnitten n, f, q, g.*

Was bedeuten diese Abschnitte des Vektors $\mathbf{v}$ nun?
An einem Hafen k, das heisst der k-te Abschnitt bedeuten die Abschnitte n, f, q, g:

- n ist eine Liste von Spalten, in die Container am Hafen k eingeladen werden.
- q ist eine Liste von Spalten, aus denen Container ausgeladen werden sollen wegen wünschenswerter Umstapelungen (Ordnen)
- f ist eine Liste von Spalten, in die Container eingeladen werden, die wegen Überstauung ausgeladen wurden („necessary shifts“)
- g ist eine Liste von Spalten, in die Container eingeladen werden, die wegen wünschenswerter Umstapelungen („voluntary shifts“) ausgeladen wurden.

q und g korrespondieren, das heisst die entsprechenden Einträge beziehen sich jeweils auf denselbe Container (Aus- und Einladespalte).
Wie gewinnt man nun einen Stauplan aus $\mathbf{v}$? Dazu werden noch weitere Listen benötigt:

- Die Hafenwarteliste W_p an einem Hafen p sei eine Liste von Containern mit ihres Zielhäfen. Zu Beginn besteht W_p aus den Containern aus der Transportmatrix.
- Die Spaltenwerteliste w_{pr} an einem Hafen p enthält Container, die in Spalte r eingeladen werden.

Einen Stauplan bei Ankunft in Hafen p kann man nun so gewinnen:

1. Entlade alle p-Container und die sie übestauenden h-Container. Hänge diese an W_p.
2. Entlade aus den Spalten, die in q spezifiziert sind Container. Und zwar von oben und so oft, wie die entsprechende Spalte in q auftritt.
3. Füge diese Container aus Schritt 2 gemäß Teilvektor g den entsprechenden w_{pg} zu.
4. Füge die Container aus W_p den Spaltenwertlisten w_{pr} zu, die in der hintereinandergehängten Listen von n und f angegeben sind.
5. Lade alle Container der Spaltenwertlisten geordnet in die entsprechenden Spalten.

Was kann hier schieflaufen? Es kann sein, dass in einer Spaltenwertlisten mehr Container gelistet sind, als dort noch hineinpassen. Dann sollen diese Container der nächsten Spalte mit noch freien Plätzen hatzugeordnet werden. Dabei entspreche Spalte $C + 1$ der Spalte 1.

Was ist sonst noch zu bemerken?

- Der Vektorteil f sollte eine Länge haben, so dass die größtmögliche Zahl notwendiger Umstauungen untergebracht werden kann. Deshalb enthält f redundante Einträge, die in der Bestimmung der Staupläne nicht verwendet werden .
- Die größtmögliche Anzahl von wünschenswerten Umstauungen ist durch die Länge von q bzw. g gegeben.
- Sollte eine Spalte in q ganz leer werden und weiterhin wünschenswerte Entladungen vorgenommen werden, dann übergehe diese Enträge in q.

3.4.2.2 Crossover

Das Crossover funktioniert bitweise und nur mit einer gewissen Wahrscheinlichkeit: Seien v_1 und v_2 zwei (Eltern-)Lösungen. Dann wird ein Gen (Vektorelement) der (Kind-)Lösung v entweder mit der entsprechenden Spaltennummer von v_1 oder von v_2 besetzt.

3.4.2.3 Parameter

Die Populationsgröße ist in [8] mit 200 angegeben und die Länge des Vektors q wird abgeschätzt.
Bleibt der Zielfunktionswert der Population (wie bekommt man den?) eine gewisse Anzahl von Iterationen lang unverändert, so werden viele Individuen aus der Population entfernt und neue Individuen zufällig erzeugt, um neues genetisches Material in die Population einzubringen.

3.4.2.4 Beispiel

Sei ein Schiff gegeben mit dem folgenden Stauplan bei Einfahrt in Hafen 1 (Container seien durch ihres Zielhafen benannt):

2		3
2	3	2
5	5	2
3	**2**	**1**

$W_2 = (4, 5, 5)$ und der entsprechende Abschnitt für Hafen 2 des Individuums sei

Hafen 2								
n			**f**		**q**		**g**	
3	2	1	1	$\cdots$	3	2	1	3

Nun werden erst alle 2-Container abgeladen und deshalb auch ein 3-Container, der zur Menge W_2 hinzugefügt wird: $W_2 = (4, 5, 5, 3)$. Nun sieht der Laderaum des Schiffes so aus:

	3	
5	5	
3	**2**	**1**

Als nächtes werde die zwei wünschenswerten Umstauungen vorgenommen: Der oberste 5-Container in Spalte 3 kommt nach Spalte 1 und der oberste 3-Container von Spalte 2 nach Spalte 3. Also sehen die Spaltenwertelisten w_{pr} so aus:

$$w_{21} = (5), w_{22} = \emptyset, w_{23} = (3)$$

und der Laderaum hat folgendes Aussehen:

	5	
3	**2**	**1**

Nun werden die Container der Hafenwarteliste W_2 gemäß der Einträge in n und f verteilt. Das heisst der erste Container in W_2 kommt nach w_{23}, der zweite nach w_{22} usw. Danach sehen die Spaltenwertelisten so aus:

$$w_{21} = (5, 3, 5), w_{22} = (5), w_{23} = (3, 4)$$

Jetzt werden die Container geordnet in ihre entsprechenden Spalten geladen und es ergibt sich der folgende Stauplan:

		3
3	5	5
4	5	5
3	**2**	**1**

Nun noch ein Beispiel für ein Crossover:
Sei $P_1 = (3, 2, 1, |1, 2, 3|3, 2, |1, 3)$ und $P_2 = (3, 2, 1, |2, 3, 1, |1, 2, |2, 1)$, dann ergeben sich solche Kinder $C_1 = (3, 2, 1, |2, 2, 3, |3, 2, |2, 1)$ oder $C_2 = (3, 2, 1, |2, 2, 1, |3, 2, |2, 3)$

Kapitel 4

Stauraumplanung realer Schiffe

Bisher haben wir ein Primitivcontainerschiff betrachtet, daß einen rechteckigen Laderaum hat und dessen Container lediglich durch einen Quell- und einen Zielhafen bestimmt waren. Nun wollen wir sehen, wie reale Containerschiffe sich davon unterscheiden und wie eine Stauraumplanung dort möglich wird.

4.1 Ein erstes zweiphasiges Verfahren

Dazu folgendes, was den Bau eines Containerschiffes berücksichtigt:

> „ (...), the cargo space of a containership is made up of a number of *bays*, collections of stacks of containers along the length of a ship. Generally, each bay in the cargo space is divided into *above-deck* and *below-deck* by *hatch covers*, and sub-areas of a bay divided by hatch covers are called *holds*. Each hold is composed of a group of *stacks*, and each stack is composed of vertically arranged groups of *cells*. Each cell is a physical location or a slot where a container is to be loaded. Contaiers loaded below-deck can be unloaded only after all containers loaded above-deck on the hatch cover above are removed as well as the hatch cover.“ (p.415 in [15])

Im folgenden versuchen die Autoren folgendes:

> „ For the problem, we develop a heuristic solution method in which the problem is divided into two subproblems, one for assigning container groups to the holds and one for determining a loading pattern of containers assigned to each hold. The former subproblem is solved by a greedy heuristic based on the transportation simplex method, while the latter is solved by a tree search method. These two subproblems are solved iteratively using information obtained from solutions of each other“ (p.416 in [15])

Wir suchen einen Stauplan, der die Lade- und Löschzeit für ein Containerschiff minimiert und dabei die technische Stabiltät des Schiffes erhält. Kang und Kim gehen in [15] nun trotzdem noch von gewissen Vereinfachungen aus:

1. Containers to be delivered on the container ship are of the same size (40ft standard container)
2. The route of the ship's tour is given.
3. The number of containers to be delivered from one port to another is known (at the starting point, i.e. the port at which the stowage plan is to be made) for all pairs of ports included in the tour.
4. The initial stowage pattern of the ship at the starting point is given.
5. The number of containers to be loaded does not exceed the capacity of the ship at any port.
6. The unit cost (time) related to crane movements is the same at all ports, and the time required to handle an overstow is the same at all ports.

(p.418, [15])

Definition 24 *Wir legen nun fest, dass ein Containerschiff aus Laderaumabteilen („bays") besteht und diese wiederrum aus Laderäumen („holds"). Es gibt pro Laderaumabteil sechs Laderäume und zwar je drei über-deck und drei unter-deck. Diese sind durch Lukendeckel („hatch cover") getrennt.*

„ It is assumend that ships with the capacities 2500, 300 and 4000 TEU have 12,15, and 20 bays. Each bay of the ships consists of three hatch covers and six holds." (p.434 in [15])

4.1.1 Vorbemerkungen

Es werden nun für jeden Hafen Gruppen-zu-Frachtraum-Zuordnungen gesucht (group-to-hold). Diese werden aus einer MIP-Formulierung gewonnen, welchen die aus einer Zuordnung zu erwartenden Zeitaufwand durch Überstauungen und Kranbewegungen abschätzt. Es werden Containergruppen identifiziert, die aus Quell-, Zielhäfen sowie grober Gewichtsklassen gebildet werden.

[15]: (p.418) Here, a container group denotes a set of containers with the same source (port of loading: POL), the same destination (port of destination: POD), and the same weight.

Es werden die folgenden Parameter geschätzt:

- a_{ij}: Zahl der notwendigen Umstauungen, wenn ein Container der Gruppe i dem Frachtraum j zugeordnet wird.
- b_{ij}: Zahl der Umstauungen an späteren Häfen, wenn ein Container der Gruppe i dem Frachtraum j zugeordnet wird.
- d_{ij}: Zeit, die benötigt wird für Kranbewegungen am Zielhafen eines Container der Gruppe i, wenn ein Container der Gruppe i dem Frachtraum j zugeordnet wird.

Diese Parameter sollen nun geschätzt werden (p.419 in [15]):

Schätzung für a_{ij}:

Fall 1: Wenn Frachtraum j über-deck ist, sei $a_{ij} = 0$.

Fall 2: Wenn Frachtraum j unter-deck ist, sei $a_{ij} = N_j$, wobei N_j die Zahl der (geladenen) Container im Frachtraum j über-deck ist.

Schätzung für b_{ij} Es sei n_T die Zahl der leeren Stellplätze eines Frachtraums und n_N die Zahl der leeren Stellplätze über Containern, deren Zielhäfen näher sind, als die Zielhäfen von Containern der Gruppe i. So werden nun die b_{ij} geschätzt:

Fall 1 Wenn Frachtraum j über-deck ist, dann sei j' der zughehörgige Frachtraum unter-deck. Dann

Fall 1a Wenn in Frachtraum j' kein Container mit Zielhafen näher als Zielhafen eines Containers der Gruppe i ist, dann sei $b_{ij} = \frac{n_N}{n_T}$.

Fall 1b Wenn in Frachtraum j' Container mit Zielhafen näher als Zielhafen eines Containers der Gruppe i ist, dann sei $b_{ij} = 1$

Fall 2 Wenn Frachtraum j unter-deck ist, sei $b_{ij} = \frac{n_N}{n_T}$.

Schätzung für d_{ij}: Sei T_S die Zeit, die ein Kran braucht um einen Container umzustauen und T_C, die ein Kran braucht um zwischen zwei Laderäumen zu wechseln. Dann sei $c = \frac{T_C}{T_S}$. Nun zur Schätzung der d_{ij}.

Fall 1 Wenn in Laderaumabteil B_j (bay) kein Container mit gleichem Zielhafen wie der Zielhafen der Container der Gruppe i, dann sei $d_{ij} = c$.

Fall 2 Wenn in Laderaumabteil B_j (bay) Container mit gleichem Zielhafen wie der Zielhafen der Container der Gruppe i, dann sei $d_{ij} = 0$.

Nun folgen die Bezeichner für die benötigten Indzies und Parameter, sowie Variablen:

Verwendete Indizies

- i: Index der Container-Gruppe, $i = 1, \ldots, I$
- j: Index des Laderaums (hold), $j = 1, \ldots, J$
- k: Index des Hafens, $k = 1, \ldots, K$
- q: Index des Laderaumabteils, $q = 1, \ldots, Q$

Gegebene Parameter

- D_i: Anzahl der Container der Gruppe i im betrachteten Hafen
- H_q: Menge der Laderäume im Laderaumabteil q
- B_j: Laderaumabteil, in dem Laderaum j ist
- e_j: Anzahl der leeren Stellplätze im Laderaum j
- E_q: Anzahl der leeren Stellplätze im Laderaumabteil q, d.h. $E_q = \sum_{j \in H_q} e_j$

- W_i: Gewicht der Container der Gruppe i
- X_j, Y_j, Z_j: Die longitudinale, transversale und vertikale Koordinate des Mittelpunktes von Laderaum j.
- c: vereinheitlichte Zeit um einen Krane von einem Ladeteilraum zu einem anderen zu bewegen
- l_L, u_l: untere bzw. obere Grenze des longitudinalen Moments des Schiffes
- u_T: obere Schranke des transversalen Moments (Quermoments) des Schiffes
- u_v: obere Schranke der vertikalen Position des Schwerpunktes („centre of gravity“) des Schiffes

Variablen

- x_{ij}: Zahl der Container der Gruppe i in Laderaum j
- y_{ij}: Binärvariable, die den Wert 1 annimmt, wenn ein Container der Gruppe i in Laderaum j ist und sonst den Wert 0 hat.
- z_q: Binärvariable, die den Wert 1 annimmt, wenn ein Container der Gruppe i in Teilladeraum q ist und sonst den Wert 0 hat.

4.1.2 Gruppen zu Laderäumen

Es ist nun möglich für einen Hafen k das folgende ganzzahlige lineare Programm $[GH_k]$ zu formulieren:

$$Min \sum_i \sum_j a_{ij} y_{ij} \sum_i \sum_j b_{ij} x_{ij} + c \sum_q z_q + \sum_i \sum_j d_{ij} y_{ij}$$

Verschiffungsmenge $(i = 1, \ldots, I)$:

$$\sum_j x_{ij} = D_i \tag{4.1}$$

Laderaumkapazität $(j = 1, \ldots, J)$:

$$\sum_i x_{ij} \leq e_j \tag{4.2}$$

Verknüpfung x **und** y $(i = 1, \ldots, J, i = 1, \ldots, I)$:

$$x_{ij} \leq e_j y_{ij} \tag{4.3}$$

Verknüpfung y **und** z $(q = 1, \ldots, Q)$:

$$\sum_i \sum_{j \in H_q} y_{ij} \leq E_q z_q \tag{4.4}$$

Krängungsbedingung:

$$-u_t \leq \sum_i \sum_j W_i Y_j x_{ij} \leq u_T \tag{4.5}$$

Schwerpunktstabilität:

$$\sum_i \sum_j W_i Z_j x_{ij} \leq u_v \tag{4.6}$$

Trimmstabilität:

$$l_L \leq \sum_i \sum_j W_i X_j x_{ij} \leq u_L \tag{4.7}$$

Variable x $(i = 1, \ldots, I, j = 1, \ldots, J)$:

$$x_{ij} \geq 0 \text{ und ganzzahlig} \tag{4.8}$$

Variable y $(i = 1, \ldots, I, j = 1, \ldots, J)$:

$$y_{ij} \in \{0, 1\} \text{ und ganzzahlig} \tag{4.9}$$

Variable z $(q = 1, \ldots, Q)$:

$$z_q \in \{0, 1\} \text{ und ganzzahlig} \tag{4.10}$$

Die Zielfunktion minimiert die Zeiten um die Überstauungen zu verarbeiten an Hafen k und den folgenden Häfen und die Kranbewegungen. Die $[GH_k]$ sind ganzzahlige Programme für die Häfen. Diese werden nun in [15] heuristisch gelöst:

> „ In this heuristic, an initial solution is obtained in a form of the basic feasible solution of the transportation problem after the objective function of $[GH_k]$ is approximated to a linear form. The the solution is improved with a method similar to the pivot operation used in the transportation simplex method“ (p.420 in [15])

Es ist zu bemerken, dass $[GH_k]$ einem Transportproblem mit festen Ladungen ähnelt. (Das ist in dem Zitat gemeint.)

4.1.3 Container zu Stellplätzen

Aus obigen ergeben sich nun für jeden Hafen die Zuordnungen von Gruppen zu Laderäumen. Jetzt wird versucht, die einzelnen Container dieser Gruppen den Stellplätze zuzuweisen und zwar unter Vermeidung von Umstauungen. Stabilitätsaspekte brauchen nicht mehr beachtet werden, da nur innerhalb eines Laderaums gestaut wird und die Stabilität durch eine Lösung der obigen MIPs eingehalten wird.
Für jeden Laderaum wird nun eine Baumsuche durchgeführt um alle möglichen Konfigurationen untereinander zu vergleichen. Im Suchbaum repräsentieren die Knoten die relativen Positionen der Containergruppen in einem Laderamum und die Tiefe eines Astes die Häfen. Es werden zudem Dummy-Container eingefügt um die leeren Stellplätze ggf. aufzufüllen. (siehe [15] auf p.422). Diese Darstellung bleibt übersichtlich, da die folgenden Regeln eingehalten werden:

- Container mit demselben Start- und Zielhafen (POL und POD) kommen in adjazente Stellplätze. Deshalb genügt die Angabe der horizontalen Position der Gruppe.
- Müssen verschiedene Container in dieselbe Spalten geladen werden, so lädt man den mit grösserem Zielhafen unter den mit kleinerem Zielhafen.
- Bei Containern desselben Ziels werden die schweren Container unter die leichten Container platziert.

Ein Beispiel hierzu wurde in der Vorlesung gegeben

4.1.4 Verbesserungsverfahren

Da bei dem „Gruppen zu Laderäumen“-Verfahren keine zukünftigen Häfen beachtet wurden, kann es zu sehr schlechten Stauplänen kommen. Deshalb werden die Gruppen zum Teil neu gebildet und damit ein iterativer Prozess in Gang gesetzt. Und zwar werden die Koeffizienten b_{ij} modifiziert und das „Gruppen zu Laderäumen“-Verfahren nochmal gerechnet und auch neue Zuordnungen der Container-zu-Stellplätzen (wenn etwas verbessert wurde?!).

Die Modifikation der b_{ij} für einen bestimmten Hafen ist wie folgt:

$$b_{ij}^{\text{neu}} = b_{ij} + \frac{n_O}{n_X}$$

mit

n_O = Zahl der Umstauvorgänge aufgrund überstauter Container und

n_X = Zahl der Container der Gruppe i in Laderaum j in den betrachteten, vorhergehenden Häfen

4.2 Ein zweites zweiphasiges Verfahren

Wilson [32] stellt in seiner Dissertation ein anderes zweiphasige Verfahren vor, das dann in einer Menge von Aufsätzen [33, 34, 35, 17] von verschiedenen Gesichtspunkten beleuchtet wird. Zuerst werden die Container jedes Hafens in Gruppen eingeteilt, dann diese Gruppen den Hatches zugeordnet und zwar so, dass n verschiedene Lösungen entstehen, dann werden einige davon ausgewählt und die Planung des nächsten Hafens berechnet. Dort wird wieder nach Lösungen gesucht usw. Insgesamt wird der so entstehende Baum gegen Ende weniger breit, weil immer mehr Lösungen weggelassen werden.
Das folgende kann geradezu als die zentrale Modellierung eines Containerschiffes gesehen werden:

> „Container-ships can be seen as consisting of Bays, Stacks and Tiers. Combining these entities gives a location of a single cell. Cells can be either on deck or below deck. Cells below deck are acessed via hatch-lids. Cells on deck are removed to reveal the hatch-covers needed to access cells below deck. Bays are grouped below deck into compartments. Compartments are separated by bulkheads. “ (Seite 292 in [32])
> Anmerkung: „bulkhead“ heißt Schott.

4.2.1 Strategische Phase

Das Schiff soll nun wiederum in gröbere Strukturen als einzelne Stellplätze unterteilt werden. Man kann zum Beispiel jeweils die Spalten („stacks“) nehmen. Dazu jedoch:

> „(...) reduced by representing the cargo-space as a set of stacks that relate to hatch-lids, the state-space is still unacceptably large when considering a multi-destination voyage. Therefore, it was determined that the model of the containership's cargo-spce usied during the strategic planning phase requires further abstraction:“ (p. 235 in [32])

Also brauchen wir eine gröbere Einteilung.

> „Blocking stacks of cells that share a common relationship to a hatch-lid.(...)“ (p.236 in [32])

Trotzdem wird es in der Tat noch gröber gemacht:

> „Firstly, the stowage location for a container can be specified by its longitudinal position. Secondly, the location can be made more specific by choosing its latitudinal position.(...) Blocking cargo longitudinally by hatch means that a location of a container is specifiied only by hatch lid (i.e. as being either above or below a particular hatch-lid).“ (p.240 in [32])

Diese longitudinale Planung ist ausreichend um die maximale Auslastung der Beladungskräne zu berücksichtigen. Auch reicht sie um das Durchbiegen („bending“) und den Trimm mit hinreichener Genauigkeit zu beachten. Auch gelingt hier die Planung von speziellem Gut.
Wie geht das nun genauer? Es werden alle Container in Klassen von Containern eingeteilt (es steht da im Text, daß man das wie in [25] macht). Dann werden erst grosse, dann kleinere Mengen der nach Kriterien sortierten Containergruppen eingeplant. Die Zuordnung wird nach folgenden Gesichtspunkten durchgeführt (p.208 in [32]):

- Schwere Container unter leichte Container
- Container vom Inneren des Schiffes nach außen einplanen
- Container mit dem weitesten Zielhafen nach unten
- Container mit dem gleichen Zielhafen zusammen in einen Stack bzw. in einen Laderaum
- und noch einige Trennungsbedingungen („segregation“) der Gefahrgutcontainer

Ab jedem Hafen wird für jede Zuordnung des vorherigen Hafens ein Pool neuer Lösungen erzeugt. An weiteren Häfen wird die Anzahl der berechneten Lösungen weniger um den Baum klein zu halten. Zum Beispiel im ersten Verzweigungsschritt zehn Zweige, dann von jeden nur fünf Verzweigungsschritte usw.
Jede dann erzeugte Lösung wird bewertet nach der Kranauslastung, der Anzahl der Lukenbewegungen und der Anzahl der Überstauungen. Dies wird berechnet durch eine Simulation des Ein- und Ausladens an jedem Hafen. Knoten, die zu unzulässigen Beladungsplänen führen, werden abgeschnitten.
Nun geht es weiter:

> „Given the containers allocated to each longitudinal block the planning process can now move to considering which block within each hatch the containers should be placed in. That is, it becomes necessary to distribute the containers between latitudinal blocks associated with each longitudinal block“ (p. 242 in [32])

Hier kann die laterale Verteilung von Gewicht akzeptabel berechnet werden und zu einem gleichmäßigen Kiel („even keel“) gebracht werden.
Jeder Container eines „Hatches“ wird nun mit Branch-and-Bound einem bestimmten Block zugeordnet, wobei eine Zuordnung bewertet wird nach

- Maximierung der Laderaumausnutzung
- Minimierung der Lukendeckelbewegung
- Minimierung der Überstauungen.

Die Krängung („heel“) eines „Hatches“ wird lokal minimiert indem die Gewichtsverteilung querschiffs („athwartships“) so ausgeglichen wie möglich gehalten wird. Gewisse Container mit eingeschränkten Stellplatzmöglichkeiten verringern den Lösungsraum weiter.

4.2.2 Taktische Phase

Dann folgt die taktische Phase. Hier wird von der besten in der strategischen Phase gefundenen Lösung ausgegangen. Als erstes werden die Container nach Größe (Größte zuerst), Ziel (Weiteste zuerst) und Gewicht (Schwerste zuerst) sortiert. Je nachdem, ob es sich bei dem Block um eine hafenseitigen, steuerbord oder zentralen Block handelt, wird in einer anderen Reihenfolge gereiht. Die **Ladeheuristik** sieht nun so aus (alles nach p.259f in [32]):

1. Wähle eine Aufstapelreihenfolge je nach Block:
   - hafenseitig: Stapel von unten nach oben, vom Heck zum Bug, von rechts nach links
   - zentral: Stapel von unten nach oben, vom Heck zum Bug, von innen nach außen
   - steuerbord: Stapel von unten nach oben, vom Heck zum Bug, von links nach rechts
2. Wähle den ersten noch leeren Stapel
3. Wähle den nächsten Container c der Liste
4. Gibt es noch Standardcontainer, so ordne c dem Stapel zu dessen oberster Container den gleichen oder einen späteren Ziehafen hat. Versuche den Stapel mit weiteren Containern der selben Klasse aufzufüllen.
5. Gibt es noch Standardcontainer, so wiederhole 3
6. Für übergroße Container suche einen (vollen) Stapel dessen oberster Container den gleichen oder einen späteren Ziehafen hat. Vertausche den obersten Container und den übergroßen Container, d.h. der obersten Container kommt in die Liste zurück.
7. Wenn es weitere übergroße Container gibt, so gehe zu 6.
8. Gibt es weitere Container, so gehe zu 3.

Dadurch werden Überstauungen minimiert, durch übergroße Container erzeugter Leerraum minimiert, schwere unter leichte Container platziert und die Stapel im Allgemeinen nur von einer Klasse gebildet.
Es werden die drei Beladeheuristiken unterschieden:

- Fülle auf vom Heck zum Bug, von innen nach außen, von Stapel zu Stapel
- Fülle auf vom Heck zum Bug, von innen nach außen, Reihe für Reihe - zuerst die inneren Stapel und wenn die Stapel voll sind nach außen
- Fülle auf vom Heck zum Bug, von innen nach außen. Beginne einen neuen Stapel bei jedem neuen Zielhafen

(Erklärendes Bild in der Vorlesung...)

Phase	Ziel	Nachbarschaft
1	Minimiere Überstauungen	Alle Vertauschungen
2	Gleichmässige Höhe der Stapel	Vertauschugen von Containern gleichen Zielhafens
3	Schwere Container unter leichte Container	Vertauschungen im gleichen Stapel
4	Gewichtsverteilung	Vertauschen von Containern gleichen Zielhafens

Tabelle 4.1: Phasen der Tabu Suche

Es wird nun nach einer der drei Heuristiken eine genaue Zuordnung der Container zu den Stellplätzen gebildet.

4.2.3 Verbesserungsverfahren

Die beste Lösung der taktischen Phase dient als Ausgangspunkt eines Optimierungsverfahrens - einer Tabu Suche und zwar in den cargo-space blocks („holds“). Es werden für die verschiedenen Ziele verschiedene Umgebungen („neighbourhoods“) definiert (siehe 4.1).

> „Since a typical cargo-space block will hold approximately 12-60 TEU'2, finding an optimum solution given the above criteria is a relatively simple combinatorial task.

Diese Tabu Suche ist notwendig, weil es noch einige unzulässige Beziehungen geben kann, wie Gefahrgutcontainertrennung oder Zuordnung von zu elektrifizierenden Containern. Ein menschlicher Planer vertauscht solange Container bis alle Bedingungen erfüllt sind. Tabu Suche adaptiert dieses Verfahren.
Tabu Suche geht von einer Lösung aus und transformiert sie in eine andere und bemisst die Verbesserung. Insofern ähnelt sie der lokalen Suche. Jedoch erlaubt die Tabu Suche auch eine Verschlechterung des Zielfunktionswertes, wobei die Transformation zur alten Lösung dann zum Beispiel nicht erlaubt wird. So kann es gelingen lokalen Optima zu entkommen.
Die Güte eines (lokalen) Stauplans wird bemessen nach Überstauungen, Anzahl gemischter Stapel, Anzahl leichter Container unter schweren Containern und Gewichtsverteilung. Dies kann man nun mit einer gewichteten Zielfunktion machen oder einem Verfahren in verschiedenen Phasen, das jeweils andere Kriterien aber auch Nachbarschaften verwendet. Ein Vertauschen (Move) ist hier das Vertauschen von zwei Containern bzw. einem großen und zwei kleinen Containern oder Versetzen eines Containers an einen anderen Stellplatz

Hier die Zitate des Orginals:

> „For the container stowage problem, s is the stowage configuration for the entire container-ship and $N(s)$ is the set of all configurations obtained by making moves within a single hatch, with each hatch being optimised separately. Here a move is the swapping of the content a of location p with the contents b of location q (where the content of one location can be 'empty')“(p.1254 in [34])

> „For the model under consideration, the following are considered salient:
>
> - Re-handles are to be minimised;

- Container weight is to be graded upwards in the cargo space, heaviest to lightest;
- Stacks (vertical collections of containers) with mixed POD are to be minimised “

(p.1253 in [34]

4.3 Kommentare aus der Praxis

Wie machen es nun die Praktiker? Dazu die folgenden Zitate

> „ In fact, up to 30 percent of the stacks contain containers of different type “ (p. 734 in [26])

Verfahren in der Praxis:

> „Nowaday, a dispatcher susequently assigns export containers in inverse order of ports to be visited. First, he chooses a bay. Then, he marks all free positions for containers of the currently considered type. For these posistions, the decision support system offers a list of not yet assigned export containers of that type. The dispatcher selects some containers from that list. The final assignment is determined by a simple heuristic in accordance woth a specified loading strategy and with regard to container weights.“ (p.734f in [26])

> „By now, a stowage plan is generated ignoring loading and transport sequence. In particular, containers are assigned to bay positions without consideration of the necessary transportation times between storage positions in the yard and the quay cranes“ (p.735 in [26])

> „It is asumed that in practice, ballast will be used, if necessary, to bring the stowage pattern to within tolerable levels.“ (p. 254 in [32])

4.3.1 Eine Variante

Eine Variante des letzten Verfahrens ist in [17] gegeben. Hier einige Auszüge:

> „Strategic planning allocates generalised containers to a blocked cargo-space in which slots corresponding to hatch-lids are grouped together. Tactical planning allocates the specific containers to specific slots within the blocks determined during the strategic planning phase“. ([17])
> The strategic planner wants to ensure maximum parallel crane usage at each destination and that any constraints are met.(...) ’Outline planning’ allocates containers within hatches on the General Arrangement Document to above or below deck stacks.(...) The primary objective at outline planning ist to minimise the removal of hatch-lids whlist minimising the amount of unused below-deck cargo space. In tactical planning, the general plan is used to guide specific placements of containers into specific slots. ([17])
> (...) A container can heuristically be allocated a slot by applying a packing algorithm designed to sequence containers into blocks, Wilson (1997), For each block

1. List containers according to size (large first), and then by destination and weigt (furthest and heaviest first)
2. The first container ist taken from the list.
3. A standard dimension container ist loaded into the first availible slot and non-standard dimensioned containers are swapped (where possible) with containers for the same POD at the top of a stack. A containers displaced is returned to the list to be placed somewhere else.
4. If the list is empty then the placement procedure is terminated, otherwise the process begins again from 2.

(...) This approach excels at producing good weight gradiation stacks, low mixing of PODs in stacks and enforcing non-standard dimensionated containers to be located at the top of stacks.
The chromosome will be a list of all containers in the load list and their associated TEU value (???) the ordering is such that chromosomes can be mapped to availible space. For each hatch there are two associated TEU capacity values (above-deck and below-deck). Therefore, containers are loaded into a given hatch up to and including the TEU capacity of a given hatch but are not allowed to exceed this capacity.

Kapitel 5

Rehandles im Container Lagerplatz

Überstauuer treten auch im Container Lagerplatz auf. Dort sind die Zugriffe aber letztlich durch die Masse der Stapel statistischer Natur.

5.1 Exakte Berechnung

Es sei $(n_1, n_2, n_3, \ldots, n_a)$ die Konfiguration einer Stapelung (bay), das heisst die Stapelung enthält Stapel der entsprechenden Höhen, die nach Größe (lexikographisch) sortiert seien. Es sind im Stapel dann $n_1 + n_2 + n_3 + \ldots + n_a$ Container. Es sei a, b, c die Zahl der Reihen, Abschnitte und Lagen einer Stapelung. Einzelne Container unterscheiden sich nicht voneinander: Beispielsweise $(3, 2, 1, 0, 0, 0)$ unterscheidet sich von $(0, 3, 0, 2, 0, 1)$ im Modell nicht. Sei nun

$v(n_1, n_2, n_3, \ldots, n_a) =$ erwartete Zahl der Umstauungen („rehandles“) um einen beliebigen Container der Konfiguration $(n_1, n_2, n_3, \ldots, n_a)$ zu entnehmen.

$c_{kj} =$ die $j$te Konfiguration der Stapelung aus k Containers

$p_k(i, j) =$ Wahrscheinlichkeit, daß die Konfiguration von Konfiguration c_{ki} zu $c_{(k-1)}j$ übergeht, wenn ein Container entnommen wird

$s_{kj} =$ die Wahrscheinlichkeit, daß eine Stapelung die j-te Konfiguration hat.

$v_k(i, j) =$ Zahl der Umstauungen beim Übergang von c_{ki} zu $c_{(k-1)j}$.

$v_k 0$ Zahl der erwarteten Umstauungen für die zufällige Entnahme eines Containers, wenn die Stapelung k Container enthält.

S_k sei ein Vektor mit den Einträgen s_{kj}.

P_k sei eine Matrix mit dem Eintrag $p_k(i, j)$ an der (i, j)ten Stelle.

Die S_k können rekursiv berechnet werden:

$$S_k = S_{k+1} P_{k+1} \tag{5.1}$$

Und für die v_k ergibt sich:

$$v_k = \sum_i s_{ki} \sum_j p_k(i,j) v_k(i,j) \tag{5.2}$$

Um die Gesamtzahl der Umstauungen bei Entnahme aller Container zu berechnen, summiert man die v_k für $k = 1$ bis $k = n$ auf.

5.2 Approximation

[19] bestimmt die zu erwartende Zahl von Umstauern in einer Stapelung aus a Containerstapeln der Höhe c. Dabei wird angenommen, dass jeder Container die gleiche Wahrscheinlichkeit hat, entnommen zu werden. Gibt es Überstauer, so werden diese auf den nächstliegenden Stapel mit der geringsten Höhe gesetzt. Es soll nun abgeschätzt werden, wieviele Umstauer es im Schnitt geben kann, wenn die Stapel komplett geleert werden und keine Container dazu kommen. Man versucht die Anzahl der Überstauer einer Anzahl k von Containern in einer Stapelung als lineare Funktion anzunehmen und braucht hierzu die Steigung der Geraden, die durch den Ursprung geht.
Angenommen, die Stapelung besteht aus $k = ac$ Containern und jeder Stapel hat die Höhe c. Dann ist die erwartete Anzahl von Überstauern

$$v(a,c,k) = \frac{a}{ac}(1+2+3+\ldots+(c-1)) \tag{5.3}$$

$$= \frac{1}{c}\left(\frac{c(c-1)}{2}\right) = \frac{c-1}{2} \tag{5.4}$$

Das unterschätzt jedoch die Anzahl der Überstauer. Sei die Stapelung hervorgegangen druch der Ausstapelung eines Containers aus einem Stapel S. Dann hat dieser Stapel S im Schnitt die Höhe von $\frac{c}{2}$. Die Überstauer werde alle auf die Stapel mit der kleinsten Höhe ausgestapelt, somit werden auf $\frac{c}{2}$ Stapel jeweils ein Container mehr gestapelt und haben damit die Höhe $c+1$. Damit muss zu (5.4) der Term $c \cdot \frac{c}{2}\frac{1}{ac}$ addiert werden. Jedoch verringern sich die zu erwartenden Überstauer im Stapel S. Waren vorher die zu erwartenden Überstauer

$$\frac{1}{ac}(1+2+3+\ldots+(c-1)) \tag{5.5}$$

so sind es jetzt

$$\frac{1}{ac}\left(1+2+3+\ldots+(\frac{c}{2}-1)\right) \tag{5.6}$$

und die Differenz davon ist $(3c^2-2c)/(8ac)$. Also unterscheidet sich die neue Konfiguration um

$$c \cdot \frac{c}{2}\frac{1}{ac} - (3c^2-2c)\frac{1}{8ac} = \frac{c+2}{8a} \tag{5.7}$$

Da die gesuchte Gerade durch den Koordinatenursprung und den Punkt $P = (ac, \frac{c-1}{2} + \frac{c+2}{8a}$ geht, ergibt sich ein Steigung von $\frac{c-1}{2ac} + \frac{c+2}{8a^2c}$.
Damit ist

$$\hat{v}(a,c,k) = \left(\frac{c-1}{2ac} + \frac{c+2}{8a^2c}\right) \cdot k \tag{5.8}$$

Die Gesamtzahl der zu erwartenden Rehandles (Überstauer) kann nun geschätzt werden als

$$\hat{V}(a,c) = \int_0^{ac} \left(\frac{c-1}{2ac} + \frac{c+2}{8a^2c} \right) x dx \tag{5.9}$$

$$= \frac{1}{4}ac(c-1) + \frac{1}{16}c(c+2) \tag{5.10}$$

5.3 Index of Selectivity

Eine sehr gebräuchliche Methode die Anzahl der Umstauer zu bestimmen ist der IOS-Wert ([19]), wobei IOS für „Index of Selectivity“. Jedem Container wird ein Bruch zugeordet und zwar 1/(Lagereihe). Dann ist die Zahl der gesamten Containerbewegungen $N = \frac{(\text{Zahl der Container})^2}{\text{Summe der IOS aller Container}}$, das heisst die Zahl der Überstauuer kann geschätzt werden als $N -$ Zahl der Container. Diese Methode basiert auf keiner systematischen Theorie, ist aber weit verbreitet.

Kapitel 6

Technische Fragen

6.1 Allgemeine Definitionen

Definition 25 ([16]) *Ein Stauraum besitzt zur Aufnahme von stapelfähigem (standfähigem), gleich- oder ungleichartigem Stückgut eine ebene Grundfläche mit bekannter Tragfähigkeit. Das zur Verfügung stehende Nutzvolumen ist bestimmt durch seine konstruktive Ausführung (Bauart) und/oder einsatzbedingte virtuelle Begrenzungsflächen (Lichtraumprofil).*

Definition 26 ([16]) *Als Stückgut werden in der Transporttechnik alle Gegenstände bezeichnet, die ohne Rüsicht auf Form und Masse während des Transportes als Einheiten behandelt werden. Stückgut wird als stapelfähig bezeichnet, wenn seine geometrischen Form und seine Festigkeit gegenüber Stapelstauchdruck eine Stapelung zum Zwecke der Lagerung, des Umschlags und des Transports zulassen.*

Bemerkung [16]: Die Stapelfäigkeit von nicht stapelbaren Gütern (Schüttgut, gasförmigem Gut, flüssigem Gut) kann durch Bildung der logistischen Einheiten Packstück oder Ladeeinheit erreicht werden, die der Definition von Stückgut genügen.

6.2 Container

Contanaier werden in den folgenden DIN/ISO-Normen festgelegt:

- DIN/ISO 830: „Container-Terminologie“
- DIN/ISO 668: „ISO-Container der Reihe 1 - Klassifikation, Außenmaße, Gesamtgewichte - “
- DIN/ISO 6346: „ISO-Container - Kodierung, Identifizierung und Kennzeichnung - “

An [18] angelehnt gilt nach DIN/ISO die folgende

Definition 27 (DIN/ISO 668) *Ein **Container** ist ein Transportbehälter, der....*

a) *von dauerhafter Beschaffenheit und daher genügend widerstandsfähig für den wiederholten Gebrauch ist.*

	Dimensions	
	$20' \times 8' \times 8'6''$	$40' \times 8' \times 8'6''$
Length (meters)	5.9	12.0
Width (meters)	2.4	2.4
Height (meters)	2.6	2.6
Cubic capacity (cubic meters)	32.9	67.0
Stacking capacity	9 high	9 high
Maximum weight (metric tons)	24	30

Abbildung 6.1: Principial dimensions of flat roof steel containers

b) *besonders dafür gebaut ist, den Transport von Gütern mit einem oder mehreren Transportbehältern, ohne Umpacken der Ladung, zu ermöglichen.*

c) *für den mechanischen Umschlag geeignet ist.*

d) *so gebaut ist, daß er leicht be- und entladen werden kann.*

e) *einen Rauminhalt von mindestens 1 Kubikmeter hat.*

Fahrzeuge und Verpackungen sind keine Container.

Definition 28 (DIN/ISO 830) *Die **Stapelfähigkeit** eines Containers ist die Fähigkeit eines Containers, eine bestimmte Anzahl voll beladener Container der gleichen Nennlänge und mit gleichem Gesamtgewicht bei den in Schiffszellen auftretenden Beschleunigungen zu tragen, wobei der Versatz zwischen Container, der durch Toleranzen der Zellabmessungen entsteht, berücksichtigt ist.*

6.3 Containertypen

Die Abmessungen für Container werden in [27] (Abb. 11.4) wie in Tabelle 6.1 angegeben

Es sind 39% der Container 20ft-Container und 53% 40ft-Container.
Die Kosten einer Containerbewegung (Ein- oder Ausladen) wird mit einheitlich mit 200$ in [27] angegeben.

Die verschiedenen Typen von Containern:

- Stückgut-Container:
  - Gewöhnlicher Stückgut-Container
  - Spezieller Stückgutcontainer: geschlossenner, belüfteter Container; Open-Top-Container; Platform mit vollständigem unvollständigen Rahmenaufbau und offenen Seiten
- Spezialgut-Container:
  - Thermal-Container: Isolierte Container; Maschinelle gekühlte Container; gekühlte und beheizte Container
  - Tank-Container
  - Schüttgut-Container (bulk)

Container-Type	20-ft equivalent ur		Per cent change
	1985	1995	
Standard	4090	8050	97
Open-Top	221	225	2
Folding Flatrack	36	42	16
Ventilated	37	39	5
Platform	38	31	-17
Bulk	27	24	-13
Other	13	19	42
Internal reefer	157	520	231
Insulated	77	72	-7
Tank	34	84	149
Total	4779	9194	93

Abbildung 6.2: World container stock by principal type in 1000 TEU

– Container für namentlich bezeichnete Güter

Ausserdem gibt es Luftverkehrs-Container und 45-Fuß-Container.

Wie sind die Anteile verschiedener Container-Typen verteilt? Dazu gibt es in [27] eine Tabelle

6.4 Terminal Betriebssysteme

Es gibt folgende Umschlagssysteme:

- Containerverladebrücken
- Portalstapler
- Mobilkrane
- Reach staker
- Gabelstapler
- Transtainer
- Zugmaschinen
- Spreader

Gebrauchliche Betriebssysteme der Terminals sind dann:

- Trailer-park-System: Container dirket vom Schiff auf ein entsprechendes Straßenchassis, wo sie während ihres Landaufenthalts bleiben
- Straddle-carrier-System: Container von Schiff auf die Kais, von Portalstaplern aufgenommen und an ein bestimmten Abstellplatz gebracht.
- Gabelstapler-System: Transport der Container auf dem Hafengelände mit schweren Frontgabelstaplern

- Stapel-System: Bei diesem System werden die Container mit den Flurfördergeräten in die Lagerzone gebracht und dort mit einem fahrbaren Bockkran (Transtainer) angehoben und bis zu sechsfach hoch unmittelbar nebeneinander und hintereinander gestapelt.
- gebräuchlichstes System: Vom Schiff auf Sattelaufleger gesetzt. Diese Auflieger werden von Zugmaschinen zum Lagerplatz gezogen. Dort dann von straddle- bzw. van-carrier überfahren und auf den vorgesehenen Stapel abgesetzt.

6.5 Richtlinien der Stauraumplanung

Die folgenden Richtlinien lassen sich bei Beobachtung der Pratiker herauslesen (diese Liste basiert auf einer Liste in [32]):

1. Die Zahl der Umstauvorgänge eines Containers ist zu minimieren. Die genauen Kosten der Umstauungen hängen vom Termial ab und vor allem vom jeweiligen Operator
2. Ballast-Wasser kann benutzt werden um Stabilitätsprobleme zu vermeiden.
3. Die Ladung, d.h. die Container müssen gleichmässig über das Schiff verteilt werden.
4. Der Trim muss während des Ladens immer Nahe 0 sein.
5. Die Verstauungsreihenfolge muss sorgfältig geplant werden.
6. Die Container eines Zieles sollten über vier Laderäume verteilt sein, damit vier Kräne am Terminals arbeiten können
7. Ein 40'-Container darf auf zwei 20'-Container plaziert werden, aber nicht umgekehrt.
8. Kühlcontainer sollten geeignet plaziert werden, d.h. mit Stromanschluss und weg vom Maschinenraum und beheizten Containern.
9. Gefahrgutcontainer müssen weit genug getrennt werden (nach der IMGM-Richtlinie) und vor allem weit genug weg von der Mannschaftsunterkünften
10. Feuchte Laderäume dürfen keine Container enthalten mit empfindlichem Ladegut
11. Überhohe Container obenauf, genauso wie leere und Open-Top-Container
12. Schwere Container eher nach unten, leichtere nach oben.
13. Manche Spalten können von manchen Kränen nur schwer erreicht werden.
14. Bewegen der Lukendeckel sollte minimiert werden.
15. Manchen Container müssen bei der Reise kontrolliert werden und geeignete Plätze an Deck bekommen.
16. Stapelhöhen müssen eingehalten werden

6.6 Vermischtes

Einiges aus [24]:

- Die Be- und Entladung der Vollcontainerschiffe erfolgt nach dem LoLo-Verfahren (Lift-on/Lift-off), d.h. die Container werden in vertiklaer Richtung druch die Umschlagsmittel oben in bzw. auf das Schiff geladen und in umgekehrter Richtung entladen. Auf dem Schiff findet kein Horizontaltransport der Container statt. (Seite 14)
- Das Zellensystem besteht aus mehreren Bays (Reihen), eine Bay wiederrum aus mehreren nebeneinander angeordneten Stacks (Stapel, Säulen). Zur eindeutigen Adressierung einer Zelle ist außerdem die Tier (Lage) notwendig. (...) Je zwei benachbarte Bays werden zu einer Doppelbay zusammengefaßt. Ein Laderaum wiederrum besteht aus maximal zwei Doppelbays und wird durch zwei feuer- und flüssigkeitsundurchlässige Trennwände begrenzt. (Seite 15)
- Außerdem unterscheidet man die beiden Staubereich unter und über Deck. Das Stellplatzkapazitätsverhältnis unter/über Deck variiert zwischen 70/30 und 50/50. (Seite 16)
- Eine Luke wird von genau einem Lukendeckel abgedeckt. Lukendeckel werde wie Container bewegt und bei Lade- und Löschvorgängen unter Deck abgesetzt. (Seite 19)
- Im gleichen Stack können 40'-Container über zwei 20'-Container gestaut werden, aus Gründen der Laschung jedoch nicht umgekehrt. (Seite 19)
- Die (Wasser-)Verdrängung D eines Schiffes entspricht dessen Gesamtgewicht. (Seite 22)
- Die Stabilität eines Schiffes kennzeichnet seine Eigenschaft, in aufrechter Lage zu schwimmen und einer Krängung aufrichtenden Momente entgegenzusetzen. Krängung ist die Drehung eines Schiffes um seine Längesachse um den Krängungswinkel ϕ. (Seite 24)
- Trimm ist der Tauchungsunterschied zwischen dem hinteren und vorderen Tiefgang. (Seite 31)
- Unter der Längsfestigkeit eines Schiffskörpers versteht man sein Vermögen den durch ungleichmässige Gewichts- und Auftriebsverteilung in Schiffslängsrichtung hervorgerufenen Beladungen Widerstandmomente entgegenzusetzen. Kennwerte für diese Belastungen sind die Querkräfte und die Biegemomente. (Seite 34)
- Unter Torsion eines Schiffskörpers versteht man seine Verdrehung oder Verwindung um die Längsachse. (Seite 39)
- Nach dem IMDG-Code (International Maritime Dangerous Goods Code) wird Gefahrgut räumlich voneinander getrennt. Man unterscheidet Trennung (Segregation) in vertikaler und in horizontaler Richtung. (Seite 42)
- Im Linienverkehr fährt ein Containerschiff in der Regel Rundreisen mit gleichem Ausgangs- und Endhafen. Bezeichnet eine Range eine Menge von Häfen deren geographische Entfernungen untereinander im Vergleich zu einer anderen Range gering sind, so lassen sich Liniendienste anhand ihrer Rundereiseausprägungen unterscheiden in
  - Ein-Relationen-Dienste, die zwei Ranges verbinden und in
  - Rund-um-die-Welt-Dienste, die mehr als zwei Ranges verbinden und in ost- oder westlicher Richtung um die Welt führen. (Seite 45)
- Planungsziele
  - Kapazitätsauslastung - Stellplatzauslastung bzw. Gewichtsauslastung
  - Umstauvorgänge

  - Hafenliegezeit - Gruppierung, Umstauvorgänge, Containerbrückenauslastung, Containerbrückenlängsbewegungen, Lukendeckelbewegungen, Flufördergerätebehinderungen
  - Brennstoffverbrauch - Ballastwassermenge, Trimm

  (Seite 55)

- (...noch einiges einfügen..) (Seite 62)
- Interdependenzen zwischen Stau- und Terminalplanung entstehen
  - bei mehrlagiger Stapelung in Lagerplatzbereich eines Terminals, wenn die Baladungssequenz nicht mit der spiegelbildlichen Lagerungssequenz übereinstimmt und
  - bei Zulauf von Containern aus Binnenland während der Beladung des Schiffes, wenn die Beladesequenz nicht mit der zeitlichen Verfügbarkeit dieser Container übereinstimmt.

  (Seite 70)

- Staufaustregeln: schwere Container unten, schwere Container mittschiffs, Leercontainer in der höchsten Lagen über Deck, Gewichtssumme backbord und steuerbord gleich groß, Gewichtssumme in spiegelbildlichen Stacks einer Bay gleich groß (Seite 121)
- Planungsreihenfolge in der Grobplanung:
  - Plattformcontainer, IMDG-Container, offene Container, Kühlcontainer, Standardcontainer
  - 20'-Container. 40'-Container
  - Container mit höchstem Zielhafenindex, Container mit zweithöchstem Zielhafenindex usw.

(Seite 117)

Das ökonomische Auftreten stark ausgelasteter Containerschiffe beleuchtet dieses Zitat:

> „Mit wachsenden Schiffsgrößen im Containerverkehr erhöht sich der Zwang zur Kapazitätsauslastung. Der break-even-Punkt liegt bei Containerschiffen der 3. Generation (2000-3000 TEU) bei etwa 85 bis 90 Prozent der Kapazität und bei Containerschiffen der 4. Generation (bis 5500 TEU) bei etwa 70 Prozent der Kapazität.“ (Seite 265f in [5])

Kapitel 7

Dreidimensionale Packungsprobleme

7.0.1 Dyckhoffs Klassifikation

Die Verallgemeinerung der einfachen Stapel findet nun bei dreidimensionalen Packungsproblemen statt.

7.1 Klassifikation von Zuschnitt- und Packungsproblemen

In [9] wird die folgende Typologie für Packungsprobleme (Zitat nach [36]) vorgeschlagen:

$$a/b/c/d$$

mit

a Dimension des Problems: 1,2,3 oder n
b Verlade- oder Beladeproblem: V oder B
c Anzahl der Containerarten („one, identical, different“): O, I oder D
d Anzahl unterschiedlicher Kistenarten
(„congruent, many , relatively few figures, few figures“): C, M, R oder F.

Es werden die folgenden Dinge unterschieden (p.178f in [9]):

- Dimensionality
- Type of Assignment
- Characteristics of objects and items
  - type of quantity measurement
  - figure (shape)
  - assortment

  - availability
- Pattern restrictions
  - geometrical characteristics
  - operational characteristics
  - dynamics of allocation
- Objectives
- Status of information
- Variability
- Solution methods
- Properties based on reality
  - type of objects and items
  - planning context

Am Interessantsten für die Fragenstellung nach den Überstauuern sind hier die **„dynamics of allocation“**. Diese werden beschrieben durch die „inclusion of time“ und unterschieden in

- static = all objects/ items have the same beginning and finishing date
  - On-Line = no all abjects/ items are known in advance
  - Off-Line = all objects/ items are known in advance
- dynamic = objects/ items have different beginning anf finishing dates
  - with reallocation = the spatial arrangement of an item ca be rearranged within the object
  - without reallocation = the spatial arrangement, once given, cannot be changed

7.1.1 Wottowas Klassifikation

Diese Typologie ist nach [36] „unvollständig und unübersichtlich“. Deshalb wird in [36] eine andere Klassifizierung von Packungsproblemen vorgeschlagen:

Definition 29 ([36]) *Jedes orthogonale Packungsproblem gehört zu einer der vier folgenden grundlegenden Problemklassen, die wir als Grundklassen bezeichnen wollen:*

- *Pallet Loading (PL): identische Kisten, ein Container, packe möglichst viele Kisten*
- *Knapsack (KS): unterschiedliche Kisten, ein oder mehrere Container, maximiere den Wert der gepackten Kisten. (Falls der Wert durch die Kistengröße bestimmt wird, heißt dies, den oder die Container möglichst voll zu packen und möglichst wenig Leerfläche übrigzulassen.)*
- *Strip Packing (SP): unterschiedliche Kisten, ein nach einer Seite (jeweils der letzten Dimension) offener Container, packe alle Kisten und minimiere unbeschränkte Containerdimensionen*

- *Bin Packing (BP): unterschieldliche Kisten, mehrere Container, packe alle Kisten in möglichst wenige Container. Falls mehrere Containertypen vorhanden sind, werden diesen Kosten zugeordnet (standardmäßig deren Fläche bzw- Volumen) und die Summe der Kosten der benötigten Container minimiert*

Definition 30 ([36]) *Als Problemklasse bezeichnen wir eine Grundklasse mit fixierter Problemdimension. Jede Problemklasse hat als Bezeichnung das Kürzel der Grundklasse, erweitert um die Dimension des Problems ($1D, 2D, 3D$).*

Alle weiteren Unterscheidungsmerkmale können als zusätzliche Nebenbedingungen aufgefasst werden.

Definition 31 ([36]) *Relevanten Nebenbedingungen von Packungsproblemen:*

- *Guillotinen-Schnitte:*
  - *GC: Es sind nur Guillotinen-Schnitte erlaubt*
  - *GC2: Es sind nur Guillotinen-Schnitte der Tiefe 2 erlaubt*
  - *GC3: Es sind nur Guillotinen-Schnitte der Tiefe 3 erlaubt*
  - *NGC: Alle Muster erlaubt*
  - *Vorgabe: NGC*
- *Heterogene Containertypen*
  - *HC: Es gibt verschiedene Containertypen*
  - *Vorgabe: Alle Container sind vom gleichen Typ*
- *Orientierung*
  - *OF: Die Orientierung einzelner Kisten ist fixiert*
  - *OU: Die Orientierung aller Kisten ist unbestimmt*
  - *Vorgabe: OU*
- *Kisten-Werte*
  - *BC: Den Kisten sind Werte zugeordnet, die nicht dem Volumen entsprechen*
  - *Vorgabe: Die Kistenwerte sind die Volumnia der Kisten*
- *Container-Kosten*
  - *CV: Den Containern sind Kosten zugeordnet, die nicht dem Volumen entsprechen*
  - *Vorgabe: Die Containerkosten sind die Volumina der Kisten*
- *Beschränke Kistenzahl*
  - *BB: Die Anzahl der Kisten eines Typs in jedem Container ist beschränkt*
  - *Vorgabe: In jeden Container können beliebig viele Kisten eines Typs gepackt werden.*
- *Minimale Anzahl Kisten*
  - *MB: Die Anzahl der Kisten eines Typs in jedem Container ist beschränkt*

  - *Vorgabe: Es gibt keine Anzahl minimaler Kisten eines Typs pro Container*

- *Unbeschränkte Kistenzahl (nur bei KS)*
  - *UB: Die Anzahl der Kisten eines oder mehrerer Kistentypen ist nicht beschränkt*
  - *Vorgabe: Von jedem Typ existiert nur eine beschränkte Anzahl Kisten*

- *Beschränkte Containerzahl (nur bei BP)*
  - *BC: Die Anzahl der Container eines Typs ist beschränkt*
  - *Vorgabe: Es gibt beliebig viele Container eines Typs*

7.2 Containerladeproblem nach Chen

Seite 70 in[7]: Gesucht ist eine Anzahl von Containern um eine gegebene Menge von Kisten zu packen. Alle Kisten sind orthogonal im Container anzuordnen, das heisst die Kanten einer Kiste sind parallel oder senkrecht zu den Achsen der Container. Die längste Kante einer Kiste bzw. eines Containers sei seine Länge, die kürzeste die Höhe und die mittlere die Breite.
Es werden nun die folgenden Parameterdefiniert:

N	Anzahl der zu packenden Kisten
m	Anzahl der verfügbaren Container
M	Eine grosse Zahl
(p_i, q_i, r_i)	Länge, Breite und Höhe der Kiste i
(L_j, Q_j, H_j)	Länge, Breite und Höhe des Containers j

Folgende Variablen werden benötigt:

s_{ij}	Binäre Variable mit dem Wert 1, wenn Kiste i in Container j gepackt wird; 0 sonst
n_j	Binäre Variable mit dem Wert 1, wenn Container j benutzt wird; 0 sonst
(x_i, y_i, z_i)	(Kontinuierliche) Ortskoordiante der Vorne-Links-Unten (front-left bottom) Koordinate von Kiste i
(l_{xi}, l_{yi}, l_{zi})	Binäre Variable, die angibt ob die Länge der Kiste i parallel zur $X-, Y-$ oder $Z-$Achse ist.
(w_{xi}, w_{yi}, w_{zi})	Binäre Variable, die angibt ob die Breite der Kiste i parallel zur $X-, Y-$ oder $Z-$Achse ist.
(h_{xi}, h_{yi}, h_{zi})	Binäre Variable, die angibt ob die Höhe der Kiste i parallel zur $X-, Y-$ oder $Z-$Achse ist.

Relative Position der Kisten zueinander

a_{ik}, b_{ik}, c_{ik}	Binäre Variable, die angibt, ob Kiste i links, recht oder vor Kiste k steht
d_{ik}, e_{ik}, f_{ik}	Binäre Variable, die angibt, ob Kiste i hinter, über oder unter Kiste k steht

Das Containerbeladeproblem kann dann als das folgende gemischt-ganzzahlige lineare Programm formuliert werden:

$$\min \sum_{j=1}^{m} L_j \cdot W_j \cdot H_j \cdot n_j - \sum_{i=1}^{N} p_i \cdot q_i \cdot r_i \tag{7.1}$$

unter den Nebenbedingungen:

$$x_i + p_i \cdot l_{xi} + q_i \cdot w_{xi} + r_i \cdot h_{xi} \leq x_k + (1 - a_{ik}) \cdot M \qquad \forall i, k, i < k \tag{7.2}$$
$$x_k + p_k \cdot l_{xk} + q_k \cdot w_{xk} + r_k \cdot h_{xk} \leq x_i + (1 - b_{ik}) \cdot M \qquad \forall i, k, i < k \tag{7.3}$$
$$y_i + q_i \cdot w_{yi} + p_i \cdot l_{yi} + r_i \cdot h_{yi} \leq y_k + (1 - c_{ik}) \cdot M \qquad \forall i, k, i < k \tag{7.4}$$
$$y_k + q_k \cdot w_{yk} + p_k \cdot l_{yk} + r_k \cdot h_{yk} \leq y_i + (1 - d_{ik}) \cdot M \qquad \forall i, k, i < k \tag{7.5}$$
$$z_i + r_i \cdot h_{zi} + q_i \cdot w_{zi} + p_i \cdot l_{zi} \leq z_k + (1 - e_{ik}) \cdot M \qquad \forall i, k, i < k \tag{7.6}$$
$$z_k + r_k \cdot h_{zk} + q_k \cdot w_{zk} + p_k \cdot l_{zk} \leq z_i + (1 - f_{ik}) \cdot M \qquad \forall i, k, i < k \tag{7.7}$$

Die Bedingungen (7.2)-(7.7) gewährleisten, dass die Kisten sich nicht überlappen.

$$a_{ik} + b_{ik} + c_{ik} + d_{ik} + e_{ik} + f_{ik} \geq s_{ij} + s_{kj} - 1 \quad \forall i, k, j, i < k \tag{7.8}$$

Die Bedingung (7.8) erzwingt, dass Überlappungen nur zwischen Kisten im gleichen Container geprüft werden.

$$\sum_{j=1}^{m} s_{ij} = 1 \quad \forall i \tag{7.9}$$

Die Bedingung (7.9) erzwingt, dass jede Kiste in genau einem Container plaziert wird.

$$\sum_{i=1}^{N} s_{ij} \leq M \cdot n_j \quad \forall j \tag{7.10}$$

Die Bedingung (7.10) gibt an, dass ein Container benutzt wird, wenn eine Kiste in ihm steht.

$$x_i + p_i \cdot l_{xi} + q_i \cdot w_{xi} + r_i \cdot h_{xi} \leq L_j + (1 - s_{ij}) \cdot M \quad \forall i, j \tag{7.11}$$
$$y_i + q_i \cdot w_{yi} + p_i \cdot l_{yi} + r_i \cdot h_{yi} \leq W_j + (1 - s_{ij}) \cdot M \quad \forall i, j \tag{7.12}$$
$$z_i + r_i \cdot h_{zi} + q_i \cdot w_{zi} + p_i \cdot l_{zi} \leq H_j + (1 - s_{ij}) \cdot M \quad \forall i, j \tag{7.13}$$

Die Bedingungen (7.11)-(7.13) erzwingen, daass die in einem Container plazierten Kisten innerhalb der Abmessungen des Containers bleiben.

$$l_{xi}, l_{yi}, l_{zi}, w_{xi}, w_{yi}, w_{zi}, h_{xi}, h_{yi}, h_{zi} \in \{0, 1\} \tag{7.14}$$
$$a_{ik}, b_{ik}, c_{ik}, d_{ik}, e_{ik}, f_{ik}, s_{ij}, n_j \in \{0, 1\} \tag{7.15}$$
$$x_i, y_i, z_i \geq 0 \tag{7.16}$$

7.3 Eigenschaften dreidimensionaler Packungsprobleme

Definition 32 (Guillotine-Schnitte) *Unter Guillotine-Schnitten versteht man gerade Schnitte, die von einer Seite zur gegenüberliegenden Seite des zu zerschneidenden Rechteckes verlauf und dabei parallel zu den beiden anderen Seiten sind. (p. 29 in [28])*

Definition 33 (k-stufige Guillotine-Schnitte) *„Die Ausführung paralleler Guillotine-Schnitte in einer festen Lage des zu zerschneidenden Rechteckes nennen wir eine Stufe. Bei r Guillotine-Schnitten entstehen so $(r+1)$ Teilrechtecke, von denen jedes in einer zweiten Stufe um 90 gedreht und (gegebenenfalls) weiter zerschnitten wird. Entsprechend der Anzahl der ausgeführten Schnitte sprechen wir von einem k-stufigen Guillotine-Zuschnittproblem. Probleme, bei denen die Anzahl der Stufen nicht beschränkt wird, nennen wir im weiteren freie Guillotine-Zuschnittprobleme. Wird zugelassen, dass die geforderten Teile erst durch zusätzliches Beschneiden an einer oder zwei Seiten entstehen, so wird in Gilmore/Gomory (1965) ([10]) vom unexakter Fall gesprochen.“ (p. 146 aus [28])*

> „Mitunter wird die Forderung erhoben, das gesamte Ausgangsrechteck oder Teilrechtecke davon durch Guillotine-Schnitte gleichzeitig in Längs- und Querrichtung zu zerschneiden. Bei diesen speziellen mehrstufigen Zuschnittproblemen sprechen wir von einem Gruppenzuschnitt. Dabei können sowohl gleiche als auch unterschiedliche Teile erhalten werden. Wird das Ausgangsrechteck zunächst in k Teilrechtecke zerlegt und jedes dieser Teilrechtecke durch Gruppen-Zuschnitt zerschnitten, so spricht man von einem $k-$Gruppen-(Zuschnitt-)Problem.“ (p. 148 aus [28])

„Unter einem Rechteckschnitt $R = \{R_1, \ldots, R_n\}$ wollen wir die Aufteilung durch seitenparallele Schnitte in Teilrechtecke $R_1, \ldots R_n$ verstehen. Zu einem Rechteckschnitt konstruieren wir einen Adjazenzgraphen G druch folgende Festlegungen:
Jeder Knoten v_i von G entspricht eindeutig einem Rechteck R_i. Zwei Knoten v_i und v_j von G werden durch eine Kante verbunden, wenn zwischen den entsprechenden Rechtecken R_i und R_j eine echte Berührung (nicht nur in einem Punkt) vorliegt.
Mit den eingführten Begriffen läßt sich das Problem A: Welche Graphen sind Adjazenzgraphen von Rechteckschnitten? formulieren, das in seiner Allgemeinheit nicht gelöst ist.“ (p. 152 aus [28])

Übungsaufgabe 12 Kann man einen Würfel mit einer Kantenlänge von 12cm durch Quader vom Ausmaß $2cm \times 4cm \times 8cm$ vollständig ausfüllen?

Definition 34 (p.33 in [36]) *Als den **starken (ungerichteten) Berührgraphen** eines d-dimensionalen Packmusters bezeichnen wir einen Graphen, dessen Knoten den Kisten des Packmusters eindeutig zugeordnet sind. Je zwei Knoten sind genau dann durch eine Kante verbunden, wenn die Punkte, an denen sich die entsprechenden Kisten berühren, eine $(d-1)$-dimensionale affine Hyperebene aufspannnen.*

In einem 2D-Packmuster liegen die Punkte, an denen sich zwei Kisten berühren, auf einer „Berührstrecke“ (diese darf nicht entartet sein, da ein einzelner Punkt nicht genügt) und in einem 3D-Packmuster in einem „Berührrechteck“ (ein Punkt bzw. eine Strecke genügen nicht)

Satz 5 (p.34 in [36]) *Jeder starke Berührgraph eines zweidimensionalen Packmusters ist planar. Die Umkehrung gilt nicht.*

Satz 6 (p.37 in [36]) *Jede planare Einbettung eines planaren Graphen ist genau dann Berührgraph eines zweidimensionalen Packmusters, wenn sie kein Dreieck enthält, in dessen Inneren ein weiterer Knoten liegt.*

Definition 35 *Als K_n bezeichnen wir den vollständigen Graphen mit n Ecken. Als $K_{n,m}$ bezeichnen wir den vollständigen bipartiten Graphen mit Knotenmengen $V = V_1 \cup V_2$ und $|V_1| = n, |V_2| = m$ und die Kanten $v = (v_1, v_2)$ sind genau die mit $v_1 \in V_1$ und $v_2 \in V_2$.*

Bemerkung 3 (p.35 aus [36]) Es lassen sich K_4 und $X_{3,3}$ nicht als (starke) Berührgraphen von zweidimensionalen Packmustern darstellen.

Bemerkung 4 (p.47 aus [36]) Es gibt dreidimensionale Packmuster, deren Berührgraph der K_4 ist. Der K_5 hingegen kann nicht Berührgraph eines dreidimensionalen Packmusters sein.
Neben dem $X_{3,3}$ und jedem $X_{3,k}$ mit $k > 3$ läßt sich auch der $X_{4,2}$ als Berührgraph eines dreidimensionalen Packmusters darstellen, nicht aber der $X_{4,3}$.

Satz 7 (p.260 in [21]) *Any packing of a bin can be replaced by an equivalent packing where no item may be moved leftward, downward, or backward.*

Satz 8 (p.260 in [21]) *An ordering of the items in an optimal solution exists such that, if $i < j$, $x_i + w_i \leq x_j$ or $y_i + h_i \leq y_j$ or $z_i + d_i \leq z_j$.*

7.4 Überstauungen in dreidimensionalen Packungsproblemen

Fakt 9 *Gibt es eine dreidimensionale Packung, die aus 1-stufigen Guillotine-Schnitten besteht, so gibt es auch eine überstauungsfreie Packung.*

Beweis 1 Sei ein Packmuster gegeben, das aus 1-stufigen Guillotine-Schnitten besteht und das eine überstaute Kiste i enthält. Das kann nur sein, wenn eine Kiste j in Entnahmerichtung davor steht und das heisst die Schnittebene liegt senkrecht dazu. Kiste i und j können vertauscht werden ohne die Abmessungen des Packmusters zu verändern. Dadurch ist nun eine Kiste weniger überstaut

Fakt 10 *Gibt es eine dreidimensionale Packung, die aus 2-stufigen Guillotine-Schnitten besteht, so gibt es auch eine überstauungsfreie Packung.*

Das gilt jedoch nicht für beliebige 3-stufige Guillotine-Schnitte! Wie sind Guillotine-Schnitte im dreidimensionalen überhaupt zu definieren?! Man schneidet in x-,y- oder z-Richtung orthogonal durch bis zur gegenüberliegenden Fläche. Anders als im zwei-dimensionalen ergeben sich aber jeweils zwei Möglichkeiten für den nächsten Schnitt.
Wie sieht es aus für Guillotine-Schnitte mit wechselnden Achsen oder Gruppierungen? Es gibt auch Überstauungen im zweidimensionalen Fall, die nicht durch ein anderes Packmuster beseitigt werden können (siehe 7.1).

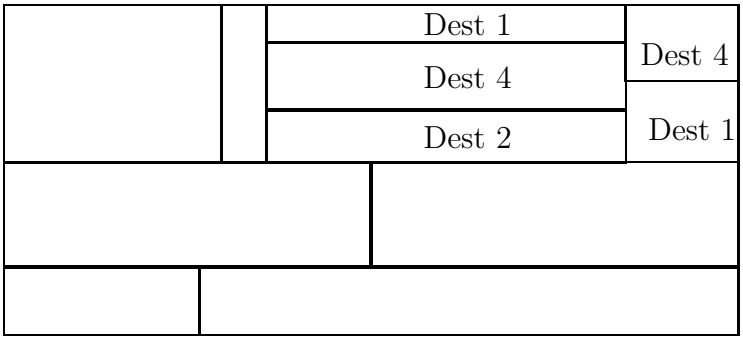


Abbildung 7.1: Unvermeidbare Überstauung

7.5 Obere Schranke für das rCSP

Wir betrachten nun noch einmal das Stauraumproblem für Containerschiffe und suchen nun eine Zuordnung eines Containers zu einer Spalte. Und zwar derart, dass die Anzahl der möglichen Überstauungen minimiert wird. Dies ist eine obere Schranke für die tatsächlichen Überstauungen. Wir gehen davon aus, dass ein Container seiner Spalte während der Fahrt treu bleibt.
Ein erster Versuch sieht so aus:

$$\min \sum_{k} \sum_{\substack{i,j \\ l_i < l_j < L_i < L_j}} x_{ik} \cdot x_{jk}$$

unter den Nebenbedingungen:

$$\sum_{k=1}^{K} x_{ik} = 1 \qquad \forall i \tag{7.17}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{K} x_{ik} = n \tag{7.18}$$

$$\sum_{l \in [l_i, L_i]} x_{ik} \leq h_k \qquad \forall k, l = 1, \ldots, L_{\max} \tag{7.19}$$

$$x_{ik} \in \{0, 1\} \tag{7.20}$$

Gegeben sind hier n Kisten mit Einladepunkten l_i und Ausladepunkten L_i, sowie K Spalten mit der Höhe h_k.
Die Variable x_{ik} gibt dann an, dass Kiste i in Spalte k gestaut wird. Die Bedingung (7.17) erzwingt, dass jede Kiste in genau eine Spalte gestaut wird, (7.18) erwzingt, daß alle Kisten verstaut sind und (7.19) sorgt dafür, daß nicht mehr Kisten in einer Spalte verstaut werden, als die Höhe der Spalte zulässt. Die Zielfunktion (7.5) zählt die Kisten in einer Spalte, die sich überstauen können, das heisst deren Ein- und Ausladeintervalle sich überlappen. Allerdings wird die Zahl der tatsächlichen Überstauungen überschätzt.
Das ist allerdings ein quadratisches Programm und solche sind im Allgemeinen schwer zu lösen. Gibt es eine Formulierung als gemischt-ganzzahliges lineares Programm (MIP)?
Dazu verwenden wir eine neue Variable y_{ij}, die angibt ob Kiste i und Kiste j in einer Spalte

gelagert werden oder nicht. Das sieht dann ingesamt so aus:

$$\min \sum_{k} \sum_{\substack{i,j \\ l_i < l_j < L_i < L_j}} y_{ij}$$

unter den Nebenbedingungen:

$$\sum_{k=1}^{K} x_{ik} = 1 \qquad \forall i \tag{7.21}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{K} x_{ik} = n \tag{7.22}$$

$$\sum_{l \in [l_i, L_i]} x_{ik} \leq h_k \qquad \forall k, l = 1, \ldots, L_{\max} \tag{7.23}$$

$$x_{ik} + x_{jk} \leq y_{ij} + 1 \qquad \forall i, j, k \tag{7.24}$$

$$x_{ik} \in \{0,1\} \quad y_{ij} in \{0,1\} \tag{7.25}$$

(7.25) ist eine Nebenbedingung der Art: Wenn a und b wahr sind, dann folgt, dass c wahr ist. Wie kann man allgemein logische Bedingungen als Nebenbedingung ausdrücken?

7.6 Auftreten von Ausladenebenbedingungen

Treten nun Ausladenebenbedingunge bei dreidimensionalen Packungsproblemen überhaupt auf? Dazu steht in [9] auf Seite 26:

> „Multi-dimensional problems occur especially in abstract C&P problems (e.g. in the case of multi-perio capital budgeting). They can also occur in loading containers, however, wehne an item is to be stored within a certain time frame, and time is considered as the fourth relevant dimension.“

In [31] wird eine Packungsproblem betrachtet und folgendes festgestellt:

> „This problem not only involves weight and dimensional constraints , but also customer unloading location constraints “ (p.1285 in [31])

Es werden Entladungspunkte definiert, die eine gewisse Entfernung zueinander haben und deren Entfernung nicht überschritten werden darf. Zwei „Bündel“, die auf demselben Wagen liegen, haben jeweils einen Einlagerungspunkt und die Entfernung der Einlagerungspunkte muss kleiner als 2 sein.

Ab hier Neues

In [6] wird auf verschiedene praktische Nebenbedingungen eingegangen, die bei Pakcungsproblemen auftreten können. Eine davon ist die sog „multi-drop situation“. Dazu das folgende Zitat:

> „If a container is to carry consignments for a number of different destinations, it is desirable not only to place items within the same consignment close together, but also to order the consignments within the container so as to avoid, as far as

> possible. having tp unload and re-load a large part of the cargo several times.“ (p.379 in [6])

Dort wird nun auch ein Verfahren angegeben, um einen Stauplan zu finden, der Überstauungen zu vermeiden versucht.

7.6.1 Wall-building-approach

Teile die Containertiefe in verschiedene Sektionen auf, die jeweils zu den Entladepunkten gehören. Dies kann nun in einem „Wall-building-approach“ geschehen, d. h. man versucht die Kisten eines Entladepunktes zu Wänden zu stauen, die man dann nach Entladepunkten absteigend in den Container lädt. Der Aufbau von gleichmässigen Wänden gelingt aber nur, wenn die Kistenmenge homogen genug ist - sonst kommt es zu relativ grossen ungenutzten Volumenbereichen.

7.6.2 Verfahren von Bischoff und Ratcliff

Die Autoren in [6] schlagen nun folgende vor:

1. Identifizierung der noch freien Räume
2. Identifizierung der noch nicht platzierten Kisten
3. Auswahl der Platzierung einer Kiste anhand von Kriterien:
   - Größte mögliche Volumenausnutzung multipliziert mit einen Maß für mögliche Stapelbildung
   - Kleinstmöglicher Zuwachs in Längsrichtung
   - Größtes Kistenvolumen
   - Kleinster Wert der Breitenkoordinate
4. Iteriere nach Update der Daten

Das Verfahren versucht nicht nur Stapel zu bilden und das wir erreicht durch geschickte Wahl des Maßes für mögliche Stapelbildung. Es wird versucht solangen wie möglich im hinteren Teil des Containers zu laden und auch um das „Profil“ des Stauplans hinreichend glatt zu halten. Um Entladepunkte zu berücksichtigen müssen nun die Kistenmengen partitioniert werden, sonst kommt es zu einer Durchmischung.
Dieses Verfahren baut tendenziell eher Stapel aus Wänden und ist deshalb für heterogene Kistenmengen eher geeignet. Dies ist für die Fragestellung mehrerer Entladepunkte von besonderem Interesse, da hier viele gleiche Kisten vorkommen können, jedoch mit verschiedenen Entladepunkten.
Nachteilig ist jedoch, daß die Stabilität der Staupläne schlecht sein kann und etwagige Gewichtsrestriktionen nicht beachtet werden. Diese müssten nun geeignet ins das Verfahren eingenbaut werden.
Wie sehen nun die Regeln genau aus? Die Identifizierung der noch freien Räume sucht nach ebenen, zusammenhängenden und rechtwinkligen Flc̈hen und die Höhe ist der Abstand bis zum Containerdach. Die Zuordnung einer Kiste erlaubt kein Überstehen über entsprechende Ladefläche, d.h. alle Kisten werden mit ihrer Grundfläche voll unterstützt. Es wird nun unter allen Kistenorientierungen und allen freien Räumen eine Kiste zugeordnet und zwar nach

den angegebenen Regeln.
Es seien x, y, z die nach der Länge, Breite, Höhe ausgerichteten Kanten einer Kiste, von der es noch m Unverstauute gebe, und X, Y, Z die Länge, Breite und Höhe des betrachteten freien Raumes mit dem Referenzpunkt (a, b, c). Eine Kiste passt nur in einen freien Raum, wenn $x \leq X, y \leq Y$ und $z \leq Z$. Dann sehen die Kriterien so aus:

$$\text{Suche maximales} \qquad u = \left(\frac{x \cdot y \cdot z}{X \cdot Y \cdot Z}\right) \cdot \min\left(\lfloor \frac{Z}{z} \rfloor, m\right) \tag{7.26}$$

$$\text{Mit minimalem Zuwachs in Längsrichtung} \qquad p = a + x \tag{7.27}$$

$$\text{Größtem Volumen} \qquad v = x \cdot y \cdot z \tag{7.28}$$

$$\text{Kleinstem Referenzwert} \qquad b \tag{7.29}$$

Literaturverzeichnis

[1] Aslidis, A.H.: Combinatorial Algorithms for Stacking Problems, Massachusetts Institute of Technology, Diss., January 1989

[2] Avriel, Mordecai ; Penn, Michal: Exact and approximate solutions of the container ship stowage problem. In: Computers and Industrial Engineering 25 (1993), Nr. 1-4, S. 271–274

[3] Avriel, Mordecai ; Penn, Michal ; Shpirer, Naomi: Container ship stowage problem complexity and connection to the coloring of circle graphs. In: Discrete Applied Mathematics 103 (2000), Nr. 1-3, S. 271–279

[4] Avriel, Mordecai ; Penn, Michal ; Shpirer, Naomi ; Witteboon, Smadar: Stowage planning for container ships to reduce the number of shifts. In: Annals of Operations Research 76 (1998), S. 55–71

[5] Biebig, Peter ; Althof, Wolfgang ; Wagener, Norbert: Seeverkehrswirtschaft. 2. München: Oldenbourg, 1995

[6] Bischoff, E.E. ; Ratcliff, M.S.W.: Issues in the Development of Approaches to Container Loading. In: Omega, International Journal of Management. Science 23 (1995), Nr. 4, S. 377–390

[7] Chen, C.S. ; Lee, S.M. ; Shen, Q.S.: An analytical model for the container loading problem. In: Europen Journal of Operational Research 80 (1995), Nr. 1, S. 68–76

[8] Dubrovsky, Opher ; Levitin, Gregory ; Penn, Michal: A Genetic Algorithm with a Compact Solution Encoding for the Container Ship Stowage Problem. In: Journal of Heuristics 8 (2002), Nr. 6, S. 585–599

[9] Dyckhoff, H. ; Finke, U.: Cutting and Packing in Produktion and Distribution. Heidelberg: Physica.Verlag, 1992 (Contributions to Management Science)

[10] Gilmore, P.C. ; Gomory, R.E.: Multistage cutting stock problems of two and more dimensions. In: Operations Research 13 (1965), S. 94–120

[11] Hammer, Gerald: Vorlesung: Moderne Heuristische Verfahren / Institut für Anwendungen des Operations Research, Universität Karlsruhe. 2003. – Forschungsbericht

[12] Ho, Y.C. ; Pepyne, D.L.: Simple Explanation of the No-Free-Lunch Theorem and Its Implications. In: Journal of optimization theory and applications 115 (2002), Nr. 3, S. 549–570

[13] Jellinghaus, Andreas: Vermeiden vom Umstapelungen in Staupläne von Containerschiffen, Fakultät für Wirtschaftswissenschaften, Universität Karlsruhe, Diplomarbeit, 2004

[14] Kämmerer, Lutz: Mathematische Modellierung und Behandlung von Stapelproblemen, Bauhaus-Universität Weimar, Diss., September 1997

[15] Kang, J-G. ; Kim, Y-D: Stowage planning in maritime container transportation. In: Journal of the Operational Research Society 53 (2002), S. 415–426

[16] Keber, Reinhard: Stauraumprobleme bei Stückguttransporten / Institut für Fördertechnik der Universität Karlsruhe. 1985 (17). – Wissenschaftlicher Bericht

[17] Kemp, Samuel E. ; Roach, Paul A. ; Ware, Andrew J. ; Wilson, Ian D.: Artificial Intelligence for Automatic Container Stowage Planmning Optimization. In: Ship Technology Research 50 (2003), S. 151–156

[18] Kienzle, Jörg: Schriftenreihe Transport und Verkehr. Bd. 1: Das Container-Transportsystem. München: Heinrich Vogel, 1991

[19] Kim, Kap H.: Evaluation of the Number pf Rehandles in Container Yards. In: Computers & Industrial Engineering 32 (1997), Nr. 4, S. 701–711

[20] Margarita, Sergio: The Towers of Hanoi: A New Approach. In: AI Expert 3 (1993), S. 22–27

[21] Martello, Silvano ; Pisinger, David ; Vigo, Daniele: The Three-Dimensional Bin Packing Problem. In: Operations Research 48 (2000), Nr. 2, S. 256–267

[22] Michalewicz, Zbigniew ; Fogel, David B.: How to Solve It: Modern Heuristics. Heidelberg: Springer, 2000

[23] Rotem, D. ; Urrutia, J.: Finding Maximum Cliques in Circle Graphs. In: Networks 11 (1981), S. 269–278

[24] Schott, Rainer: Verkehrswissenschftliche Studien. Bd. 39: Stauplanung für Containerschiffe. Göttingen: Vandenhoeck u. Ruprecht, 1989

[25] Shields, Jonathan J.: Containership Stowage: A Computer-Aided Preplanning System. In: Marine Technology 21 (1984), Nr. 4, S. 370–383

[26] Steenken, Dirk ; Winter, Thomas ; Zimmermann, Uwe T.: Stowage and Transport Optimization in Ship Planning. In: Grötschel, M. (Hrsg.) ; Krumke, S.O. (Hrsg.) ; Rambau, J. (Hrsg.): Online Optimization of Large Scale Systems. Berlin et al.: Springer, 2002, S. 731–745

[27] Stopford, Martin: Maritime Economics. 2. London: Routledge, 1997

[28] Terno, Johannes ; Lindemann, Rudolf ; Scheithauer, Guntram ; al., Birnbaum et (Hrsg.): Zuschnittprobleme und ihre praktische Lösung. Mathematische Modelle von Layout-Problemen (Layout problems and their practical solution. Mathematical models of layout problems). (Lizenzausg. des VEB Fachbuchverlag Leipzig). Thun-Frankfurt/Main: Verlag Harri Deutsch., 1987 (Mathematik für Ingenieure)

[29] Todd, David S. ; Sen, Pratyush: A Multiple Criteria Genetic Algorithm for Containership Loading. In: Bäck, Thomas (Hrsg.): Proceedings of the Seventh International Conference on Genetic Algorithms. San Francisco, California : Morgan Kaufmann, 1997, S. 674–681

[30] Unger, Walter: On the k-colouring of circle-graphs. In: STACS 88, Theoretical aspects of computer science, Proc. 5th Annu. Symp., Bordeaux/France 1988 Bd. 294. New York et al.: Springer, 1988, S. 61–72

[31] Vasko, Francis J. ; Mcnamara, John A. ; Newhart, dennis D. ; Wolf, Floyd E.: A Practical Solution to a Cargo Loading Problem at Bethlehem Steel. In: Journal of the Operationl Research Society 45 (1994), Nr. 11, S. 1285–1292

[32] Wilson, Ian D.: The Application of Artificial Intelligence Techniques to the Deep-Sea Container-Ship Cargo Stowage Problem, University of Glamorgan - School of Accounting and Mathematics, Diss., May 1997

[33] Wilson, I.D. ; Roach, P.A.: Principlies of Combinatorial Optimization Applied to Container-Ship Stowage Planning. In: Journal of Heuristics 5 (1999), S. 403–418

[34] Wilson, I.D. ; Roach, P.A.: Container stowage planning: a methology for generating computerised solutions. In: Journal of the Operational Society 51 (2000), S. 1248–1255

[35] Wilson, I.D. ; Roach, P.A. ; Ware, J.A.: Container stowage pre-planning: using search to generate solutions, a case study. In: Knowledge-Based Systems 14 (2001), Nr. 3, S. 137–145

[36] Wottawa, Michael: Struktur und algorithmische Behandlung von praxisorientierten dreidimensionlalen Packungsproblemen, Mathematisch-Naturwissenschaftliche Fakultät der Universität Köln, Diss., 1996