Optimales Scheduling mit Hilfe von Constraint-Netzen

Susanne Heipcke[1+2], Josef Kallrath[1], Matthias Bücker[3] und Stefan Brode[1]

[1] BASF-AG, ZXA/ZC, Kaiser-Wilhelm-Str. 52, 67056 Ludwigshafen
[2] Katholische Universität Eichstätt, Math.-Geogr. Fakultät, 85071 Eichstätt
[3] Georg Heeg, Objektorientierte Systeme, Baroper Str. 337,44227 Dortmund

Zusammenfassung Constraint-Netze sind eine aus der KI Forschung entliehene Technik, die verwendet werden kann, um auch komplizierte Lösungsräume auf natürliche Art und Weise durch lokale Beschreibung der Beziehungen zwischen Variablen (Constraints) darzustellen. Es gibt keinerlei prinzipielle Beschränkung auf bestimmte, z.B. lineare Beziehungen. Die Constraints sind durch lokale Propagierungs- Mechanismen bei der Suche nach optimalen Lösungen eine wertvolle Hilfe, da durch die Propagierung einzelner Werte und Wertebereiche der Variablen in einem Constraint-Netz die Menge der noch zu untersuchenden Lösungen erheblich eingeschränkt werden kann.
In diesem Beitrag wird die Darstellung von zwei Optimierungsproblemen und ihre Zerlegung in einzelne Komponenten zur Erstellung eines "computertauglichen Modells" vorgestellt. Durch Rezepturbedingungen verknüpfte Aufträge werden chemischen Anlagen unter Berücksichtigung detaillierter Personalanforderungen zugeordnet, wobei die Gesamtbearbeitungsdauer minimiert wird. Eine gemischt-ganzzahlige Formulierung dieses Problems und der Versuch, es mit einem auf LP-Relaxierung beruhenden Branch&Bound-Verfahren zu lösen, liefert bei restriktiven Personalressourcen infolge eines signifikanten *duality gaps* in vertretbarer Zeit keine zulässige Lösung. Mit Hilfe der Constraint-Netz-Propagierung konnten jedoch Lösungsmannigfaltigkeiten effizient untersucht werden und zum Beispiel die Konsequenzen kurzdauernder Überauslastungen diskutiert werden.
Abschließend wird als weitere industrielle Anwendung der beschriebenen Technik ein Modell zur optimalen Reihenfolgeplanung quantenmechanischer Berechnungen mehrerer Moleküle auf einem Cluster aus parallel arbeitenden Workstations diskutiert. Diese Rechnungen zerfallen in mehrere Teilschritte mit unterschiedlichem Parallelisierungsgrad. Mathematisch führt dieses Modell auf ein nicht-lineares, beschränktes, gemischt-ganzzahliges Optimierungsproblem.

1 Einleitung

Auf mathematischen Modellen basierende Optimierung kann in fast allen Bereichen der Industrie wertvolle Beiträge leisten, beginnend mit der Anregung und Bewertung neuer Ideen in Forschung und Entwicklung über Prozeßoptimierung, Steuerung in Produktion und Logistik, Erstellung von Marktprognosen, optimale Preisfindung bis zur strategischen Planung. Die **mathematische Optimierung** kann garantieren, daß eine gefundene Lösung sämtliche Randbedingungen erfüllt und die bestmögliche Lösung überhaupt unter diesen Bedingungen ist. Erst durch den Einsatz mathematischer Optimierung wird es möglich, auch komplexe Systeme optimal zu steuern, zu konfigurieren oder in ihnen optimale Entscheidungen zu treffen. Der Begriff der **Optimierung** —im mathematischen Sinne verwendet— bedeutet die Bestimmung des Maximums oder Minimums einer bestimmten Funktion, die auf einem (beschränkten) Bereich S oder Zustandsraum definiert ist. Die klassische Optimierungstheorie (Differentialrechnung, Variationsrechnung, Optimale Steuerung) behandelt die Fälle, in denen S kontinuierlich ist. Die **gemischt-ganzzahlige**, **kombinatorische**, oder kurz **diskrete Optimierung**, bis vor wenigen Jahren noch ein Randgebiet der mathematischen Optimierung, spielt eine zunehmend wichtige Rolle [6] und bietet Algorithmen [11], wie z.B. auf LP-Relaxation beruhende Branch&Bound Verfahren, die inzwischen auch in kommerziellen Software-Paketen implementiert sind. Der Definitionsbereich S ist teilweise *diskret*, d.h. einige Variablen sind auf ganze Zahlen beschränkt. Die Ganzzahligkeit der Problemstellungen rührt z.B. daher, daß sogenannte Null-Eins-Entscheidungen — etwa die Entscheidung ob ein Arbeitsschritt von

einem bestimmten Mitarbeiter zu einem Zeitpunkt bearbeitet wird oder nicht — zu treffen sind oder Größen — z.B. Standort, Stufenzahl einer Kolonne, Containergröße — nur ganzzahlige oder nur bestimmte Werte annehmen können. Ein Überblick über den Einsatz diskreter Optimierungsverfahren in der chemischen Industrie findet sich bei [9]. Im Rahmen des von der Europäischen Gemeinschaft geförderten ESPRIT-Projektes "PAMIPS", bei dem die BASF Mitglied in einem aus 4 Industriepartnern und 3 Universitäten bestehenden Konsortium ist, werden parallele Algorithmen zur Lösung gemischt-ganzzahliger Formulierungen von Scheduling-Problemen behandelt. Die Notwendigkeit paralleler Hardware und Software deutet schon den enormen Rechenaufwand an, der mit der Lösung von Scheduling-Problemen ([3],[4]) verbunden ist; in der Tat gehören sie zu den NP-schweren Problemen; oft ist die Bestimmung einer Lösung oder der Nachweis der Optimalität in vertretbarer Zeit nicht möglich. Wie in Kapitel 3 beschrieben und in [8] ausführlicher gezeigt wird, läßt sich für das hier aufgeführte Problem bei niedriger Personalkapazität mit einem auf LP-Relaxierung basierenden Branch&Bound Verfahren (siehe z.B. [11]) in vertretbarer Zeit keine Lösung der gemischt-ganzzahligen Formulierung des in Kapitel 2 dargestellten Problems gewinnen. Ein Ansatz ganz anderer Art, welcher der KI Forschung entliehen ist, beruht auf Constraint-Netzen (siehe z.B. [5] oder [7]). Diese Technik kann verwendet werden, um auch komplizierte Lösungsräume einfacher, auf natürliche Art und Weise durch lokale Beschreibung der Beziehungen zwischen Variablen (Constraints), darzustellen. Es gibt dabei keinerlei prinzipielle Beschränkung auf bestimmte, z.B. lineare Beziehungen. Constraints sind durch lokale Propagierungs-Mechanismen bei der Suche nach optimalen Lösungen eine wertvolle Hilfe, da durch die Propagierung einzelner Werte und Wertebereiche der Variablen in einem Constraint-Netz die Menge der noch zu untersuchenden Lösungen erheblich eingeschränkt werden kann, was ausführlicher in Kapitel 4 beschrieben wird. Recht effizient lassen sich "Constraints" im Zusammenhang mit objektorientierten Darstellungen nutzen; das eingesetzte Werkzeug basiert auf Smalltalk-80 mit der Klassenbibliothek COME ([5]). Dieses Werkzeug wird seit einigen Jahren erfolgreich bei CIBA-GEIGY in Basel zur Modellierung, Simulation und Optimierung kontinuierlicher Prozesse genutzt.

Die Anwendung dieses Ansatzes auf das im folgenden beschriebene personalbeschränkte Scheduling-Problem führt, wie in Kapitel 5 dargestellt zu zufriedenstellenden Resultaten. Als Ergebnis der Untersuchung wird auf drei Punkte besonders eingegangen: Qualität der Lösungen (Zeitbedarf zur Abarbeitung sämtlicher Kampagnen bei Planung mit Constraint-Netz-Propagierung - im folgenden mit *CNP* abgekürzt), erforderliche Rechenzeit und quantitative Auswirkungen kurzzeitiger Personalüberauslastungen. In Kapitel 6 wird noch kurz ein weiteres zur Zeit in Arbeit befindliches schwieriges Scheduling-Problem vorgestellt. Dabei handelt es sich um ein Reihenfolgeplanungsproblem quantenmechanischer Berechnungen mehrerer Moleküle auf einem Cluster aus parallel arbeitenden Workstations. Diese Rechnungen zerfallen in mehrere Teilschritte mit unterschiedlichem Parallelisierungsgrad. Mathematisch führt dieses Modell auf ein nicht-lineares, beschränktes, gemischt-ganzzahliges Optimierungsproblem. Kapitel 7 faßt schließlich die Ergebnisse dieser Arbeit zusammen.

Variablen/Unbekannte sind mit kleinen Buchstaben gekennzeichnet, wobei kleine griechische Buchstaben auf binäre, d.h. (0,1)-Variablen hinweisen, die in der Regel als Entscheidungs- oder Belegungsvariablen verwendet werden. Mit großen Buchstaben werden die vor der Optimierung bekannten Größen (direkte Eingabedaten oder aus Eingabedaten ableitbare Daten) bezeichnet.

2 Problembeschreibung: Personalbeschränktes Scheduling-Problem

Im Rahmen dieser Untersuchung soll eine Planung zur optimalen Personalauslastung für einen Gesamtzeitraum von $N_T = 840$ Stunden (5 Wochen) auf Stundenbasis erstellt werden. Betrachtet werden hierbei $N_K = 10$ Kampagnen, die jeweils auf einer fest bestimmten von 9 Anlagen bearbeitet werden sollen; für die Kampagnen 6 und 7 steht nur eine Anlage zur Verfügung. Die Kampagnen bestehen aus je $N_{c(k)}$, ($N_{c(k)} \approx 3 - 30$), Chargen (Ansätze), die jeweils 1 bis 6 Arbeitsschritte mit unterschiedlichem, bekanntem Personalbedarf und unterschiedlicher Dauer erfordern. Ferner ist gegeben, wann die letzte Charge einer Kampagne fertiggestellt sein muß

(= $Liefertermin$) sowie die Dauer (Tabelle 1). Zur Verfügung stehen bis zu 4 Mitarbeiter in kontinuierlicher Wechselschicht. Die Zuordnung von bestimmten Arbeitskräften zu bestimmten Anlagen oder Arbeitsschritten ist nicht vorgesehen, Pausenzeiten und Schwankungen des Personalstandes (Urlaub, Krankheit) werden nicht berücksichtigt. Die verfügbare Personalzahl N_P wird über den gesamten Zeitraum als konstant angenommen. Alternativ zu untersuchende Optimierungsziele sind die möglichst frühe Fertigstellung der letzten (oder auch aller) Chargen bzw. eine Personalauslastung, die möglichst nahe bei 100% liegt. Die Produktion der Kampagnen 2-6 und

Tabelle1. Die Tabelle gibt für die Kampagnen k die gewünschten Fertigstellungstermine Z_k sowie die Attribute $a_{ks}-e_{ks} : p_{ks}$ eines einzelnen Arbeitsschrittes, d.h. Startzeit-Endzeit: Personalbedarf in Einheiten von 1/6 Arbeitskraft

k	Z_k	$a_{ks} - e_{ks} : p_{ks}$					
1	264	1- 2:12	3 : 6	4- 9:2	10-11:6	12-24:3	25-28:12
2	762	1 :12	2- 7: 2	8:6			
3	432	1- 2:12	3-19:12	20-21:2	22-23:9		
4	552	1- 2:12	3- 9: 2	10-11:6	12-24:2	25-26:12	
5	624	1- 2: 6	3- 9: 6	10-12:2	13-24:2		
6	744	1-12: 6					
7	840	1- 2: 6	3- 9: 3	10-12:6			
8	840	1- 2:12	3 : 6	4-24:2	25-27:6	28-34:3	35-37:12
9	744	1- 2:12	3-19: 3	20-24:6	25-29:3	30-31:6	
10	840	1- 5: 6	6 :12	7-15:2	16-21:6		

9-10 ist durch ein *Rezepturgefüge* verkettet, d.h. der Output einer Kampagne wird als Vorprodukt für die nächste Kampagne benötigt. Für alle übrigen Kampagnen bestehen keine Restriktionen bezüglich der Ausgangsprodukte. Die Produktion einer Charge c_{k_1} von Kampagne k_1 erfordert $p = p(k_1, k_2)$ Chargen c_{k_2} von Kampagne k_2, symbolisch $(k_1 \leftarrow p \cdot k_2)$. Im vorliegenden Fall wird das Rezepturgerüst durch $\{(3 \leftarrow 2.5 \cdot 2), (4 \leftarrow 3 \cdot 3), (5 \leftarrow 1 \cdot 4), (6 \leftarrow 1 \cdot 5), (10 \leftarrow 0.3 \cdot 9)\}$ repräsentiert. Aus Rezepturgerüst und Aufträgen ergibt sich der Vektor der zu produzierenden Chargen (7,30,12,4,4,4,10,4,3,10), d.h. insgesamt $N_C = 88$ Chargen. Eine Charge besteht aus einer Zahl von Verfahrensschritten s, die durch Personalbedarf P_{cs} und Dauer D_{cs} charakterisiert sind.

Aus der Dauer $D_c := \sum_s D_{cs}$ der Chargen einer Kampagne, den vorgegebenen Fertigstellungsterminen und den Verkettungen zwischen mehreren Chargen können bereits Restriktionen der erlaubten Startzeitintervalle berechnet werden, indem vor Beginn des Branch&Bound-Prozesses die zulässigen Bereiche für die Variablen des Constraint-Netzes auf lokal konsistente Bereiche eingeschränkt und somit früheste und späteste Startzeiten für alle Chargen berechnet werden. Bei nicht verketteten Kampagnen berechnen sich der späteste Startzeitpunkt aus der Dauer der Charge und dem vorgegebenen Endtermin (Liefertermin). Neben dem Fertigstellungstermin bedeuten die vorgegebenen Fertigstellungstermine für Zwischenprodukte innerhalb der verketteten Produktionen bei fast allen Kampagnen zusätzliche Einschränkungen.

3 Gemischt-ganzzahlige Problemformulierung

Die gemischt-ganzzahlige Formulierung des in Abschnitt 2 beschriebenen Job-Shop-Problems mit limitierter Personalressource lehnt sich an das Projekt-Planungsbeispiel in [2] an und ist vollständig in [8] beschrieben. Entscheidungsvariablen δ_{tc} geben im Falle $\delta_{tc} = 1$ an, daß die Charge c zur Zeit t gestartet wird. Die Startzeiten s_c der Chargen werden mit den Entscheidungsvariablen über die

Bedingung

$$s_c = \sum_{t=t_c}^{t^c} t \cdot \delta_{tc} \tag{1}$$

verknüpft, wobei die Grenzen t_c und t^c den Bereich möglicher Startzeiten für eine bestimmte Charge c eingrenzen. Diese Grenzen ergeben sich aus dem Rezepturgefüge, das den einzelnen Chargen unterliegt und werden mit Hilfe der Konsistenztests bzw. der Domain-Einschränkungen des Constraint-Netz Formalismus' bestimmt. Je enger die Grenzen, desto stärker reduziert sich die Menge der Entscheidungsvariablen. Die Binärvariablen δ_{tc} werden in N_c Mengen $\mathcal{S}_c$

$$\mathcal{S}_c := \{\delta_{tc} | t_c \leq t \leq t^c\}. \tag{2}$$

vom Typ SOS-1 zusammengefaßt, um das Branch&Bound-Verfahren effizienter zu gestalten. Eine weitere Entscheidungsvariable η steuert, ob auf Anlage 6 zuerst die Chargen des Verfahrens 6 oder die des Verfahrens 7 gefertigt werden. Das Rezepturgefüge, das den einzelnen Chargen unterliegt, wurde durch Ungleichungen der Form

$$s_{c_i} \geq e_{c_j} + 1, \tag{3}$$

dargestellt, wobei s_{c_i} den Startzeitpunkt einer Charge c_i aus einer Kampagne k_1 repräsentiert und e_{c_j} den Endzeitpunkt einer Charge c_j aus einer anderen Kampagne k_2 bedeutet. Der Endzeitpunkt e_{c_j} ergibt sich einfach aus Startzeit s_{c_j} und bekannter Chargendauer D_c gemäß

$$e_{c_j} = s_{c_j} + D_c. \tag{4}$$

Die Personalbeschränkung wurde ähnlich wie in [2] formuliert; siehe [8] für geïngfügige Änderungen. Als Zielfunktion wurden wahlweise die Fertigungsdauer t_e aller Kampagnen

$$t_e := \min \max_k e_k \Longleftrightarrow \min \{t_e | \ t_e \ \geq e_k\} \quad \forall k \tag{5}$$

oder die Summe Z aller Kampagnenendzeiten

$$Z := \min \sum_k e_k \tag{6}$$

minimiert. Aus dem Produktionsgefüge läßt sich für das vorliegende Beispiel $t_e \geq 362$ ableiten. Bezeichnet N_P die zur Verfügung stehende Anzahl von Mitarbeitern, so liefert ein numerisches Experiment [8], welches in oben skizziertem MILP N_P als zusätzliche Variable aufnimmt und die Restriktionen um die Bedingung $t_e = 362$ ergänzt, beim Zielfunktional $\min N_P$ approximativ den Wert 2.04. Damit ist gezeigt, daß die LP-Relaxierung des MILPs für sämtliche Personalstände $N_P \geq 2.04$ den Wert 362 ergibt. Dieser Endzeitpunkt kann, wie verschiedene Optimierungsläufe (siehe [8]) mit dem Optimierungssystem XPRESS-MP (siehe [3]) und COME gezeigt haben, mindestens bei einem Personalstand von $4\frac{1}{2}$ Mitarbeitern eingehalten werden.

Mit Hilfe einer Homotopie, die beginnend bei $N_P = 10$ die Lösung der LP(N_P) bestimmt, die Basislösung speichert und als Startwert für LP($N_P - a$), $a \in [0, 1]$, verwendet, läßt sich innerhalb weniger Minuten eine optimale Lösung bis zu $N_P = 6\frac{1}{2}$ gewinnen. Für kleinere Personalstände wächst der Rechenaufwand erheblich. Ursache dafür ist die Symmetrie des Problems. Die meisten LP-Relaxierungen liefern den Wert 362 auf Kosten nicht-ganzzahliger δ_t, die jedoch sämtliche linearen Restriktionen erfüllen. Zur Zeit ist ein Branch & Cut Ansatz in Bearbeitung, um das Problem möglichst doch exakt zu lösen.

4 Propagierung in Constraint-Netzen

4.1 Definitionen

Betrachtet wird ein Optimierungsproblem der Form

$$\min_{(v_1,...,v_n)\in S} Z(v_1,...,v_n), \tag{7}$$

z.B. die quadratische Form

$$\min_{(v_1,v_2,v_3)\in S} \sum_{i=1}^{3} c_i \cdot {v_i}^2. \tag{8}$$

Jeder **Variablen** v_i muß ein **Domain** D_i zugewiesen werden, dies kann z.B. ein Integer-Intervall ($\{1,...,10\}$) oder eine beliebige endliche Menge wie etwa $\{$*rot, gelb, grün*$\}$ sein. Ein Domain sollte endlich sein. Dies garantiert, daß theoretisch alle Lösungen untersucht werden können. In der Praxis können auch unendliche Intervalle angegeben werden, wenn sichergestellt ist, daß durch die Constraints endliche Grenzen vorgegeben werden. Domains müssen darüberhinaus abzählbar sein (in den meisten Fällen ist diese Bedingung hinreichend, um mindestens eine Lösung bzw. die optimale Lösung zu finden). Die **zulässige Menge** S ist eine n-dimensionale Menge, $S \subseteq D_1 \times ... \times D_n$, die durch die (lokalen) Beziehungen (Constraints) zwischen den Variablen charakterisiert ist. Ein Werte$-n$-Tupel aus S heißt **Lösung** des Optimierungsproblems. Es heißt **optimale Lösung** des Optimierungsproblems, wenn es kein anderes Werte$-n$-Tupel mit kleinerem Wert der Zielfunktion (7) gibt. Die **Zielfunktion** $Z(v_1,...,v_n)$ ist eine i.a. reellwertige Funktion auf einem n-dimensionalen Bereich zur Bewertung einzelner Lösungen. Ein **Constraint** $c_i(i_1,....,i_l;R_i)$ stellt eine Beziehung zwischen den Werten von Variablen her. Erforderlich dazu ist neben der Angabe der Variablen die Spezifizierung einer **Relation** R_i, die für die zulässigen Werte der Variablen erfüllt sein muß:

$$c_i(i_1,...,i_l;R_i)(D_{i_1},...,D_{i_l}) := (D_{i_1} \times ... \times D_{i_l}) \cap R_i. \tag{9}$$

Im Sinne der Definition (9) handelt es sich bei einem Constraint c_i um eine Abbildung, die dem Mengen$-l$-Tupel $(D_{i_1},...,D_{i_l})$ die Menge $(D_{i_1},...,D_{i_l}) \cap R_i$ zuordnet. Auch die Relation R_i wird als eine Menge aufgefaßt. Die Indexmenge $\{i_1,...,i_l\}$ kann mehrfach denselben Index enthalten, $l > n$ ist möglich. Für die Formulierung von Constraints ist nicht in jedem Fall Stetigkeit vorauszusetzen (z.B. Rechnen mit Integer-Intervallen), es ist keine vollständige Ordnungsrelation erforderlich (z.B. für Vergleiche), und die Variablen müssen nicht notwendig Zahlen sein, z.B. $v_3 \in \{$*rot, gelb, grün*$\}$). Für jeden Constraint $c_i(i_1,...,i_l;R_i)$ können für alle $j \in \{1,...,l\}$ **Projektionen** pr_j definiert werden:

$$pr_j(c_i(i_1,...,i_l;R_i)(D_{i_1},...,D_{i_l})) \rightarrow \tilde{D}_{i_j} \tag{10}$$

Diese Projektionen identifizieren aus der durch den Constraint c_j definierten Menge durch Projektion der $j-ten$ Stelle Teilmengen des Domains D_j, so daß D_{i_j} ohne Informationsverlust durch $\tilde{D}_{i_j}$ ersetzt werden kann. Wenn für alle $j \in 1,...,l$

$$pr_j(c_i(i_1,...,i_l;R_i)(D_{i_1},...,D_{i_l})) = D_{i_j} \tag{11}$$

gilt, dann heißt $c_i(i_1,...,i_l;R_i)$ **lokal konsistent**. Für jeden beliebigen Wert $w_{\hat{j}} \in D_{\hat{j}}$ gibt es in diesem Fall Werte $w_k \in D_k$, $k = i_1,...,\hat{j},...,i_l$, so daß $(w_{i_1},...w_{\hat{j}},...w_{i_l})$ den Constraint c_i erfüllt (greift man zwei oder mehr Werte heraus, ist dies nicht mehr gewährleistet). Durch die Verwendung der Projektionen pr_j wird die Berechnung erleichtert, da nicht weiter mit n- oder l-dimensionalen Mengen, sondern mit einfachen Mengen oder Werten gerechnet werden kann. Da das kartesische Produkt der Projektionen $pr_{i_1} \times ... \times pr_{i_l}(c_i(i_1,...,i_l;R_i))$ i.d.R. echte Obergrenze von $(D_{i1} \times ... \times D_{il}) \cap R_i$ ist, ergibt sich jedoch gleichzeitig ein Informationsverlust. Das **Constraint-Netz CN** definiert den zulässigen Bereich S eines Optimierungsproblems in Abhängigkeit von den

Domains $D_1, ...D_n$ aller Variablen und der Menge C aller Constraints c_i, die durch die logische *Und-Bedingung* verknüpft sind:

$$CN(D_1, ..., D_n; C) = S = \wedge_i c_i \tag{12}$$

Man spricht von **globaler Konsistenz** des Constraint Netzes, wenn für jedes n-Tupel $(s_1, ...s_n) \in S$ und für alle Constraints $c_1, ...c_m \in C$ gilt:

$$c_i(i_1, ..., i_l; R_i)(\{s_{i_1}\} \times ... \times \{s_{i_l}\}) = (s_{i_1}, ..., s_{i_l}), \tag{13}$$

d.h. jede Lösung erfüllt alle Constraints. Entsprechend ergibt sich **Inkonsistenz** für jedes n-Tupel $(\bar{s}_1, ...\bar{s}_n) \notin S$:

$$\exists c_i \in C \text{ so daß } c_i(i_1, ..., i_l; R_i)(\{\bar{s}_{i_1}\} \times ... \times \{\bar{s}_{i_l}\}) = \{\} \tag{14}$$

Globale Konsistenz ist eine stärkere Forderung als lokale Konsistenz für alle Constraints, wie am Beispiel von drei Ampeln ersichtlich: Jede Ampel v_1, v_2, v_3 zeigt rot oder grün; die Constraints $v_1 \neq v_2$, $v_1 \neq v_3$, $v_2 \neq v_3$ können jeweils jede für sich, d.h. lokal, erfüllt werden, eine Lösung, die allen Ungleichungen gleichzeitig genügt, d.h. global konsistent ist, gibt es aber offensichtlich nicht. **Ziel der Propagierung** von Constraint-Netzen ist das Erreichen globaler Konsistenz, d.h. die Bestimmung derjenigen Werte l-Tupel, die allen gegebenen Beschränkungen (Constraints) genügen ([12]). Dazu wird zunächst lokale Konsistenz für die einzelnen Constraints hergestellt. Die Domains der Variablen werden dann durch Berücksichtigung von immer mehr Constraints ständig weiter eingeschränkt. Für tiefergehende Details sei auf [5] verwiesen.

4.2 Algorithmische Aspekte

In COME steht einerseits ein **Branch&Bound**-Verfahren zur exakten Optimierung zur Verfügung, andererseits können aber auch **generierende Heuristiken** (Monte-Carlo-Methode, Simulated Annealing) oder unvollständiges Branch&Bound aufbauend auf Propagierung in Constraint-Netzen angewandt werden.

Vor Beginn des eigentlichen Branch&Bound-Verfahrens sowie nach jedem Verzweigungsschritt erfolgt eine Überprüfung auf lokale Konsistenz. Ist allen Variablen ein fester Wert zugewiesen, so wird anhand der Constraints überprüft, ob das n-Tupel $(v_1, ...v_n) \in S$ ist, d.h. ob globale Konsistenz gegeben ist. Ist dies der Fall, so wird $(v_1, ...v_n)$ als zulässige Lösung registriert. Die Konsistenzprüfung für Teillösungen verhindert, daß zu viele unzulässige Lösungen getestet werden müssen, Informationen werden möglichst umfassend ausgenutzt, die Laufzeit wird verkürzt. Beim **Branch&Bound** besteht die Möglichkeit, das Verfahren nach einer bestimmten Laufzeit bzw. nach einer festzulegenden Anzahl von Schritten, d.h. von Lösungen, zu stoppen (unvollständiges Branch&Bound). In diesem Fall ist nicht sicher, ob das bis zu diesem Zeitpunkt als beste Lösung festgestellte n-Tupel aus S das Optimum ist. In den meisten Fällen wird allerdings schon nach ca. 10% der Rechenzeit das exakte Optimum gefunden, die übrige Zeit ist notwendig, um zu prüfen, ob diese Wertekombination tatsächlich das Optimum bildet.

Die **generierenden Algorithmen** in COME arbeiten im Gegensatz zu anderen Verfahren nicht mit Penalisierung, um die Zulässigkeit der Lösungen zu gewährleisten, statt dessen wird direkt überprüft, ob alle Constraints erfüllt sind. Damit vermeidet man weitere durch die Penalisierung erzeugte lokale Nebenminima. Allerdings hat man so viele unzulässige Tupel zu prüfen. Für sämtliche Heuristiken kann dasselbe Constraint Netz zugrunde gelegt werden wie für die exakte Optimierung; dies erleichtert ein "Umschalten" zwischen exakten und heuristischen Verfahren. Für generierende Algorithmen müssen allerdings zusätzlich Startlösungen, Umgebungen und Entscheidungsfunktionen spezifiziert werden.

5 Constraint-Netz und Ergebnisse

5.1 Modellbildung mit COME

Als Zielfunktionale werden wieder die in Abschnitt 3 vorgestellten Ansätze (5) und (6) betrachtet, d.h. die Endzeit t_e der letzten Charge oder die Summe Z aller Kampagnen-Endzeiten e_k wird minimiert. Das Rezepturgefüge wird wieder gemäß (3) erfaßt. Die richtige Verknüpfung von Chargen innerhalb einer Kampagne wird durch

$$s_{c+1} - s_c \geq D_c \quad ; \quad \forall k = 1, ..., N_K, \quad \forall c = N_f(k), ..., N_l(k) - 1 \tag{15}$$

gewährleistet, wobei $c = N_f(k)$ bzw. $c = N_l(k)$ die erste bzw. die letzte Charge der Kampagne k bezeichnet. Die Kampagnen 6 und 7 sollen hintereinander auf Anlage 6 laufen, d.h.

$$s_f(6) \geq s_l(7) + D_l(7) \quad \vee \quad s_f(7) \geq s_l(6) + D_l(6) \tag{16}$$

Schließlich wird noch eine Bedingung benötigt, die sicherstellt, daß alle J_c Schritte s einer Charge c unmittelbar aufeinander folgen müssen:

$$s_{c(s+1)} - s_{cs} = D_{cs} \quad ; \quad s = 1, ..., J_c, \ \forall c \tag{17}$$

Sämtliche auf Binärvariablen beruhendene Personalrestriktionen sowie (1-3) entfallen, da in COME spezielle Objekte zur Modellierung von Ressourcen zur Verfügung stehen. Die Anlagen werden als Objekte definiert, die mit einem Zeitstrahl von 1 bis N_T versehen sind, auf dem Aktivitäten eingeplant werden können. Da die einzelnen Arbeitsschritte der Chargen (insgesamt 329) jeweils unterschiedlichen Personalbedarf haben, werden sie als eigenständige Aktivitäten behandelt. Für die Modellierung des Personalstandes ist ebenfalls eine mit einem Zeitstrahl versehene Klasse vorgesehen. Eine Aktivität wird gleichzeitig auf einer Maschine und auf dem Personalzeitstrahl eingeplant. Das Arbeiten mit Belegungsvariablen —wie bei XPRESS-MP notwendig— entfällt also bei COME, es müssen allerdings stattdessen die verschiedenen Zeitstrahlen verwaltet werden.

5.2 Ergebnisse

Mit COME können Lösungen gefunden werden, sofern mindestens $N_P = 3$ Mitarbeiter zur Verfügung stehen. Die Rechenzeit Δt auf einer IBM $RS/6000$ beträgt etwa eine Stunde; es handelt sich dabei jeweils nur um eine erste zulässige Lösung, nicht unbedingt um das Optimum — innerhalb der folgenden Stunden (bis zu 3 Tagen) konnte bisher allerdings in keinem Fall eine bessere Lösung gefunden werden. In den Fällen, in denen der Optimalitätsbeweis nicht möglich ist, kann mit Hilfe der LP-Relaxierung und einer mit *CNP* gefundenen Lösung die Qualiät der Lösung bewertet werden. Die LP-Relaxierung und ihr gefundener Lösungswert z_{LP} dient als untere Schranke z_u, die *CNP*-Lösung z_{CNP} als obere Schranke z_o. Dann gibt

$$G := \frac{z_o - z_u}{z_u} \tag{18}$$

einen garantierten Maximalabstand der *CNP*-Lösung von der optimalen Lösung $z_*, z_u \leq z_* \leq z_o$ in dem Sinne an, daß $\frac{z_{CNP} - z_*}{z_*} \leq G$ ist. Ziel weiterer Untersuchungen ist es daher, möglichst realistische Schranken z_u und z_o zu gewinnen. Die Ergebnisse einer Untersuchung der Abhängigkeit der gesamten Fertigungsdauer von der Personalkapazität sind in Tabelle 2 auf Basis der besten gefundenen Lösungen zusammengefaßt. Die Art der Aufzählung — immer beginnend mit den kleinsten Werten im zulässigen Bereich — legt die Vermutung nahe, daß zumindest in einigen Fällen bereits das Optimum gefunden ist, ohne dies exakt bewiesen zu haben. Zur Abschätzung können aber einige Schranken angegeben werden. Für beliebige Personalzahlen N_P bildet 362 eine untere Schranke. Für Werte $N_P \leq 2.48$ läßt sich diese Schranke verbessern, indem man die Summe aller Mannstunden durch den Personalstand dividiert. Für $Np = 2$ ergibt sich dabei der Wert 450.1. Die Chargen der Kampagnen 1,2,3,4,8 und 9 beginnen alle mit Personalbedarf 2, diese Chargen

Tabelle2. Diese Tabelle gibt die Rechenzeiten Δ_t (auf IBM RS/6000) und erzielten Fertigungsdauer t_e *bzw.* Z bei gegebener Personalzahl N_p und kurzdauernder Überauslastung U in Prozent an. G gibt die Qualitätsbewertung der Lösung in Prozent gemäß (18) an; 0 bedeutet bewiesene Optimalität.

N_p	U	$\Delta_t[h]$	t_e	$t_e[h]$	$G[\%]$	$Z[h]$
3	100.0%	1:00:48	22^d17^h	545 (100 %)	50.6	4186 (100 %)
$3\frac{1}{3}$	111.0%	0:30:07	19^d11^h	467 (85.7%)	29.0	3813 (91.1%)
4	100.0%	4:45:48[1]	16^d01^h	385 (100.0%)	6.4	3242 (100 %)
$4\frac{1}{2}$	112.5%	0:43:32	15^d02^h	362 (94 %)	0.0	2988 (92.2%)

[1] mit 486er PC, 66 MHz

können bei einem Personalstand von 2 Personen nicht gleichzeitig bearbeitet werden. Außerdem haben Chargen der Kampagne 10 in der 6. Betriebsstunde Personalbedarf 2, können sich also nur in den 1. bis 5. Betriebsstunden mit Chargen der Kampagnen 1,2,3,4,8,9,10 überschneiden. Dies ergibt insgesamt 1219 Stunden Bearbeitungszeit allein für Kampagnen 1,2,3,4,8,9 und 10. Mit $N_P = 2$ ist das Zeitlimit von 840 Std. also nicht mehr einzuhalten. Besonders hinzuweisen ist auf das Ergebnis für $N_P = 4\frac{1}{2}$ in Tabelle 2: Deutet man diese Zahl so, daß 4 Arbeitskräfte kurzfristig über 100% ausgelastet sein dürfen (nämlich in diesem Fall bis zu 112.5%), so kann man einen wesentlich früheren Endtermin erzielen. In 44 Stunden liegt die Auslastung dabei über 100%, in 12 Std. ist $N_P = 4\frac{1}{6}$, in 23 Std. ist $N_P = 4\frac{1}{3}$ und nur in 9 Std. ist $N_P = 4\frac{1}{2}$, d.h. die Auslastungsgrenze wird nur in seltenen Fällen erreicht. 360 Stunden entsprechen 30 Schichten à 12 Stunden, d.h. im Durchschnitt besteht in 3 Stunden im Laufe von zwei aufeinanderfolgenden Schichten eine Überauslastung - tatsächlich sind es bis zu 7 Stunden während 2 aufeinanderfolgenden Schichten. Ähnliches gilt für eine Personalzahl $Np = 3\frac{1}{3}$ statt $N_P = 3$. In diesem Fall ist in insgesamt 58 Stunden die Auslastung von 100% überschritten, d.h. im Durchschnitt entfallen wieder auf jeweils 2 der 39 zur Bearbeitung notwendigen Schichten 3 Stunden mit Überauslastung des Personals. Weitere Ergebnisse im Hinblick auf die Auswirkung der Reihenfolgebedingung und den Einfluß der Zielfunktionale finden sich in [8].

6 Job-Scheduling in einem Cluster paralleler Workstations

In diesem Abschnitt sei noch ein Modell für optimales Job-Scheduling zur Berechnung von Molekülen in einem Cluster aus Workstations beschrieben. Dabei soll an dieser Stelle auf die spezielle Modellrealisierung nicht eingegangen werden, sondern eine generische, nur die logischen Verknüpfungen berücksichtigende Formulierung präsentiert werden.

6.1 Problembeschreibung

6.1.1 Moleküle, Arbeitspakete: Eine Gleichgewichtsreaktion ist beispielsweise eine Reaktion des Typs

$$A + B + C \rightleftharpoons D + E$$

d.h. die Moleküle A, B und C reagieren ohne Verluste zu Molekülen D und E. Die Anzahl der unterschiedlichen bzw. resultierenden Moleküle ist dabei variabel. Die quantenmechanischen Berechnungen für sämtliche an einer Reaktion beteiligten Moleküle werden als ein **Arbeitspaket** (Anzahl Arbeitspakete N_P, $N_P \approx 4-5$) bezeichnet; ein Arbeitspaket p umfaßt $N_M(p)$ (zwischen 4 und ca.30) **Moleküle**, die wiederum jeweils aus $N_A(p,m)$ **Atomen** ($N_A(p,m) \approx 5-50$) bestehen. Für die Bearbeitung jedes Arbeitspaketes p sollen eine **Priorität** P_p und eine **Zielzeit** T_p festgelegt werden können. Für jedes Molekül ($m = 1, ..., N_M(p)$) des Arbeitspaketes p werden in einem ersten Schritt die Geometrie und im zweiten Schritt bestimmte physikalische und chemische Eigenschaften berechnet, wobei sich dieser Schritt in drei Teilschritte untergliedert. Nach der

Berechnung der Geometrie (S_1) ist eine manuelle Überprüfung des Ergebnisses (S_2) notwendig, bevor die Ableitung der Moleküleigenschaften (S_3-S_5) begonnen wird. Die manuelle Überprüfung soll nur während der Arbeitszeit, also von 8 bis 17 Uhr an Wochentagen, stattfinden und wird als eigener Bearbeitungsschritt geführt, der keine Maschinenressourcen benötigt; die Daten aus S_1 werden allerdings bis zu dieser Prüfung auf den Festplatten der beteiligten Rechner gespeichert. Die Reihenfolge von S_3 und S_4 ist beliebig, sie müssen aber beide vor Beginn von S_5 abgeschlossen sein. Werden die S_3 und S_5 eines Arbeitspaketes auf verschiedenen Rechnern durchgeführt, so entstehen **Umrüstverluste** bzw. **Transferzeiten** V_{pm}; diese Umrüstzeit wird im Modell als S_6 bezeichnet. Umrüstverluste zwischen den übrigen Schritten sind vernachlässigbar. Jeder der Schritte S_1, S_4 und S_5 kann parallel mehrere Rechner nutzen, wobei für S_1 die Aufteilung so vorgenommen wird, daß alle beteiligten Workstations gleichzeitig rechnen, während für S_4 und S_5 in Abhängigkeit von der Anzahl $N_A(p, m)$ der Atome eines Moleküls die Berechnungen verschiedene Dauern haben und zu unterschiedlichen Zeitpunkten beginnen können. Die **erforderlichen Rechenzeiten** (typischerweise 10^2 bis 10^6 CPU-Sekunden) werden für die weitere Bearbeitung in CPU-Zeit vorgegeben. Für S_2 wird grundsätzlich eine Viertelstunde veranschlagt.

6.1.2 Beschreibung des Workstation-Clusters: Zur Verfügung steht derzeit ein Cluster von $N_R = 16$ Maschinen. Die Bearbeitungszeit für S_1 wird bei paralleler Bearbeitung auf mehreren Rechnern nichtlinear verkürzt in Abhängigkeit davon, welche und wie viele Rechner eingesetzt werden. Der Zusammenhang zwischen der Anzahl u_{pms} der zur Berechnung des $s-ten$ Schrittes ($s = 1, 3, 4, 5$) für Molekül m des Arbeitspaketes p eingesetzten Rechnern und der **reduzierten Rechenleistung** wird durch die Funktion $F(u_{pms})$, $0 \leq F_{pms} \leq 1$, spezifiziert, die zunächst als unabhängig davon angesehen wird, welche Rechner für die Berechnung eingesetzt werden. Im Vektor $\rho_{pms} = [\delta_{pms}(1), ..., \delta_{pms}(N_R)]$, $\delta_{pms}(r) \in \{0, 1\}$ wird die Information, welche Rechner für einen bestimmten Rechenschritt im Einsatz sind, festgehalten, wobei noch die Belegungsvariable δ_{pmsr} und die detaillierte, die Zeit mit einbeziehende Belegungsvariable δ_{pmsrt} verwendet werden:

$$\delta_{pmsr} := \delta_{pms}(r) = \begin{cases} 1, \textit{ pms} \text{ läuft im Planungszeitraum auf Rechner } r \\ 0, \text{ sonst} \end{cases} \tag{19}$$

$$\delta_{pmsrt} = \delta_{pms}(r, t) := \begin{cases} 1, \textit{ pms} \text{ läuft auf } r \text{ im Zeitintervall } t \\ 0, \text{ sonst} \end{cases} \tag{20}$$

Die spezifische Rechenleistung des Rechners r bzgl. des langsamsten Rechners wird durch den **Performance-Faktor** L_r, $L_r \geq 1$, ausgedrückt. Zu berücksichtigen sind weiter die **Kapazität der Arbeitsspeicher** A_r und der **Festplatten** $F_r(r = 1, ..., N_R)$. Die Kapazität der Arbeitsspeicher wird vorerst nicht in die Optimierung miteinbezogen, da $z.Zt.$ die Festplattenkapazitäten für die betrachteten Rechenschritte jeweils eine stärkere Beschränkung darstellen, eine Restriktion bezüglich der Arbeitsspeicher also nicht aktiv wäre.

6.1.3 Optimierungsziele: Die Berechnung der Arbeitspakete soll entweder

1. hinsichtlich der gesamten Bearbeitungszeit oder
2. hinsichtlich der Prioritäten oder
3. bezüglich der Überschreitung festgelegter Endtermine

aller vorliegenden Arbeitspakete über einen Zeitraum von mehreren Wochen optimiert werden. Ein mit diesen konkreten Zielfunktionalen verbundener Aspekt ist es, die Auslastung der Rechner zu erhöhen und die planerische Arbeit des Cluster-Koordinators zu unterstützen bzw. erheblich zu reduzieren. Außerdem sollen diejenigen Moleküle und diejenigen Rechner (Plattenplatz, Arbeitsspeicher) identifiziert werden, die als kritische "bottle necks" anzusehen sind, sowie das gesamte Modell unter dem Blickwinkel einer Sensitivitätsanalyse untersucht werden. Langfristig angestrebt ist ein dynamisches Planungssystem, das den momentanen evtl. durch nicht vorhersehbare Störungen modifizierten Zustand des Clusters erfaßt und darauf aufbauend optimiert.

Mit den einfachen Annahmen $N_P \approx 5$, $N_M(p) \approx 10$ und $Ns = 5$ folgen formal 920 Unbekannte. Einige dieser Variablen sind jedoch eng und direkt gekoppelt, so folgt aus der Endzeit e_p des

Arbeitspaketes p direkt aus der Kenntnis aller e_{pm5}, diese wiederum aus Startzeiten s_{pms} und Dauern d_{pms}. Dies führt zu erheblichen Einschränkungen von Freiheitsgraden und erhöht damit die Chance für die Rechenbarkeit des Problems, zumindest auf Basis der Constraint-Netze. Die Belegungsvariablen sind nicht in die genannte Zahl mit einbezogen. Die δ, ρ sind hier im Hinblick auf die Bildung eines gemischt-ganzzahligen Modells eingefügt. Formal ergeben sich in einem gemischt-ganzzahligen Modell $\approx 4000 \cdot (N_T + 1)$ binäre Variablen, wobei N_T die Anzahl der betrachteten Zeitintervalle ($N_T \approx 10^5$) repräsentiert. Zwar ist auch dies nur eine Obergrenze, aber die Größenordnung der Anzahl binärer Variablen zeigt die Schwierigkeit, das Problem in der vorliegenden Form als gemischt-ganzzahliges Modell zu behandeln.

6.2 Mathematisches Modell

6.2.1 Zielfunktionen: Die drei genannten Zielfunktionen (minimale Gesamtbearbeitungszeit, minimale Summe der gewichteten Endzeiten der Arbeitspakete, minimale Überschreitung der Zielzeiten) nehmen folgende mathematische Form an:

$$\min(\max_p e_p), \quad \min \sum_{p=1}^{N_A} \frac{1}{P_p} e_p, \quad \min \sum_{p=1}^{N_A} z_p \tag{21}$$

Die Prioritäten des zweiten Zielfunktionals sind hierbei so geordnet, daß dasjenige Arbeitspaket, das als erstes fertig sein soll, mit einer niedrigen Zahl, z.B. 1, belegt wird. Arbeitspakete niedrigerer Priorität erhalten höhere Zahlen, z.B. zwischen 1 und 100. Bei Zielfunktion 3 gibt z_p die Überschreitung der gewünschten Zielzeit T_p an; die Zielzeiten T_p sind so gut wie möglich einzuhalten. Die Überschreitung kann durch eine Obergrenze T_p^+ beschränkt werden. Hierfür sind folgende zusätzliche Nebenbedingungen erforderlich:

$$e_p \leq T_p + z_p \quad , \quad z_p \leq T_p^+ \quad ; \quad \forall p \tag{22}$$

6.2.2 Nebenbedingungen: Sie bilden ein System zehn gekoppelter Relationen. 01) Festlegung der Reihenfolge der Rechenschritte für ein Molekül.

$$s_{pm(s+1)r} \geq e_{pmsr} \quad ; \quad \forall p \forall m \forall r \; s = 1, 2, 4 \tag{23}$$

$$s_{pm4r} \geq e_{pm2r}, s_{pm5r} \geq \quad e_{pm6r}, s_{pm6r} \geq e_{pm3r} \quad ; \quad \forall p \forall m \forall r \tag{24}$$

02) Endzeitpunkt der Berechnung eines Schrittes für ein Molekül m

$$e_{pmsr} = s_{pmsr} + d_{pmsr} \quad ; \quad \forall p \forall m \forall s \forall r \tag{25}$$

$$e_{pms} = \max_r \{e_{pmsr}\} \quad ; \quad \forall p \forall m \forall s \tag{26}$$

03) Endzeitpunkt der Berechnung eines Arbeitspaketes p

$$e_p = \max_m \{e_{pm5}\} \quad ; \quad \forall p \tag{27}$$

04) Beschränkung für die parallele Aufteilung eines Rechenschrittes für Molekül m aus Arbeitspaket p, auf mehrere Rechner; der Job pms darf auf höchstens U_{pms} Workstations bearbeitet werden.

$$\sum_{r=1}^{N_R} \delta_{pms}(r) = u_{pms} \quad , \quad u_{pms} \leq U_{pms}, \quad ; \quad \forall p \forall m \; s = 1, 4, 5 \tag{28}$$

05) Wenn S_1 auf mehrere Rechner aufgeteilt wird, dann muß ein besonders leistungsstarker Rechner — diese werden im folgenden mit Rechner 1 oder Rechner 2 bezeichnet — dabei sein.

$$u_{pms} > 1 \Rightarrow [\delta_{pms}(1) = 1 \vee \delta_{pms}(2) = 1] \quad ; \quad \forall p \forall m \quad s = 1 \tag{29}$$

06) Auf jeder Maschine höchstens 1 Job pro Zeiteinheit. Später soll auch zugelassen sein, daß ein Rechner gleichzeitig 2 oder 3 Jobs bearbeiten kann. Wartezeiten, d.h. Zeiten während derer Daten auf der Festplatte gespeichert bleiben, werden nicht als Jobs gerechnet, die reduzierte Festplattenkapazität wird allerdings berücksichtigt.

$$\sum_{s} \delta_{pms}(r,t) \leq 1 \quad ; \quad \forall r \forall t \forall p \tag{30}$$

$$\delta_{pms}(r) \leq 1 \Longleftrightarrow \ [\forall t : s_{pms} \leq t \leq e_{pms} \ , \ \delta_{pms}(r,t) \leq 1] \tag{31}$$

07) Berechnung der Gesamtdauer d_{pmsr} eines Rechenschrittes *pms* aus der auf den langsamsten Rechner bezogenen Referenzdauer Δ_{pms} eines Schrittes bei gegebener *Datentransferzeit* V_{pm} und *Skalierungsfaktor* A.

$$\forall p \forall m \forall r \quad d_{pm1r} = A \cdot \frac{\Delta_{pm1}}{L_{1oder2} F(u_{pm1})} + \frac{(1-A) \cdot \frac{\Delta_{pm1}}{L_r F(u_{pm1})} \cdot \delta_{pm1}(r)}{\sum_{i=1}^{Nr} (\frac{\Delta_{pm1}}{L_i F(u_{pm1})} \cdot \delta_{pm1}(i))} \tag{32}$$

$$d_{pm2r} = \frac{1}{4} Stunde \quad ; \quad \forall p \forall m \forall r \tag{33}$$

$$d_{pm3r} = \frac{\Delta_{pm3}}{L_r F(\rho_{pm3})} \cdot \delta_{pm3}(r) \quad ; \quad \forall p \forall m \forall r \tag{34}$$

$$d_{pmsr} = \frac{\Delta_{pmsr}}{L_r F(\rho_{pms})}, \sum \Delta_{pmsr} = \Delta_{pms} \quad ; \quad \forall p \forall m \forall r \ s = 4,5 \tag{35}$$

$$d_{pm6r} = V_{pm} \quad ; \quad \forall p \forall m \forall r \tag{36}$$

08) Beschränkung des Plattenbedarfs R_{pms} durch die jeweilige Festplattenkapazität F_r

$$R_{pms} > F_r \Rightarrow \delta_{pms}(r) = 0 \quad , \quad s = 3,4,5,6 \tag{37}$$

$$R_{pm1} \cdot \frac{\Delta_{pm1}}{L_r F(u_{pm1})} \cdot \delta_{pm1}(r) / \sum_{i=1}^{N_r} (\frac{\Delta_{pm1}}{L_i F(u_{pm1})} \cdot \delta_{pm1}(i)) \leq F_r \quad ; \quad \forall p \forall m \forall r \tag{38}$$

Die Plattenplatzrestriktionen gehen von der Annahme aus, daß zu Beginn eines Arbeitsschrittes der gesamte durch R_{pms} angeforderte Plattenplatz unmittelbar und für die gesamte Zeitdauer belegt wird; es wird also nicht berücksichtigt, inwieweit der zeitlich aufgelöste benötigte Plattenplatz $R_{pms}(t')$ während der Arbeitsphase eines Schrittes variiert. Es muß also lediglich sichergestellt werden, daß $R_{pms} \geq \max_{t'}\{R_{pms}(t')\}$.

09) Ausführung von S_2 nur während der Arbeitszeit (8-17 Uhr wochentags)

10) Die Umrüstzeit (=Datentransferzeit) V_{pm} zwischen S_3 und S_5 berechnet sich aus der Anzahl der hinzukommenden Rechner multipliziert mit dem Plattenplatzbedarf und dividiert durch die Geschwindigkeit des Netzwerkes.

Für einen gemischt-ganzzahligen Ansatz wird eine *Zeitdiskretisierung* so vorgenommen, daß der Job kleinster Dauer bezogen auf eine Zeiteinheit etwa die Dauer 1 hat. Für die auf Constraint-Netzen basierte Formulierung kann als Zeiteinheit $\Delta t = 1$ Minute bis zu $\Delta t = 15$ Minuten gewählt werden.

6.3 Mathematischer Lösungsansatz mit Constraint-Netzen

Bei dem in Abschnitt 6.2 beschriebenen Modell handelt es sich um ein diskretes, nicht-lineares, beschränktes Optimierungsproblem. Grundsätzlich kann das Problem im Rahmen der in COME zur Verfügung gestellten *CNP* behandelt werden, jedoch läßt die Komplexität des Modells (Zahl der Freiheitsgrade, Zahl der Nebenbedingungen und Art der Verknüpfung) auch hier Schwierigkeiten erwarten. Aus diesem Grunde und auch, um zu einer recht schnellen Lösung des Problems zu gelangen, soll zunächst der folgende Dekompositionsansatz verfolgt werden; seine Realisierung und damit erzielte Ergebnisse sind bei [8] beschrieben.

6.3.1 Dekompositionsansatz: Für einen bestimmten Rechenschritt s eines Moleküls m im Paket p wird der in Abschnitt 6.1.2 definierte Vektor ρ_{pms} als Teilfreiheitsgrad einer generierenden Heuristik, z.B. Simulated Annealing (SA) ([10], [1]) verwendet; er spezifiert die Rechnerzuweisung bezüglich der Aufgabe "pms". Der Vektor $\mathbf{x}$ sei aus sämtlichen Aufgaben, d.h. aus allen *pms*-Kombinationen zusammengesetzt. Als Zielfunktional $f(\mathbf{x}, \mathbf{p}, \mathbf{s}(\mathbf{x}))$ sei ein Kandidat aus (21) gewählt. Mit Hilfe des SA wird nun das Minimierungsproblem

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{p}; \mathbf{s}) \tag{39}$$

gelöst, wobei $\mathbf{p}$ verschiedene Parameter wie Festplattenkapazitäten, CPU Daten, benötigte Rechenzeiten etc., aber auch Rechendauern auf den beteiligten Rechnern beinhaltet. Zu beachten ist aber, daß sich $f(\mathbf{x}, \mathbf{p}; \mathbf{s}(\mathbf{x}))$ als Lösung eines Minimierungsproblems, genauer eines Schedulingproblems bestimmt. Der Vektor $\mathbf{s}$ ergibt sich bei gegebenen $\mathbf{x}$ und $\mathbf{p}$ aus

$$\mathbf{s} = \mathbf{s}(\mathbf{x}, \mathbf{p}) = argmin\{f(\mathbf{x}, \mathbf{p}; \mathbf{s})\} \tag{40}$$

als Lösung des inneren Minimierungsproblems und repräsentiert die Startzeiten s_{pms} der einzelnen Jobs "pms" auf den durch $\mathbf{x}$ vorgegebenen Workstations.

7 Diskussion

Die bisherigen Untersuchungen zeigen die grundsätzliche Eignung der ConstraintNetz-Propagierung zur Lösung von Scheduling-Problemen insbesondere auch von nicht-linearen Problemen. Der Nachweis der Optimalität erweist sich auch auf Basis der *CNP* als schwierig. Jedoch bietet es sich bei den vorgelegten Minimierungsproblemen an, mit Hilfe der LP-Relaxierung der gemischt-ganzzahligen Formulierung eine untere Schranke und mit Hilfe einer via *CNP* abgeleiteten zulässigen Lösung eine obere Schranke zu bestimmen. Damit ist in den Fällen, in denen die Optimalität nicht bewiesen ist, doch wenigstens eine garantierte Schranke berechnet und damit eine Bewertung der Lösungsqualität möglich.

References

1. Aarts E., Korst J. : Simulated Annealing and Boltzman Machines. Chichester, (1993), New York .
2. Ashford R.W. , Daniel R. : Some Lessons in Solving Practical Integer Problems. J. Opl. Res. Soc. **43(5)** (1992), 425–433.
3. Ashford R.W. , Daniel R. : Mixed Integer Programming in Production Scheduling: A Case Study. in: Ciriani T.A. & Leachman R.C. (Eds.) Optimization in Industry, John Wiley & Sons, New York, 1993.
4. Blasewicz J., Ecker K., Schmidt G., Weglarz J. : Scheduling in Computer and Manufactoring Systems. (1993), Berlin, Heidelberg.
5. Bücker M. : Ein allgemeines Konzept zur Modellierung und Lösung diskreter Optimierungsprobleme. Habilitationsschrift, Fakultät für Wirtschaftswissenschaften der Universität Karlsruhe, Karlsruhe, 1995.
6. Grötschel M. , Lovász L. : Combinatorial Optimization. in: Graham R., Grötschel M. & Lovász L. (eds.) Handbook on Combinatorics, North Holland, 1994.
7. Güsgen H.-W. : Constraints. Eine Wissensrepräsentationsform. Arbeitspapiere der Gesellschaft für Mathematik und Datenverarbeitung mbH, (1985), St.Augustin.
8. Heipcke S. : Optimales Scheduling mit Contraint Netzen. Diplomarbeit, Katholische Universität Eichstätt, Eichstätt, 1994.
9. Kallrath J. : Diskrete Optimierung in der chemischen Industrie. in: Mathematik in der Praxis - Fallstudien aus Industrie, Wirtschaft, Naturwissenschaften und Medizin. Springer Verlag, Heidelberg, 1994.
10. Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E. : Equation of state by fast computing machines. Journal of Chemical Physics **21** (1953), 1087–1092.
11. Nemhauser G.L. , Wolsey L.A. : Integer and Combinatorial Optimization. John Wiley & Sons, New York, 1988.
12. Winston P.H. : Künstliche Intelligenz. Bonn, MA Reading Mass., 1987.

This article was processed using the LaTeX macro package with LLNCS style