Angewandte Mathematik und Informatik
Universität zu Köln

Report No. 96-234

Verbundprojekt PARALOR:
Parallele Algorithmen für Routing-Probleme
im Flug– und Straßenverkehr

by

A. Bachem, B. Monien, H. J. Prömel, R. Schrader, B. Voigt

1996

Beitrag zur Statustagung des BMBF
HPSC 95
„ Stand und Perspektiven des Parallelen Höchstleistungsrechnens
und seiner Anwendungen “

1991 Mathematics Subject Classification: 90B06,90B35,90C08
Keywords: parallel algorithms, fleet assignment, crew scheduling, vehicle routing, 3-dimensional packing, branch-and-bound, simulated trading

Verbundprojekt PARALOR: Parallele Algorithmen für Routing-Probleme im Flug– und Straßenverkehr

A. Bachem, B. Monien, H. J. Prömel, R. Schrader, B. Voigt

13. Mai 1996

Zusammenfassung

Im Verbundprojekt PARALOR wird untersucht, wie parallele Algorithmen der kombinatorischen Optimierung zur Lösung großer Optimierungsprobleme aus der industriellen Praxis eingesetzt werden können. Dabei werden insbesondere konkrete Aufgabenstellungen aus dem Bereich der Flugplanoptimierung und der integrierten Steuerung von Fertigungslagern bearbeitet. Der Beitrag gibt einen Überblick über die jeweiligen Problemstellungen, die verwendeten Algorithmen und die bisher erzielten Resultate. Insbesondere werden mit dem *Parallelen Simulated Trading* und dem *Parallelen Branch-and-Bound* parallele Methoden betrachtet, mit denen eine breite Klasse kombinatorischer Optimierungsprobleme behandelt werden kann.

1 Einleitung

Die Lösung kombinatorischer Optimierungsprobleme ist in vielen Bereichen von Wirtschaft und Technik der Schlüssel zur Steigerung der Effizienz technischer Abläufe, zur Verbesserung der Produktqualität und zur Verringerung von Produktions–, Material– und Transportkosten.

Im Rahmen des Verbundprojektes PARALOR wird untersucht, wie parallele Algorithmen der kombinatorischen Optimierung in Anwendungen aus der industriellen Praxis effizient eingesetzt werden können. Die Projektpartner sind:

- Lufthansa Systems Berlin, AG Voigt,
- Profi.L, Gesellschaft für professionelle Logistik und Managementberatung mbH, Remscheid,
- Humboldt–Universität zu Berlin, Institut für Informatik, AG Prömel,
- Universität zu Köln, Mathematisches Institut, AG Bachem,
- Universität zu Köln, Institut für Informatik, AG Schrader,

- Universität Paderborn, Fachbereich Informatik, AG Monien.

Die bearbeiteten konkreten Aufgabenstellungen stammen aus der betrieblichen Praxis der beiden industriellen Projektpartner.
Ein wichtiges Planungsproblem bei der Lufthansa Systems Berlin ist die Flugplanoptimierung. Im Projekt PARALOR werden speziell die Teilprobleme des *Fleet Assignment* und des *Crew Scheduling* bearbeitet.
Profi.L ist eine Unternehmensberatung, deren Schwerpunkt im Bereich Transport und Logistik liegt. Eine komplexe Problematik von hoher wirtschaftlicher Bedeutung ist hier die Optimierung der integrierten Steuerung von Fertigungslagern. Dazu werden im Rahmen von PARALOR Algorithmen zur *Touren– und Speditionsplanung* und effiziente *Packverfahren* entwickelt und integrierte Lösungen erarbeitet.

Im folgenden wollen wir jedes der vier Anwendungsfelder genauer betrachten, wobei wir insbesondere eine Übersicht über die jeweilige Problemstellung, die verwendeten Algorithmen und die bisher erzielten Resultate geben wollen.

Bei allen in Angriff genommenen Aufgaben handelt es sich um Probleme der *kombinatorischen Optimierung*. Als Lösung eines solchen Problems erhält man beispielsweise eine Reihenfolge von Flügen, eine Anordnung von Kisten in einem Container oder einen Ablaufplan zur Maschinenbelegung. In praktischen Problemen unterliegt diese Lösung einer großen Anzahl von Nebenbedingungen. So muß z.B. ein Einsatzplan für Flugpersonal zahlreiche tarifliche Festlegungen beachten. Die Menge der möglichen Lösungen ist dann zwar endlich, aber schon für mittlere Probleminstanzen in der Regel immens groß. In der kontinuierlichen Mathematik werden Optimallösungen mit Hilfe von Konzepten wie Konvexität, Stetigkeit und Lokalität gekennzeichnet. Analoge Charakterisierungen in der diskreten Mathematik gibt es nicht, so daß im wesentlichen die ganze Menge zulässiger Lösungen durchsucht werden muß, um das Optimum zu finden. Da die hier behandelten Probleme der kombinatorischen Optimierung komplexitätstheoretisch zur Klasse der NP–vollständigen Probleme gehören, wird auch die geschickte Ausnutzung der Problemstruktur zu keiner substantiellen Vereinfachung dieser Suche führen. Der Einsatz herkömmlicher sequentieller Verfahren ist für praxisrelevante Probleme aufgrund der enormen Rechenzeiterfordernisse nur sehr eingeschränkt möglich. Parallele Systeme, die zur Zeit schon mehrere tausend leistungsfähige Prozessoren enthalten können, bieten eine Möglichkeit, derartige Probleme in vertretbarer Zeit zu lösen.

Die Parallelisierung von Algorithmen der kombinatorischer Optimierung gestaltet sich sehr schwierig, da die bekannten generellen Ansätze zur Parallelisierung klassischer numerischer Verfahren wie *Domain Decomposition* hier nicht angewandt werden können. Obwohl die Parallelisierung deshalb oft sehr auf das spezielle Problem zugeschnitten werden muß, konnten doch mit allgemeineren Ansätzen gute Ergebnisse erzielt werden. In PARALOR werden das von der Arbeitsgruppe Bachem entwickelte *Parallele Simulated Trading* und die von der Arbeitsgruppe Monien entwickelte Toolbox für *Paralleles Branch–and–Bound* in den genannten Anwendungsbereichen eingesetzt. Wir wollen diese parallelen Methoden und ihre Einsatzmöglichkeiten jeweils kurz darstellen.

2 Flugplanoptimierung

Eine große Fluggesellschaft, wie die Deutsche Lufthansa, bietet wöchentlich knapp 10.000 Flüge an, jeder einzelne muß von einem der rund 250 Flugzeuge der Flotte geflogen werden. Jeder Flug wird von einer 4-8 köpfigen Crew begleitet, die aus dem 11000 Menschen zählenden Flugpersonal des Unternehmens zusammengestellt wird.

Soll ein neuer Flugplan entworfen werden, so wird die Gesellschaft zunächst bestimmen, welche Flugverbindung zu welchem Zeitpunkt angeboten werden soll. Bereits die Überprüfung, ob ein solcher Flugplan mit dem vorhandenen Material und den Beschäftigten realisiert werden kann, erfordert einen hohen Einsatz von Erfahrung und Rechnerpotential. Dabei sind Anschlußzeiten, Wartungszeiträume, Ruhepausen des Personals und viele andere Restriktionen zu beachten.

Selbst wenn ein Flugplan prinzipiell realisierbar ist, muß zusätzlich geklärt werden, welche Kosten er verursacht und welche Erträge zu erwarten sind. Bei der Ertragsberechung spielt die Analyse von Zubringerflügen die wichtigste Rolle, da es nicht unbedingt klar ist, wieviele Menschen, die beispielsweise die gewinnbringende Route von Köln über Frankfurt nach New York gebucht haben, überhaupt zum richtigen Zeitpunkt in Frankfurt sein können, und wieviele Passagiere, die anschließend weiter nach Seattle, Miami oder San Francisco fliegen möchten, ebenfalls mitgenommen werden können. Außerdem kann ein Flugplan, der eine *Crew* zwingt, in New York eine Pause zu machen, die länger dauert als die Pause, die dem Flugzeug gegönnt wird, den Ertrag auf dieser Strecke reduzieren, da zusätzliche Crews eingesetzt werden müssen.

Die Komplexität des Problems ergibt sich dann, wenn beachtet wird, daß jede Woche eine *halbe Million* solcher Reiserouten angeboten wird, für jede einzelne und alle Kombinationen daraus, müssen diese Analysen getätigt werden.

Meist wird so vorgegangen, daß ein historisch gewachsener Flugplan an wenigen Stellen leicht modifiziert wird, und so die Realisierung des Flugplanes sowie seine Kostenbewertung sich an früheren Ergebnissen orientieren kann.

Zwingt der Wettbewerb dazu, Kosten zu sparen, beziehungsweise Erträge zu verbessern, führt dieses Vorgehen bald nicht mehr weiter. Hier müssen völlig neue Flugpläne entworfen werden und auf ihre Realisierbarkeit sowie den zu erwartenden Ertrag untersucht werden. Rechner, wegen der Datenfülle insbesondere Parallelrechner, helfen dem Flugplandesigner dabei, ein *völlig neues* Angebot der Gesellschaft an den Kunden, den Fluggast, zu machen.

2.1 Fleet Assignment

Nachdem geklärt wurde, zu welchem Zeitpunkt eine Flugverbindung (leg) angeboten wird, wird als nächstes festgelegt, welcher Flugzeugtyp (equipment), beziehungsweise welches konkrete Flugzeug (tail) einen Flug bedienen soll. Bild 1 zeigt einen Ausschnitt aus einem typischen wöchentlichen Angebot.

Zu diesem Zeitpunkt wird ebenfalls festgelegt, wieviel Ertrag ein Kunde auf einer Reiseroute (itinerary) einbringen wird. Da mehrere Reiserouten denselben Flug benutzen können, werden in ein und demselben Flugzeug Passagiere nebeneinander in derselben Klasse sitzen können, die völlig unterschiedliche Preise bezahlt haben, jeweils nach der

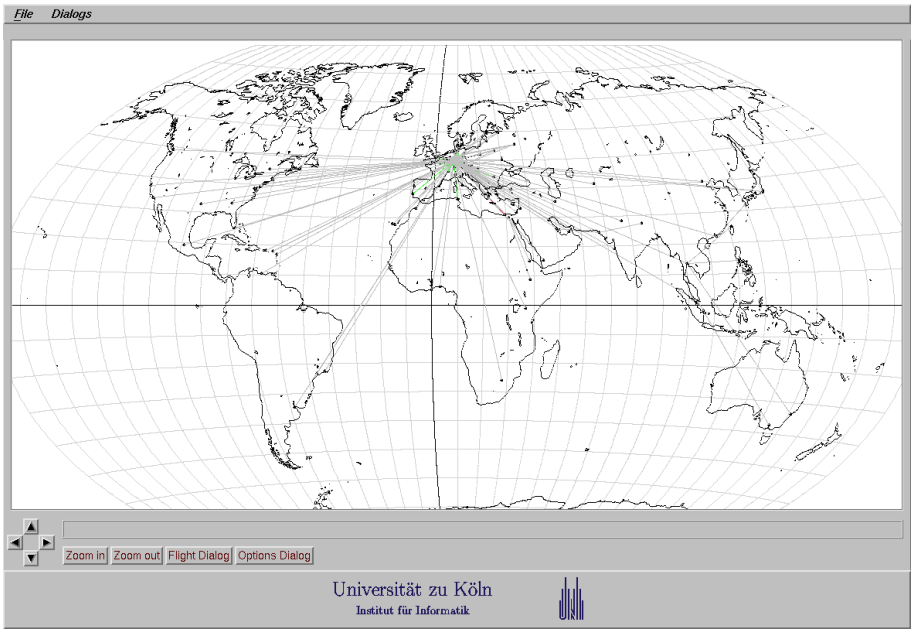


Abbildung 1: Ein Auschnitt aus dem wöchentlichen Flugplan

vollständigen Reiseroute differenziert. Sonderangebote oder andere Aktionen verändern das Preisspektrum zusätzlich.

Darüberhinaus unterscheiden sich Flugzeuge nicht nur in der Kapazität, also der Anzahl der Passagiere, die ein Flugzeug mitnehmen kann, oder der maximalen Reichweite, sondern auch in den Kosten, die ein Flugzeug pro geflogenem Kilometer und befördertem Passagier verursacht. Durch das Fleet-Assignment wird festgelegt, welcher Flugzeugtyp für die Gültigkeitsdauer des Flugplans jede Woche für eine bestimmte Flugverbindung verwendet wird.

Die Kosten, die beim Fleet-Assignment minimiert werden, sind vor allem Energiekosten. Diese liegen bei einem *wöchentlichen* Flugplan bei mehreren Millionen Mark.

2.1.1 Die Zuweisung

Eine graphentheoretische Modellierung des Fleet-Assignment-Problems wird durch einen gerichteten Graphen $D = (V, A)$ gegeben, wobei V, die Knoten, aus den einzelnen direkten Flügen, den legs, gebildet werden. Zwei Knoten $u, v \in V$ werden verbunden, wenn es möglich ist, erst das leg u und anschließend das leg v mit demselben Flugzeug zu bedienen. Ein Fleet-Assignment $F : V \mapsto T$ weist jedem leg $v \in V$ einen Flugzeugtyp $t \in T$ zu.

Zur Lösung von Fleet-Assignment-Problemen kleinerer Fluggesellschaften schlägt Abrara [4] ganzzahlige lineare Programmierung vor. Die Komplexität des zugrundeliegenden Mehrgüterflußproblems [14] (die equipments werden als Güter aufgefaßt, die durch den Graphen D transportiert werden müssen) läßt dieses Vorgehen jedoch bei größeren Problemen nicht zu. Bei drei oder mehr verschiedenen Flugzeugtypen ist das Problem NP-schwer [22].

Eine Zuweisung kann aber gefunden werden, wenn die betrachtete Flottenmenge sukzessive in jeweils zwei Untermengen unterteilt wird, und ein Fluß durch den Graphen festlegt, welcher Flug von einer der beiden Mengen bedient wird. Bestehen die einzelnen Untermengen im Verlauf dieser Heuristik nur noch aus einem Flugzeugtyp, hat man eine Lösung des Fleet-Assignment-Problems gefunden. Unsere Untersuchungen ergaben, daß eine solche Heuristik auf einer mittleren Workstation etwa 30 Minuten benötigt, um ein Fleet-Assignment zu bestimmen. Dabei wird der Wert der Zuweisung durch schnelle Heuristiken abgeschätzt.

Ein gegebenes Assignment wird anschließend mit Austauschheuristiken [29] oder Simulated Annealing mit einigen Stunden Rechenzeit noch nachgebessert.

2.1.2 Rotationelle Verknüpfung

Ist das Fleet-Assignment festgelegt, so wird ein wöchentlicher Einsatzplan einzelner Flugzeuge bestimmt, wobei wieder neue Restriktionen, insbesondere Wartungsintervalle beachtet werden müssen. Hierbei sei für jeden Flugzeugtyp $t \in T$ eine Zahl b_t gegeben, die angibt, wieviele einzelne Flugzeuge des Typs t vorhanden sind.

Für ein Fleet-Assignment seien D_t die durch das Assignment induzierten Subgraphen von D mit $v \in D_t$, wenn $F(v) = t$. Für jeden induzierten Subgraphen ist die minimale Anzahl von Ketten, die ihn überdecken, gleich der Anzahl der zu verwendenden Flugzeuge. Diese Zahl läßt sich mit Matchingalgorithmen [37] in polynomieller Zeit berechnen. Ist sie kleiner als b_t für alle $t \in T$, so ist das Fleet-Assignment unter dem Gesichtspunkt der rotationellen Verknüpfung zulässig. Eine überdeckende Kette entspricht dann einem Wochenplan eines Flugzeugs. Diese Berechnungsphase braucht pro Flugzeugtyp wenige Sekunden Rechenzeit auf einer mittleren Workstation.

2.1.3 Marktanalyse

Bereits während des Aufbaus eines Fleet-Assignment werden die Erträge, die sich durch eine Zuweisung erwarten lassen, überprüft. Hierbei gilt es vor allem, die Passagierzusammensetzung auf legs, die von verschiedenen itineraries benutzt werden, zu kontrollieren. Eine Reiseroute wird im graphentheoretischen Modell als Hyperkante aufgefaßt, die aus mehreren Knoten, den zugehörigen legs, besteht. Diese Kanten des Hypergraphen $\mathcal{H} = (V, \mathcal{E})$ werden durch den Ertrag, den ein Passagier auf dieser Route einbringt, bewertet. Auf den Knoten gibt die Kapazität des Flugzeugtyps die Zahl der Passagiere an, die auf diesem Flug transportiert werden können.

Auf den Reiserouten werden jetzt — graphentheoretisch — Güter, die Passagiere, möglichst ertragsreich durch das Netzwerk transportiert, so daß für jedes Gut in jedem Knoten die Flußerhaltung gewährleistet bleibt. Die Lösung dieses Mehrgüterflußproblemes gibt den Wert des Fleet-Assignments an. Aufgrund der Datenmenge ist vor allem in dieser Berechnungsphase der Einsatz von Höchstleistungsrechnern unerläßlich, auf denen dann parallele Flußtechniken zum Einsatz [20] gebracht werden.

2.2 Crew Scheduling

Crew-Personalkosten stellen nach Treibstoff die zweithöchsten Betriebskosten einer Luftfahrtgesellschaft dar. Das Scheduling von Cockpit- und Kabinen-Crews ist daher eins der wichtigsten Probleme im Flugplanungsbereich (siehe z.B. [33]). Das Problem besteht darin, zu gegebenem gefleeteten und rotationell verknüpften Flugplan eine Zuweisung des Crew-Personals auf die sich aus dem Flugplan ergebenden Dienste zu finden. Für jeden Flug ist dabei ein Crew-Komplement definiert, das angibt, welche Crew-Positionen zu besetzen sind. Planungshorizont ist ein Monat. Eine typische Probleminstanz einer kleineren europäischen Airline umfaßt beispielsweise etwa 7100 Flugsegmente, denen etwa 200 Kapitäne, 160 Erste Offiziere und 360 Cabin-Attendants zuzuweisen sind.

Schwierig wird das Problem durch die mannigfaltigen Nebenbedingungen. Vornehmlich sind jedoch eine Reihe von gesetzlichen und tarifrechtlichen Bestimmungen einzuhalten [1, 2, 3]. Um die Komplexität des Crew-Scheduling-Problems zu reduzieren, wird es typischerweise in zwei Phasen gelöst. In einer ersten Phase, der Pairing-Zerlegung, wird die Menge der zu bedienenden Flugsegmente (legs) in (anonyme) Pairings (auch Besatzungsumläufe) zerlegt. Ein Pairing besteht aus einer räumlich und zeitlich konsistenten Kette von Flugsegmenten, Proceedings und Stand-by-Diensten, die eine Crew in Folge ableisten kann, und entspricht meist einer Rundreise. In einer zweiten Phase erst, dem Crew-Assignment oder -Rostering, werden die Crew-Positionen eines Pairings bestimmten physischen Personen zugewiesen und die Dienstpläne des Flugpersonals erstellt.

2.2.1 Pairing-Zerlegung

Das Problem der Pairing-Zerlegung kann als klassisches Set-Partitioning-Problem formuliert werden, bei dem die Zeilen der Matrix der in Pairings zu partitionierenden Menge der Flugsegmente, die Spalten den zulässigen, gepriceten Pairings entsprechen. Gesucht ist eine günstigste Teilmenge der Pairings, die jedes Flugsegment genau einmal überdecken. Das Problem Set-Partitioning ist wie viele kombinatorische Optimierungsprobleme NP-schwer, so daß es für dieses Problem vermutlich keinen effizienten (d.h. polynomiellen) Algorithmus gibt. Schon für kleinere Probleme ist darüberhinaus die Anzahl der Pairings nicht mehr enumerierbar. Ein Lösungsansatz besteht hier darin, die Spalten der Matrix erst während der Optimierung nach und nach zu erzeugen (Column Generation), siehe [25, 13]. American Airlines löst sukzessive handhabbare Teilproblem mit einem Integer-Programming Ansatz [17, 5]. Hoffman und Padberg [23] testen Schnittebenen-Methoden für Branch-and-Cut-Löser des Crew-Scheduling-Problems, innere Punkt-Methoden werden in [9] verwendet. United Airlines zykelt zwischen einem Pairing Generator und einem Schnittebenen-Integer-Programming-Optimierer unter einem elastischen Set-Partitioning-Ansatz [21].

Da die Qualität des Crew-Assignments letztendlich auch davon abhängt, wie gut die Pairings später in der Rostering-Phase dem Flugpersonal zuzuweisen sind unter Nebenbedingungen, wie sie sich aus Urlaub, Off-days und diversen Bodendiensten ergeben, ist es u.U. gar nicht notwendig oder sinnvoll, das Set-Partitioning-Problem optimal zu lösen. Entscheidender ist es, auf lokale Änderungen des Flugplans schnell reagieren zu können.

Primale Heuristiken. Durch eine geeignete Wahl der Kostenfunktion lassen sich primale Heuristiken gewinnen, die gute Startlösungen für nachfolgende Verbesserungsheu-

ristiken berechnen. Als wesentliche Optimierungsziele sind hier zu berücksichtigen:

- Pairings sollen in „home-bases“ des Flugpersonals starten und enden;
- Proceedings und auswärtige Übernachtungen minimieren;
- Arbeitszeit, die keine Flugzeit ist, minimieren;
- möglichst enge Anlehnung der Pairings an die Flugzeug-Rotationen.

Globale Optimierung. Die erhaltene Lösung wird durch in ein parallelisiertes Simulated-Annealing-Verfahren eingebettete lokale Verbesserungsheuristiken nachgebessert. Zwei Strategien scheinen in diesem Zusammenhang erfolgversprechend zu sein:
Extended 2-Opt. Hier wählt man einen Flughafen aus und bricht alle Pairings, die diesen besuchen, dort auf. Durch Lösung eines gewichteten bipartiten Matching-Problems werden die in diesen Flughafen ein- und ausgehenden Pairings neu zusammengefügt [8].
Subproblem Selection. Hier wählt man einige wenige Pairings der aktuellen Lösung aus und löst auf den von ihnen überdeckten Flugsegmenten ein kleines Set-Partitioning-Problem durch Erzeugung aller dort möglichen Pairings. Verbessert die Lösung dieses kleinen Set-Partioning-Problems die aktuelle globale Lösung, wird sie in diese aufgenommen [30]. Diese Methode ist im System TRIP, das von American Airlines und Continental Airlines genutzt wird, realisiert worden, siehe [17].

2.2.2 Crew Rostering

Neben den erwähnten gesetzlichen und tarifrechtlichen Bedingungen zu Ruhezeiten, maximalen Flugdienstzeiten etc. sind bei der Zuweisung des Flugpersonals auf die sich aus den Pairings gemäß Crew-Komplement ergebenden Dienste weitere Nebenbedingungen zu beachten wie z.B.:

- jedes Crew-Mitglied hat eine Home-Base, wo seine Dienste beginnen und enden;
- Crew-Mitglieder haben nur für bestimmte Flugzeugtypen oder nur für bestimmte Flughäfen eine Lizenz bzw. nur für bestimmte Länder Visa;
- für jedes Crew-Mitglied sind diverse Bodendienste zu planen wie Training zur Lizenz–Erhaltung, Erste–Hilfe-Kurse, ärztliche Untersuchungen, etc.;
- Preassignments im Dienstplan durch freie Tage / Urlaub.

Darüberhinaus sollen die Dienstpläne der Crew-Mitglieder hinsichtlich unliebsamer Dienste ausgeglichen sein; dies trägt dem besonderen europäischen Modell Rechnung, wo die meisten Crew-Mitglieder dieselben oder ähnliche Arbeitsverträge haben.

Das Problem wird als verallgemeinertes Assignment-Problem mit Nebenbedingungen modelliert (siehe auch [11]). Der bipartite Graph hat als Partitionsmengen die Menge D der Dienste und die Menge C der Crewmember. Eine Kante zwischen einem Dienst $d \in D$

und einem Crewmember $c \in C$ besagt dann, daß der Dienst d dem Crewmember c zulässig zugewiesen werden kann. $A = (a_{cd})_{cd}$ bezeichne die Adjazenzmatrix dieses Graphen.

$$
\begin{aligned}
& \min \sum_{\substack{c:\ Crewmember \\ d:\ Dienst}} c_{cd} x_{cd} & \\
s.t. \quad & \sum_{c:\ Crewmember} x_{cd} = 1 & \text{für alle Dienste } d \in D \\
& \sum_{d:\ Dienste} a_{cd} x_{cd} \leq b_c & \text{für alle Crewmember } c \in C \\
& x_{cd} \in \{0,1\} & \text{für alle } c \in C,\ d \in D
\end{aligned}
$$

Die zweite Nebenbedungung modelliert hierbei, daß einem Crewmember nur (noch) gewisse (Rest-) Kapazitäten zur Übernahme von (weiteren) Dienste zugeschrieben werden. Das Problem wird nun in mehreren Phasen gelöst, in welchen jeweils durch ein kostenminimales Matching jedem Crewmember Dienste zugewiesen werden und anschließend der Graph aktualisiert wird.

2.2.3 Entwicklung eines Prototyps

Die entwickelten Algorithmen wurden in einen Prototypen einer Crew-Scheduling-Workbench eingebettet, die das Scheduling-Problem und die Lösungsschritte in Form einer GANTT-Chart graphisch visualisieren kann. Das zugrundeliegende objekt-orientierte Datenmodell wurde mit Hilfe der Object-Modeling-Technique entworfen (siehe [31]). Zur Regelüberprüfung wird zur Zeit noch das XCrew-System der Lufthansa-Systems Berlin herangezogen.

3 Integrierte Fertigungslager

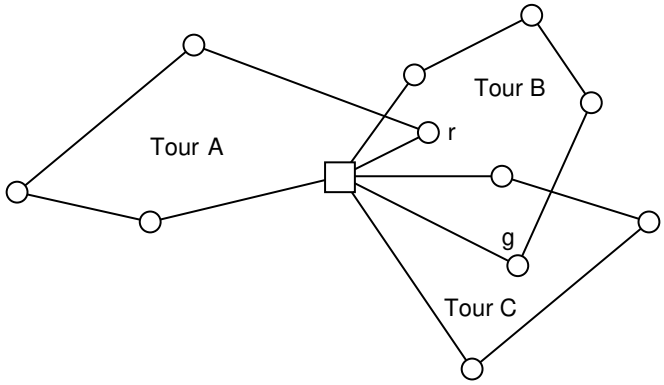
Im Bereich der internen Logistik (Lagerorganisation) als auch im Bereich der externen Logistik (Transport) treten häufig komplexe Optimierungsprobleme auf, in denen das bekannte Vehicle–Routing–Problem – die Bestimmung kostenoptimaler Touren, auf denen Kunden ausgehend von einem Depot beliefert werden – mit einem dreidimensionalen Verpackungsproblem oder mit einem Scheduling–Problem kombiniert wird. So muß beispielsweise bei der Auftragszusammenstellung ein Kommissionierungswagen auf kürzestem Weg durch das Fertigungslager gesteuert und mit unterschiedlichen Paketen beladen werden. In einem anderen Fall werden Produkte nach ihrer Fertigstellung – ohne Zwischenlagerung – direkt in LKW verpackt und an die Kunden ausgeliefert. Hier besteht ein enger Zusammenhang zwischen Vehicle–Routing–Problem (in welcher Reihenfolge werden die Kunden beliefert) , Stauraumoptimierung und Scheduling Problem (in welcher Reihenfolge werden die Produkte erstellt).

3.1 Tourenplanung

3.1.1 Austauschheuristik für das Vehicle–Routing–Problem

Die Arbeitsgruppe Monien hat einen neuen Algorithmus zur Lösung des Vehicle–Routing–Problems entwickelt, der sich zur späteren Parallelisierung und für die Einbeziehung zusätzlicher Packungsheuristiken besonders eignet.

Der Algorithmus bestimmt zunächst eine zulässige Startlösung [12, 32] und versucht anschließend durch komplexe Knotenaustauschoperationen die Lösung weiter zu optimieren. Die Vorgehensweise kann am einfachsten anhand eines kleinen Beispiels erläutert werden. Abbildung 3.1.1 zeigt drei Touren, auf denen Kunden (als Kreise dargestellt) ausgehend von einem zentralen Depot (als Rechteck dargestellt) angefahren werden.



Es sei nun angenommen, daß Kunde r günstiger auf Tour B beliefert werden kann. In diesem Fall wäre Tour B jedoch nicht mehr zulässig, weil beispielsweise Kapazitäts- oder Zeitrestriktionen verletzt sind. Die Zulässigkeit der Tour B kann jedoch durch das Verschieben des Kunden g in eine andere Tour wiederhergestellt werden. Es ergibt sich so ein rekursiver Ansatz, mit desen Hilfe es möglich ist, den Nachbarschaftsraum einer gegebenen Lösung bis zu einer gewissen Tiefe zu durchsuchen. Die Idee des Algorithmus kann wie folgt zusammengefaßt werden:

> Erlaube kurzfristig eine unzulässige Lösung, wenn sie Gewinn bringt. Versuche anschließend durch weitere Kundenverschiebungen die Zulässigkeit der Lösung wiederherzustellen.

Zur Überwindung lokaler Extrema ist die Austauschheuristik in einen Simulated Annealing [24] Ansatz integriert. Die zur Wiederherstellung der Zulässigkeit durchgeführten Kundenverschiebungen können den anfänglichen Gewinn vernichten, ja sogar zu einer Verschlechterung der ursprünglichen Lösung führen. In diesem Fall wird unter Berücksichtigung des verwendeten Abkühlungsschemas ausgewürfelt, ob die schlechtere Lösung trotzdem akzeptiert wird. Um im nächsten Schritt ein Zurückkehren zur ursprünglichen Lösung zu verhindern, wird eine Tabuliste benutzt [18, 19]. Hierin ist für jeden Kunden aufgelistet, welche Kunden *nicht* vor ihm beliefert werden dürfen.

3.1.2 Parallelisierung

Eine Parallelisierung soll die Lösung großer Probleminstanzen ermöglichen. Dazu werden die Kunden gleichmäßig auf die Prozessoren verteilt. Jeder Prozessor überprüft, ob einer

der ihm zugeteilten Kunden günstiger auf einer anderen Tour beliefert werden kann. Ist dies der Fall, so wird der Kunde der entsprechenden Tour zugeordnet. Wie im sequentiellen Fall muß nun eventuell die Zulässigkeit der Lösung durch weitere Kundenverschiebungen wiederhergestellt werden. Die so erhaltene Lösung wird jedoch nicht sofort realisiert wie es bei einer First-Fit-Strategie der Fall wäre. Vielmehr werden alle weiteren Kunden überprüft und schließlich wird die Verschiebungssequenz mit dem größten Einsparungspotential ausgewählt (Best-Fit-Strategie). Die von den einzelnen Prozessoren bestimmten optimalen Verschiebungssequenzen werden an einem ausgezeichneten Prozessor gesammelt und nach Gewinn geordnet. In absteigender Reihenfolge werden alle voneinander unabhängigen Sequenzen zur Realisierung ausgewählt. Die so ermittelte Folge wird allen Prozessoren mitgeteilt (globale Synchronisation). Anschließend kann jeder Prozessor die ausgewählten Verschiebungssequenzen realisieren. Hierdurch wird sichergestellt, daß alle Prozessoren zu jedem Zeitpunkt auf derselben Lösung arbeiten. Die Best-Fit-Strategie wurde gewählt, um die Anzahl der globalen Synchronisationsschritte, welche beim Übergang von einer Lösung zur nächsten notwendig sind, so gering wie möglich zu halten. Außerdem hat sich herausgestellt, daß Best-Fit-Strategien zu besseren Lösungen des Vehicle–Routing–Problems führen [27].

Ein erster Prototyp des parallelen Algorithmus wurde bereits auf dem *Parsytec GCel 3/1024* der Universität Paderborn implementiert. Laufzeittests zeigten, daß auf bis zu acht Prozessoren ein zufriedenstellender Speedup erreicht wird, bei höheren Prozessorzahlen jedoch ein Einbruch in der Effizienz stattfindet. Dieser Effekt kann wie folgt erklärt und vermieden werden. Die einzelnen Verschiebungssequenzen sind unterschiedlich lang und damit unterschiedlich schwer zu berechnen. Für verschiedene Probleminstanzen konnte beobachtet werden, daß ein Großteil der Rechenzeit für die Bestimmung einiger weniger Verschiebungssequenzen verwendet wird. Zur Erhöhung der Skalierbarkeit müssen daher die Sequenzen selbst parallel berechnet werden. Dazu werden einem Prozessor nicht nur Kunden sondern auch gewisse Positionsangaben zugeordnet. Diese Positionsangaben bestimmen, welche Kundenverschiebungen von dem Prozessor ausgeführt werden dürfen. Wird beispielsweise der Kunde r zusammen mit den Positionsangaben $X = \{x_1, \ldots, x_s\}$ dem Prozessor P_1 zugeordnet, so darf P_1 den Kunden r nur vor einem Kunden aus X in einer anderen Tour einordnen. Alle weiteren Positionen werden von Prozessoren überprüft, denen ebenfalls r, jedoch eine Menge X', $X' \cap X = \emptyset$, zugeordnet ist. Die optimale Verschiebungssequenz ausgehend von r wird so von genau einem Prozessor gefunden.
Diese verfeinerte Parallelisierungsstrategie wird insbesondere bei der geplanten Integration der Packalgorithmen von Nutzen sein, da sich hierbei der Aufwand zur Berechnung einer Verschiebungssequenz nocheinmal erheblich erhöht.

3.2 Packverfahren

Im Rahmen des Teilprojektes „Integrierte Steuerung von Fertigungslagern“ untersuchen wir Verfahren zum Packen gleicher oder verschiedenartiger zwei– oder drei–dimensionaler Objekte in einen oder mehrere Behälter (z.B. Container, Euro–Paletten) und ihre Parallelisierung. Diese Verfahren sollen zusammen mit Algorithmen zur Wegeminimierung und Plazierung im Lager sowie mit Tourenplanungsalgorithmen in einer integrierten Lageroptimierung vereinigt werden.

3.2.1 Praxisproblem: Packen von Badewannen

In einer industriellen Fallstudie wurde das dreidimensionale Packen von Artikeln mit komplexer Geometrie behandelt. Ein Hersteller von Sanitärartikeln liefert täglich auf etwa 20 Lastzügen über 600 Euro–Paletten mit Bade– und Duschwannen aus. Um eine gute Auslastung der Fahrzeuge zu erreichen, muß zu den den gegebenen Auftragspositionen eine Palettierung gefunden werden, die möglichst wenig Paletten benutzt.

Als Kernproblem identifizierten wir die *homogene Kommissionierung von Wannen*, ein Packproblem, das wir sowohl als *Bin–Packing–Problem* mit nachfolgerabhängigen Gewichten als auch als *Vehicle–Routing–Problem* mit Kapazitätsbeschränkung formuliert haben. Letzteres ist bemerkenswert, da die ursprüngliche Interpretation des Vehicle–Routing–Problems aus einem völlig anderen Anwendungsbereich stammt, der Planung einer optimalen Auslieferungstour z.B. für LKWs.

Aus der betrieblichen Realität ergeben sich mehrere Anforderungen an ein praxistaugliches Packverfahren. So sind bei der Palettierung eine Reihe von Nebenbedingungen (z.B. Gewichts– und Stablitätsrestriktionen) zu beachten. Es muß mit sehr geringem Aufwand möglich sein, diese in den Algorithmus zu integrieren und bei Bedarf zu ändern oder zu ergänzen. Außerdem muß das Verfahren schnell genug sein, um im Dialogbetrieb eingesetzt werden zu können.

3.2.2 Algorithmische Ansätze

Ausgehend von der Formulierung als Bin–Packing–Problem wurde ein neuer, auf Greedy–Heuristiken basierender Algorithmus entwickelt und implementiert. Alternativ dazu erlaubte die Formulierung Vehicle–Routing–Problem den Einsatz des bekannten Savingsverfahrens und des in der Arbeitsgruppe Bachem entwickelten *Simulated Tradings* [7, 6]. Um die Verfahren zu vergleichen, haben wir die Laufzeiten und Palettenzahlen für 13 Real–World–Probleminstanzen ermittelt. Die folgende Tabelle enthält die aus den Einzelergebnissen akkumulierten Werte:

	Greedy	Savings	Sim.Trad.
Anzahl Paletten	198	219	190
Laufzeit	1,7 s	0,65 s	130 s

Alle Verfahren lieferten praktisch verwendbare Lösungen. Die besten Lösungen lieferte das Simulated Trading, wobei die Laufzeit allerdings wesentlich höher lag als beim Greedy– und beim Savings–Verfahren, da hier sehr komplexe Austausch–Operationen vorgenommen werden.

3.2.3 Parallelisierung

Zur Parallelisierung des Problems wurden drei Möglichkeiten untersucht: eine kundenweise Aufspaltung (d.h. die Auftragsbestände verschiedener Kunden werden gleichzeitig auf verschiedenen Rechnerknoten bearbeitet), eine funktionale Dekomposition (also die Zerlegung des Programms in unabhängige Funktionseinheiten, die gleichzeitig abgearbeitet werden können) und das parallele Simulated Trading.

Mit einer Implementierung der kundenweise Aufspaltung wurde auf dem Parsytec GCel–3/1024 des Zentrums für Paralleles Rechnen der Universität zu Köln Real–World–Probleme mit bis zu 8300 Posten effizient bearbeitet. Diese Art der Parallelisierung lieferte schon bei kleinen Problemen gute Ergebnisse, so daß man sie in der Praxis auf jeden Fall einsetzen sollte, falls Performanceprobleme auftreten und insbesondere, wenn die Probleminstanzen größer werden sollten.

Die funktionale Dekomposition erwies sich erst bei Kunden mit sehr großer Postenanzahl als sinnvoll einsetzbar. Sie sollte dann zusätzlich zur kundenweisen Aufteilung verwendet werden.

Im parallelen Simulated Trading werden mehrere Simulated Trading Prozesse parallel auf mehrere Paletten–Cluster angewendet. In diesem speziellen Fall fehlt die geometrische Information der euklidischen "Nähe", die in Tourenplanungsproblemen, für die das Simulated Trading ursprünglich entwickelt wurde, gegeben ist. Dennoch kann dieser Ansatz auch hier angewendet werden, da gewissermassen alle Paletten gleich "nah" zusammen liegen.

Für die Fallstudie „Packen von Badewannen" kann insgesamt festgehalten werden, daß für die momentan aktuellen Praxisprobleme eine Parallelisierung nicht notwendig ist, was sich aber bei der sich abzeichnenden Vergrößerung der Produktpalette in den nächsten Jahren ändern kann.

4 Parallele Methoden

4.1 Parallele Branch and Bound Library

Zur exakten Lösung kombinatorischer Optimierungsprobleme werden im wesentlichen Branch-&Bound-Verfahren eingesetzt. In den letzten Jahren wurde viel Arbeit darauf verwendet, sequentielle Branch-&-Bound-Algorithmen zu parallelisieren. Ein Großteil der Implementierungen waren auf spezielle Anwendungen zugeschnitten, die nur auf einzelnen Parallelrechnerachitekturen mit eigenen Programmierumgebungen liefen.

Im Rahmen von PARALOR wurde daher eine Library entwickelt, die eine einfache Portierung sequentieller Branch-&-Bound-Algorithmen auf beliebige Parallelrechnersysteme ermöglicht. Dabei wird insbesondere beabsichtigt, wesentlich größere Probleminstanzen als mit den zugrundeliegenden sequentiellen Algorithmus bei guter Speed–Up–Werten lösen zu können.

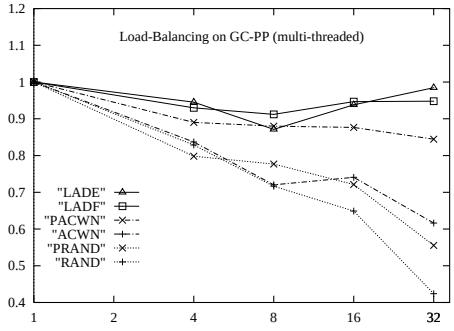
Insgesamt besteht der portierbare parallele Branch-&-Bound Programmrahmen aus folgenden Teilen:

- Eine definierte Anwendungsschnittstelle, die die Unabhängigkeit der Anwendung von Lastverteilung und Message-Passing-Funktionen garantiert.
- Eine definierte Lastverteilungsschnittstelle, mit deren Hilfe die Unabhängigkeit der Lastverteilungsstrategien von den Kernfunktionen des Message-Passing garantiert wird.
- Ein systemabhängiges Kernel, das unter Verwendung von PVM oder Parix Funktionen zum Message-Passing, für Pufferung, Terminierungserkennung sowie Broadcasting zur Verfügung stellt.

- Lokales Queue Managemant mit Funktionen um Stacks, Queues und Priority Queues (Heaps) zu verwalten. Diese sind f—r die verschiedenen Typen von Branch-&-Bound (Depth-first, Breadth-first und Best-first) notwendig.

- Monitoring und I/O Funktionen.

Mit Hilfe der Library wurden verschiedene Anwendungen aus dem Gebiet der kombinatorischen Optimierung parallelisiert: das Set Partitioning Problem, das zweidimensionale Cutting Stock Problem sowie das dreidimensionale Bin-Packing Problem. Für das Set-Partioning Problem, das ein Basisalgorithmus des Fleet Assignment und des Crew Scheduling ist, sei hier exemplarisch die Effizienzmessung bei der Bestimmung der optimalen Lösung unter Verwendung verschiedener Lastverteilungsalgorithmen gezeigt (siehe auch [39]). Es zeigt sich das bei geeigneter Wahl des Lastverteilungsalgorithmus Effizienzen von nahezu 1 erreichen lassen.



Die Implementierungen aller Anwendungen basieren auf effizienten sequentiellen Algorithmen und benutzen jeweils nur Funktionen der definierten Anwendungsschnittstelle, die aus sequentieller Sicht im wesentlichen nur das Queue Management zur Verfügung stellt. Daher benötigt ein Anwender, der sein sequentielles Programm auf einem Parallelrechner starten möchte, im Prinzip weder Wissen über die Hardwarearchitektur noch über die Lastverteilungsstrategie.

Entscheidend für die Performance des Algorithmus ist im wesentlichen die Lastverteilungsstrategie. Hier kann der Benutzer unter verschiedenen Verfahren auswählen, sowie die Verfahren derart parametrisieren, daß die speziellen Bedürfnisse einer Anwendung optimal reflektiert werden.

Unsere Parallele Branch-&-Bound-Programmierumgebung ist in C geschrieben und so entworfen, da sie auf jeder Distributed Memory Multicomputer (DMM) Architektur läuft. Lediglich im Kernel müssen die Message Passing Funktionen auf das jeweilige System portiert werden. Alle anderen Teile, wie die Lastverteilungsmethoden und die Branch-&-Bound-Anwendungen sind systemunabhängig. Getestet wurde die Programmierumgebung auf dem Parsytec GCPP-128 unter PARIX und einem Workstation-Cluster unter PVM.

4.2 Paralleles Simulated Trading

Hauptanwendungsgebiet des Simulated Trading ist sicherlich die Tourenplanung im Straßen– und Flugverkehr, es kann aber auch auf andere, dem Vehicle–Routing–Problem verwandte

Praxisprobleme wie die oben erwähnte homogene Kommissionierung von Wannen, *dial–a–ride* aus dem Speditionswesen oder das *capacitated tree problem* aus dem VLSI–Design angewandt werden. Insbesondere ist Simulated Trading ein paralleles Verfahren, in dem die für die adäquate Behandlung von Real–World–Problemen sehr wichtigen Nebenbedingungen wie Zeitfenster, mehrere Depots, Tour– und Kapazitätsbeschränkungen sehr leicht integriert werden können.

Ausgangspunkt der Optimierung ist ein Starttourenplan, der zufällig oder durch eine schnelle Heuristik erzeugt wird. Um den aktuellen Tourenplan zu verbessern, werden im Simulated-Trading-Verfahren komplexe Kundenaustauschvorgänge zwischen den Touren durchgeführt. Die grundlegende Idee ist die Simulation einer Warenbörse, an der die einzelnen Touren als Händler auftreten; sie können Kunden „kaufen "oder „verkaufen ". Die Kauf- und Verkaufsangebote und ihre Beziehungen werden in einem gewichteten Graphen, dem sogenannten *Trading–Graphen* abgebildet. Ein zulässiger Kundenaustausch entspricht dann einem Matching im Trading-Graphen unter Nebenbedingungen; einer Verbesserung des Tourenplans entspricht ein Matching mit positivem Gewicht.

Die Entwicklung neuer Clusterungsstrategien führte zu einer Variante des Verfahrens, deren deutlich gesenkter Bedarf an Kommunikationsleistung auch den effizienten Einsatz auf Workstation–Clustern erlaubt. Jedem Rechenknoten werden mehrere Touren sowie eine lokale Börse zugeordnet, an der versucht wird, günstige Austäusche von Kunden zu vermitteln. Dieses Mapping ist allerdings nicht fest, sondern wird von einer übergeordneten Instanz – dem *Tourpartitioner* – dynamisch verändert. Dabei ist es sinnvoll, eng aneinanderliegende Touren auf einem Prozessor zu bearbeiten, weil die Wahrscheinlichkeit eines gewinnbringenden Kundenaustausches zwischen diesen Touren am größten ist.

Intensive Testläufe haben gezeigt, daß Simulated Trading bei fast allen Testproblemen bessere Ergebnisse liefert als andere aus der Literatur bekannte Algorithmen. Auch bei Real–World–Problemen eines Fuhrparks mit über hundert LKWs konnten die bisherigen Lösungen, die mit einer Variante des Saving-Verfahrens erzeugt wurden, um etwa 10% verbessert werden.
In der Fallstudie „Packen von Badewannen“ erzielte Simulated Trading im Schnitt um 15% bessere Ergebnisse als das Savingsverfahren und um 4% bessere Ergebnisse als der speziell für das Problem entworfene Bin–Packing–Algorithmus.

Literatur

[1] *Explanatory note covering the joint aviation authorities flight and duty time limitations and rest requirements proposals,* Flight Time Limitations Study Group (FTLSG) of the Joint Aviation Authorities Operations Committee, 1995.

[2] *Zweite Durchführungsverordnung zur Betriebsordnung für Luftfahrtgerät (2. DV LuftBO),* Luftfahrt-Bundesamt, 2/94.

[3] *Manteltarifvertrag Nr. 4 für das Bordpersonal,* Deutsche Angestellten-Gewerkschaft, Hamburg, 1992.

[4] Abrara J., *Applying integer linear programming to the fleet assigment problem,* Interfaces 19, 20-38, 1989.

[5] Anbil, R., Gelman, E., Patty, B., Tanga, R., *Recent advances in crew-pairing optimization at American Airlines,* Interfaces 21, 62-74, 1991.

[6] Bachem, A., Hochstättler, W. und Malich, M., *Simulated Trading – Eine kurze Einführung,* in PARS Mitteilungen 11 in Parallele–Algorithmen, Rechnerstrukturen und Systemsoftware, 81-87, 1993.

[7] Bachem, A., Hochstättler, W. und Malich, M., *Simulated Trading – A New Parallel Approach for Solving Vehicle Routing Problems,* in G. R. Joubert and D. Trystram and F. J. Peters and D. J. Evans (eds.): Parallel Computing: Trends and Applications, 471–475, 1994.

[8] Baker, E., Bodin, L., Finnegan, W., Ponder, R., *Efficient heuristic solutions to an airline crew scheduling problem,* AIIE Trans. 11, 79-85, 1979.

[9] Bixby, R.E., Gregory, J.W., Lustig, I.J., Marsten, R.E., Shanno, D.F., *Very large-scale linear programming: a case study in combining interior point and simplex methods,* Oper. Res. 40, 885-897, 1992.

[10] Boesch, F.T., Gimpel, J.F., *Covering the points of a digraph with point-disjoint paths and its application to code optimization,* J. ACM 24, 192-198, 1977.

[11] Carraresi, P., Gallo, G., *Network models for vehicle and crew scheduling*, Europ. J. Oper. Res. 16, 139-151, 1984.

[12] G. Clarke, J.W. Wright, *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points*, Oper. Res. 12, 568-581, 1964.

[13] Desrosiers, J., Dumas, Y., Desrochers, M., Soumis, F., Sanso, B., Trudeau, P., *Exiting airline crew scheduling results produced using GENCOL,* Paper presented at the TIMS/ORSA Joint National Meeting, Nashville, May 1991.

[14] Even, S., Itai, A., und Shamir, A., *On the complexity of timetable and multicommodity flow problems,* SIAM Journal of Computing, 5, 691-703, 1976.

[15] Fulkerson, D.R., *Note on Dilworth's decomposition theorem for partially ordered sets,* Proc. AMS 7, 78-85, 1958.

[16] Gendron, B. und Crainic, T.G.,*Parallel branch-and-bound algorithms: survey and synthesis*, Operations Research 42, 1042-1066, 1994.

[17] Gershkoff, I., *Optimizing flight crew schedules*, Interfaces 19 (4), 29-43, 1989.

[18] Glover, F., *Tabu Search, Part 1*, ORSA Journal on Computing, Vol. 1, Nr. 3, (1989), 190-206.

[19] Glover, F., *Tabu Search, Part 2*, ORSA Journal on Computing, Vol. 2, Nr. 1, 4-32, 1990.

[20] Goldberg, A. V. , *Parallel algorithms for network flow problems,* in *John H. Reif (Ed.), Synthesis of Parallel Algorithms.* Morgan Kaufmann, 1993.

[21] Graves, G., McBride, R., Gershkoff, I., Anderson, D., Mahidhara, D., *Flight crew scheduling*, Management Science 39, 736-745, 1993.

[22] Gu, Z., Johnson, E. L., Nemhauser, G. L., und Wang, Y., *Some properties of the fleet assignment problem*, Operations Research Letters, 15, 1994.

[23] Hoffman, K., Padberg, M., *Solving airline crew scheduling problems by branch-and-cut*, Management Science 39, 657-682, 1993.

[24] Kirkpatrik, S., Gelatt, C.D., Vecchi, M.P., *Optimization by Simulated Annealing*, Science 220, 671-680, 1983.

[25] Lavoie, S., Minoux, M., Odier, E., *A new approach for crew pairing problems by column generation with an application to air transportation,* Europ. J. Oper. Res. 35, 45-58, 1988.

[26] Lüling, R. und Monien, B.*Load Balancing for distributed branch and bound algorithms*, Proceedings of 6th International Parallel Processing Symposium, 543-548, 1992.

[27] Osman, I.H., *Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem*, Annals of Oper. Res. 41 421-451, 1993.

[28] Radicke, U., *Algorithmen für das Fleet Assignment von Flugplänen: Optimierung auf Marktmodellbasis*, Verlag Shaker, Aachen, 1994.

[29] Radicke, U. D., *Algorithmen für das Fleet Assigment von Flugplänen: Optimierung auf Marktmodellbasis*, Dissertation, Universität zu Köln, 1993.

[30] Rubin, J. *A technique for the solution of massive set covering problems, with application to airline crew scheduling,* Transport. Sci. 2, 34-48, 1973.

[31] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., *Object-oriented modeling and design,* Prentice-Hall Intern., Englewood Cliffs, 1991.

[32] Solomon, M.M., *Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints*, Oper. Res. 35(2), 254-265, 1987.

[33] Suhl, L., *Computer-aided scheduling - an airline perspective,* Gabler Edition Wissenschaft, 1994.

[34] Tschöke,, S., Lüling, R. and Monien, B.,*Solving the traveling salesman problem with a distributed branch-and-bound algorithm on a 1024 processor network*, Proceedings of International Parallel Processing Symposium, 1995.

[35] Tschöke, S., and Polzer, T.,*Portable parallel branch-and-bound library: User manual*, Technical report, University of Paderborn, 1995.

[36] Tschöke, S., and Holthöfer, N.,*A New Parallel Approach to the Constrained Two-Dimensional Cutting Stock Problem*, Proceedings of IRREGULAR 95, Springer LNCS, 1995.

[37] Vazirani, V. V., *Parallel graph matching,* in Synthesis of Parallel Algorithms Morgani, Hrsg. Reif, R.H., Morgan Kaufmann Publ., San Mateo CA, 783-811, 1993.

[38] Xu, C.-Z., Lüling, R., Monien, B., and Lau, F. C. M., *An analytical comparison of nearest neighbor algorithms for load balancing in parallel computers*, Proceedings of 9th International Parallel Processing Symposium, 1995.

[39] Xu, C.-Z., Tschöke, S. and Monien, B., *Performance Evaluation of Load Distribution Strategies in Parallel Branch and Bound Computations*, Proceedings of IEEE Symposium on Parallel and Distributed Processing (SPDP), 1995.