



TECHNISCHE
UNIVERSITÄT
WIEN

Vienna University of Technology

DIPLOMARBEIT

Mathematische Programmierung bei Continuous Casting Problemen

Ausgeführt am Institut für
Wirtschaftsmathematik
der Technischen Universität Wien

betreut von
Dr. Josef Haunschmied

unter der Leitung von
Dr. Gernot Tragler

eingereicht an der
Technischen Universität Wien durch

Stefan Zojer
Gauermannstraße 40
2542 Kottlingbrunn

Wien, am 7. August 2011

Zusammenfassung

Das Steelmaking und Continuous Casting (SCC) ist einer der Engpässe in der Stahlindustrie. Da Stahl in vielen Industrien zum Einsatz kommt, herrscht ein großes wirtschaftliches Interesse daran, den Output eines Stahlwerks zu optimieren. Es wurden viele Verfahren entwickelt, um sich diesem Problem zu stellen. Einige sind nur für bestimmte Werke anwendbar, andere überall einsetzbar. In dieser Arbeit beschäftige ich mich mit verschiedenen Lösungsansätzen und werde speziell einen, den Column Generation Ansatz, herausgreifen und genauer untersuchen.



Abbildung 1: Stahlstränge beim Casting.

Quelle: <http://www.matsoc.com/assets/pastevents/concast.jpg>

Abstract

Steelmaking and continuous casting (SCC) is one of the major bottlenecks of the steel industry. As steel is used in many different industries, there is a great economical interest in optimizing the output of a steel plant. Many different methods to solve this problem have been investigated. While some of them are optimal for specific plants only, others can be applied universally. In this paper, I will examine various approaches to solve this issue and deal with one, the column generation approach, in greater detail.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die eingereichte Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe. Ich erkläre weiters, dass ich diese Diplomarbeit bisher weder im In- noch im Ausland in irgendeiner Form als wissenschaftliche Arbeit vorgelegt habe.

Wien,

Stefan Zojer

Danksagung

Ich möchte mich an dieser Stelle bei all jenen Personen bedanken, die mir bei der Erarbeitung dieser Diplomarbeit geholfen haben.

Mein herzlicher Dank gilt Herrn Dr. Josef Haunschmied, dem Betreuer meiner Arbeit.

Ein großes Dankeschön geht an meinen Eltern und Geschwistern, die in jeder Phase meines Studiums für mich da waren.

Ich möchte auch all jene Freunde und Verwandte dankend erwähnen, die das Zustandekommen meiner Arbeit durch ihre Hilfestellung ermöglicht haben. Der Dank gilt meinen Studienkollegen Horst und Fabian, ebenso wie meinen Eltern, die sich die Zeit genommen haben, meine Arbeit mathematisch und grammatikalisch korrekturzulesen.

Abschließend geht ein sehr herzliches Dankeschön an all meine Freunde und Kollegen, die mir während meiner härtesten Prüfungen der letzten Jahre eine helfende Hand reichten.

Vielen Dank,

Stefan

Inhaltsverzeichnis

1	Einleitung	1
2	Das Continuous Casting Problem von Tang und Wang	4
3	Lösungsansätze	15
3.1	Mathematische Optimierungsverfahren	15
3.1.1	Mathematische Programmierung	15
3.1.1.1	Heuristische Methoden	15
3.1.1.2	MIP/LP	28
3.2	Expertenbasierte Systeme	40
4	Empirische Ergebnisse bei der Anwendung von Column Generation	43
4.1	Lösung eines linearen Modells	45
4.2	Formulierung des cast Zuordnungsproblems mittels Column Generation	45
4.3	Numerische Implementierung	48
4.3.1	Parameteränderungen	49
4.3.2	Lösungsgenauigkeit	51
5	Ein paar Ideen zu einer Benutzeroberfläche	56
6	Conclusio	58
A	Gams code: Column Generation	62

1 Einleitung

Die Stahlindustrie liefert das Rohmaterial für einige der größten Industrien der Welt. Das macht sie zu einem wichtigen Faktor für Wirtschaftsleistung eines Landes oder eines Kontinents. Durch die schnell aufstrebenden Industrienmächte Ostasiens wird es zunehmend wichtiger, dass Produktionsmechanismen verbessert werden, um den wachsenden Anforderungen standzuhalten.

Die Herstellung von Stahl kann grob in drei Schritte unterteilt werden:

Eisenherstellung: Aus Eisenerz, Kohle und einem Bindemittel wird flüssiges Eisen erzeugt.

Stahlherstellung: Das flüssige Eisen wird mit bestimmten Chemikalien zu Stahl verschiedener Qualitätsklassen verarbeitet und in Quadern, *slabs* genannt, verfestigt.

Finalgüter: Weitere Verarbeitung der slabs mittels hot rolling, cold rolling, galvanizing, etc..

Die Erzeugung des Stahls aus Eisen wird weiter in vier Phasen unterteilt: Zuerst wird das Eisen entschlackt und entschwefelt. Eine Transportkelle bringt es zum converter, der zuvor mit Schrott gefüllt wurde. Eine Füllung des converter wird *charge* genannt. In der nächsten Stufe, dem *refining*, wird der Stahl je nach gewünschter Qualität weiter verfeinert. Dies kann zum Beispiel eine Legierung oder weiteren Entzug von Schwefel oder Phosphor beinhalten. Nach dem refining wird der Stahl mit der Kelle zum *caster* gebracht.

Dieses ist eine Kelle mit einer zugespitzten Öffnung nach unten, aus der der flüssige Stahl rinnt. Die Fließgeschwindigkeit beträgt ungefähr 1.8 m/min. Der Stahl wird durch Rollen in die Länge gezogen und abgekühlt, bis er geschnitten werden kann. Wird dieser Vorgang unterbrochen, oder ist der caster einmal leer, so dauert das wieder in Stand setzen ungefähr eine Stunde.

Dieser letzte Schritt, das *casting* ist die Engstelle der Stahlproduktion. Das Ziel ist es, den caster ohne Unterbrechung laufen zu lassen, da jede Pause mit Kosten und Gewinnentgang verbunden ist.

Beim casting muss man darauf aufpassen, dass nach einander produzierte slabs von gleicher, oder zumindest ähnlicher Qualität sind. Die Füllung eines tundish, im weiteren Verlauf werde ich nur noch den Begriff cast benutzen, setzt sich aus mehreren charges zusammen. Ein charge wiederum besteht aus

mehreren slabs. Sowohl die Zuordnung von slabs zu charges, als auch jene von charges zu casts wird in der Literatur behandelt.

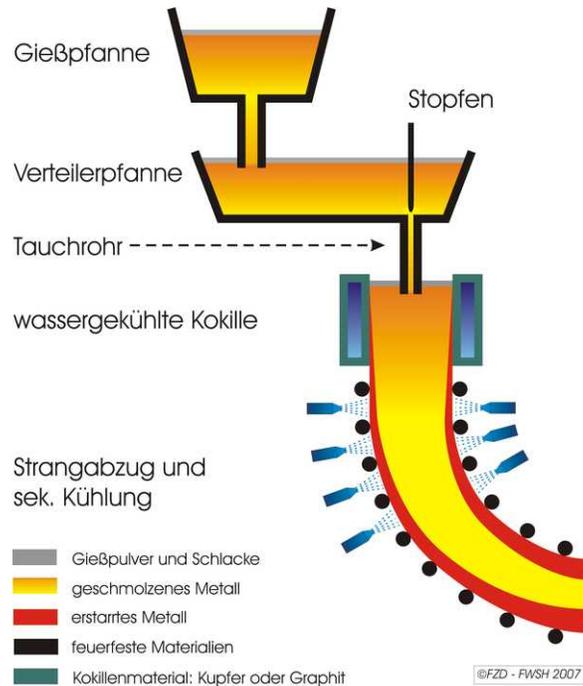


Abbildung 2: Der Castingprozess.

Quelle: <http://www.hzdr.de/db/PicOri?pOid=25714>

Ich setzte mich in meiner Arbeit lediglich mit dem cast Zuordnungsproblem auseinander. Die Zuordnungen unterliegen bestimmten Beschränkungen, auf die ich im Verlauf dieser Arbeit noch eingehen werde.

Ich werde verschiedene mathematische Verfahren beschreiben, die versuchen, dieses Problem optimal zu lösen. Die Optimierungsziele sind auf Grund der speziellen Beschaffenheit des Problems von verschiedener Gestalt. Neben der Maximierung des Profits oder des Outputs werden auch ununterbrochenes casting, über die gesamte Lebensdauer eines tundish, beziehungsweise die Minimierung der Kosten, bestehend aus Produktions- und Strafkosten, betrachtet. Die Produktionskosten werden bei den Verfahren ignoriert, da deren Berechnung keine Herausforderung für die Produktionskontrolleure be-

ziehungsweise die Buchhalter der Firmen darstellen. Die Strafkosten sind jedoch sehr variabel und gehen in den Optimierungsprozess ein. Sie können bestehen aus Preisabschlägen wegen zu schlechter Qualität der Ware, Gewinnentgang durch zu billig verkaufte, überqualifizierte Ware, nicht Termingemäßer Lieferung, der Unterbrechung des castings et cetera.

Die Stahlindustrie ist, wie bereits erwähnt, ein Eckpfeiler jeder großen Wirtschaftsmacht, deshalb arbeiten viele Firmen und Forscher fieberhaft daran, das Produktionsverfahren zu verbessern. Einige publizierte Verfahren sind nur für ganz bestimmte Werke optimal. Dabei spielt die Größe der Firma, etwa ob sie mehrere converter, refining-Maschinen oder caster haben, aber auch die Möglichkeit, Ware während der Produktion oder auch als fertiges Gut lagern zu können, eine Rolle.

2 Das Continuous Casting Problem von Tang und Wang

Um das Problem des continuous casting modellieren zu können, muss man zuerst feststellen, dass slabs, die gemeinsam in einen charge gemischt werden, gewisse Bedingungen erfüllen müssen. Tang und Wang [2008] [1] geben die Folgenden vor.

- Slabs müssen von zumindest ähnlicher Stahlqualität sein. Sind nicht alle slabs innerhalb eines charges identisch, werden jene mit schlechterer Qualität upgraded.
- Das kumulierte Gewicht aller slabs darf einen bestimmten Wert nicht überschreiten. Sind zu wenige slabs in einem charge, wird dieser mit nicht nachgefragten slabs aufgefüllt.
- Innerhalb eines charges darf es keine zu großen Unterschiede in der slab width geben.

Des Weiteren muss die Gesamtzahl der produzierten charges eine vorgegebene Schranke überschreiten. Dasselbe gilt für das kumulierte Gewicht der charges auf den einzelnen Produktionsschienen, den charges, die RH-refining durchlaufen, und das warm-up Material. Dieses sind spezielle slabs, die benutzt werden, um die Rollen des Warmbandes auf Betriebstemperatur zu bringen.

Hat man alle Bedingungen notiert, ist die nächste Frage jene nach der optimalen Zielsetzung. Tang und Wang benutzen als Zielfunktion eine gewichtete Summe von Strafkosten. Andere Möglichkeiten umfassen die Maximierung des Profits, die Minimierung der Produktionszeit oder eine möglichst kleine Anzahl von produzierten casts.

Die von Tang und Wang betrachteten Strafkosten bestehen aus Verfehlungen durch:

- Qualitätsupgrades von slabs.
- Überproduzierte slabs.
- Prioritäten nicht produzierter slabs.

Tang und Wang formulieren das Problem in zwei Schritten. Zuerst wird die Einteilung der slabs in charges in einem gemischt ganzzahligen Optimierungsproblem (Englisch: mixed integer problem, MIP) beschrieben, danach die Zuordnung von charges zu casts, ebenfalls durch ein MIP. Bezüglich der mathematischen Modelle besteht kein Unterschied zwischen MIP-1 und MIP-2, die Notation lässt sich teilweise übernehmen.

Notation MIP-1

N	Menge der bestellten slabs.
F	Menge der Produktionsschienen in cold rolling.
g_j	Gewicht des slabs j .
G	Gewicht eines charges.
q_i^{RH}	Binär. Gibt an, ob der charge mit mittlerem slab i ein RH-refining benötigt.
q_j^{pre}	Binär. Gibt an, ob slab j warm-up benötigt.
q_j^f	Binär. Gibt an, ob slab j durch Produktionsschiene f geht.
L_{chr}	Untere Grenze der produzierten slabs.
L_{RH}	Untere Grenze der slabs mit RH-refining.
L_{pre}	Untere Grenze für das Gewicht des warm-up Material.
L_f	Untere Grenze für das Gewicht der slabs, die durch Produktionsschiene f laufen.
d_j^{min}	Minimal erlaubte slab width.
d_j^{max}	Maximal erlaubte slab width.
Ω^1	Menge der Strafkoeffizienten.
c_{ij}	Strafkosten, wenn slab j mit der Stahlqualität von slab i produziert wird.
c_{ij}	$\begin{cases} 0 & \text{Identische Qualität.} \\ \omega_{ij} & \text{Kompatible Qualitäten.} \\ \infty & \text{Inkompatible Qualitäten.} \end{cases}$
p_j	Priorität des slab j .
B	Eine große Zahl.
Y_i	Gewicht der überschüssig produzierten slabs zum charge i .

w_i^{max}	Minimum der Maximalen width aller slabs zum charge i .
w_i^{min}	Maximum der Minimalen width aller slabs zum charge i .
$\lambda_{1,2,3}$	Gewichte.

Ein charge entsteht, wenn eine Menge von slabs einem medianen slab zugeordnet wird. Die Notation x_{ij} bedeutet, dass slab i in einem charge mit medianem slab j ist. $x_{ii} = 1$ heißt, dass slab i zum medianen slab i gehört. Eine Trivialität, die beschreibt, dass es den entsprechenden charge gibt. Die binären Zuordnungen x_{ij} sind die Entscheidungsvariablen.

Das Modell MIP-1

$$\min \lambda_1 \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} + \lambda_2 \sum_{i \in N} \omega_i Y_i + \lambda_3 \sum_{i \in N} \sum_{j \in N} p_j (1 - x_{ij}) \quad (1)$$

st.:

$$\sum_{i \in N} x_{ij} \leq 1 \quad i, j \in N, \quad (2)$$

$$x_{ij} \leq x_{ii} \quad i, j \in N \quad (3)$$

$$\sum_{j \in N} g_j x_{ij} + Y_i = x_{ii} G \quad i \in N \quad (4)$$

$$\sum_{i \in N} x_{ii} \geq L_{chr} \quad (5)$$

$$\sum_{i \in N} q_i^{RH} x_{ij} \geq L_{RH} \quad (6)$$

$$\sum_{i \in N} \sum_{j \in N} q_j^{pre} g_j x_{ij} \geq L_{pre} \quad (7)$$

$$\sum_{i \in N} \sum_{j \in N} q_j^f g_j x_{ij} \geq L_f \quad f \in F \quad (8)$$

$$w_i^{min} - d_j^{min} \geq B(x_{ij} - 1) \quad i, j \in N \quad (9)$$

$$w_i^{max} - d_j^{max} \leq B(1 - x_{ij}) \quad i, j \in N \quad (10)$$

$$w_i^{min} - w_i^{max} \leq 100 \quad i \in N \quad (11)$$

$$x_{ij} \in \{0, 1\} \quad i, j \in N \quad (12)$$

$$Y_i \geq 0 \quad i \in N \quad (13)$$

$$w_i^{min}, w_i^{max} \geq 0, \text{ ganzzahlig} \quad i \in N \quad (14)$$

Die Zielfunktion (1) enthält die gewichteten Strafkosten. Der erste Term betrachtet die Kompatibilität der slabs innerhalb eines charges, der zweite Term bewertet überproduzierte Ware und der dritte Term befasst sich mit der Priorität nicht produzierter slabs.

Bedingung (2) stellt sicher, dass jeder slab nur einem charge zugeordnet wird. Laut (3) darf ein slab nur in einen existierenden charge gefüllt werden. (4) kontrolliert die Gewichtsgrenze für einen charge. Die Gleichungen (5) bis (8) stellen sicher, dass die geforderten Untergrenzen eingehalten werden. Bedingungen (9) bis (11) beachten die width Differenzen innerhalb der charges. Die Werte, welche die Variable annehmen können, stehen in (12) bis (14).

Die Bedingungen für das cast Zuordnungsproblem:

- Die Stahlqualität zweier aufeinander folgender charges muss kompatibel sein. Sind sie nicht ident, wird dies durch Strafkosten geahndet.
- Die castingdauer aller charges in einem cast darf die Lebenszeit des tundish nicht überschreiten. Nicht genutzte tundish Zeit wird ebenfalls bestraft.
- Die Zahl der width Sprünge zwischen aufeinander folgenden charges und innerhalb eines casts ist beschränkt.
- Die Zahl der charges innerhalb eines casts muss den Bedingungen für die einzelnen Produktionsschienen, caster Kapazität, RH-refining und warm-up Material genügen. Speziell für die einzelnen Produktionsschienen und das warm-up Material ist es erstrebenswert, dass gewisse Zielgewichte erreicht werden.

Die Zielfunktion belegt durch Strafkosten:

- Nicht genutzte tundish Zeit.
- Qualitätsupgrades.
- Die Priorität nicht produzierter slabs.
- Abweichung zum Zielgewicht bei den Produktionsschienen und dem warm-up Material.

Notation MIP-2

$F, L_{chr}, L_{RH}, L_{pre}, L_f$ wie in MIP-1.

P	Menge der charges $k, l \in P$.
M	Menge der casts $r \in M$.
Q_l^{RH}	Binär. Gibt an, ob charge l RH-refining benötigt.
Q_l^{pre}	Gewicht des warm-up Materials in charge l .
Q_l^f	Gewicht der slabs in charge l in Produktionsschiene f .
U_{chr}	Untergrenze für die gewählten charges.
U_{RH}	Untergrenze für die charges, die RH-refining benötigen.
U_{pre}	Untere Grenze für das Gewicht der charges, die warm-up benötigen.
O_{pre}	Zielwert für das Gewicht der charges, die warm-up benötigen.
U_f	Untergrenze für das Gewicht der slabs in Produktionsschiene f .
O_f	Zielwert für das Gewicht der slabs in Produktionsschiene f .
t_l	Produktionszeit für charge l im tundish.
T	Lebensdauer eines tundish.
S_l	Menge der verschiedenen width in charge l $S_l = \{(B_l^1, E_l^1), \dots, (B_l^m, E_l^m)\}$.
Ω^2	Menge der Strafkoeffizienten.
C_{kl}	Strafkosten für unterschiedliche Stahlqualität der charges k und l .
C_{kl}	$\begin{cases} 0 & \text{Identische Qualität.} \\ \omega_{kl} & \text{Kompatible Qualität.} \\ \infty & \text{Inkompatible Qualität.} \end{cases}$
p_l	Priorität von charge l . $p_l = \sum_{j \in N} (\lambda_3 p_j - \lambda_1 c_{ij}) x_{ij} - \lambda_2 \omega_i Y_i$
δ_{pre}	Strafkosten für die Abweichung zum Zielgewicht des warm-up Materials.
δ_f	Strafkosten für die Abweichung zum Zielgewicht der slabs in Produktionsschiene f .

- u_l Binär. Gibt an, ob es einen width Sprung in charge l gibt.
 u_{kl} Binär. Gibt an, ob zwischen charge k und charge l ein width Sprung ist.
 $\mu_{1,2,3,4}$ Gewichte.

Die Variable x_{klr} bedeutet, dass charge l nach charge k in cast r produziert wird. y_{lr} beschreibt die Tatsache, dass charge l in cast r ist. (B_l, E_l) ist die casting width von charge l .

Das Modell MIP-2

$$\begin{aligned}
 \min \mu_1 \omega_T & \left(\sum_{r \in M} \sum_{l \in P} T x_{0lr} - \sum_{r \in M} \sum_{l \in P} t_l y_{lr} \right) + \mu_2 \sum_{r \in M} \sum_{l \in P} p_l (1 - y_{lr}) \quad (15) \\
 & + \mu_3 \sum_{r \in M} \sum_{k \in P} \sum_{l \in P / \{k\}} C_{kl} x_{klr} + \mu_4 \left(\delta_{pre} + \sum_{f \in F} \delta_f \right)
 \end{aligned}$$

s.t.:

$$\sum_{k \in \{0\} \cup P / \{l\}} x_{klr} = \sum_{k \in \{0\} \cup P / \{l\}} x_{lkr} = y_{lr} \quad l \in P, r \in M \quad (16)$$

$$\sum_{l \in P} x_{0lr} = \sum_{l \in P} x_{l0r} \leq 1 \quad r \in M \quad (17)$$

$$\sum_{r \in M} y_{lr} \leq 1 \quad l \in P \quad (18)$$

$$\sum_{l \in P} t_l y_{lr} \leq T \quad r \in M \quad (19)$$

$$0 \leq \sum_{r \in M} (E_k - B_l) x_{klr} \leq 100 \quad k, l \in P \quad (20)$$

$$\sum_{l \in P} u_l y_{lr} + \sum_{k \in P} \sum_{l \in P / \{k\}} u_{kl} x_{klr} \leq 1 \quad r \in M \quad (21)$$

$$L_{chr} \leq \sum_{r \in M} \sum_{l \in P} y_{lr} \leq U_{chr} \quad (22)$$

$$L_{RH} \leq \sum_{r \in M} \sum_{l \in P} Q_l^{RH} y_{lr} \leq U_{RH} \quad (23)$$

$$L_{pre} \leq \sum_{r \in M} \sum_{l \in P} Q_l^{pre} y_{lr} \leq U_{pre} \quad (24)$$

$$L_f \leq \sum_{r \in M} \sum_{l \in P} Q_l^f y_{lr} \leq U_f \quad f \in F \quad (25)$$

$$\begin{aligned} \delta_{pre} = & \max \left\{ 0, \omega_{pre}^+ \left(\sum_{r \in M} \sum_{l \in P} Q_l^{pre} y_{lr} - O_{pre} \right) \right\} \\ & + \max \left\{ 0, \omega_{pre}^- \left(O_{pre} - \sum_{r \in M} \sum_{l \in P} Q_l^{pre} y_{lr} \right) \right\} \end{aligned} \quad (26)$$

$$\begin{aligned} \delta_f = & \max \left\{ 0, \omega_f^+ \left(\sum_{r \in M} \sum_{l \in P} Q_l^f y_{lr} - O_f \right) \right\} \quad f \in F \\ & + \max \left\{ 0, \omega_f^- \left(O_f - \sum_{r \in M} \sum_{l \in P} Q_l^f y_{lr} \right) \right\} \end{aligned} \quad (27)$$

$$(B_l, E_l) \in S_l \quad l \in P \quad (28)$$

$$u_l, u_{kl}, x_{klr}, y_{lr} \in \{0, 1\} \quad k, l \in P, r \in M \quad (29)$$

Die Terme in der Zielfunktion (15) beschreiben, gewichtet durch μ_1 , die Strafkosten für übrig gebliebene tundish Zeit, μ_2 die Prioritätskosten durch nicht produzierte slabs, μ_3 die Strafkosten bezüglich Qualitätsdifferenzen und μ_4 die Strafkosten für die Abweichungen von den Zielgewichten pro Produktionsschiene und warm-up Material.

Bedingung (16) stellt sicher, dass jeder charge nur einen Vorgänger und einen Nachfolger haben kann. Bedingung (17) bestimmt, dass jeder cast einen ersten und einen letzten charge hat und Gleichung (18) besagt, dass jeder charge nur in einem cast vorkommen kann. Beschränkung (19) beschäftigt sich mit der Lebensdauer eines tundish. In Gleichung (20) stehen die erlaubten width Wechsel zwischen aufeinander folgenden charges, in (21) findet man die Zahl der erlaubten width Wechsel innerhalb eines casts. Beschränkungen (22)-(25) stellen sicher, dass die Menge der produzierten charges die vorgegebenen Grenzen einhält. In den Gleichungen (26) und (27) werden die Strafkosten bezüglich der Abweichung vom Zielgewicht ermittelt. Beschränkung (28) besagt, dass eine der width in charge l , ein Element aus S_l , als casting width gewählt wird. (29) gibt den Rahmen der Variablen an.

Der Produktionsablauf erlaubt es, die Probleme hierarchisch zu ordnen. In einem charge befinden sich 10 bis 20 slabs, somit ist die Zahl der charges in MIP-2 rund $\frac{1}{15}$ mal der Zahl der slabs in MIP-1, was einen erheblichen zeitlichen Vorteil bringt. Da das Problem schon für kleine Bestellgrößen nicht mehr in zufriedenstellender Zeit exakt gelöst werden kann, werden die Modelle heuristisch gelöst.

Der Algorithmus für die auf dynamischer Programmierung basierende Heuristik für das charge Zuordnungsproblem, MIP-1, verläuft schrittweise:

1. Die slabs werden nach Stahlgrad gruppiert.
2. Auf jede Gruppe N_s mit identischem Grad wird eine auf dynamischer Programmierung basierende Heuristik angewandt, um charges zu bilden. Die ausgewählten slabs werden aus der Gruppe entfernt.
3. Sind alle Mengen leer, endet das Verfahren. Andernfalls gehe zu Schritt 4.
4. Zwischen ungewählten charges und solchen, die schon einem charge zugewiesen sind, wird ein lokaler Austausch durchgeführt.
5. Wenn Beschränkung (8) erfüllt ist, endet das Verfahren. Andernfalls gehe zu Schritt 6.
6. Mische zwei oder mehr nicht-leere Gruppen in eine Menge, wähle slabs aus ihr und generiere charges, bis Bedingung (8) erfüllt ist, oder die Strafkosten sinken. Wiederhole diesen Vorgang, bis Gleichung (8) erfüllt ist.
7. Wiederhole Schritt 6 für die Beschränkungen (5), (6) und (7).

Die in Schritt 2 erwähnte Heuristik, nach der die charges gebildet werden:

1. Ordne die slabs in N_s in absteigender Reihenfolge nach d_j^{min} , d_j^{max} und p_j .
2. Konstruiere ein Netzwerk $G = (N, A)$. $N = N_s \cup \{0\}$, $A = \{(i, j) | i, j \in N, i < j, \sum_{k=i+1}^j g_k \leq G, \max_{k=i+1}^j \{d_k^{min}\} - \min_{k=i+1}^j \{d_k^{max}\} \leq 100\}$. Das Gewicht eines Pfeiles wird beschrieben durch $w_{ij} = \min\{\lambda_2 \omega_j(G - \sum_{k=i+1}^j g_k), \lambda_3 \sum_{k=i+1}^j p_k\}$.
3. Suche den kürzesten Weg auf G . Benutze dabei die rekursive Funktion $f_j = \min_{(i,j) \in A} \{f_i + w_{ij}\}$.

4. Transformiere den kürzesten Weg in eine intuitiv gute Lösung.
Ist $(0, j_1, j_2, \dots, j_h, n_s)$ der kürzeste Weg, so hat man $h + 1$ Pfeile mit $w_{j_i j_{i+1}} = \lambda_2 \omega_{j_{i+1}} (G - \sum_{k=j_i+1}^{j_{i+1}} g_k)$. Die slabs $j_i + 1$ bis j_{i+1} werden in einen charge gruppiert.

Für das cast Zuordnungsproblem, MIP-2, wird eine Tabu Search basierte Heuristik angewandt.

Tabu Search ist ein Näherungsverfahren, das ausgehend von einer initialen Lösung alle Nachbarn, das sind Lösungskonstellationen, die sich nur in einer oder wenigen Komponenten unterscheiden, nach einer möglichen Verbesserung der Zielfunktion absucht und diese gegebenenfalls aktualisiert. Schrittweise wird die aktuelle Lösung verbessert, bis ein vom Benutzer vorgegebenes Abbruchkriterium erfüllt ist. Eine genaue Beschreibung dieses Verfahrens erfolgt in Kapitel 3.1.1.

Die Beschränkungen (22)-(25) können nur sehr schwer eingehalten werden. Will man auch noch die Strafkosten minimieren, so ist das Problem nicht in annehmbarer Zeit lösbar. Deshalb werden die Bedingungen (24) und (25) als Straffunktionen in die Zielfunktion geschrieben. Das relaxierte Modell sieht folgendermaßen aus.

$$\begin{aligned} \min \mu_1 C_T \left(\sum_{r \in M} \sum_{l \in P} T x_{0lr} - \sum_{r \in M} \sum_{l \in P} t_l y_{lr} \right) + \mu_2 \sum_{r \in M} \sum_{l \in P} p_l (1 - y_{lr}) \\ + \mu_3 \sum_{r \in M} \sum_{k \in P} \sum_{l \neq k, l \in P} C_{kl} x_{klr} + \mu_4 \left(\delta_{pre} + \sum_{f \in F} \delta_f \right) \end{aligned}$$

δ_f und δ_{pre} beschreiben die Strafkosten bezüglich warm-up Material beziehungsweise pro Produktionsschiene. $W_{pre} = \sum_{r \in M} \sum_{l \in P} Q_l^{pre} y_{lr}$ ist das Gewicht der charges, die warm-up benötigen. $W_f = \sum_{r \in M} \sum_{l \in P} Q_l^f y_{lr}$ jenes der charges, die in Produktionsschiene f erzeugt werden.

$$\delta_f = \begin{cases} \beta_f^- (L_f - W_f) & W_f < L_f \\ \omega_f^- (O_f - W_f) & L_f \leq W_f < O_f \\ \omega_f^+ (W_f - O_f) & O_f \leq W_f \leq U_f \\ \beta_f^+ (W_f - U_f) & W_f > U_f \end{cases}$$

$$\delta_{pre} = \begin{cases} \beta_{pre}^- (L_{pre} - W_{pre}) & W_{pre} < L_{pre} \\ \omega_{pre}^- (O_{pre} - W_{pre}) & L_{pre} \leq W_{pre} < O_{pre} \\ \omega_{pre}^+ (W_{pre} - O_{pre}) & O_{pre} \leq W_{pre} \leq U_{pre} \\ \beta_{pre}^+ (W_{pre} - U_{pre}) & W_{pre} > U_{pre} \end{cases}$$

Berücksichtigt werden die Beschränkungen (16) bis (20) und (28) bis (29). $\beta_f^+, \beta_f^-, \beta_{pre}^+$ und β_{pre}^- sind Strafkoeffizienten. Die Koeffizienten β sind weitaus größer als $\omega_f^+, \omega_f^-, \omega_{pre}^+, \omega_{pre}^-$.

Aus der Menge der generierten casts werden so lange welche ausgewählt, bis alle Bedingungen erfüllt sind. Ist dies nicht trivial möglich, so wird ein Austauschverfahren angewandt. Hierbei werden charges, die in einem cast sind, der bereits der Lösung angehört, mit solchen charges getauscht, die einem cast zugeteilt wurden, der noch nicht ausgewählt wurde.

Die Nachbarschaftsstruktur des Verfahrens ist definiert über die Schrittmöglichkeiten.

- Direkter charge Tausch: Ein charge aus einem ausgewählten cast wird mit einem aus einem ungewählten cast getauscht. Dabei muss die Position des neuen charges nicht unbedingt der des alten entsprechen. Es ist nur festgelegt, welcher charge in welchen cast kommt.
- Direkte charge Einsetzung: Ein ungewählter charge wird in einem cast an einer passenden Stelle eingesetzt.
- Indirekter charge Tausch: Ein charge wird aus cast A gelöscht. Ein anderer charge aus cast B wird an seiner Stelle neu in cast A eingesetzt. Ein ungewählter charge ersetzt den fehlenden in cast B.
- Indirekte charge Einsetzung: Ist noch tündish Zeit verfügbar, wechselt ein charge aus cast B nach cast A. Ein ungewählter charge wird in cast B eingesetzt.
- Charge Tausch: Ein Paar gewählter charges tauscht die zugehörigen casts.

Das eben vorgestellte MIP und vergleichbare Formulierungen sind Inhalt zahlreicher Arbeiten. Andere Optimierungsansätze des Modells umfassen Dynamisierung, mathematische Programmierung, Heuristiken und Simulationsverfahren. Im ersten Teil dieser Arbeit werden nun einige dieser Möglichkeiten beschrieben.

Der Leser wird feststellen, dass in den Modellen mancher Verfahren nicht alle Nebenbedingungen ausformuliert werden. Das hat den Grund, dass die Verfahren hier nur schematisch behandelt werden und diese Bedingungen bei der Lösungsmethodik nicht weiter relevant sind.

3 Lösungsansätze

Wie bereits erwähnt, kann man für das continuous casting Problem durch verschiedene Lösungsansätze ein gutes Ergebnis erarbeiten. Es wird unterschieden zwischen preskriptiven Verfahren, wie mathematischen Optimierungsverfahren, transskriptiven Verfahren, durch Simulation, oder Experten basierten Verfahren.

Aus der ersten Gruppe, den mathematischen Optimierungsverfahren, werde ich ein paar Heuristische Methoden vorstellen. Meine Auswahl umfasst dabei Tabu Search, Simulated Annealing, Harmony Search, Ant Colony Optimization, Auction based Approach und ein Alternative Graph Verfahren.

Als Alternative zu den Heuristiken, die sich der Lösung nur annähern, bietet die mathematische Programmierung mehrere Tools zur Optimierung. Um die Rechenzeit im Rahmen zu halten, weicht man aber auch in diesem Fall manche Bedingungen derart auf, dass die Lösung nicht mehr optimal, sondern nur gut ist. Die von mir beschriebenen Verfahren sind Column Generation, Lagrange Relaxation und zwei Möglichkeiten, ein aufgestelltes gemischt ganzzahliges Problem zu vereinfachen.

Zuletzt beschreibe ich noch anhand eines Beispiels einen Experten basierenden Lösungsansatz.

3.1 Mathematische Optimierungsverfahren

3.1.1 Mathematische Programmierung

3.1.1.1 Heuristische Methoden

Tabu Search (TS) mit “Cannibalization”

Lopez, Carter und Gendreau [1998] [2] postulieren in ihrer Arbeit eine Anordnung der slabs in Form eines Sarges. Diese Form hat den Vorteil, das Übergänge in width, hardness, gauge und residence time sehr sachte vonstatten gehen können. Die Form besteht aus einer Aufwärmphase, dem breitesten Punkt und der Abschlussphase. Es ergeben sich weitere Bedingungen für manche slabs, etwa dass sie in der Mitte des Sarges platziert sein sollen, da mit fortlaufender Benutzung die Verarbeitungsgeräte Verschleißerscheinungen zeigen und die gewünschte Qualität nicht mehr gewährleistet werden kann.

Das Problem wird als erweitertes Travelling Salesman Problem (TSP) formuliert. Die Erweiterung besteht darin, dass Preise für besuchte Städte (verarbeitete slabs) zuerkannt werden, jedoch auch Strafkosten für nicht besuchte Städte (nicht produzierte slabs). Die binäre Variable x_{ij} gibt an, ob slab j nach slab i produziert wird. Mit c_{ij} werden die Strafkosten bezeichnet, die anfallen, wenn die slabs i, j nacheinander produziert werden. w_i ist der Gewinn, der durch den Verkauf von slab i lukriert wird. L, U sind Beschränkungen dafür, wieviel in einem Stück, ohne Austausch der Verarbeitungsgeräte, produziert werden kann. x_{00} ist eine technische Notwendigkeit, um einen ersten slab zu gewährleisten.

Das Modell

$$\min \sum_{i=0}^n \sum_{j=0}^n c_{ij} x_{ij}$$

s.t:

$$\sum_{j=0}^n x_{ij} = 1 \quad i = 1, \dots, n$$

$$\sum_{i=0}^n x_{ij} = 1 \quad j = 1, \dots, n$$

$$L \leq \sum_{i=1}^n w_i x_{ij} \leq U$$

$$x_{00} = 0 \quad x_{ij} \in \{0, 1\}$$

∃! Zyklus mit Länge ≥ 2

Die übrigen Beschränkungen für width, hardness, gauge und Zeit werden der Einfachheit halber weggelassen. Sie sind in einer kurzen, schematischen Beschreibung des Verfahrens nicht von Bedeutung.

Das Verfahren Tabu Search mit Cannibalization besteht aus mehreren Schritten. Zuerst beschreibe ich kurz die allgemeine Tabu Search (TS) Idee.

TS ist ein heuristisches Verfahren, das vermeidet, in einem lokalen Optimum hängen zu bleiben. Die Eckpfeiler des Verfahrens sind die *initiale Lösung*, der *Schritt*, die *Nachbarschaft*, die *Tabu Liste* und die *Abbruchkriterien*.

Ausgehend von der initialen Lösung verändert man eine Eigenschaft der aktuellen Lösung gemäß der Bewegungsvorschrift und erhält die Nachbarschaft. Dabei sind manche Veränderungen ausgeschlossen, wie zum Beispiel

die Invertierung des letzten Schrittes. Welche Elemente aus der Nachbarschaft ausgenommen werden, steht in der Tabu Liste. Unter den übrigen Möglichkeiten wird die bezüglich einer Zielfunktion beste Veränderung gewählt und zur neuen Lösung gemacht.

Bei jedem Schritt verändert sich die Lösung! Im Gegensatz zu anderen Verfahren wird auch eine neue Lösung akzeptiert, bei der sich die Zielfunktion im Vergleich zur alten Lösung verschlechtert.

Das Verfahren wird so lange durchgeführt, bis ein Abbruchkriterium erfüllt ist.

Der einfache Tabu Search:

Im vorliegenden Fall wird in einem *Schritt* von der aktuellen Lösung ein slab eliminiert und ein anderer eingetauscht. Hierbei müssen schon einige Nebenbedingungen beachtet werden. Aus der slab Formation i, j, k soll slab j durch einen slab r ausgewechselt werden. Gesucht wird dabei jener slab r , der $width_i \leq width_r \leq width_k$ erfüllt und $c_{ir} + c_{rk} - c_{ij} - c_{jk}$ minimiert. Die dargestellte width Bedingung gilt für die Aufwärmphase, in der Abschlussphase drehen sich die Ungleichungszeichen um.

Die *Nachbarschaft* besteht aus allen durch die Bewegungsvorschrift erreichbaren Konstellationen.

Die *Tabu Liste* verbietet die Elimination eines eingetauschten slabs für die nächsten θ Schritte. Ein aus der Lösung eliminiertes slab darf ebenfalls θ Schritte nicht wieder eingewechselt werden. Die Tabu Liste hat Platz für zehn Einträge.

Abbruchkriterien sind eine maximale Anzahl von 50 Iterationen, oder 20 Iterationen ohne eine Verbesserung der Zielfunktion.

Die Autoren verwenden in ihrer Arbeit neben dem einfachen TS auch noch den CROSS Tausch. Hierbei werden nicht nur einzelne slabs getauscht.

CROSS Tausch:

Es werden ganze Gruppen von slabs getauscht. Dazu wird eine in Sargform gegebene Lösung auf gute und schlechte Sektionen untersucht. Den Anfang eines Blocks schlechter slabs bildet ein slab mit Strafkosten. Von diesem schlechten slab ausgehend wird so lange gesucht, bis zehn gute slabs aufeinander folgen. Der letzte schlechte slab beendet die schlechte Gruppe. Der slab vor dem ersten der Sektion (*vorhergehender*) und der erste nach dem letzten (*nächster*) bestimmen die width Beschränkungen für diese Gruppe.

Da die Bemessungen durch die slabs *vorhergehender* und *nächster* bekannt sind, werden Blöcke mit den passenden Verhältnissen gesucht. Jene Gruppe, die den größten Nutzengewinn, die Reduktion der über die Gruppe aufsummierten Strafkosten, liefert, wird für die vorhandene schlechte Sektion getauscht.

Das von Lopez, Carter und Gendreau entwickelte Verfahren benötigt neben der Tabu Liste für den einfachen TS auch eine für den CROSS Tausch. Die Funktion dieser Liste, genannt Kannibalen Liste, ist analog zur Tabu Liste, es stehen jedoch Blöcke und nicht einzelne slabs in den Einträgen. Zusätzlich ermöglicht der CROSS Tausch ein weiteres Abbruchkriterium. Ist die Zahl der Iterationen ohne Verbesserung der aktuellen besten Lösung gleich der Anzahl der initialen Lösungen, so bricht der Algorithmus ab, da alle Tauschmöglichkeiten bereits untersucht wurden, ohne eine Verbesserung zu finden.

Der Algorithmus:

1. K initiale Lösungen werden erstellt: Durch simple Heuristik werden m Versuchslösungen generiert. Die beste wird akzeptiert und die verwendeten slabs von der Liste der einsetzbaren gestrichen. Dies wird K mal wiederholt. Setze $k = 1$; *besteLösung* = k ; Iterationen ohne Verbesserung, $IoV = 0$.
2. Auf die K initialen Lösungen wird ein kurzer TS angewandt. Es werden nur fünf Iterationen eines einfachen TS durchgeführt.
3. Zwischen der k -ten Lösung und den $K - 1$ übrigen wird ein CROSS Austausch für jeden schlechten Block gemacht und die Kannibalen Liste angepasst.
4. Für die k -te Lösung wird der oben beschriebene einfache TS durchgeführt.
5. Verbessert sich die beste Lösung in 3. und 4., so setze $IoV = 0$ und *besteLösung* = k . Andernfalls setze $IoV = IoV + 1$.
6. Setze $k = k + 1$ und gehe zu Schritt 3.

Der Algorithmus bricht ab, wenn $IoV = K$. Ist das Abbruchkriterium für den einfachen TS in Schritt 4 erfüllt, so springt der Algorithmus zu Schritt 5.

Guided Variable Neighbourhood Search (G-VNS)

Ein neighbourhood search Verfahren sucht in der Umgebung der aktuellen Lösung nach Verbesserungen. Diese Methode ist abhängig von einer gut gewählten Anfangslösung und benötigt effiziente Suchalgorithmen, will man in annehmbarer Zeit ein akzeptables Ergebnis erhalten. Die Autoren Dong, Huang, Ip und Wang [2009] [3] verwenden in ihrer Arbeit die Verfahren Simulated Annealing und Harmony Search.

Bevor man das neighbourhood search Verfahren anwenden kann, muss man das zu Grunde liegende Problem entsprechend formulieren. Ich beginne mit der Notation:

n	Anzahl der slabs.
SG_i	Stahlqualität.
C_i^{code}	CC code.

$[WT_i^{min}; WT_i^{max}]$	Gewichtsbegrenzungen.
$[W_i^{min}; W_i^{max}]$	Breitenbegrenzungen.
W_i^{roll}	Rolling Breite.
T_i	Thickness.

Es wird unterschieden zwischen schweren und leichten Beschränkungen. Schwere Beschränkungen sind solche, die unbedingt eingehalten werden müssen, während leichte nur mit Strafen in der Zielfunktion bedacht werden.

Schwere Beschränkungen:

1. Die Kompatibilität der slabs in einem charge muss gegeben sein.
2. Das Gesamtgewicht eines charges darf nicht überschritten werden, z.B: 300 Tonnen.
3. Die thickness der slabs muss identisch sein.
4. Das Fenster für die Breite $[W_{ik}^{LB}; W_{ik}^{UB}]$ des charges k ist definiert als der Durchschnitt von allen Breitenbegrenzungen der slabs innerhalb des charges.

5. $W_i^{roll} \leq W_i^{min} \leq W_i^{max} \leq W_i^{roll} + \Delta^W$ mit Δ^W einer festgeschriebenen Größe, z.B: $\Delta^W = 200mm$.

Anforderungen, die nicht notwendigerweise erfüllt sein müssen (Leichte Beschränkungen):

1. Die Priorität der slabs gemäß dem just in time (JIT) Prinzip.
2. Möglichst viele slabs mit der selben Beschaffenheit in denselben charge gruppieren.
3. Das Breitenfenster ($W^{UB} - W^{LB}$) soll möglichst groß sein, die verlorene Breite ($W_{ik}^{LB} - W_i^{min}$) pro slab soll so klein wie möglich sein.
4. Die Stahlqualität sollte innerhalb eines charges so ähnlich wie möglich sein.

Das Problem wird nun wieder vereinfacht, indem man die Bestellung nach Stahlqualität und thickness gruppiert (vgl. Decomposition Algorithmus).

Ablauf des GVNS: Zuerst wird mittels next fit decreasing (NFD) nach einer Initiallösung gesucht, danach startet der variable neighbourhood search. In diesem Schritt wird dem Verfahren geholfen, indem es nicht von einer zufälligen Position aus starten muss.

Die Struktur des GVNSA und des GVNHS sieht nun folgendermaßen aus:

Initialisierung: Als erstes werden eine Nachbarschaftsstruktur sowie Abbruchkriterien definiert und mittels der Fitnessfunktion der Startwert berechnet.

Eine Lösung x ist ein Vektor mit ganzzahligen Einträgen, den charge Nummern. Sie erfüllt immer alle schweren Beschränkungen. Etwaige Verletzungen von leichten Beschränkungen verstecken sich im Zielfunktionswert, sind also nicht an der Zusammensetzung der charges ablesbar. Die Distanz zwischen zwei Lösungen wird über die Zahl der Komponenten definiert, in denen sie sich unterscheiden. Eine Nachbarschaft ist nun die Menge aller x' innerhalb eines gewissen Radius: $N_p(x) = \{x' : \rho(x, x') = p\}$, $p = 1, \dots, p_{max}$. $\rho(x, x')$ ist ein Maß dafür, in wie vielen Komponenten sich x und x' unterscheiden.

Der Algorithmus:

1. $p = 1$.
2. Bis $p = p_{max}$ werden folgende Schritte wiederholt:
 - Untersuchung der Nachbarschaft:* Der "erste beste Nachbar" $x' \in N_p(x)$ von x wird gesucht.
 - Lokale Suche:* Mittels Harmony Search (HS) oder Simulated Annealing (SA) sucht man ausgehend von x' nach einem lokalen Optimum x'' .
 - Annahme oder nicht:* Ist der Wert von x'' besser als jener von x , so übernimmt man das neue Optimum $x = x''$ und setzt $p = 1$. Andernfalls setze $p = p + 1$.

Angewandt auf das continuous casting Problem sieht dies nun folgendermaßen aus:

Ad Initiaillösung: Die slabs werden in absteigender Reihenfolge bezüglich WT_i^{min} gereiht. Als nächstes werden sie mittels "next fit decreasing" in charges gefüllt, wobei

$$\sum_{i=1}^n WT_i^{min} x_{ik} \leq WT \quad k = 1, \dots, m$$

die Gewichtsgrenze für charges beachtet werden muss. Die auf diese Weise entstandenen charges werden ganzzahlig nummeriert. Eine erste Lösung ist geboren.

Ad Untersuchung der Nachbarschaft: Auf dieser Stufe befinden sich in der Menge aller möglichen Nachbarn nur solche, die sich in genau p Stellen von der aktuellen Lösung unterscheiden. In allen charges werden nun die slabs Kosten minimierend geordnet. Jene Allokation mit den p teuersten slabs, im Sinne der Strafkosten, an vorderster Stelle wird als initialer bester Nachbar x' genommen.

Ad lokale Suche: Die Verfahren Harmony Search und Simulated Annealing suchen, ausgehend von der soeben generierten initialen Lösung, nach einem lokalen Optimum in $N_p(x')$. Es werden maximal t_{max} Iterationen pro x' ausgeführt. Wird bei einer solchen Iteration ein neues Optimum x''' gefunden, so setzt man $x = x'''$ und fährt mit $N_p(x), p = 1$ fort, andernfalls setzt man $t = t + 1$.

Als Abbruchkriterien werden p_{max} , bzw. innerhalb des HS oder SA t_{max} , verwendet. Die Fitnessfunktion ist

$$z' = \sum_{k=1}^m \sum_{i=1}^n c_{ik} x_{ik} + \sum_{k=1}^m c_k^{wt} | WT - \sum_{i=1}^n WT_i^{max} x_{ik} | .$$

Die Austauschregel: Es wird zwischen zwei verschiedenen Arten von Austausch unterschieden. Innerhalb einer Gruppe oder Gruppen übergreifend.

$x_{ik} \rightarrow x_{ik'}$: Slab i kommt in charge k' . k und k' können aus derselben Gruppe sein oder nicht. Welcher Fall eintritt entscheidet der Zufall. Es bleibt dem Anwender überlassen, hier mehrere slab-Wechsel durchzuführen oder nur einen.

Die folgenden zwei Heuristiken Simulated Annealing und Harmony Search sind vollwertige eigenständige Verfahren, um Probleme bestimmter Größe zu lösen. Sie sind hier innerhalb des GVNS gelistet, da auch die Autoren in ihrer Arbeit auf beide Verfahren zurückgreifen und sie, angewandt auf ein und dieselbe Problemformulierung leichter zu vergleichen sind.

Simulated Annealing (SA)

SA wurde ursprünglich entwickelt, um die Annäherung eines instabilen physikalischen Systems zu einem stabilen thermischen Gleichgewicht bei einer festen Temperatur zu simulieren. Das Verfahren kann verhindern, in einem lokalen Optimum hängenzubleiben. Es besteht aus einer äußeren Schleife, die die Temperatur schrittweise senkt, und einer inneren, die für jedes Temperaturniveau ein Optimum sucht.

Initialisierung: Das Verfahren startet mit dem in GVNS in “Untersuchung der Nachbarschaft” gefundenen x' . Weiters werden die Anfangstemperatur $T_{k^{out}}^{SA} = T_0^{SA}$ gewählt und der Zähler $k^{out} = 0$ gesetzt. T_f^{SA} ist eine untere Schranke für $T_{k^{out}}^{SA}$.

Die folgenden Schritte werden wiederholt bis $T_{k^{out}}^{SA} < T_f^{SA}$ gilt:

- Man bestimmt eine maximale Anzahl von Schritten $n^{SA}(T_{k^{out}}^{SA})$ und setzt den Zähler $n^{SA} = 0$.
- Bis $n^{SA} > n^{SA}(T_{k^{out}}^{SA})$ gilt wird gesucht:
 - Die Strafkosten der charges werden nach absteigendem Wert geordnet. Mit den Austauschregeln wird die Lösung x' adaptiert, $n^{SA} = n^{SA} + 1$ gesetzt und $\Delta z' = z'(x) - z'(x')$ berechnet.
 - Ist $\Delta z' < 0$ setze $x = x'$, andernfalls wird ein $\xi \in U(0, 1)$ erzeugt und wenn $e^{\Delta z' / T_{k^{out}}^{SA}} > \xi$ setze ebenfalls $x = x'$. Dieser zweite Teil, der eine schlechtere Lösung mit einer zufälligen Wahrscheinlichkeit übernimmt, führt das Verfahren aus einem lokalen Optimum heraus.

- Die Temperatur wird gesenkt: $T_{k^{out}}^{SA} = T_{k^{out}-1}^{SA} * r$, mit der Abkühlungsrate r , meistens gilt $r \in [0.95, 0.99]$. Erhöhe den Zähler $k^{out} = k^{out} + 1$ und kehre zum ersten Schritt zurück.

Das Verfahren liefert jene Lösung, die nach Eintritt eines Abbruchkriteriums aktuell ist. Diese Abbruchkriterien sind in der Regel eine festgesetzte Zahl an Iterationen und eine maximale Schrittzahl ohne Verbesserung der Lösung.

Harmony Search (HS)

Mit HS simuliert man den Versuch von Musikern durch Improvisation eine perfekte Harmonie zu erzeugen. Im Optimierungsprozess wird ein bezüglich einer Zielfunktion bester Vektor gesucht. HS kann sowohl im diskreten als auch im kontinuierlichen Fall angewandt werden. Der schrittweise Verlauf des Verfahrens:

- Initialisierung eines Harmony Memory (HM). Dieses ist ein Vektor, der die Zusammenstellung mehrerer Lösungen und deren Zielfunktionswerte speichert.
- Eine neue Harmony wird generiert.
- Überprüfung, ob die neue Harmony besser ist als der schlechteste Eintrag im bisherigen HM. Falls ja, wird dieser Eintrag gelöscht und die generierte Harmony in das HM aufgenommen.
- Es wird so lange eine neue Harmony generiert, bis ein Abbruchkriterium erfüllt ist.

Zur Initialisierung werden die Größe des Harmony Memory sowie die Harmony Memory considering rate $p(HMCR) \in [0, 1]$ und die pitch adjusting rate $p(PAR) \in [0, 1]$ initialisiert. $p(HMCR) = 0.8$ bedeutet, dass der Algorithmus im nächsten Schritt mit einer Wahrscheinlichkeit von 80% eine Variable aus dem HM wählt. Bei $p(PAR) = 0.1$ wird mit 10% Wahrscheinlichkeit ein Wert aus der Nachbarschaft genommen.

Angewandt auf das continuous casting Problem bedeutet dies: Die Lösungen x sind Harmonies. Die initiale Harmony für das Verfahren ist das im Schritt "Untersuchung der Nachbarschaft" ermittelte x' . Mit anderen $x'' \in N_p(x)$ wird das HM aufgefüllt. Ist dies nicht möglich, so müssen an

den leeren Stellen hohe Strafkosten eingefügt werden, da Einträge, die keinen Wert aufweisen, vom Algorithmus automatisch als optimal angesehen werden und nie aus dem HM ausgetauscht würden.

Für jeden charge in x' werden die Strafkosten berechnet und in absteigender Reihenfolge für einen charge ein $\delta^{HMCR} \in U(0,1)$ und ein $\delta^{PAR} \in U(0,1)$ erzeugt. Ist $\delta^{HMCR} < p(HCMR)$, so wird ein charge einer Harmony aus dem aktuellen HM ausgewählt, andernfalls einer aus dem Lösungsraum. Ist $\delta^{PAR} < p(PAR)$, wird ein charge von einem Nachbarn der Harmony gewählt, sonst ein zufälliger. Auf diese zwei charges wird die Austauschregel angewandt. Dieser Schritt wird so oft wiederholt, bis eine bessere Harmony gefunden wird oder das Schrittemaximum erreicht ist.

Ant Colony Optimization (ACO)

Wie schon der Name suggeriert, ist dieses von Ferretti, Zanoni und Zanovella [2006] [4] vorgestellte Verfahren dem Arbeitsverhalten der Ameisen nachgestellt. Diese produzieren ein Pheromon, das sie auf dem Weg, auf dem sie krabbeln, hinterlassen. Je mehr Pheromone auf dem Weg sind, desto attraktiver ist er. Wird der Weg zur Futterstelle blockiert, so entscheidet sich die Ameise zufällig, ob sie links oder rechts geht. Jene Ameise, die schneller an das Futter kommt, ist schneller wieder zurück und kann den Pfad ein weiteres Mal benutzen. So lagert sie auf dem kürzeren Weg mehr Pheromone ab, wodurch dieser attraktiver wird und ihn mehr Ameisen benutzen.

Das ACO Verfahren ist eine Form des Travelling Salesman Problem (TSP).

Notation:

n	Städte.
d_{ij}	Länge des Weges von i nach j .
τ_{ij}	Intensität der Pheromone.
η_{ij}	Erreichbarkeit einer Stadt.
α, β	Gewichte.
$tabu_k$	Vektor der bereits besuchten Städte.

Die k -te Ameise wählt ihren Weg mit der Wahrscheinlichkeit

$$p_{ij}^k(t) = \begin{cases} \sum_{\omega \notin tabu_k} \frac{\tau_{ij}(t)^\alpha (\eta_{ij})^\beta}{\tau_{i\omega}(t)^\alpha (\eta_{i\omega})^\beta} & j \notin tabu_k \\ 0 & j \in tabu_k \end{cases}$$

Eine Tour dauert so lange, bis jede Ameise jede Stadt einmal besucht hat. Am Ende einer Tour wird die Pheromonintensität aktualisiert: $\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij}$. ρ ist ein Parameter für die Vergänglichkeit der Pheromonintensität. $\Delta\tau_{ij}$ sind die in der vergangenen Tour dazugekommenen Pheromone.

Angewandt auf das Problem des continuous casting entsprechen die Städte den Bestellungen und die Länge der Pfeile der Zeit, die benötigt wird, um zwischen verschiedenen Produktionsstellen zu wechseln.

In kleinen TSPs erzielt ACO mitunter sogar bessere Ergebnisse als andere heuristische Verfahren, es ist daher vorzugsweise auf kleinere Werke anzuwenden.

Auction based Approach

Die Autoren Kumar V, Kumar S und Tiwari [2006] [5] formulieren das Problem der Reihung der slabs bzw. charges in diesem Verfahren als lineares ganzzahliges Problem. Die Reihenfolge der charges wird über eine Auktion ermittelt. Das Gebot mit der höchsten Auszahlung wird angenommen und implementiert. Ein Gebot entspricht einer bestimmten Reihenfolge der charges. Ausgewertet werden die Gebote durch ein heuristisches Verfahren.

Die hierbei angewandte Auktion ist eine kombinatorische. Das heißt, es wird nicht ein Gut nach dem anderen verkauft, sondern es können Gebote für Kombinationen abgegeben werden.

Zwei wichtige Konzepte: *Komplementarität*: Der Wert für die Summe zweier Güter ist höher als der kumulierte Wert der einzelnen. $V(\{a, b\}) > V(\{a\}) + V(\{b\})$.

Substitutierbarkeit: Die Summe der Werte der einzelnen Güter ist größer als der Wert des Güterbündels. $V(\{a, b\}) < V(\{a\}) + V(\{b\})$.

In einer kombinatorischen Auktion werden alle Gebote angenommen und ausgewertet. Wie schon erwähnt, ist es möglich, Gebote für Kombinationen von Gütern abzugeben, was für den Auktionär von Vorteil sein kann. Er kann sich unter allen Geboten jene Mischung herausuchen, die für ihn den größten Profit liefert.

Die Kriterien für die Profitmaximierung sind die Minimierung der Wartezeit, das ist im Arbeitsprozess verlorene Zeit, und die Maximierung des outputs gemessen in produzierten slabs.

Als Nebenbedingungen müssen eine nicht zu überschreitende Wartedauer zwischen zwei Verarbeitungsschritten, die kontinuierliche Benutzung der

Maschinen und eine eventuell benötigte Aufbereitungszeit für bestimmte Maschinen beachtet werden.

Der schrittweise Ablauf des Verfahrens ist nun folgender:

1. In einer Tabelle werden charge Nummer, benötigte Maschinen und die nötige Zeit festgehalten.
2. Alle möglichen Kombinationen von zwei charges werden ausgewertet.
3. Die beste Kombination wird als eine Gruppe festgelegt. Gibt es mehrere Kandidaten, so wählt man jene, die die kleinere charge Nummer beinhaltet.
4. Die Gruppe wird mit den noch verbliebenen charges kombiniert und der charge, der den größten Nutzensgewinn bringt, wird in die Gruppe aufgenommen.
5. Die Schritte drei und vier werden so lange wiederholt, bis kein kombinierbarer charge mehr übrig ist.
6. Die benötigte Wartezeit und der Output werden berechnet.

Alternative Graph + Beam Search

Im Alternative Graph Verfahren von Pacciarelli und Pranzo [2004] [6] wird der Herstellungsprozess eines Stücks Stahl in die einzelnen Arbeitsschritte zerlegt. Jede Arbeit an oder von einer Maschine stellt eine Operation dar. Das Ziel ist die Minimierung des Startzeitpunkts der letzten Operation. Für Wartezeiten vor den Engpässen in der Produktion gibt es Puffer. Diese werden als eine Maschine ohne Prozessdauer modelliert.

Das Triple (N, F, A) wird als Alternative Graph bezeichnet. N ist die Menge der Knoten (Maschinen) im Netzwerk, F ist die Menge aller Pfeile von einem Knoten zum nächsten, wobei die Pfeillänge die Operationsdauer angibt. A ist die Menge aller alternativen Pfeile, das sind jene, die nicht gleichzeitig ausgeführt werden können, zum Beispiel weil dieselbe Maschine benötigt wird.

Es wird nun nach einer Auswahl $S \subseteq A$ gesucht, sodass der Graph $(N, F \cup S)$ keine alternativen Pfeile und keine Zyklen (ein geschlossener Kreis von Pfeilen) von positiver Länge enthält. Ein Zyklus mit positiver Länge $\Delta > 0$ würde bedeuten, dass eine Operation zum Zeitpunkt $t + \Delta$, also strikt nach ihrer eigentlichen Beginnzeit startet. S' heißt eine Erweiterung von S , wenn

gilt $S \subseteq S'$. Eine sogenannte komplette Erweiterung bekommt man, wenn man von allen ungewählten Tupeln alternativer Pfeile einen auswählt. Nur ein kompletter Graph ist ein Lösungskandidat.

Zwei Propositionen erklären, wie man die Menge der zu beachtenden Graphen verkleinern kann. $l(i, j)$ ist definiert als der längste Weg von i nach j .

Proposition 1 (Entfernung eines Knotens): Sei $G = (N, F, A)$ ein alternativer Graph und $i \in N$ ein Knoten, der mit keinem alternativen Pfeil verbunden ist. Sei F_i^- die Menge aller Pfeile, die mit dem Knoten i verbunden sind und $F_i^+ = \{(h, j) | (h, i) \in F, (i, j) \in F\}$. Die Länge des Pfeiles (h, j) sei gleich der Summe der Länge der Pfeile (h, i) und (i, j) , $d_{hj} = d_{hi} + d_{ij}$. Dann ist der alternative Graph $G' = (N', F', A')$ mit $N' = N \setminus \{i\}$, $F' = F \setminus \{F_i^- \cup F_i^+\}$ äquivalent zum Graph G .

Proposition 2 (Entfernung eines Pfeiles): Gegeben sei ein Pfeil $(i, j) \in F$. Existiert innerhalb von F ein Weg von i nach j , dessen Länge $l(i, j)$ größer oder gleich der Pfeillänge d_{ij} ist, so kann man den Pfeil (i, j) aus dem Graphen entfernen. Der entstehende Graph G' ist wieder äquivalent zum ursprünglichen Graphen G .

Weitere Propositionen engen den Suchhorizont ein, indem sie zum Beispiel gewisse Pfeile verbieten oder deren Gebrauch vorschreiben. Der Rechenaufwand des Computers kann auf diese Weise drastisch reduziert werden. $l(0, i)$ heißt der Kopf von Knoten i , $l(i, n)$ ist der Schwanz.

Proposition: Gegeben seien eine Auswahl S und die ungewählten alternativen Pfeile $((i, j), (h, k))$. Wenn $l(h, k) + d_{hk} > 0$, so wird der Pfeil (h, k) verboten und (i, j) von S impliziert.

Proposition: Gegeben seien eine Auswahl S , eine Schranke UB und die alternativen Pfeile $((i, j), (h, k))$. Ist $l(0, h) + d_{hk} + l(k, n) > UB$, so wird der Pfeil (h, k) verboten und (i, j) von S impliziert.

Auf die Menge aller Auswahlen wird nun das Beam Search Verfahren angewandt. Dieses ist ein Artificial Intelligence Verfahren, das erstmals von Fox [1983] auf ein scheduling Problem angewandt wurde. Beam Search ist ein abgeschnittenes Branch and Bound (B&B), in dem nur Zweige weiterverfolgt werden, deren Wert eine vorgegebene Schranke β übersteigt. Dieses β nennt man die *beam width*. Ein höheres β führt zu einem besseren Ergebnis, allerdings auch zu längerer Arbeitszeit des Computers.

3.1.1.2 MIP/LP

Column Generation (CG)

Das Hauptaugenmerk meiner Arbeit liegt auf dem von Chang SY, Chang MR und Hong [2000] [7] beschriebenen Verfahren. Bei diesem Lösungsansatz wird anfangs ein sehr großes, ganzzahliges Problem aufgestellt, das danach schrittweise vereinfacht wird. Dabei kommt auch das duale Problem zur Anwendung.

Angenommen, es gebe m verschiedene Typen von charges, die sich in thickness, width, carbon und manganese Gehalt unterscheiden.

z_j, y_{ij} sind binäre Hilfsvariablen. y_{ij} gibt an, ob ein charge vom Typ i in cast j ist und z_j ist eins, wenn cast j nicht leer ist. t_i steht für die benötigte Zeit für charge i , T bezeichnet das vorhandene Zeitlimit pro cast. N_i ist die Zahl der charges vom Typ i , J die Gesamtzahl der vorhandenen charges. Chang et al. stellen zu allererst das ganzzahlige Optimierungsproblem auf (Model-IP)

$$\min \sum_{j=1}^J z_j$$

s.t:

$$\begin{aligned} \sum_{j=1}^J y_{ij} &= N_i & i = 1, \dots, m \\ \sum_{i=1}^m t_i y_{ij} &\leq T z_j & j = 1, \dots, J \\ & \vdots \end{aligned}$$

sowie den entsprechenden Nebenbedingungen für width, carbon und manganese Gehalt. Da nicht die explizite Darstellung, sondern nur die Existenz der restlichen Nebenbedingungen für den Verlauf des Verfahrens von Bedeutung ist, werden diese nur erwähnt, nicht aber angeschrieben.

Angenommen, die Menge aller möglichen Kombinationen von charges wird nummeriert und mit $k = 1, \dots, K$ bezeichnet. Der Spaltenvektor $a_k = (a_{1k}, \dots, a_{mk})'$ stellt eine solche Kombination dar. In a_{ik} steht, wie viele charges vom Typ i in der k -ten Kombination gecastet werden. Mit dieser Schreibweise lässt sich das Problem umformulieren zu (Model-LP)

$$\min \sum_{k=1}^K x_k$$

s.t:

$$\begin{aligned} \sum_{k=1}^K a_{ik}x_k &= N_i & i = 1, \dots, m \\ x_k &\geq 0, \text{ ganzzahlig} & k = 1, \dots, K \\ & & \vdots \end{aligned}$$

x_k ist die Anzahl der verwendeten k -ten Lösungskombinationen. Dieses Modell ist analog zum “trimming problem”. Noch immer ist das Problem zu groß, um vollständig gelöst werden zu können, also wird es relaxiert. In den Arbeiten über das trimming problem von Eisemann [1957], Gilmore und Gomery [1961] und Murty [1983] wurde gezeigt, dass die gerundete Lösung des relaxierten Problems ein zufrieden stellendes Ergebnis für das ganzzahlige Problem liefert. Das übrig gebliebene Modell ist jenes von oben, es fällt nur die Beschränkung “ganzzahlig” aus der zweiten Nebenbedingung hinaus. Weiterhin ist es zu groß, um exakt gelöst zu werden, deshalb wenden Chang et al. den column generation Ansatz an.

Angenommen es existiert eine zulässige Lösung mit der Basis B zu Modell-LP, dann definiere $\pi = (\pi_1, \dots, \pi_m) = c_B B^{-1}$ als die entsprechende duale Lösung. c_B sind die Koeffizienten der Zielfunktion bezüglich B . In dem gegebenen Fall sind sie alle eins. Die Lösung ist optimal wenn gilt:

$$\pi a_k \leq 1 \quad k = 1, \dots, K$$

Diese Überprüfung nennt man *pricing* oder *column generation* und um sie durchzuführen, stellt man das duale Problem auf. s_i bezeichnet die Anzahl der charges vom Typ i (Modell- CG_π).

$$\max \sum_{i=1}^m \pi_i s_i$$

s.t:

$$\begin{aligned} \sum_{i=1}^m t_i s_i &\leq T & i = 1, \dots, m \\ s_i &\geq 0, \text{ ganzzahlig} & i = 1, \dots, m \\ & & \vdots \end{aligned}$$

Wenn der Zielfunktionswert für dieses duale Modell nicht größer ist als eins, ist die Basis B für Model-LP optimal.

Nun definiert man Teilmengen S_{pqr} von $M = \{1, \dots, m\}$ als Gruppen von charges, die sich in width, carbon und manganese Gehalt nicht um mehr als d_w, d_c bzw. d_g unterscheiden. Durch diese Definition werden sämtliche oben weggelassenen Nebenbedingungen automatisch eingehalten. Für $p, q, r \in M$ sieht die Menge so aus: $S_{pqr} = \{i : i \in M, w_p \leq w_i \leq w_p + d_w, c_q \leq c_i \leq c_q + d_c, g_r \leq g_i \leq g_r + d_g\}$. Von den entstandenen Teilmengen werden alle leeren und redundanten entfernt und die restlichen mit $P_l, l = 1, \dots, L$ bezeichnet. Löst man oben stehendes Model- CG_π für P_l , also

$$\max \sum_{i \in P_l} \pi_i s_i$$

s.t:

$$\begin{aligned} \sum_{i \in P_l} t_i s_i &\leq T \\ s_i &\leq N_i & i \in P_l \\ s_i &\geq 0, \text{ ganzzahlig} & i \in P_l \end{aligned}$$

so ist ein gefundenes Optimum ebenfalls für Model- CG_π optimal. Für das letzte Modell gibt es einen effizienten, pseudo polynomischen Algorithmus. Man vergleicht L -viele Ergebnisse und wählt jene Kombination, die den höchsten Zielfunktionswert liefert.

Von der optimalen Lösung des Model- CG_π kommt man leicht auf jene des Model-LP. Die Lösung des Model-LP muss man nun noch runden, um eine zufriedenstellende Lösung von Model-IP zu erhalten.

Decomposition Strategie

Anstatt ein riesiges, unlösbares MILP aufzustellen, zerlegen Harjunkoski

und Grossmann [2001] [8] das Problem in diesem Verfahren in kleinere, exakt lösbare Modelle. Dabei werden zum Teil komplizierte Nebenbedingungen umgangen.

Vorsortierung: Die Bestellungen werden sortiert nach casting Dauer, Fertigstellungszeitpunkt, slab width, Qualität und thickness. Innerhalb vorzugebender Toleranzgrenzen werden die slabs in Gruppen aufgeteilt. Das wichtigste Merkmal ist die thickness.

Gruppenbildung: In dieser Phase wird die Anzahl der Gruppen minimiert und die casting Folge maximiert. Beachtet werden muss dabei, dass in einer Gruppe nicht mehr als M_{max} Jobs sein dürfen, sowie dass Qualität, width und thickness innerhalb der Gruppe und bei aufeinander folgenden Gruppen kompatibel sind.

Ordnung innerhalb einer Gruppe: Hier wird speziell auf die Fertigungsdauer und die Betriebszeit der Maschinen in den einzelnen Arbeitsgängen geachtet. Es werden Wartezeiten, während denen der Stahl abkühlt, minimiert. Bei zwei parallelen Electric Arc Furnaces (EAF) kann das Problem in diesem Schritt sehr groß werden. In der Praxis behilft man sich, indem die Belegung der EAFs a priori fixiert wird.

Ordnung aller Gruppen: Die Konzentration liegt nicht mehr auf den einzelnen Bestellungen, sondern es werden nur noch die Gruppen als Blöcke nacheinander gereiht. Es werden die Gesamtarbeitsdauer und die Startzeitpunkte an den einzelnen Maschinen minimiert. Die wichtigsten Nebenbedingungen, außer width und der Aufnahme chemischer Bestandteile aus dem Verarbeitungsmaterial, sind die Beachtung spezieller Qualitätsanforderungen und die Minimierung von thickness Differenzen im Caster.

LP Verbesserung: Die Reihung und Zusammenstellung der Gruppen ist fixiert. Man kann nun den letzten Schritt wiederholen, muss jetzt jedoch nur noch auf die Beschränkungen eingehen, die an aufeinander folgende Bestellungen gestellt werden. Hierdurch wird erreicht, dass eventuelle unnötige Wartezeiten verkürzt werden. In der Praxis bringt dieser Schritt eine Verbesserung um 5-15 %.

Lagrange Relaxation

Zu allererst eine kurze Beschreibung des von Tang, Luh, Liu und Fang [2002] [9] aufgestellten Problems. Die Aufgabe besteht darin, die Kontinuität der Produktion, insbesondere des castings zu erreichen. Aus diesem Grund ergeben sich für die Zielfunktion folgende Strafkosten:

- Unterbrechung des castings.
- Temperaturabfall des Stahls, der zur weiteren Verarbeitung wieder aufgewärmt werden muss.
- Zeitliche Verzögerungen bei der Fertigung. Zu früh produzierte Ware muss gelagert werden, zu spät fertiggestellte muss mit Preisabschlägen verkauft werden.

Die Nebenbedingungen umfassen Stahlqualität, width, width Sprünge innerhalb eines casts, Lieferungszeitpunkte und die Anzahl der charges in einem cast. Die Aufbereitungszeit vor und zwischen zwei casts wird separat vom Optimierungsprozess berücksichtigt. Die Autoren unterteilen die Fertigung in drei Stufen: Stahlherstellung, refining und continuous casting.

Notation:

i	Index der charges. $i \in \{1, \dots, N\}$.
j	Index der Stufen eins bis drei.
g	Index der casts. $g \in \{1, \dots, M\}$.
k	Index der Zeitpunkte. $k \in \{1, \dots, K\}$.
Ω	Menge aller charges.
Ω_g	Menge aller charges in cast g .
s_{pg}	Der p -te charge in cast g .
d_i	Lieferzeitpunkt von charge i .
$C1_g$	Strafkosten für casting Unterbrechungen.
$C2_{ij}$	Strafkosten für die Wartezeit nach der Fertigstellung von charge i auf Stufe j .
$C3_i$	Strafkosten für zu frühe Fertigstellung.
$C4_i$	Strafkosten für zu späte Fertigstellung.
T_{ij}	Bearbeitungsdauer für charge i auf Stufe j .
$t_{j,j+1}$	Transportdauer von Stufe j zu Stufe $j + 1$.
S_{ij}	Set-up Zeit für charge i auf Stufe j . $S_{ij} = 0$ für alle $(i, j) \neq (\text{erster charge in cast}, 3)$.
R_{ij}	Entfernungszeit für charge i nach Bearbeitung auf Stufe j . $R_{ij} = 0$ für alle $(i, j) \neq (\text{letzter charge in einem cast}, 3)$.

- M_{jk} Zahl der verfügbaren Maschinen auf Stufe j zum Zeitpunkt k .
 K Zahl der verfügbaren Zeiteinheiten.

Entscheidungsvariablen

$$\delta_{ijk} = \begin{cases} 1 & \text{wenn charge } i \text{ auf Stufe } j \text{ zum Zeitpunkt } k \text{ bearbeitet wird} \\ 0 & \text{sonst} \end{cases}$$

$$i \in \Omega; j = 1, 2, 3; k = 1, \dots, K$$

$$C_{ij} \quad \text{Fertigungszeitpunkt von charge } i \text{ auf Stufe } j.$$

Das Modell

$$\begin{aligned} Z = \min & \sum_{g=1}^M \sum_{p=1}^{|\Omega_g|-1} C1_g (C_{s_{g,p+1},3} - T_{s_{g,p+1},3} - C_{s_{gp},3}) \\ & + \sum_{i=1}^N \sum_{j=1}^2 C2_{ij} (C_{i,j+1} - T_{i,j+1} - t_{j,j+1}) \\ & + \sum_{i=1}^N C3_i \max \{0, d_i - C_{i3}\} + \sum_{i=1}^N C4_i \max \{0, C_{i3} - d_i\} \end{aligned}$$

s.t:

$$C_{ij} + t_{j,j+1} \leq C_{i,j+1} - T_{i,j+1} \quad i \in \Omega; j = 1, 2 \quad (30)$$

$$C_{s_{gp},3} \leq C_{s_{g,p+1},3} - T_{s_{g,p+1},3} \quad p = 1, 2, \dots, |\Omega_g| - 1; g \in M \quad (31)$$

$$\sum_{k=1}^K \delta_{ijk} = T_{ij} + S_{ij} + R_{ij} \quad i \in \Omega; j = 1, 2, 3 \quad (32)$$

$$k\delta_{ijk} \leq C_{ij} + R_{ij} \quad i \in \Omega; j = 1, 2, 3; k = 1, \dots, K \quad (33)$$

$$C_{ij} - T_{ij} - S_{ij} + 1 \leq k + K(1 - \delta_{ijk}) \quad i \in \Omega; j = 1, 2, 3; k = 1, \dots, K \quad (34)$$

$$\sum_{i \in \Omega} \delta_{ijk} \leq M_{jk} \quad j = 1, 2, 3; k = 1, \dots, K \quad (35)$$

$$\delta_{ijk} \in \{0, 1\} \quad i \in \Omega; j = 1, 2, 3; k = 1, \dots, K \quad (36)$$

$$C_{ij} \in \{1, \dots, K\} \quad i \in \Omega; j = 1, 2, 3 \quad (37)$$

Beschränkung (30) stellt sicher, dass die charges nicht auf zwei Stufen gleichzeitig bearbeitet werden können. (31) verhindert, dass zwei charges innerhalb eines casts zeitgleich gecastet werden. Gleichung (32) beschreibt die

gesamte benötigte Zeit inklusive set-up und Entfernungszeit für jede Maschine auf jeder Stufe. In (33) und (34) steht, wie lange eine Maschine auf einer Stufe in Anspruch genommen wird. Gleichung (35) behandelt die Kapazität der Maschinen.

Nur die Gleichungen (31) und (35) befassen sich mit zwei charges gleichzeitig. Relaxiert man diese, so kann man das Problem in Subprobleme aufteilen, in denen jeweils nur ein charge vorkommt. Für das Modell heißt das, dass in den Bedingungen (30), (32), (33), (34), (36), (37) und der Zielfunktion $i \in \Omega$ fest ist. Die dafür nötige Relaxierung erreicht man, indem man die Nebenbedingungen (31) und (35) mit nicht negativen Lagrange Multiplikatoren $u_{s_{gp}}, v_{jk}$ in die Zielfunktion schreibt. Die Zielfunktion Z wird um die Terme

$$\begin{aligned} & \sum_{g=1}^M \sum_{p=1}^{|\Omega_g|-1} u_{s_{gp}} (C_{s_{gp},3} - C_{s_{g,p+1},3} + T_{s_{g,p+1},3}) \\ & + \sum_{k=1}^K \sum_{j=1}^3 v_{jk} (\sum_{i \in \Omega} \delta_{ijk} - M_{jk}) \end{aligned}$$

erweitert zu Z_{LR} . Die Nebenbedingungen (31) und (35) fallen weg, dafür kommen die Beschränkungen

$$u_{s_{gp}} \geq 0 \quad p = 1, \dots, |\Omega_g| - 1; g = 1, \dots, M \quad (38)$$

$$v_{jk} \geq 0 \quad j = 1, 2, 3; k = 1, \dots, K \quad (39)$$

neu hinzu.

Die Subprobleme können durch dynamische Programmierung mit Rückwärtsinduktion gelöst werden. Angefangen bei der letzten Stufe, dem continuous casting, optimiert die DP Schritt für Schritt jede Stufe bis zur ersten, der Stahlherstellung. Da das Problem relaxiert wurde, erfüllen die gefundenen Lösungen meist die Bedingungen (31) und (35) nicht. Dies wird nun in zwei Phasen repariert.

In der ersten Phase wird die Reihenfolge der charges innerhalb eines casts durch das Paar (g, p) (p -te Position in cast g) definiert. In der zweiten Phase wird auf die Maschinenkapazität eingegangen.

Erste Phase:

1. Setze $g = 1$.
2. Für cast g überprüfe, ob (31) erfüllt ist. Wenn ja, dann gehe zu Schritt 5.

3. Sei $B_{s_g 13}^n = \min_{j \in \Omega_g} \{B_{i3}^n\}$; $B_{s_{g+1} 3}^n = B_{s_g, p-1 3}^n + T_{s_g, p-1 3}$, $p = 2, \dots, |\Omega_g|$. In diesem Schritt wird der früheste Startzeitpunkt für einen charge innerhalb des gegebenen casts gesucht und für alle folgenden charges deren Startzeitpunkte definiert. Es wird hier nichts an dem Beginnzeitpunkt des casts geändert, aber sichergestellt, dass die Reihenfolge der charges gemäß (31) übereinstimmt.
4. Basierend auf dem Startzeitpunkt für die letzte Stufe B_{i3}^n , $i \in \Omega_g$ können die Beginnzeitpunkte für die charges auf den Stufen eins und zwei festgelegt werden. Dabei wird noch nicht auf die Auslastung der Maschinen geachtet.
5. Setze $g = g + 1$. Solange $g \leq M$ gehe zu Schritt 2. Anderfalls gehe zu Phase zwei.

Zweite Phase:

1. Setze $j = 1$, $M1_{kj} = M_{jk}$ für $k = 1, \dots, K$.
2. Die charges werden gemäß ihren Startzeitpunkten indiziert und geordnet. Für i von 1 bis N : Sei $B_{[i]j}^n = \min_{i1 \in \pi_j} \{B_{i1,j}^n\}$. Suche $k = \min \{k' | M1_{k'j} \neq 0 \wedge k' \geq B_{[i]j}^n\}$. Charge $[i]$ beginnt nun auf Stufe j sofort nach Zeitpunkt k , $B_{[i]j}^n = k$. Setze $M1_{k1,j} = M1_{k1,j} - 1$ für $k1 = k + 1, \dots, k + T_{[i]j}$; $\pi_j = \pi_j - \{[i]\}$.
3. Für die folgenden Stufen werden die Startzeitpunkte aktualisiert. Setze $j = j + 1$.
Wenn $j < 3$ erneuere $B_{ij}^n = \max \{B_{i,j-1}^n + T_{i,j-1} + t_{j-1,j}, B_{ij}^n\}$ für $i \in \Omega$ und gehe zu Schritt 2.
4. Aktualisiere die Startzeitpunkte der ersten charges in allen casts auf der letzten Stufe. Dabei wird darauf geachtet, dass die verschwendete Zeit am caster, jene zwischen zwei charges innerhalb eines casts, möglichst gering ist. Sei $B1_{ij} = \max \{B_{i,j-1}^n + T_{i,j-1} + t_{j-1,j}, B_{ij}^n\}$ für $i \in \Omega$ und $B_{s_{g+1} j} = B_{s_g j} + \sum_{i1 \in \Omega_g} \max \{B1_{i1,j} - B_{i1,j}^n, 0\}$, $g = 1, \dots, M$.
5. Analog zu Schritt 2 werden nun die casts nach ihrem Anfangszeitpunkt indiziert und geordnet. Für g von 1 bis M berechne $B_{s_{[g]1} j} = \min_{g'1 \in \pi_j} \{S_{s_{g'1} j} + B_{s_{g'1} j} 0\}$, $k = \min \{k' | M1_{k'j} \neq 0 \wedge k' \geq B_{s_{[g]1} j}\}$.

Der erste charge in cast $[g]$ soll auf Stufe j sofort nach Zeitpunkt k starten und alle anderen charges in diesem cast entsprechend danach: $B_{s_{[g]1}j} = k, B_{s_{[g]p}j} = B_{s_{[g],p-1}j} + T_{s_{[g],p-1}j}, p = 2, \dots, |\Omega_{[g]}|$. Setze $M1_{k1,j} = M1_{k1,j} - 1$ für $k1 = k + 1, \dots, k + S_{[g]1} + \sum_{i \in \Omega_{[g]}} T_{ij} + R_{[g]|\Omega_{[g]}}; \pi_j = \pi_j - \Omega_{[g]}$.

6. $\{B_{ij}^n\}$ beschreibt eine gültige Lösung.

Die Lagrange Multiplikatoren werden in jeder Iteration nachadjustiert. Ihre optimalen Werte sind die Lösung des dualen Problems zum relaxierten Problem: $\max Z_D(u_i, v_{jk}) \equiv \min Z_{LR}$ unter den Nebenbedingungen (38) und (39).

Das gesamte Verfahren verläuft wie folgt:

1. Initialisierung: $n = 0; \alpha_n = 2; Z^U = +\infty, Z^L = -\infty; u_{s_{gp}}^n = 0, v_{jk}^n = 0$.
2. Löse das relaxierte Problem. Falls $Z_{D(u^n, v^n)} > Z^L$ so setze $Z^L = Z_{D(u^n, v^n)}$.
3. Erzeuge mittels den oben beschriebenen Phasen eine gültige Lösung. Falls $Z^n < Z^U$, setze $Z^U = Z^n$.
4. Überprüfe, ob Abbruchkriterien erfüllt sind. Diese richten sich nach der Zahl der Iterationen, dem Abstand der Schranken Z^U, Z^L und dem Schrittgrößenparameter t^n .
5. Aktualisiere die Lagrange Multiplikatoren.

$$\begin{aligned} u_{s_{gp}}^{n+1} &= \max \{0, u_{s_{gp}}^n + t^n \gamma^n(u_{s_{gp}}^n)\} & p = 1, \dots, |\Omega_g| - 1; g = 1, \dots, M \\ v_{jk}^{n+1} &= \max \{0, v_{jk}^n + t^n \gamma^n(v_{jk}^n)\} & j = 1, 2, 3; k = 1, \dots, K \end{aligned}$$

Wobei $\gamma^n(u_{s_{gp}}^n) = C_{s_{gp},3}^n - C_{s_{g,p+1},3}^n + T_{s_{g,p+1},3}^n$ für $p = 1, 2, \dots, |\Omega_g| - 1; g = 1, \dots, M$ und $\gamma^n(v_{jk}^n) = \sum_{i \in \Omega} \delta_{ijk} - M_{jk}; j = 1, 2, 3; k = 1, \dots, K$. Setze $n = n + 1$ und gehe zu Schritt 2.

MIP auf LP gebracht

Das Modell ist dem vorhergehenden sehr ähnlich, in der Zielfunktion werden Strafkosten für casting Unterbrechungen, Temperaturabfall sowie zeitliche Verfehlungen bei der Fertigung berechnet.

Tang, Luh, Liu und Fang [2002] [9] haben eine Relaxierung angewandt, um das Problem zu vereinfachen, Tang, Liu, Rong und Yang [2000] [10] erreichen dies, indem sie es linearisieren. Im Unterschied zur Zielfunktion von Tang, Luh, Liu und Fang [2002] tauchen die Kosten $C3$ und $C4$ bei Tang, Liu, Rong und Yang [2000] jedoch in einer Doppelsumme auf. Dafür sind die Nebenbedingungen so formuliert, dass in keiner zwei charges gleichzeitig auftreten.

Notation:

$SI(i, j)$ Nachfolgender charge von charge i auf Maschine j .

$SP(i, j)$ Nachfolgende Maschine von charge i auf Maschine j .

Π Menge aller Maschinen.

Φ Menge aller casting Maschinen $\Phi \subseteq \Pi$.

Die übrigen Variablen sind analog zu Tang, Luh, Liu und Fang [2002].

Das Modell:

$$\begin{aligned}
Z = & \sum_{k=1}^M \sum_{i \in \Omega_k, j \in \Pi_j \cap \Phi, SI(i,j) \in \Omega_k} C1_k (X_{SI(i,j),j} - X_{i,j} - T_{i,j}) \\
& + \sum_{i=1}^N \sum_{\substack{j \in \Pi_i \\ SP(i,j) \in \Pi_i}} C2_{ij} (X_{i,SP(i,j)} - X_{i,j} - T_{i,j} - t_{j,SP(i,j)}) \\
& - \sum_{i=1}^N \sum_{j \in \Pi_i \cap \Phi} C3_i \min \{0, X_{i,j} + T_{i,j} - d_i\} \\
& - \sum_{i=1}^N \sum_{j \in \Pi_i \cap \Phi} C4_i \max \{0, X_{i,j} + T_{i,j} - d_i\}
\end{aligned}$$

s.t:

$$\begin{aligned}
X_{SI(i,j),j} - X_{ij} & \geq T_{ij} & i \in \Omega; j \in \Pi_i; SI(i, j) \in \Omega \\
X_{i,SP(i,j)} - X_{ij} & \geq T_{ij} + t_{j,SP(i,j)} & i \in \Omega; j \in \Pi_i; SP(i, j) \in \Pi_i \\
X_{SI(i,j),j} - X_{ij} & \geq T_{ij} + S_{SI(i,j),j} + \mu & i \in \Omega_k; j \in \Pi_i \cap \Phi; SI(i, j) \in \Omega_p \\
& & k, p = 1, \dots, M; k \neq p \\
X_{ij} & \geq 0 & i \in \Omega; j \in \Pi_i
\end{aligned}$$

Die nichtlinearen Terme in diesem Modell stehen in der dritten und vierten Zeile der Zielfunktion. In diesen Termen werden nur casting-Maschinen

benutzt, also teilen die Autoren die Menge der Maschinen, die für charge i gebraucht werden, in die caster $\{j : j \in \Pi_i, j \in \Phi\}$ und alle anderen $\{j : j \in \Pi_i, j \notin \Phi\}$. In der Zielfunktion werden die Summen für die Strafkosten $C3$ und $C4$ bezüglich oben genannter Unterscheidung aufgetrennt. Entsprechend wird mit den Nebenbedingungen verfahren.

Es werden zwei neue Variablen definiert: $Z_{ij} = -\min\{0, X_{ij} + T_{ij} - d_i\}$ und $Y_{ij} = \max\{0, X_{ij} + T_{ij} - d_i\}$. Nun gilt $Y_{ij} - Z_{ij} = X_{ij} + T_{ij} - d_i$ beziehungsweise $X_{ij} = Y_{ij} - Z_{ij} - T_{ij} + d_i$.

X_{ij} kann durch die neuen Variablen ersetzt werden und man erhält ein Modell, das linear in Y_{ij} und Z_{ij} ist.

MIP Formulierung eines belgischen Werks.

Bellabdaoui und Teghem [2006] [11] formulieren das gegebene Problem derart, dass es von Standard Software Packages gelöst werden kann. Ihr Modell ist zugeschnitten auf ein belgisches Stahlwerk mit je zwei converter (CV), refiner (RF) und caster (CC).

Notation:

$i \in \{1, \dots, n_1\}$	Index für die charge Reihenfolge für converter 1.
$j \in \{1, \dots, n_2\}$	Index für die charge Reihenfolge für converter 2.
k	Index für den converter.
m_1	Index für die charge Position in converter 1.
m_2	Index für die charge Position in converter 2.

Je nachdem, ob Gesamtzahl der charges gerade ist oder nicht, berechnen sich die Anteile für die converter. $M_1 = M_2 = (n_1 + n_2)/2$ oder $M_1 = (n_1 + n_2 + 1)/2$ und $M_2 = (n_1 + n_2 - 1)/2$.

Variable:

$z_{i,k,m}$	Gibt an, ob charge i auf Position m an converter k liegt.
$z'_{j,k,m}$	Gibt an, ob charge j auf Position m an converter k liegt.
x_i^s bzw. y_j^s	Startzeitpunkt von charge i auf Stufe s .
	$s \in \{1, 2, 3\}$ entspricht den Stufen CV, RF und CC.
p_i^3 bzw. q_j^3	Produktionsdauer auf der 3. Stufe für charge i .

Parameter:

Δ_1 bzw. Δ_2	Startzeitpunkt für CV1 respektive CV2. O.b.d.A. $\Delta_1 < \Delta_2$.
$[\alpha_1, \alpha_2]$ bzw. $[\beta_1, \beta_2]$	Variationsintervall für den Startzeitpunkt des ersten charges auf CC1 bzw. CC2.
p_i^1 bzw. q_j^1	Produktionsdauer für charge i, j in CV1 bzw. CV2.
p_i^2 bzw. q_j^2	Produktionsdauer für charge i, j in RF1 bzw. RF2.
$[\underline{p}_i^3, \overline{p}_i^3]$ bzw. $[\underline{q}_j^3, \overline{q}_j^3]$	Variationsintervall für p_i^3, q_j^3 auf CC1 bzw. CC2.
D_1 bzw. D_2	Maximale Wartezeit für einen charge zwischen CV und CC.
t_{12} bzw. t_{23}	Transportzeit von Stufe 1 auf Stufe 2 bzw. von Stufe 2 auf Stufe 3.

Beschränkungen:

$$\sum_{m_1=1}^{M_1} z_{i,1,m_1} + \sum_{m_2=1}^{M_2} z_{j,2,m_2} = 1 \quad \forall i \quad (40)$$

$$\sum_{i=1}^{n_1} z_{i,k,m} + \sum_{j=1}^{n_2} z'_{j,k,m} = 1 \quad m = 1, \dots, M_{1,2}; k = 1, 2$$

$$z_{1,1,1} + z'_{1,1,1} = 1 \quad (41)$$

$$z_{i+1,1,r+1} \leq \sum_{m_1=1}^r z_{i,1,m_1} + \sum_{m_2=1}^r z_{i,2,m_2} \quad i = 1, \dots, n_1 - 1; \quad (42)$$

$$r = 1, \dots, M_1 - 1$$

$$z_{i+1,2,r} \leq \sum_{m_1=1}^r z_{i,1,m_1} + \sum_{m_2=1}^r z_{i,2,m_2} - z_{i,2,r} \quad i = 1, \dots, n_1 - 1; \quad (43)$$

$$r = 1, \dots, M_2$$

$$x_i^1 = \sum_{m_1=1}^{M_1} (\Delta_1 + p^1(m_1 - 1))z_{i,1,m_1}$$

$$+ \sum_{m_2=1}^{M_2} (\Delta_2 + p^1(m_2 - 1))z_{i,2,m_2} \quad \forall i \quad (44)$$

$$x_i^2 \geq x_i^1 + p^1 + t_{12} \quad \forall i \quad (45)$$

$$x_i^3 \geq x_i^2 + p^2 + t_{23} \quad \forall i \quad (46)$$

$$x_{i+1}^2 \geq x_i^2 + p^2 \quad i = 1, \dots, n_1 - 1 \quad (47)$$

$$x_{i+1}^3 = x_i^3 + p_i^3 \quad i = 1, \dots, n_1 - 1 \quad (48)$$

$$\underline{p}_i^3 \leq p_i^3 \leq \overline{p}_i^3 \quad \forall i \quad (49)$$

$$x_i^3 - (x_i^1 + p^1) \leq D_1 \quad \forall i \quad (50)$$

$$\alpha_1 \leq x_1^3 \leq \alpha_2 \quad (51)$$

Die Bedingungen (40) und (42)-(51) gelten jeweils auch für die zweite Produktionsschiene. Die Gleichungen (40) und (41) befassen sich damit, dass ein charge nur in einem der converter gefertigt werden kann und dass es nur einen ersten charge geben kann. Bedingungen (42) und (43) stellen sicher, dass ein charge erst bearbeitet werden kann, wenn die entsprechende Maschine frei ist. Beschränkung (44) beschreibt den Startzeitpunkt aller charges in einem converter. Die Bedingungen (45), (46) und (47) garantieren, dass eine Operation erst beginnen kann, wenn der charge auf der Maschine angekommen ist und dass nicht zwei charges gleichzeitig in einem RF sein können. Gleichung (48) stellt die Kontinuität auf dem CC sicher. In Beschränkung (49) steht das Intervall, das ein charge auf einem caster braucht. Gleichung (50) beschränkt die Wartezeit auf dem caster. Bedingung (51) definiert das Zeitintervall für den Beginn des ersten charges auf einem caster.

Die Zielfunktion minimiert die Produktionsdauer. Der letzte charge auf CC1 endet zum Zeitpunkt $x_{n_1}^3 + p_{n_1}^3$. Analog endet der letzte charge auf CC2 zum Zeitpunkt $y_{n_2}^3 + p_{n_2}^3$. Somit ergibt sich die Zielfunktion

$$\min x_{n_1}^3 + p_{n_1}^3 + y_{n_2}^3 + p_{n_2}^3$$

3.2 Expertenbasierte Systeme

Job shop problem: kombiniert Erfahrung mit Computerunterstützten Modellen

In ihrer Arbeit beschreiben McKay und Buzacott [2000] [12] ein Expertenbasiertes System, das in Kapfenberg in Zusammenarbeit mit den Böhlerwerken entwickelt wurde. Beschränkungen, die beachtet werden müssen:

- *Kompatibilitätsbeschränkung:* Bei der Bearbeitung im Ofen bleibt ein Teil der chemischen Bestandteile in der Wand zurück. Mit der Daumenregel, dass dies etwa 3% der Elemente sind, ist man immer auf der

sicheren Seite. Die Experten relaxieren diese Regel in gewissen Fällen später im Verfahren noch. Sie gilt für 42 Elemente, wird aber nur bei acht angewandt: Nickel Ni, Chrom Cr, Mangan Mn, Molybden Mo, Wolfram W, Kobalt Co, Vanadium V und Eisen Fe.

- *Zeitliche Beschränkung*: Da die Produktionsstätte über genügend Lagerraum verfügt, ist diese Beschränkung nicht in vielen Fällen relevant. Es gibt jedoch Qualitäten, die nicht abkühlen sollen, oder zu einem fixen Zeitpunkt geliefert werden müssen. Der Produktionsplan muss deshalb mit einer Genauigkeit von maximal zwei Stunden Toleranz erstellt werden.
- *Kapazitätsbeschränkung*: Im Idealfall sollte es keine Pausen geben, sondern der caster ohne Unterbrechung arbeiten. Es gibt zwei Produktionsschienen, die je nach Bestellung ausgenutzt werden können oder müssen.

In Kapfenberg gibt es zwei Verfahren der Stahlherstellung: continuous casting oder casting in ingots.

Die zeitliche Abfolge der Produktionsplanerstellung:

Der Experte erhält zwei Listen für die Produktionsschienen mit je 35 Bestellungen. Der Computer braucht fünf bis zehn Minuten, um eine erste Lösung zu generieren. In der *preprocessing* Phase wird jeder Bestellung ein Wert für die Schwierigkeit der Herstellung zugewiesen. Beim *processing* sortiert der Computer die Bestellungen beginnend mit den schwierigsten. Der Experte kann das Programm jederzeit unterbrechen und den Plan umordnen. Der Computer überprüft dann noch einmal, ob alle Nebenbedingungen eingehalten werden.

Bewertung der Jobs: Verwendet werden symbolische Variablen, die Engstellen deutlicher aufzeigen als Zahlen. Meistens ist es nicht so wichtig zu wissen, wieviele Bestellungen einer Kategorie es genau gibt, sondern nur die Größenordnung.

Diese Variablen sind: Für die Zahl der Bestellungen: *too_many*, *many*, *normal*, *few*, *no*. Für den Schwierigkeitsgrad: *very_difficult*, *difficult*, *normal*, *easy*, *very_easy*. Kategorien: *CC*, *DoubleCastings*, *SmallIngots*, *DeliveryDates*, *Tunaways*, *BigForgingGradeIngots*.

Evaluierung: Das System erstellt eine dynamische Kompatibilitätsmatrix. Sie gibt durch die symbolischen Variablen *very_high*, *high*, *medium*, *low* und *not_compatible*, für alle Bestellungen an, wie schwer es ist, je zwei Jobs

nacheinander zu ordnen. Diese Matrix muss nicht symmetrisch sein! Wenn eine Bestellung verplant ist, werden die entsprechende Zeile und Spalte aus der Matrix gelöscht.

Planerstellung: Die Jobs werden nicht in einer langen Wurst geordnet, sondern in drei bzw. vier Stunden Abschnitten. Das berücksichtigt die Schichten der Arbeiter. Bei Bestellungen mit fixem Lieferdatum wird die Arbeitszeit geschätzt und dementsprechend der Startzeitpunkt festgelegt. Lücken zwischen zwei Jobs sollten gefüllt werden. Manchmal muss man dazu die Ordnungsstrategie ändern. So können auch Jobs mit einer guten Bewertung schwierig zu koordinieren sein.

Die Bestellungsliste kann auch double castings oder Sequenzen enthalten. Der Experte entscheidet, ob es möglich ist, diese in Serie zu produzieren. Zwei denkbare Szenarien: b hat als einzig kompatiblen Nachfolger b_1 . Wird nun b vom Computer eingereiht, so muss b_1 automatisch danach kommen. Hat b zwei mögliche Nachfolger b_1 und b_2 , ist aber b_1 schon verplant, so sind b und b_2 genauso verbunden wie im ersten Beispiel. Je kleiner die Kompatibilitätsmatrix wird, desto schwieriger wird das casting.

Muss ein Job vorgezogen werden oder kommt ein neuer mit zeitlicher Bedingung dazu, so sucht das System nach einer Möglichkeit diesen unterzubringen. Findet das System kein very high oder high compatible, so schlägt es ein medium vor, oder wenn auch ein solches nicht vorhanden ist, ein low vor und der Experte entscheidet, ob diese Anordnung möglich ist. Am Ende der Planerstellung sucht das System noch nach Verbesserungen, indem es zum Beispiel schwierige Jobs samt Nachfolger tauscht.

Das in Kapfenberg implementierte System hat den Vorteil, dass der Benutzer den Computer jederzeit anhalten und selbst Ordnungen vorschlagen kann. So ist es möglich, aufschlussreiche "Was wäre Wenn" Spiele zu spielen.

4 Empirische Ergebnisse bei der Anwendung von Column Generation

Wie im vorigen Kapitel dargelegt, kann das Problem des continuous casting als MIP zur Minimierung der Castinganzahl dargestellt werden. Meine Aufgabe in dieser Arbeit ist es, innovative Lösungsansätze dieser Modelle zu untersuchen.

Man muss in der Modellbildung zwei Typen von Restriktionen unterscheiden. Einerseits muss die Nachfrage erfüllt werden, was zu linearen Nebenbedingungen führt. Andererseits unterliegen die einzelnen casts gewissen Einschränkungen, wie der Dauer des castings oder den Maßabweichungen der einzelnen charges. Letztere Einschränkungen werden nicht durch Nebenbedingungen modelliert, sondern dienen zur Überprüfung der Zulässigkeit von casts.

Ähnlich wie bei Verschnittproblemen definieren sich die Entscheidungsvariablen über die zulässigen casts, im Folgenden auch Spalten oder Muster genannt. Im Entscheidungsproblem wird dann ermittelt, welche casts wie oft ausgeführt werden sollen, sodass die Nachfrage erfüllt und die Anzahl der produzierten casts minimal gehalten wird.

Ich gebe ein kleines Beispiel zur Verdeutlichung an:

charge Typ	castingdauer	width
1	30	550
2	35	550
3	40	600
4	45	700

Tabelle 4.1

Unter den oben angeführten charge Angaben, sowie einer maximalen casting Dauer von 120 Minuten und einer maximalen width Differenz von 100 mm in einem cast, sind beispielsweise die casts

$$\begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \text{ und } \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \text{ erlaubt. Der cast } \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \text{ ist nicht zulässig,}$$

da die Kombination der charges vom Typ 1 und 4 sowie 2 und 4 auf Grund der zu großen width Differenz ausgeschlossen ist.

Jeder zulässige cast wird im MIP Modell durch eine Spalte dargestellt.

Bei typischen Anwendungsproblemen ist mit einer großen Anzahl von Spalten zu rechnen, welche vor Verwendung eines gängigen Lösungsalgorithmus (z.B. Branch and Bound) alle spezifiziert werden müssen. Nicht nur der Aufwand für das Lösen des diskreten Optimierungsproblems, sondern auch der Aufwand für die Modellbildung muss beachtet werden.

Um den Aufwand zu reduzieren, bietet sich aufgrund der Struktur der Problemstellung an, reduzierte Optimierungsaufgaben zu lösen. Dieses Vorgehen verwirklicht der Column Generation [8] Ansatz, welcher bereits im Kapitel 3 vorgestellt wurde. Diesen und einen ähnlichen Ansatz, das Column Selection, habe ich hergenommen, um sie in dieser Arbeit mit der direkten Lösung des MIP zu vergleichen.

Alle von mir verwendeten Lösungsansätze gehen aus von einem MIP mit den Bedingungen

- Einhaltung der maximalen casting Dauer
- Berücksichtigung der maximalen width Differenz in einem cast
- Befriedigung der Nachfrage

und der Zielsetzung, dabei möglichst wenige casts zu produzieren.

Die von mir verwendeten Lösungsansätze sind:

1. Das geradlinige Lösen des MIP. Ich gebe alle möglichen Varianten, wie ein cast unter Einhaltung aller Beschränkungen mit charges gefüllt werden kann, vor. Der Algorithmus sucht unter allen gegebenen Mustern jene Kombinationen aus, die die Zahl der produzierten casts minimiert.
2. Der Column Generation (CG) Ansatz. Der Algorithmus löst das Problem für eine initiale, kleine Menge von casts und berechnet ein optimales, neu hinzuzunehmendes Muster. Auf Grund des schrittweisen Informationsgewinns wird die Lösung immer besser.
3. Der Column Selection (CS) Ansatz. Dieser funktioniert wie CG, er startet mit wenigen casts und nimmt schrittweise neue hinzu. Der Unterschied zu CS ist, dass die neuen Muster nicht generiert werden, sondern aus einem Pool an vorhandenen Möglichkeiten ausgewählt werden.

Alle Lösungsansätze haben als Grundlage ein MIP Modell.

4.1 Lösung eines linearen Modells

Zum Lösen eines linearen Optimierungsproblems schreibt man alle Nebenbedingungen und die Zielfunktion in ein Tableau. m Nebenbedingungen bei n Variablen werden durch die $m \times n$ Matrix \tilde{A} und den $m \times 1$ Vektor b dargestellt. Die Zielfunktionskoeffizienten sind durch den $1 \times n$ Vektor c gegeben. x bezeichnet den $n \times 1$ Vektor der Variablen. Die Nebenbedingungen des Optimierungsproblems

$$\begin{aligned} Z &= \min_x cx \\ \text{st.} \quad \tilde{A}x &\leq b \\ x &\geq 0 \end{aligned}$$

können durch die Hinzunahme von Schlupfvariablen auf die Form $Ax = b$ gebracht und das Problem durch das Tableau

$$\begin{pmatrix} c & Z \\ A & b \end{pmatrix}$$

dargestellt werden. Die Matrix A hat nun die Form $(I_m \tilde{A})$, beziehungsweise (BD) , wobei mit B die m Basisspalten und mit D die $n - m$ übrigen Spalten bezeichnet werden. Die Teilmatrix (A, b) wird auf die Form $(I_m B^{-1}D, B^{-1}b)$ gebracht. In der Zielfunktionszeile (c, Z) stehen die relativen Kostenkoeffizienten der Variablen und der Zielfunktionswert. Um eine Verbesserung des Zielfunktionswerts zu erreichen, wird eine Variable, die einen negativen relativen Kostenkoeffizient aufweist, in die Basis aufgenommen.

4.2 Formulierung des cast Zuordnungsproblems mittels Column Generation

Wie oben dargestellt, sind die Entscheidungsvariablen in diesem Problem die verschiedenen casts. Welche und wie oft sie ausgeführt werden, wird anhand zweier Modelle festgestellt, dem Mastermodell und dem Rucksackmodell.

Das Masterproblem minimiert die Summe der verwendeten casts unter der Nebenbedingung, dass die Nachfrage erfüllt werden muss. Dazu werden alle Muster verwendet, die initialisiert, beziehungsweise generiert wurden.

Welche Muster über die initialen hinaus erzeugt werden, ist Aufgabe des Rucksackproblems.

Im Rucksackproblem werden alle Beschränkungen, die die Kombination von charges zu casts betreffen, betrachtet. Das sind im gegebenen Fall die maximale casting Dauer und width Bedingungen. Dadurch wird sichergestellt, dass jeder neu generierte cast alle Beschränkungen erfüllt und er bei der nächsten Iteration im Masterproblem als zusätzliche Entscheidungsvariable zur Verfügung steht.

Notation:

- i Ein charge Typ.
- I Menge aller charge Typen.
- k Index über die Menge aller casts. Diese wächst mit jeder Iteration.
- A Matrix aller zur Verfügung stehenden casts. Jede Spalte entspricht einem cast.
- m Entscheidungsvariable. Läuft über k .
- N Nachfrage pro charge Typ. Läuft über i .

Das Mastermodell

$$\min \sum_k m(k)$$

s.t:

$$\sum_k m(k)A(i, k) \geq N(i) \quad \forall i \in I$$

minimiert die Zahl der benötigten casts $m(k)$ unter der Nebenbedingung, dass die Nachfrage $N(i)$ befriedigt wird.

Das Problem wird gelöst und liefert eine erste Abschätzung für die Zahl der nötigen casts. Dieses Ergebnis wird verbessert, indem man geeignete Spalten zu A hinzufügt, was dem Algorithmus eine größere Auswahl an erlaubten Kombinationen zur Verfügung stellt.

Wie im linearen Modell beschrieben, kann eine Variable die Zielfunktion nur dann verbessern, wenn ihr relativer Kostenkoeffizient negativ ist. Die

reduzierten Kosten sind definiert durch $\sigma_k = 1 - \pi^T A_k$. Hier bezeichnet A_k eine Spalte der Matrix A und π den Vektor der dualen Variablen zur Bedingung $\sum_k A(i, k)m(k) \geq N(i)$. Es muss also $\sigma_k < 0$ gelten. Der kleinste Wert σ_k mit zulässiger Spalte wird nun durch das Rucksackproblem ermittelt.

Notation:

y_{bin_j}	Binärer, zweidimensionaler Vektor.
j	Ein Paar inkompatibler charges. Sind die charges 1 und 7 aufgrund ihrer width Werte inkompatibel, so ist $j = (1, 7)$.
J	Menge aller inkompatibler Paare j .
T_{max}	Maximale casting Dauer eines casts.
$t(i)$	Casting Dauer von charge i .
B	Eine große Zahl.
y	Entscheidungsvariable. Läuft über i .

Das Rucksackmodell

$$\min_y \left\{ 1 - \sum_{i \in I} \pi(i)y(i) \right\}$$

s. t:

$$\begin{aligned} \sum_{i \in I} y(i)t(i) &\leq T_{max} & i \in I \\ y(j) &\geq y_{bin_j}(j) & \forall j \in J \\ y(j) &\leq y_{bin_j}(j)B & \forall j \in J \\ \sum_{\hat{j} \in j} y_{bin_j}(\hat{j}) &\leq 1 & \forall j \in J \end{aligned}$$

Die erste Nebenbedingung befasst sich mit der casting Dauer. Die letzten drei Gleichungen verhindern, dass zwei charges mit zu großer width Differenz in einen cast gelöst werden. Je mehr charges eine zu große width Differenz aufweisen, umso größer wird J . Für jedes inkompatible Paar gibt es ein Element $j \in J$, eine Variable y_{bin_j} und drei Gleichungen.

Der Lösungsvektor y aus diesem Problem erweitert die Matrix A um eine Spalte zu \hat{A} .

Mit diesem neuen \hat{A} wird wieder das Master Problem gelöst und ein neuer dualer Vektor π berechnet. Dieser Vorgang wird so lange fortgesetzt, bis für die Zielfunktion im Rucksackproblem $1 - \sum_{i \in I} \pi_i y_i \geq 0$ gilt. Das bedeutet, dass es keine Spalte gibt, die zur Verbesserung der Zielfunktion im Masterproblem etwas beitragen kann. Eine optimale Lösung ist gefunden.

Eine Mischform der vollständigen Implementierung als MIP und des CG ist das Column Selection. Die Grundidee hierbei ist dieselbe wie bei CG: Man startet mit einer kleinen Anzahl an Spalten und gestattet dem Solver in jedem Schritt, eine zusätzliche Spalte zu benutzen.

Für eine initiale Matrix werden Spalten aus einem Pool vorher erzeugter casts genommen und ein Optimierungsproblem formuliert. Das Problem wird gelöst und von allen übrigen, nicht verwendeten casts der relative Kostenkoeffizient berechnet. Im Unterschied zum CG wird keine ideale Spalte generiert, sondern der Algorithmus sucht unter allen vorhandenen nach derjenigen, die der Zielfunktion den größten Nutzensgewinn bringt. Der Vorgang wird so lange durchgeführt, bis es keine Spalte mit negativen relativem Kostenkoeffizienten mehr gibt oder ein anderes Abbruchkriterium erfüllt ist.

4.3 Numerische Implementierung

Ich habe in GAMS die drei Algorithmen Column Generation, Column Selection und das vollständige MIP implementiert. Vorweg die wichtigsten Unterschiede der Methoden:

- Für MIP und CS müssen zuerst alle möglichen Spalten generiert werden. CG hingegen startet mit einer Initillösung von einigen wenigen Spalten und generiert in jedem Schritt eine neue.
- MIP findet in jedem Fall die beste Lösung. CG und CS suchen eine brauchbare Lösung.
- MIP kann für das Auffinden der Lösung sehr lange Rechenzeiten in Anspruch nehmen. Vor allem das Erstellen der cast Muster nimmt viel Zeit in Anspruch. Der Zeitaufwand für das CG ist durch Abbruchkriterien beschränkt.

- Im CG Ansatz werden alle Beschränkungen im Rucksackproblem berücksichtigt. In MIP und CS werden alle Nebenbedingungen im Masterproblem beziehungsweise in einem separaten Generierungsproblem, behandelt.

Zum Vergleich der von mir implementierten Algorithmen habe ich Probleme mit 10, 12 und 14 verschiedenen charge Typen gewählt. Die casting Dauer der charges reicht von 25 bis 45 Minuten. Die width Abmessungen bewegen sich zwischen 500 und 700 Millimeter, die größte zugelassene Abweichung innerhalb eines casts beträgt 100 Millimeter.

Für die maximale casting Dauer eines casts habe ich drei verschiedene Werte (120, 220, 245) genommen. Durch die verlängerte casting Dauer ist es möglich, mehr charges in einem cast unterzubringen. Auf die Algorithmen hat das deutliche Auswirkungen.

Als Abbruchkriterium habe ich bei CG und CS jeweils 25 Iterationen gewählt.

4.3.1 Parameterveränderungen

Veränderung der casting Dauer

Beim CG ist die Änderung der maximalen casting Dauer nur eine Parameteränderung, der zeitliche Mehraufwand des Rechners ist vernachlässigbar.

Beim MIP und CS muss man die cast Generierung umschreiben, was bei der Erweiterung von 120 auf 220 ungefähr 15 Minuten benötigt hat. Bei wachsendem Tmax wird der Zeitaufwand, den der Rechner braucht, um alle möglichen casts zu erzeugen, zur relevanten Größe. GAMS braucht für die Generierung aller möglichen 248 Muster bei $T_{max}=120$ ungefähr 9.5 Sekunden.

Bei $T_{max}=220$ wächst die Zahl aller möglichen Kombinationen auf 3696 und der Zeitaufwand steigt auf rund 650 Sekunden an. Ist es bei $T_{max}=120$ noch möglich, die cast Generierung im Algorithmus zu integrieren, macht dies bei $T_{max}=220$ keinen Sinn mehr. Die cast Generierung muss in ein eigenes File geschrieben und der Output eingelesen werden.

Bei $T_{max}=245$ dauert die Erzeugung aller 6267 Muster bereits mehr als 6500 Sekunden.

Veränderung der Anzahl an verschiedenen charges

Dieser Eingriff hat sowohl auf MIP/CS als auch auf CG Auswirkungen. Außer der trivialen Tatsache, dass die neuen charges dem Algorithmus bekannt gegeben werden, müssen diverse Kombinationen inkompatibler charges aktualisiert werden. Dazu gehört als Erstes eine Neuordnung der bestehenden und der neuen charges. Hierdurch ändert sich auch die Numerierung alter charges, auf dies muss bei der Benennung inkompatibler Paare Rücksicht genommen werden.

Column Generation

Kommen neue charges hinzu, wird die Liste der charges, die sich nicht gemeinsam in einem cast befinden dürfen, länger. Jedes solche Paar benötigt im Rucksackproblem einen binären Vektor der Länge zwei und drei Gleichungen.

Bei meinem ursprünglichen Modell mit zehn verschiedenen charges gibt es neun inkompatible Paare, also neun binäre Vektoren und 27 dazugehörige Gleichungen. Bei der Erweiterung auf zwölf unterschiedliche charges stieg die Anzahl unerlaubter Kombinationen auf 15 und somit 45 Gleichungen im Rucksackproblem. Eine weitere Erhöhung der charge Zahl auf 15 ist an der von mir verwendeten GAMS Lizenz gescheitert.

Bei 14 verschiedenen charges gab es ebenso wie bei zwölf keinen signifikanten zeitlichen Mehraufwand gegenüber zehn charges.

Die Bearbeitung des Algorithmus pro Erweiterung hat zehn beziehungsweise 20 Minuten in Anspruch genommen.

MIP und CS

Im Fall des MIP und des CS werden die inkompatiblen charge Kombinationen bei der Generierung der casts verboten. Der Aufwand bei der Aktualisierung des Algorithmus auf zwölf charges ist mit 20 Minuten ein wenig länger als bei CG. Die nächste Erweiterung auf 14 charges dauert ebenso 20 Minuten. Relevanter als die Zeit, die zum Programmieren aufgewendet wird, ist jene, die der Rechner benötigt. Während bei $T_{max}=120$ noch keine signifikante Veränderung zu sehen ist, steigt die Zeit, die der Computer für die Generierung bei $T_{max}=220$ in Anspruch nimmt, von rund zwölf Minuten bei zehn charges auf über eine Stunde bei zwölf und auf fast 3.5 Stunden bei 14 charges.

	10 Charges		12 Charges		14 Charges	
T_{max}	Spalten	Zeit (sek.)	Spalten	Zeit	Spalten	Zeit
120	258	1.8	341	2.8	539	4.4
220	3696	703	6028	3777	15677	12317
245	6267	6546				

Tabelle 4.2

4.3.2 Lösungsgenauigkeit

Worauf man beim Arbeiten mit dem Computer aufpassen muss

In den Beispielen habe ich immer denselben solver (coincbc) verwendet. Ebenso sind alle Probleme derart formuliert, dass Überschussproduktion zugelassen wird. Den Gründen, warum ich nicht nach einer Lösung ohne Überschuss suche, widme ich mich im Folgenden.

In meiner Arbeit konzentriere ich mich auf die Zahl der produzierten casts und nicht auf deren Kosten beziehungsweise den Profit. Überschüssige Ware, Lagerhaltungskosten und dergleichen werden von mir nicht berücksichtigt. Es kann auch unter Einbindung von Lagerhaltungskosten vorkommen, dass Überschussproduktion der exakten Erfüllung der Nachfrage vorzuziehen ist. Andererseits ist es aufgrund der vom Algorithmus erzeugten Muster mitunter gar nicht möglich, die Nachfrage punktgenau zu erfüllen.

Ich liste kurz die Vor- und Nachteile des (\geq) und des (=) auf und gebe ein Beispiel für einen Fall, in dem der CG Algorithmus mit (\geq) in kürzerer Zeit ein besseres Ergebnis liefert als das MIP mit (=).

- Die Nebenbedingungen sind bei (\geq) relaxiert. Ein größerer Lösungsraum lässt bessere Lösungen zu.
- Die Lösungssuche ist bei (=) aufwendiger. In den von mir verwendeten Beispielen dauert das Lösen des selben Problems bei (=) bis zu zehnmal länger als bei (\geq)¹.
- CG arbeitet in manchen Problemstellungen mit weniger als 20 cast Mustern. Hier kommt es vor, dass eine exakte Lösung gar nicht möglich ist.

¹

– Diese Angabe bezieht sich nur auf das MIP. Die Lösungszeit für CG und CS variiert nicht mit (\geq) oder (=).

- Initialisiert man das Problem mit den trivialen Mustern $(1,0,0,\dots)$, $(0,1,0,\dots)$, $(0,0,1,\dots)$ usw., ist zwar eine Lösung unter $(=)$ möglich, die Ergebnisse von CG und CS werden jedoch sehr schlecht.

Es folgt das zuvor erwähnte Beispiel: Gegeben sind $T_{\max} = 245$ und 10 charges. Bei einer Nachfrage von 5000000 Stück pro charge Typ ergeben sich folgende Werte:

	Lösung	Zeit
MIP	7142859	38.3 sek.
CG	7142858	2.0 sek.

Tabelle 4.3

Lässt man auch bei dem MIP (\geq) zu, so findet der Algorithmus ebenfalls die bessere Lösung.

Ein anderes Problem, wenn man mit dem Computer arbeitet, stellen große Zahlen dar. Im continuous casting bei der Stahlherstellung sind solche Werte nicht relevant, der Vollständigkeit halber möchte ich jedoch erwähnen, dass bei einer sicherlich utopischen Nachfrage von 1000000000 Stück der MIP Algorithmus versagt. Es kommt zu Auslöschungseffekten bei den letzten Stellen. CG und CS sind auch bei solch großen Werten noch stabil und liefern zum Teil deutlich bessere Ergebnisse als das MIP.

Vergleich der Lösungsqualität

In den nachstehenden Tabellen sind die Ergebnisse der verschiedenen Methoden festgehalten. Sie sind nach der maximalen casting Dauer T_{\max} und der nachgefragten Menge pro charge Typ geordnet. Die Zahl in der Spalte "Spalten" gibt an, wie viele casts vom jeweiligen Algorithmus generiert beziehungsweise verwendet wurden. Bei MIP und CG ist die Zahl eindeutig. Bei CS müssen freilich erst alle Muster erzeugt werden, also kann man hier auch dieselbe Zahl wie bei MIP in die Tabelle schreiben. Ich gebe nur die Zahl an, die der Algorithmus nach dem letzten Iterationsschritt zur Verfügung hat, um eine möglichst gute Lösung zu erzeugen.

$T_{max} = 120$, 10 charges, Nachfrage jeweils 123.			
	Zeit	Spalten	Lösung
MIP	0.3 sek.	258	359
CS	0.75 sek.	17	360
CG	1.4 sek.	17	360

Tabelle 4.4

$T_{max} = 120$, 10 charges, Nachfrage: 123456.			
	Zeit	Spalten	Lösung
MIP	0.2 sek.	258	360080
CS	0.7 sek.	17	360080
CG	1.3 sek.	17	360080

Tabelle 4.5

$T_{max} = 245$, 10 charges, Nachfrage: 123.			
	Zeit	Spalten	Lösung
MIP	4.4 sek.	6267	177
CS	1 sek.	19	177
CG	1.4 sek.	18	177

Tabelle 4.6

$T_{max} = 245$, 10 charges, Nachfrage: 123456.			
	Zeit	Spalten	Lösung
MIP	4.8 sek.	6267	176367
CS	1.1 sek.	19	176367
CG	2 sek.	18	176367

Tabelle 4.7

Vergleich CS mit MIP

Sowohl für CS als auch das MIP müssen, bevor eine Lösung des Problems möglich ist, alle zulässigen casts generiert werden. Während das MIP nach deren Erzeugung ein riesiges Simplex Tableau mit allen Spalten löst, bleibt

der Aufwand bei CS ähnlich jenem bei CG, kleiner. In jedem Schritt wird ein moderates Tableau gelöst, das pro Durchlauf um eine Spalte erweitert wird. Der Zeitaufwand bei CS bleibt konstant, während bei dem MIP Schwankungen auftreten können. Der Vorteil des CS im Gegensatz zum MIP besteht darin, nicht in jedem Schritt alle Spalten betrachten zu müssen.

Verleich CG mit CS

Beide Verfahren starten mit wenigen Spalten. Sie erweitern die Auswahl an casts so lange, bis alle nützlichen gefunden sind und keine weitere Hinzunahme eines casts mehr zu einer Verbesserung der Lösung führt.

Bei CG wird im Rucksackproblem ein Optimierungsproblem gelöst. Die Gewichte der Zielfunktion sind dabei die relativen Kostenkoeffizienten der einzelnen charges. Unter Beachtung der Nebenbedingungen wird so ein neuer cast erzeugt. CS verwendet bei den Optimierungen nicht alle casts, trotzdem werden in jedem Iterationsschritt alle Muster gebraucht. Nach Berechnung einer Lösung werden alle vorhandenen nicht verwendeten casts mittels der relativen Kostenkoeffizienten bewertet und derjenige ausgewählt, der die größte Verbesserung verspricht.

Die Methodik ist ähnlich und oft werden bei identischer Lösung ebenso viele casts von CG generiert wie von CS gewählt. Trotzdem stimmen diese Spalten nur teilweise überein. Auch die Reihenfolge, in der die Spalten hinzugenommen werden, ist nicht ident. Die Übereinstimmung ist nicht einmal größer als zwischen MIP und CG, wie im nächsten Punkt gezeigt wird.

Ich habe die Übereinstimmung in einer Tabelle festgehalten. Die Zahl in der Spalte "CG&CS" gibt an, wie viele Muster sowohl von CG als auch von CS verwendet werden. Unter CG und unter CS steht, wie viele casts bei nur einem Algorithmus auftreten.

In der rechten Seite der Tabelle sind die initialen casts nicht mit eingerechnet. In der ersten Zeile beispielsweise werden zu zehn initialen casts jeweils sieben zusätzliche casts erzeugt, beziehungsweise gewählt.

Casts		Erzeugt bzw. Gewählt		
initial	neu	CG	CG&CS	CS
10	7	3	4	3
10	11	7	4	7
14	15/14	7	8	6

Tabelle 4.8

Vergleich CG mit MIP

Das Hauptaugenmerk dieser Arbeit liegt im Vergleich dieser beider Methoden: Im einen Fall die sture Auflistung aller Möglichkeiten mit anschließender Evaluierung, im anderen Fall ein schrittweises Annähern an eine Lösung.

CG ist bezüglich Parameterveränderungen leichter Hand zu haben und die Wartezeit, wenn eine neue Muster Generierung nötig ist, entfällt. Die Lösungsfindung bei den Verfahren ist ganz verschieden, die Ergebnisse sind es aber größtenteils nicht.

CG generiert zum Teil casts, die in der MIP Lösung vorkommen, verwendet diese aber nicht. Vergleicht man die von CG erzeugten casts mit jenen, die in der Lösung des MIP auftreten, so ergibt sich in einigen Fällen eine ähnliche Übereinstimmung wie es bei dem Vergleich zwischen CG und CS der Fall war.

5 Ein paar Ideen zu einer Benutzeroberfläche

In diesem Kapitel will ich mich kurz damit auseinandersetzen, wie man die Eingabe der verschiedenen Daten zur Berechnung eines Produktionsplans gestalten kann. Mit programmiertechnischen Problemen, die im weiteren Verlauf entstehen können, wie zum Beispiel, dass die Liste der charge Typen nicht mehr in die vorgesehene Box passt, werde ich mich im Rahmen dieser Ideensammlung nicht beschäftigen.

Will man eine Heuristik oder ein mathematisches Verfahren benutzen, um die optimale Produktionspolitik zu berechnen, muss man dem Rechner alle nötigen Daten in geeigneter Weise zur Verfügung stellen.

Bei dem von mir verwendeten Programm GAMS bietet sich als Schnittstelle zwischen dem Solveraufruf und der Eingabe das Programm EXCEL an. Es ist möglich, ein EXCEL file in einen GAMS Code einzulesen beziehungsweise den Output als Excel file auszugeben. Die Bestellmengen können in einer Tabelle an der entsprechenden Stelle eingegeben werden und wenn eine hinreichend gute Lösung gefunden wird, wird diese in Tabellenform wieder ausgegeben.

Vorüberlegungen

Als erstes müssen werksspezifische Daten eingegeben werden. Dies sind gar nicht oder nur selten veränderliche Werte, wie die maximale casting Dauer, das maximale Gewicht des caster, die maximale width Differenz, usw..

Danach werden alle charge Typen angegeben. In meinem Fall waren das anfangs zehn Stück.

Die Oberfläche im laufenden Betrieb

Neben der Eingabe der laufenden Bestellgrößen sollte es auf dem Arbeitsbildschirm die Möglichkeit geben, einen neuen charge Typ anzugeben. Das ist für Sonderproduktionen oder generell die Erweiterung der Produktpalette notwendig.

Die laufenden Bestellungen können, wie schon erwähnt, in Form einer Tabelle eingegeben werden. In dieser Tabelle sollte auf jeden Fall der charge Typ und die nachgefragte Menge aufscheinen. Optional könnte man zusätzlich zur Typenbezeichnung auch noch andere Signalgrößen angeben, dies ermöglicht eine schnellere Auffassung.

Die Menge der casts, mit denen der Algorithmus startet, kann entweder eine voreingestellte, triviale Menge sein oder bei der Eingabe der Bestellgrößen durch den Benutzer definiert werden.

Darüber hinaus sollte es auch die Möglichkeit geben, dem Programm die Verwendung bestimmter casts vorzuschreiben oder zu verbieten. Ein Experte kann zum Beispiel die Abnutzung der Produktionsmittel, welche von den voreingegebenen Schranken nicht erfasst wird, minimal halten.

Die letzten zwei Punkte können über eine simple Abfrage am Rand des Bildschirms erfolgen. Die Oberfläche muss folgende Eingabemöglichkeiten bieten:

- Eingabe der Bestellungen.
- Wahl der initialen cast Menge.

Optional, zum Beispiel durch Anklicken eines Buttons:

- Eingabe eines neuen charge Typ.
- Verpflichtende Verwendung oder Nichtberücksichtigung diverser cast Muster.

Sind die Bestelldaten eingegeben, kann der solver aufgerufen werden. Die Ausgabe des Ergebnisses erfolgt wieder in Form einer Tabelle, welche das Cast Muster und die Zahl der Ausführungen enthält.

Die Produktionsreihenfolge ebenso wie die Wahl zwischen gleichwertigen Lösungen bleibt dem Experten überlassen.

6 Conclusio

Viele verschiedene Herangehensweisen wurden versucht, um das continuous casting Problem optimal zu lösen. Eben so viele haben den Test bestanden und liefern brauchbare, gute Ergebnisse.

Der Column Generation Ansatz, wie in dieser Arbeit gezeigt wurde, ist ein optimales Tool zur Lösung des continuous casting Problems. Oft kommt CG auf das selbe Ergebnis wie das MIP und wenn nicht, unterscheiden sich die Lösungen um ein bis drei casts. Das entspricht einer relativen Abweichung von weniger als zwei Promille.

Die Lösungszeit des CG wächst mit der Größe des Problems, bleibt jedoch immer unter sechs Sekunden und sehr konstant. Die Rechenzeit für das Lösen des MIP schwankt deutlicher, ist meistens schneller als die des CG, in einigen Fällen aber auch deutlich langsamer.

Veränderungen an diversen Parametern sind im CG leicht und schnell durchzuführen, bei der MIP Formulierung muss man lange Rechenzeiten in Kauf nehmen.

Das Zulassen von Überschüssen, um passable Ergebnisse zu erhalten, kommt auch bei anderen Methoden zum Einsatz. Der CG Ansatz verlangt keine weiteren Einschränkungen, ist also nicht anspruchsvoller als andere mathematische Verfahren oder Heuristiken.

Weitere interessante und wertvolle Ergebnisse für eine Analyse können entstehen, indem man andere Nebenbedingungen betrachtet. Dies kann zum Beispiel eine Bedingung für das Gesamtgewicht aller charges in einem cast sein oder chemische Merkmale, die ein gemeinsames casten bestimmter charges verbieten. Man könnte anstatt anderer beziehungsweise zusätzlicher Beschränkungen auch eine andere Zielsetzung betrachten. Vorstellbar ist hier die Zuweisung eines Auszahlungswerts für jeden charge und als Optimierungsziel die Maximierung des Profits.

Die Tatsache, dass bei gleichen Lösungswerten die Lösungszusammensetzungen von unterschiedlicher Gestalt sind, kann auch ein Ansatz für Erweiterungen meiner Arbeit sein. Interessant wäre es hier zu betrachten, ob sich die Lösungszusammensetzungen unter der Hinzunahme weiterer Nebenbedingungen angleichen oder different bleiben. Eventuell ist es sinnvoll, hier nach der Berechnung verschiedener Lösungsmöglichkeiten einen Experten heranzuziehen. Dieser könnte auf Grund seiner Erfahrung zwischen rechnerisch gleichwertigen Ergebnissen noch eine Differenzierung durchführen.

Zusammenfassend kann man sagen, dass der CG Ansatz eine bessere Option ist als das MIP. Es werden nur jene casts erzeugt, die man zur Lösung des Problems benötigt. Es bedarf keiner zusätzlichen zeitraubenden Generierungsalgorithmen. Bei größtenteils gleichen Ergebnissen ist CG auch bezüglich Veränderungen in den Parametern leichter zu handhaben. Im Gegensatz zu anderen mathematischen Verfahren oder Heuristiken werden die Ergebnisse von CG mit wachsender Problemgröße nicht schlechter, sondern die Vorteile gegenüber dem vollständigen MIP immer deutlicher.

Literatur

- [1] Lixin Tang, Gongshu Wang. Decision support system for the batching problems of steelmaking and continuous-casting production. *Omega* 2008;36:976-991.
- [2] Lopez L, Carter MW, Gendreau M. The hot strip mill production scheduling problem: a tabu search algorithm. *European Journal of Operational Research* 1998;106:317-35.
- [3] Dong Hongyu, Huang Min, Ip W.H., Wang Xingwei. On the integrated charge planning with flexible jobs in primary steelmaking processes. *International Journal of Production Research* 2009 48;21:6499-6535.
- [4] Ferretti I, Zanoni S, Zavanella L. Production-inventory scheduling using an ant system metaheuristic. *International Journal of Production Economics* 2006;104:317-26.
- [5] Kumar V, Kumar S, Tiwari MK. Auction-based approach to resolve the scheduling problem in the steel making process. *International Journal of Production Research* 2006;44(8):1503-22.
- [6] Pacciarelli D, Pranzo M. Production scheduling in a steelmaking continuous casting plant. *Computers and Chemical Engineering* 2004;28:2823-35.
- [7] Chang SY, Chang MR, Hong YS. A lot grouping algorithm for a continuous slab caster in an integrated steel mill. *Production Planning and Control* 2000;11:363-8.
- [8] Harjunkoski I, Grossmann IE. A decomposition approach for the scheduling of steel plant production. *Computers and Chemical Engineering* 2001;25:1647-60.
- [9] Tang LX, Luh PV, Liu JY, Fang L. Steelmaking process scheduling using Lagrangian relaxation. *International Journal of Production Research* 2002;40(1):55-70.
- [10] Tang LX, Liu JY, Rong AY, Yang ZH. A mathematical programming model for scheduling steelmaking-continuous casting production. *European Journal of Operational Research* 2000;120:423-35.

- [11] Bellabdaoui A, Teghem J. A mixed-integer linear programming model for the continuous casting planning. *International Journal of Production Economics* 2006;104(2):260-70.
- [12] McKay KN, Buzacott JA. The application of computerized production control systems in job shop environments. *Computers in Industry* 2000;42(2-3):79-97.

A Gams code: Column Generation

```
$ontext
Column Generation angewandt auf das Continuous Casting Problem
Stefan Zojer
$offtext

set
    i      'Chargentypen'           /charge1*charge10/
    il(i)  'Verbotenes Paar'       /charge1, charge8/
*-----
*Die Liste der verbotenen Paare ist je nach charge Anzahl kürzer oder
*länger. Hier ist exemplarisch nur ein Paar angeführt.
*-----
    j      'Merkmale'              /width, castingdauer, Nachfrage/
    k      'Casts'                 /cast1*cast100/
    kl(k)  'Teilmenge von k'       /cast1*cast10/
    iter   'Maximale Iterationszahl' /iter1*iter25/
    kk(k)  'Dynamische Teilmenge'
;

*-----
*Daten
*-----

table Bestellungsdaten(i,*)
        width castingdauer Nachfrage
charge1  500      30      123
charge2  500      35      123
charge3  550      25      123
charge4  550      30      123
charge5  550      35      123
charge6  600      30      123
charge7  600      45      123
charge8  650      40      123
charge9  650      40      123
charge10 700      40      123
;

parameter
    w(i)
    t(i)
    N(i)
;

w(i)=Bestellungsdaten(i,'width');
t(i)=Bestellungsdaten(i,'castingdauer');
N(i)=Bestellungsdaten(i,'Nachfrage');

parameter
    w_min 'Minimale width in einem cast'           /500/
    w_max 'Maximale width in einem cast'           /700/
    d_w   'Maximale width Differenz in einem cast' /100/
    Tmax  'Maximale castingdauer eines casts'       /245/
;
```

```

*-----
*Master Model
*-----

parameter mip(i,k)      'Matrix wachsend in k';

variable z              'Zielfunktionswert';

positive variable m(k) 'Verwendete casts';

equations
    master_objective 'Zielfunktionsgleichung'
    master_castall(i) 'Erfüllung der Nachfrage'
    ;

master_objective..      z =e= sum(kk,m(kk));
master_castall(i)..    N(i) =e= sum(kk,m(kk)*mip(i,kk));

model master /all/;

*-----
*In den nächsten vier Zeilen befinden sich ein paar gams Feinheiten zur
*Erleichterung des Arbeitsvorganges. Sie haben keine spezielle
*Bedeutung bei der Lösungsfindung des Algorithmus.
*-----

master.solprint=2;
master.limrow=0;
master.limcol=0;
master.solvelink=2;

*-----
*Knapsack Model
*-----

parameter B            'Eine große Zahl' /200000000/;

variable knap_z        'Zielfunktionswert';

integer variable knap_m(i) 'Neuer cast';

binary variable knap_bin1(i1) 'Binäre Variable zur Menge i1';

equations
    knapsack_obj      'Zielfunktionsgleichung'
    knapsack_cast_time 'Einhaltung der max castingdauer'
    ;

*-----
*Die folgenden drei Gleichungen verhindern, dass beide charges eines
*verbotenen Paares in den selben cast kommen.
*-----

knapsack_binary1(i1)
knapsack_binary2(i1)
knapsack_binary3
;

```

```

knapsack_obj..      knap_z =e= 1+sum(i,master_castall.m(i)*knap_m(i));
knapsack_cast_time.. Tmax =g= sum(i, t(i)*knap_m(i));
knapsack_binary1(i1).. knap_m(i1) =g= knap_bin1(i1);
knapsack_binary2(i1).. knap_m(i1) =l= knap_bin1(i1)*B;
knapsack_binary3..  sum(i1, knap_bin1(i1)) =l= 1;

model knapsack /all/;

*-----
*Wieder einige gams Spezialitäten.
*-----

knapsack.solprint=2;
knapsack.solvelink=2;
knapsack.optcr=0;
knapsack.limrow=0;
knapsack.limcol=0;

*-----
*Initialisierung
*-----

set ki(k);
ki('cast1')=yes;

scalar
    counter /0/
    index /0/
    done1 /0/
    ;

parameter CastMuster(i,k);
*-----
*Pro charge Typ wird ein cast generiert...
*-----
index=1;
loop(i,
    counter=0;
    done1=0;
    loop(k1 $(not done1),
        counter=counter+1;
        if(counter=index,
            CastMuster(i,k1)= floor((Tmax*(t(i))(-1)));
            index=index+1;
            done1=1;
        );
    );
);
*-----
*...und initialisiert.
*-----
loop(i,
    mip(i,ki)=CastMuster(i,ki);
    kk(ki)=yes;
    ki(k)=ki(k-1);
);

```

```

loop(i,
    abort$(w_min > w(i))
    'Problem ist nicht lösbar. Dimensionen sind nicht kompatibel.';
    abort$(w_max < w(i))
    'Problem ist nicht lösbar. Dimensionen sind nicht kompatibel.';
);

*-----
*Beginn Column Generation
*-----

scalar
    done /0/
    iteration
    ;

loop(iter$(not done),

*-----
*solve master
*-----

    option lp=bdlp;
    solve master using lp minimizing z;

*-----
*solve knapsack
*-----

    solve knapsack using mip minimizing knap_z;

*-----
*Verbesserung durch Hinzunahme eines neuen Musters
*-----

    if(knap_z.l < -0.000000000000001,
        mip(i,ki) = knap_m.l(i);
        kk(ki) = yes;
        iteration = ord(iter);
        display '—Iteration—',
            kk, mip;
        ki(k) = ki(k-1);
    else
        done = 1;
    );
);

abort$(not done) 'Zu viele Iterationen.';

*-----
*solve final mip
*-----

master.optcr=0;

```

```

variable z_fin;

integer variable m_fin(k) 'Verwendete Muster';
m_fin.up(k)=1000000000;

equation
    fin_obj          'Zielfunktion'
    fin_demand(i)   'Erfüllung der Nachfrage'
    ;

fin_obj..          z_fin =e= sum(kk, m_fin(kk));
fin_demand(i)..   N(i) =e= sum(kk, m_fin(kk)*mip(i,kk));

model master_fin /all/;

*-----
*Startlösung für MIP
*-----

loop(k,
    m_fin.l(k)=floor(m.l(k));
);

option mip=coincbc;
solve master_fin using mip minimizing z_fin;

*-----
*Berechnung des Überschusses
*-----

parameter too_much(i);
too_much(i) = sum(kk,m_fin.l(kk)*mip(i,kk))-N(i);

*-----
*z_fin.l liefert den Zielfunktionswert.
*m_fin.l gibt an, wie oft welches Muster ausgeführt wird.
*In mip stehen alle zur Verfügung stehenden cast Muster.
*too_much gibt den Überschuss an.
*-----

display z_fin.l, m_fin.l, mip, too_much;

```
