Angewandte Mathematik und Informatik
Universität zu Köln

Report No. 97-268

Verbundprojekt PARALOR:
Parallele Algorithmen zur Wegeoptimierung in Flugplanung und Logistik

by

A. Bachem, M. Bodmann, G. Bolz, T. Emden-Weinert,
A. Erdmann, M. Kiahaschemi, B. Monien, H.J. Prömel,
J. Schepers, R. Schrader, J. Schulze, S. Tschöke

1997

Beitrag zur Statustagung des BMBF
HPSC 97
„ Paralleles Höchstleistungsrechnen
und seine 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 zur Wegeoptimierung in Flugplanung und Logistik

A. Bachem, M. Bodmann, G. Bolz, T. Emden-Weinert,
A. Erdmann, M. Kiahaschemi, B. Monien, H.J. Prömel,
J. Schepers, R. Schrader, J. Schulze, S. Tschöke

20. März 1997

Zusammenfassung

Die Lösung kombinatorischer Optimierungsprobleme ist in vielen Bereichen von Wirtschaft und Technik der Schlüssel zur Steigerung der Effizienz technischer Abläufen, zur Verbesserung der Produktqualität und zur Veringerung von Produktions–, Material– und Transportkosten. Der Einsatz herkömmlicher sequentieller Verfahren ist für praxisrelevante Probleme aufgrund der enormen Rechenzeiterfordernisse nur sehr eingeschränkt möglich. Parallele Systeme bieten eine Möglichkeit, derartige Probleme in vertretbarer Zeit zu lösen. Im Rahmen des Verbundprojektes PARALOR wird untersucht, wie parallele Algorithmen der kombinatorischen Optimierung in konkreten, industriellen Anwendungen aus der Flugplanung sowie der Speditionslogistik effizient eingesetzt werden können. In diesem Artikel werden wesentliche Ergebnisse des Projekts exemplarisch vorgestellt.

1 Einleitung

Die fortschreitende Globalisierung der Märkte zwingt international tätige Unternehmen in immer stärkerem Maße zur Reduzierung von Produktions-, Material- und Transportkosten. In diesem Zusammenhang kommt der Entwicklung effizienter Verfahren zur Lösung großer kombinatorischer Optimierungsprobleme eine besondere Bedeutung zu. Der Einsatz herkömmlicher sequentieller Verfahren ist für praxisrelevante Probleme, die in der Regel durch eine komplexe Zielfunktion und vielfältige Nebenbedingungen charakterisiert sind, aufgrund der enormen Rechenzeitanforderungen nur eingeschränkt möglich. Parallele Systeme bieten eine Möglichkeit, derartige Problemstellungen in vertretbarer Zeit zu lösen. Im Rahmen des Verbundprojektes PARALOR wird untersucht, wie parallele Algorithmen der kombinatorischen Optimierung in Anwendungen aus der industriellen Praxis effizient eingesetzt werden können. Ein zentrales Problem bei der Lufthansa Systems ist

die Ertragsoptimierung bei der Flugplanung, die mit schärfer werdendem internationalen

Wettbewerb unter den Luftfahrtgesellschaften eine immer größere Rolle einnimmt. Im PARALOR Projekt wird speziell die Erstellung von ökonomischen Einsatzplänen für die Flugzeuge *(Fleet Assignment)* und für das Flugpersonal *(Crew Management)* behandelt. Bereits die Überprüfung, ob ein gegebener Flugplan mit dem vorhandenen Material und den Beschäftigten realisiert werden kann, erfordert ein hohes Maß an Erfahrung und Rechnerpotential. Im Rahmen des Projektes werden Flugpläne mit einer Größe von bis zu 10.000 Legs, 300 Flugzeugen, 40 Flugzeugtypen, 200 Airports und 400.000 möglichen Reiseverbindungen betrachtet; die Personalstärken betragen 3.000 für die Cockpitbesatzungen und 8.000 für die Kabinenbesatzungen. Profi.L ist eine Unternehmensberatung, deren Schwer-

punkt im Bereich Transport und Logistik liegt. Von besonderer wirtschaftlicher Bedeutung ist hier die Optimierung der Touren- und Speditionsplanung bei gleichzeitiger Integration effizienter dreidimensionaler Packverfahren. In dem zugrunde liegenden mathematischen Modell wird das bekannte *Vehicle-Routing–Problem* – die Bestimmung kostenoptimaler Touren, auf denen Kunden ausgehend von einem zentralen Depot beliefert werden – mit Verpackungs- oder Reihenfolgeplanungsproblemen gekoppelt. Derart kombinierte Optimierungsprobleme treten sowohl bei der Steuerung von Kommissionierungswagen zur Auftragszusammenstellung im Fertigungslager (interne Logistik) als auch bei Speditionsunternehmen zur Verbesserung der Ausnutzung von Transportketten (Stauraumoptimierung) auf. In den Abschnitten 2 und 3 werden die im Rahmen der Flugplanoptimierung und

der integrierten Steuerung von Fertigungslagern betrachteten Problemstellungen detailliert vorgestellt sowie die erzielten Resultate präsentiert. Die in Abschnitt 4 beschriebenen parallelen Algorithmen werden als Bausteine in verschiedenen Anwendungen sowohl aus dem Bereich Flugplanoptimierung als auch aus dem Bereich Logistik benutzt. Es handelt sich hierbei um einen effizienten parallelen *Cholesky Löser*, eine parallele *Branch-and-Bound Library* und eine parallele *Simulated Annealing Library*.

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 vier- bis achtköpfigen Crew begleitet. Das gesamte Flugpersonal umfaßt etwa 11.000 Personen. Produktplanung und Ressourcenallokation einer Fluggesellschaft sind mithin Teil eines komplexen Optimierungsprozesses, der sich in mehrere Phasen untergliedert (vergleiche auch [30]). Am Anfang steht die Angebotsbestimmung

oder Netzwerkplanung, bei der festgelegt wird, welche Flüge *(Legs)*, d.h. non-stop Verbindungen zwischen zwei Flughäfen zu einer bestimmten Zeit, angeboten werden sollen. Die Bestimmung dieses *Flight Schedules* geschieht mit Hilfe des *Marktmodells*, einem probabilistischem Modell zur Nachfrage- und Angebotsprognose. Das Marktmodell liefert zu jeder Reiseroute *(Itinerary)* den Erlös, den ein Kunde einbringen wird. Als nächstes wird

bestimmt, welcher Flugzeugtyp *(Equipment)* bzw. welches konkrete Flugzeug *(Tail)* einen Flug bedienen soll. Die Flugzeuge unterscheiden sich nicht nur in der Kapazität, also der

Anzahl der Passagiere, die ein Flugzeug befördern kann, oder der maximalen Reichweite, sondern auch in den Kosten, die ein Flugzeug pro geflogenem Kilometer und befördertem Passagier verursacht. Ziel des sogenannten *Fleet Assignments* ist es, einen Einsatzplan für die Flugzeugflotte zu bestimmen, der bei möglichst geringen Kosten möglichst hohe Erträge erwirtschaftet. Beim *Crew Scheduling* schließlich wird den Legs das erforderliche

Crewpersonal zugeordnet, und es werden die Dienstpläne des fliegenden Personals erstellt. Es ist klar, daß es hier eine Rückkopplung vom Fleet Assignment und Crew Scheduling zur Netzwerkplanung gibt: wenn sich nämlich herausstellt, daß ein Flight Schedule nur teuer oder u.U. unter Verwendung der vorhandenen Ressourcen gar nicht zu realisieren ist, muß der Prozeß iteriert werden.

2.1 Fleet Assignment

Eine graphentheoretische Modellierung des Fleet-Assignment–Problems wird durch einen gerichteten Graphen $D = (V, A)$ gegeben, wobei die Knotenmenge V aus den einzelnen Legs gebildet wird. Zwei Knoten $u, v \in V$ sind durch eine Kante 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. Eine solche

Zuweisung kann wie folgt berechnet werden: Die betrachtete Flottenmenge wird sukzessive in jeweils zwei Untermengen aufgeteilt. Ein Fluß durch den Graphen legt fest, welche Flugverbindung von einer der beiden Mengen bedient wird. Auf den Reiserouten werden jetzt – graphentheoretisch – Güter, die einzelnen Passagiere, möglichst ertragsreich durch den Graphen D transportiert, so daß für jedes Gut in jedem Knoten die Flußerhaltung gewährleistet bleibt und die Kapazitäten der den Legs zugewiesenen Flugzeugtypen nicht überschritten wird. Durch die Flußerhaltung ist gesichert, daß bei Umsteigevorgängen an einem Flughafen alle ankommenden Passagiere weiterbefördert werden können. 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 [24], *Simulated Annealing*, oder *Genetischen Algorithmen* mit einigen Stunden Rechenzeit weiter optimiert. Die zu einer Lösung benachbarte Lösungsmenge ist in unserem Simulated Annealing Ansatz wie folgt definiert: Zwei Flugpläne sind benachbart, wenn sie sich auf genau einem Kreis (also einer zyklischen Folge von Legs) in dem darauf eingesetzten Flugzeugtyp unterscheiden. Dieses Vorgehen gewährleistet zu jedem Zeitpunkt die Flußerhaltung. Mit diesem Verfahren erreichen wir Verbesserungen von über 20 % gegenüber der Startlösung (siehe hierzu auch Abschnitt 2.1.1). Bei den Genetischen

Algorithmen [16] handelt es sich um ein aus der Natur abgeleitetes Verfahren, das auf dem Prinzip „survival of the fittest“ beruht. Hierbei wird nicht nur ein einziger Flugplan, sondern ein ganzer „Pool“ *(Population)* von Flugplänen betrachtet. Empirische Resultate zeigen ein ähnlich gutes Verhalten wie für das Simulated Annealing.

2.1.1 Parallelisierung

Zur Parallelisierung der Simulated-Annealing–Heuristik verfolgen wir verschiedene Ansätze. Auffällig in der sequentiellen Version ist die große Anzahl nicht akzeptierter Austauschschritte. Daher verteilen wir die Nachbarschaftssuche auf die vorhandenen Prozessoren. Ein ausgezeichneter Prozessor *(Master)* widmet sich ausschließlich dem Sammeln der Austauschvorschläge aller anderen Prozessoren *(Slaves)*. Gibt es keinen Vorschlag, wird mit einer erneuten Suche begonnen. Andernfalls bestimmt der Master einen Austauschschritt und teilt diesen allen Prozessoren mit *(Global Update)*. Dadurch wird gewährleistet, daß alle Prozessoren auf derselben Lösung operieren. Es zeigt sich, daß bei diesem Verfahren die Kommunikation des Master-Prozesses einen Engpaß darstellt. Ein anderes Konzept, das wir verfolgen, zeichnet keinen Prozessor zum Master aus: Alle Prozessoren betreiben die Nachbarschaftssuche und bestimmen den Prozessor mit der besten gefundenen Lösung. Dieser schickt dann den gefundenen Nachbarn an alle Prozessoren *(Broadcast)*. In einem

weiteren Ansatz verteilen wir die durch die aktuell zugewiesenen Equipments induzierten Subgraphen auf die zur Verfügung stehenden Prozessoren, so daß im Idealfall jedes Equipment von einem einzelnen Prozessor behandelt wird. Genetische Algorithmen sind implizit

parallel [11, 17]. Es können die „genetischen Operationen“ sowie die Berechnung der Erlöse der einzelnen Flugpläne der Bevölkerung verteilt werden. Alle Parallelisierungen werden

unter Verwendung der Message-Passing-Bibliothek MPI realisiert. Abbildung 1 zeigt die Lösungsqualtität in Abhängigkeit von der Laufzeit. Dargestellt sind die Kurven für das parallele Simulated Annealing (ohne dedizierten Master, mit globalem Update) auf einem, zwei, vier, sechs und acht Prozessoren einer SUN Ultra Sparc. Die y-Achse zeigt noch die Zielfunktion mit linearisierten Kosten und Erlösen.

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 vom Typ t. Diese Zahl läßt sich mit Matchingalgorithmen [32] 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 benötigt pro Flugzeugtyp wenige Sekunden Rechenzeit auf einer mittleren Workstation.

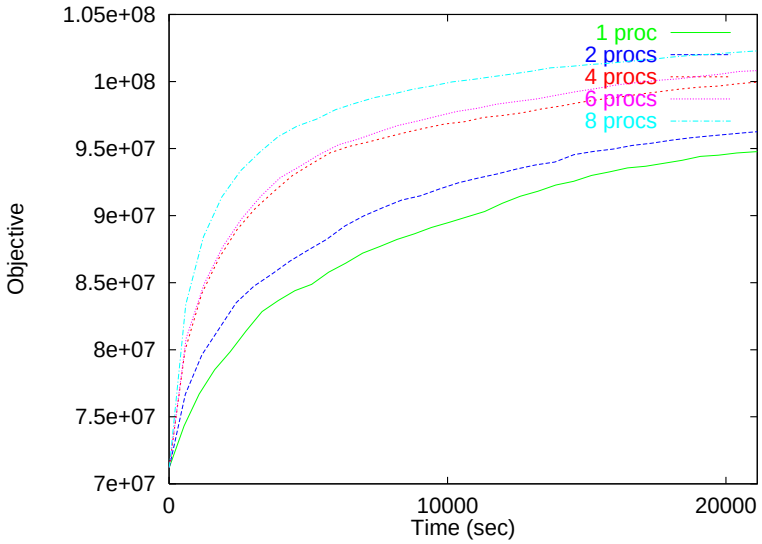

Abbildung 1: Zielfunktion in Abhängigkeit von der Laufzeit

2.2 Crew Management

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. Das Problem besteht darin, zu einem bereits 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 in der Regel ein Monat. Schwierig wird das Problem durch die mannigfaltigen Nebenbedingungen. Während ein Flugzeug – abgesehen von einigen wenigen Checks – nahezu pausenlos fliegen kann, sind beim Crew-Personal eine Reihe von gesetzlichen und tarifrechtlichen Bestimmungen, z.B. zu maximalen Flugdienstzeiten in bestimmten Zeitintervallen oder zu minimalen Ruhezeiten, einzuhalten (hier wurden die wichtigsten europäischen JAR-OPS Regeln [12] implementiert), so daß man mehrere Crews benötigt, um ein Flugzeug zu bedienen. Auch haben die Crews *Homebases*, von denen aus sie ihren Dienst beginnen und wohin sie nach abgeschlossenem Flugdienst wieder zurückkehren. 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 um eine Homebase. In einer zweiten Phase erst, dem *Crew Assignment* oder *Crew Rostering*, werden die Crew-Positionen eines Pairings bestimmten physischen Personen zugewiesen und die Dienstpläne des Flugpersonals erstellt.

Pairing Das Problem der Pairing-Zerlegung kann als klassisches *Set-Partitioning–Problem* formuliert werden, bei dem die Zeilen der Matrix der Menge der in Pairings zu partitionierenden Flugsegmente und die Spalten den zulässigen, gepriceten Pairings entsprechen. Gesucht ist eine günstigste Teilmenge der Pairings, die jedes Flugsegment genau einmal überdecken. Das Set-Partitioning–Problem 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 in vertretbarer Zeit enumerierbar. Ein Lösungsansatz besteht hier darin, die Spalten der Matrix erst während der Optimierung nach und nach zu erzeugen *(Column Generation)*, siehe [21, 8]. American Airlines löst sukzessive handhabbare Teilproblem mit einem *Integer-Programming Ansatz* [14, 1]. Hoffman und Padberg [19] testen Schnittebenen-Methoden für *Branch-and-Cut Löser* des Crew-Scheduling–Problems, *Innere-Punkt-Methoden* werden in [5] verwendet. United Airlines zykelt zwischen einem Pairing Generator und einem Schnittebenen-Integer-Programming-Optimierer unter einem elastischen Set-Partitioning–Ansatz [18]. Da die Qualität des Crew-Assignments letztendlich davon abhängt, wie gut die Pairings später in der Rostering-Phase dem Flugpersonal zuzuweisen sind, wurde hier eine Modellierung gewählt, die die beiden Phasen stärker untereinander verschränkt und Interaktion zwischen beiden Phasen unterstützt. In dieser Modellierung stellt sich das Pairing-Problem eher als ein spezielles *Multi-Depot Vehicle-Routing–Problem* (siehe [9] und Abschnitt 3) dar, bei dem die Flüge den Kunden, die Homebases den Depots, die Crews den Lkw's und die Pairings den Touren entsprechen. (Die Kunden haben hier allerdings feste Lieferzeiten, und die Touren gehen im allgemeinen über mehrere Tage.) Neben der Gesamtarbeitszeit und den Kosten für Hotelübernachtungen stellen *Proceedings*, d.h. Transfers des Crew-Personals zum Einsatzort oder vom Einsatzort zurück zur Homebase, den wichtigsten Kostenfaktor dar. Jedes Pairing fragt als Rundreise um eine Homebase dort Crew-Kapazität nach. Die Crew-Kapazitäten sind in der Regel jedoch ganz unterschiedlich über die verschiedenen Homebases der Fluggesellschaft verteilt, so daß es naheliegt, diese Verteilung schon beim Pairing zu berücksichtigen, um nicht dort Arbeitszeit nachzufragen, wo später kein Crew-Personal mehr vorhanden ist und wohin es erst kostenaufwendig proceedet werden müsste. In unserer Modellierung gehört daher zu jedem Pairing ein Vorschlag, von welcher Homebase aus dieses Pairing zu bedienen ist, um global die Proceeding-Kosten zu minimieren. Die erhaltene Lösung respektiert dann die Crew-Kapazitäten an den Homebases über die Tage des Planungszeitraums als Nebenbedingung, womit sich auch Personalengpässe durch *Preassignments* wie Urlaub, Off-days etc. behandeln lassen. Je nachdem, ob man diese Verteilung berücksichtigt oder nicht, unterscheiden sich die Lösungen zu unseren real-world Test-Instanzen hinsichtlich der Auslastung des Crew-Personals an den Homebases drastisch (siehe dazu Abbildung 2). Die gewählte Lösungsmethode ist ein Simulated-Annealing–Ansatz auf der Menge aller Zerlegungen der Legs in Pairings. Die Nachbarschaft ist dabei definiert durch die Menge aller möglichen *Joins* (d.h. zeitlich und räumlich konsistente Verkettungen zweier Pairings zu einem) und *Splits* (Aufbrechen eines Pairings zu zwei Pairings). Diese Wahl ist insofern natürlich, als Pairings in der Regel kleine Abschnitte von Flugzeugrotationen sind. Um die Größe der Nachbarschaft klein zu halten, ist die Homebase eines Pairings kein freier

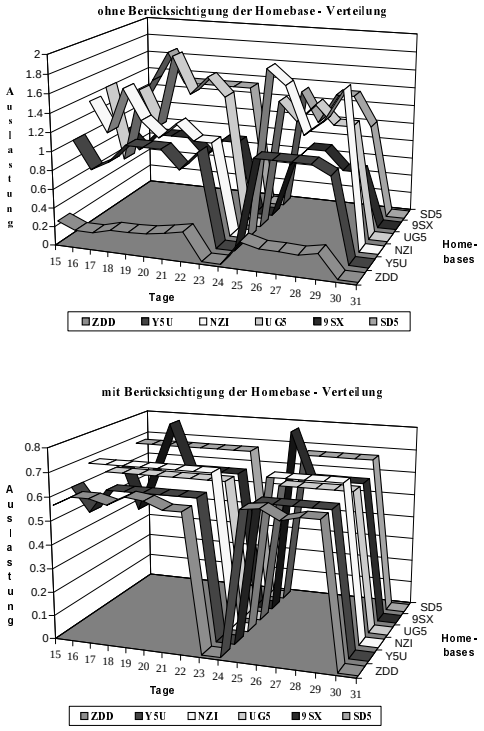

Abbildung 2: Auslastung der Crew-Kapazitäten

Parameter, sondern bei jedem Join oder Split werden für die entstehenden Pairings die lokal optimalen Homebases bestimmt. Wir verwenden einen geometrischen *Cooling-Schedule* (vergleiche [20]). Die erhaltenen Lösungen sind Lösungen aus Konstruktionsheuristiken und durch Verbesserungsheuristiken gewonnenen Lösungen deutlich überlegen. Zur Parallelisierung wird der *Clustering-Ansatz* verwendet (vergleiche Abschnitt 4.3 und [10]).

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.:

- die Dienstpläne der Crew-Mitglieder sollen hinsichtlich Arbeitskapazität (und unliebsamer Dienste) ausgeglichen sein; dies trägt dem besonderen europäischen Modell Rechnung, wo die meisten Crew-Mitglieder dieselben oder ähnliche Arbeitsverträge haben;
- Preassignments im Dienstplan durch freie Tage / Urlaub;
- Crew-Mitglieder haben nur für bestimmte Flugzeugtypen eine Lizenz oder nur für bestimmte Flughäfen eine Landeerlaubnis bzw. nur für bestimmte Länder Visa;
- für jedes Crew-Mitglied sind diverse Bodendienste zu planen wie Stand-By–Dienste, Bürodienste, Training zur Lizenz-Erhaltung, Erste-Hilfe-Kurse, etc.

Durch Berücksichtigung der von der Pairing-Phase vorgeschlagenen Homebases der Pairings verringert sich die Komplexität des Problems. Die Information über die Auslastung der Homebases aus der Pairing-Phase kann zudem dazu verwendet werden, an den kritischen Punkten zuerst zu rostern. Das Problem wird als *Generalized-Assignment–Problem*

mit Nebenbedingungen modelliert (siehe auch [6]). Hier ist eine kostengünstigste Zuordnung der Dienste auf die Crew-Mitglieder gesucht, die Kapazitätsbeschränkungen auf Seiten der Crew-Mitglieder berücksichtigt. Wegen der oben genannten Nebenbedingungen können einem Crew-Mitglied jedoch Dienste nicht unabhängig voneinander zugewiesen werden. Das Problem wird daher iterativ in Phasen gelöst, wobei in jeder Phase die Menge der zulässigen Zuordnungen von Diensten zu Crew-Mitgliedern bestimmt und mit Hilfe eines kostenminimalen Matchings jeweils ein Dienst pro Crew-Mitglied verteilt wird.

Workbench Die in PARALOR entwickelten Algorithmen wurden in einen Prototypen einer Crew-Scheduling-Workbench eingebettet, die Probleminstanz und Lösungen in Form eines *Gantt-Chart* graphisch visualisieren kann und die es erlaubt, Lösungsverfahren und Parameter einzustellen.

3 Integrierte Fertigungslager

Im Bereich der internen Logistik (Lagerorganisation) als auch im Bereich der externen Logistik (Transport) treten komplexe Optimierungsprobleme auf, in denen das bekannte

Vehicle-Routing–Problem mit zwei- und dreidimensionalen Verpackungsproblemen kombiniert wird. So werden bei der Auftragszusammenstellung mehrere Kommissionierungswagen durch ein Fertigungslager gesteuert und mit unterschiedlichen Paketen beladen. Ein anderes Beispiel, das im folgenden ausführlich dargestellt werden soll, ist die effiziente Ausnutzung von Transportkapazitäten bei Speditionen. Im Idealfall ist die geometrische Anordnung der Waren im Laderaum von vornherein festgelegt. So werden viele Güter auf genormten Paletten (z.B. Euro-Paletten) ausgeliefert, die einfach in Doppelreihen hintereinandergestellt werden. In diesem Fall reicht es aus, die Anzahl der Paletten aufzusummieren, um zu entscheiden, ob alle Aufträge einer Tour im LKW Platz finden. Das Vehicle-Routing–Problem wird dann um eine Kapazitätsbeschränkung erweitert. Dieser überschaubare Rahmen wird jedoch bereits verlassen, wenn – was für viele Arten von Gütern der Normalfall ist – die Paletten „überpackt“ werden. Die palettierte Ware ragt dann über die Palettenfläche hinaus, so daß deren Normung ihre problemvereinfachende Wirkung verliert [25]. Häufig sind neben den Paletten auch kartonverpackte Waren zu berücksichtigen, deren Ausmaße stark differieren. Um eine effiziente Ausnutzung der Transportkapazitäten zu erreichen, muß also für den Zulässigkeitstest „Paßt die Tour?“ tatsächlich ein dreidimensionales Packungsproblem gelöst werden. Bisher haben wir nur die Auswirkung der Laderaumkapazitäten auf die Tourenfestlegung betrachtet. In einem realistischen Szenario ist die Kopplung der Probleme Vehicle-Routing und dreidimensionales Packen wechselseitig: Eine Packreihenfolge, bei der vor jedem Ausladen erst eine Reihe blockierender, für spätere Tourkunden bestimmte Teile entnommen und danach wieder eingeladen werden müssen, ist nicht akzeptabel. Bei jedem Kunden müssen alle auszuladenden Waren sofort frei zugänglich sein. Somit definiert die Tourreihenfolge auch eine Nebenbedingung für das Packmuster im Laderaum. In den Abschnitten 3.1 und 3.2 werden wir zunächst die im Rahmen des Projektes entwickelten Verfahren zur Lösung des Vehicle-Routing–Problems und zum Packen zwei- und dreidimensionaler Objekte in einen oder mehrere Behälter (z.B. Container, Euro-Palette) vorstellen. In Abschnitt 3.3 wird schließlich gezeigt, wie die Verfahren zu einem integrierten Ansatz kombiniert werden.

3.1 Tourenplanung

Wie bereits in der Einleitung erwähnt, ist das Ziel des Vehicle-Routing die Bestimmung kostenoptimaler Touren, auf denen eine Menge von Kunden ausgehend von einem zentralen Depot beliefert werden. Für die Auslieferung steht eine bestimmte Anzahl von Lkw's zur Verfügung. Eine wichtige Nebenbedingung ist, daß Kunden nur innerhalb eines Zeitfensters beliefert werden können. Dies ist typisch für die *Just-in-Time Produktion*. Zur Lösung des Vehicle-Routing–Problems wurden im Rahmen des Projektes zwei Ansätze entwickelt, die *Simulated Trading Heuristik* [3] und die *Shift-Heuristik* [27]. Ausgangspunkt der Optimierung ist jeweils ein Starttourenplan, der durch eine schnelle Initialheuristik [29] erzeugt wird. Um den aktuellen Tourenplan zu verbessern werden komplexe Kundenaustauschvorgänge zwischen den Touren durchgeführt. Die grundlegende Idee der Simulated Trading Heuristik ist dabei die Simulation einer Warenbörse, an der die einzelnen Touren als Händler auftreten. Die Shift-Heuristik basiert auf der Idee, kurzzeitig unzulässige

Lösungen zuzulassen und soll im folgenden näher vorgestellt werden. In einer Iteration

der Shift-Heuristik wird für jeden Kunden untersucht, ob es eine Tour gibt, von der aus der Kunde günstiger beliefert werden kann. Falls dies der Fall ist, wird der Kunde dieser Tour zugeordnet. Die so entstandene neue Lösung kann jedoch unzulässig sein, weil Zeitfenster oder Kapazitätsrestriktionen verletzt werden. Durch das Verschieben weiterer Kunden wird dann versucht, die Zulässigkeit wiederherzustellen. Als Ergebnis erhält man in jeder Iteration eine Sequenz von Kundenverschiebungen *(Shift-Sequenz)*, die – eventuell über unzulässige Zwischenlösungen – zu einer zulässigen, benachbarten Lösung führt. Zur Überwindung lokaler Optima ist die Heuristik in einen *Tabu-Search Prozeß* [15] eingebettet. Zur Auffindung guter Tourpläne liegt es nahe, den riesigen Lösungsraum an

mehreren Stellen gleichzeitig zu durchsuchen. Daher arbeitet die Shift-Heuristik auf k Lösungen gleichzeitig, $k \in \{3, \ldots, 8\}$. Die während des Optimierungsprozesses generierten Touren werden alle in einem *Pool* gespeichert. Nach 100 Optimierungsschritten auf allen Lösungen wird eine schnelle *Set-Covering–Heuristik* benutzt, um aus den Touren im Pool einen neuen Tourplan zu konstruieren. Der neue Tourplan ersetzt die schlechteste der k Lösungen. Durch diesen *Rekombinations-Mechanismus* wird erreicht, daß sich die k unabhängigen Suchvorgänge periodisch ergänzen. Tabelle 1 verdeutlicht die Leistungsfähigkeit der Shift-Heuristik im Vergleich zur Initialheuristik (SOL85), zur Simulated Trading Heuristik (BHM96) und zu der kürzlich von Chiang und Russel [7] veroffentlichten Methode. Dargestellt ist die durchschnittliche Tourenlänge und -zahl für die 6 Klassen der *Solomon-Testbench* [29]. Die Shift-Heuristik kann einfach parallelisiert werden, indem die

k Lösungen auf k Prozessoren verteilt werden. Jeder Prozessor rechnet auf einer Lösung und verwaltet einen *lokalen* und einen *globalen* Tourenpool. Der Prozessor generiert aus beiden Pools einen Tourplan und führt – wie im sequentiellen Fall – eine iterative Verbesserung des Tourplans mittles komplexer Shift-Sequenzen durch. Nach jeder Modifikation des Tourplans speichert der Prozessor die entsprechenden Touren im lokalen Pool. Nach 100 Iterationen legt der Prozessor die Touren des iterierten Tourplans in den globalen Pools der anderen Prozessoren ab und generiert sich einen neuen Tourplan. Die Aufteilung in einen globalen und einen lokalen Pool erlaubt (wie im sequentiellen Fall) eine periodische Ergänzung der unabhängigen Suchvorgänge bei minimalem Kommunikations-Overhead.

3.2 Packverfahren

Obwohl es sehr vielfältige Algorithmen für zweidimensionale Packverfahren gibt, ist die Anzahl der Veröffentlichungen über Ansätze zur Lösung von dreidimensionalen Problemen noch äußerst gering. Dreidimensionale Praxisprobleme wurden bisher im allgemeinen auf den zweidimensionalen Fall zurückgeführt oder mittels des einzigen verfügbaren effizienten Verfahrens, des *George-Robinson–Algorithmus* [13], gelöst. Im folgenden stellen wir zwei neuartige Heuristiken vor, die im Rahmen von PARALOR entwickelt wurden [33].

	SOL85		BHM96		CR96		SHIFT	
	Länge	Touren	Länge	Touren	Länge	Touren	Länge	Touren
R1	1695,5	13,6	1391,97	12,58	1444,90	12,50	**1264,24**	**12,25**
C1	1104,2	10,0			929,10	10,00	**829,50**	**10,00**
RC1	1775,0	13,5	1486,73	12,25	1558,70	12,38	**1414,63**	**11,75**
R2	1578,1	3,3	1199,64	3,00	1258,25	2,91	**1100,33**	**2,91**
C2	921,3	3,1			686,10	3,00	**591,88**	**3,00**
RC2	1955,3	3,9	1506,00	3,38	1572,70	3,38	**1341,35**	**3,38**

Tabelle 1: Durchschn. Tourenlänge und -zahl für die 6 Klassen der Solomon Testbench

3.2.1 Teilfolgenheuristik

Die *Teilfolgenheuristik* ist eine neue, layerbasierte Heuristik für dreidimensionale Packungsprobleme. Hierbei werden iterativ Layer generiert, die nur eine Kistenschicht tief sind und deren Kisten möglichst gleiche Tiefe besitzen. Für jeden dieser Layer erhält man nun durch das Lösen eines zweidimensionalen *Knapsack-Problems (KS2D)* eine Packung, die bezüglich zweier Dimensionen optimiert ist. Für KS2D kann auf eine Reihe effizienter Verfahren zurückgegriffen werden. Die möglichst einheitliche Tiefe der Kisten eines Layers sorgt dafür, daß auch die verbleibende Dimension (hinsichtlich der die Layer gebildet wurden) möglichst wenig zum Verschnitt beiträgt. Die Parallelisierung des zweidimensionalen

Packens eines Layers wurde auf zwei Stufen angegangen: Jeder neue Layer wird versuchsweise simultan in drei verschiedenen Orientierungen gepackt, bevor die beste resultierende Packung ausgewählt wird. Für jede Orientierung bearbeitet eine Gruppe von Prozessoren das entsprechende KS2D-Subproblem mit einer neuartigen, parallelen Version des bekannten *Bengtsson-Verfahrens* [4]. Der parallele Algorithmus wurde unter Verwendung von MPI (Message-Passing-Interface) implementiert. Für unsere numerischen Tests benutzten wir drei verschiedene Parallelrechner: einen SparcServer 1000 mit vier Prozessoren (sisy), eine SGI Power Challenge mit 16 Prozessoren (sgi) und einen Parsytec GCel3/1024 (gc) mit 1024 Prozessoren. Der Algorithmus wurde mit Probleminstanzen mit einer Kistenanzahl zwischen 100 und 2500 Kisten getestet. Für hinreichend große Kistenanzahlen ergaben sich dabei Effizienzen von über 50 % auf bis zu 32 Prozessoren. In Abbildung 3 sind die Effizienzen für das größte gerechnete Beispiel mit 2500 Kisten graphisch dargestellt.

3.2.2 Push-Straight–Heuristik

Bei den layerbasierten Packverfahren nimmt man den Nachteil in Kauf, daß in jedem Layer Leervolumina übrigbleiben, die nicht effizient schichtübergreifend zusammengeführt und gemeinsam genutzt werden können. Um dieser Schwierigkeit zu begegnen, entwickelten wir ein Verfahren, das solche Lösungen nachträglich schrittweise verbessert. Dabei handelt es sich nach unserem Wissen um das erste Verbesserungsverfahren für allgemeine dreidimensionale Packungsprobleme überhaupt. Die allgemeine Struktur von Verbesserungsver-

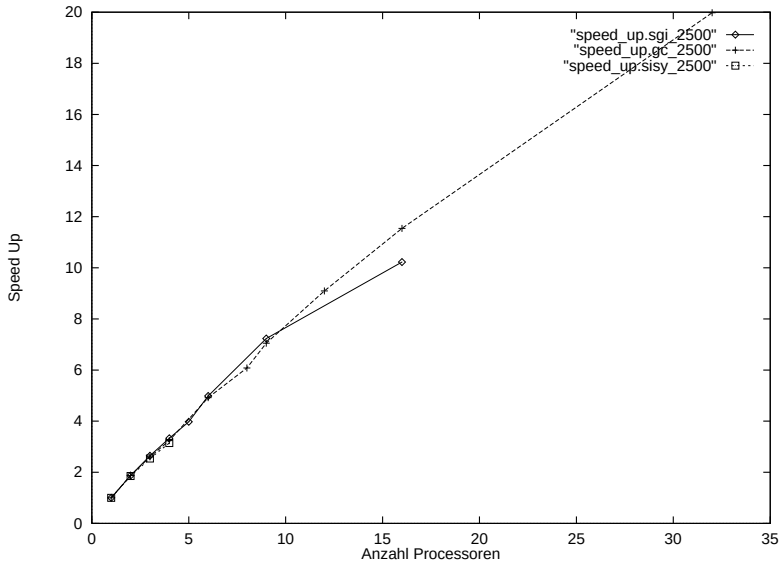

Abbildung 3: Speed Up der Teilfolgen-Heuristik für ein Beispiel mit 2500 Kisten

fahren für andere kombinatorische Optimierungsprobleme, die durch Graphen dargestellt werden können, basiert oft auf der Existenz effizienter Einfüge- und Löschoperationen von einzelnen oder mehreren Kanten oder Knoten. Auch bei Packungsproblemen, die wir durch Sichtbarkeitsgraphen darstellen können, ist dieser Ansatz möglich, indem Kisten aus einem Packmuster gelöscht und andere Kisten an deren Stelle eingefügt werden. Durch geeignete Verschiebungen wird versucht, den nach Entnahme der Kisten entstehenden Leerquader zu vergrößern. Diese Verschiebungen können effizient durchgeführt werden, wenn wir uns auf Folgen von Verschiebungen parallel zu den Koordinatenachsen (push-straight) beschränken. Daher bezeichnen wir das neue Verfahren als *Push-Straight–Verfahren*. Die entsprechenden Einfüge- und Entfernungsoperationen bilden eine Grundlage für weitere Austauschverfahren für Packprobleme analog zu den bekannten Austauschverfahren für TSP oder Vehicle-Routing. Die Parallelisierung dieser Klasse von Algorithmen mit generischen Methoden ist Thema des Abschnitts 4. Tabelle 2 verdeutlicht die Leistungsfähigkeit der neuen Packheuristiken im Vergleich zur George-Robinson–Heuristik. Dargestellt ist der durchschnittliche Verschnitt in % und die Anzahl der zusätzlich gepackten Kisten für 3 Problemtypen mit je 10 Instanzen.

3.3 Stauraumoptimierung

Wie bereits eingangs beschrieben, werden bei der Stauraumoptimierung Tourenplanungs- und Packungsprobleme miteinander verknüpft. Ziel ist die Berechnung einer Auslieferungstour und einer Packreihenfolge, so daß das Ausladen der Kisten bei keinem Kunden ein Umsortieren erfordert. Wesentlich für die Stauraumoptimierung ist neben dem verwendeten Packverfahren die Art, wie Kisten, die zu Aufträgen verschiedener Kunden gehören räumlich voneinander getrennt werden. Diese bezeichnen wir im folgenden als *Kundense-*

	Typ 1	Typ 2	Typ 3
Anzahl Kisten (Maße aus $[10, 80]$)	100	200	300
Containermaß (alle Seiten gleich)	200	250	350
George-Robinson	23,5%	25,5%	29,4%
Teilfolgen-Heuristik	**20,7%**	**16,6%**	**17,3%**
George-Robinson mit Push-Straight	17,1% (10,0)	12,8% (22,6)	14,6% (27,5)
Teilfolgen-Heuristik mit Push-Straight	**14,3% (13,2)**	**10,4% (30,0)**	**12,6% (39,0)**

Tabelle 2: Durchschnittlicher Verschnitt in % (und zusätzlich gepackte Kisten)

parierung. Folgende Arten der Kundenseparierung wurden betrachtet:

Trennwände entlang einer festen Achse In vielen Fällen werden in der Praxis innerhalb des Laderaums Trennwände eingesetzt, um verschiedene Ladepositionen abzutrennen. Somit sind zwei Dimensionen des Quaders, in den die Kisten eines Kunden gepackt werden, durch die entsprechenden Laderaumdimensionen festgelegt. Nur noch die dritte, variable Dimension ist für das Stauraumproblem relevant. Die Berechnung der entsprechenden Größen kann im Preprocessing erfolgen. Für jeden Kunden ist hier ein dreidimensionales *Strip-Packing–Problem* zu lösen. Diese Werte können dann wieder als Kapazitäten für ein kapazitätsbeschränktes Vehicle-Routing–Problem (s.o.) verwendet werden.

Trennwände entlang variabler Achsen Hier werden die Trennwände parallel zu einer ausgewählten Seitenfläche des alle bisher gepackten Kisten umgebenden Quaders gezogen. Diese Variante ist erheblich aufwendiger als die vorangegangene. Sie erfordert bei jeder Neuberechnung einer Laderaumbelegung im Vehicle-Routing Algorithmus die Lösung dreier dreidimensionaler Strip-Packing–Probleme.

Rekombination „angebrochener" Layer Fast alle dreidimensionalen Packverfahren basieren auf einer Dimensionsreduktion durch die Verwendung von Layern, die nur eine Kiste hoch sind. In dieser Art der Kundenseparierung werden die „angebrochenen" Layer zweier benachbarter Kunden kombiniert. Ob die Stauraumbedingung hier erhalten bleibt, hängt von den konkreten praktischen Einsatzbedingungen ab. Wie bei der Separierung durch Trennwände entlang einer festen Achse werden für jeden Kunden „volle" Layer im Preprocessing berechnet. Im Vehicle-Routing–Problem ist dann neben der Überprüfung der Kapazitätsrestriktion für jede Einfügeoperation ein zweidimensionales Knapsack-Problem mit relativ wenigen Kisten zu lösen.

Das kapazitätsoptimierende Tourenplanungsverfahren setzt sich also aus drei Komponenten zusammen: dem Vehicle-Routing–Algorithmus, dem Packverfahren und (wenn mit primalen Heuristiken gepackt wird) der Art der Kundenseparierung. Ein Vergleich der von uns implementierten Verfahren der integrierten Tourenplanung/Stauraumoptimierung untereinander ergab, daß der erhaltene Zielfunktionswert im wesentlichen von der verwendeten VRP-Heuristik abhängt. Vergleicht man für eine gegebene VRP-Heuristik die

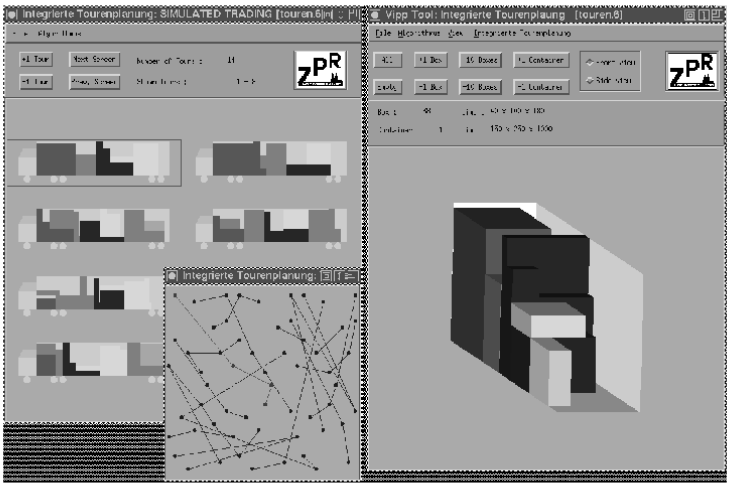

Abbildung 4: Integrierte Stauraumoptimierung/Tourenplanung

verschiedenen Separationsarten, so zeigt sich, daß man mit der Methode der Rekombination angebrochener Layer für die untersuchten Instanzen um 10 bis 20 % bessere Resultate erzielt als mit festen Trennwänden. Die Wahl der Packheuristik hatte keinen erkennbaren Einfluß auf die Ergebnisse. Das ist plausibel, da die auftretenden Packprobleme nicht allzu groß werden. Abbildung 4 zeigt die graphische Ausgabe der integrierten Tourenplanung/Stauraumoptimierung.

4 Parallele Methoden

4.1 Parallele Cholesky-Zerlegung

Für eine effiziente Parallelisierung der Inneren-Punkt-Methode, einem Verfahren zur Lösung linearer Programme, stellt die Entwicklung eines skalierbaren, parallelen Algorithmus zur Lösung eines dünn besetzten, linearen Gleichungssystems $A \cdot x = b$ eine wichtige Voraussetzung dar. Aufgrund der besonderen Eigenschaften der Koeffizientenmatrix A (symmetrisch und positiv definit), kann das Gleichungssystem mit Hilfe einer Cholesky-Zerlegung von A gelöst werden. Hierbei wird eine untere Dreiecksmatrix L bestimmt, so daß gilt $A = L \cdot L^T$. Die Matrix L ist in der Regel sehr viel dichter besetzt als die Ausgangsmatrix

A. Die zusätzlichen Einträge $l_{ij} \neq 0$ heißen *fill-in*. Durch geeignete Zeilen- und Spaltenvertauschungen kann der fill-in minimiert werden. Die Bestimmung einer optimalen Zeilen- und Spaltenpermutation ist jedoch ein NP-vollständiges Problem. Daher muß auf Heuristiken zurückgegriffen werden. Die bekanntesten Heuristiken sind das *Minimum Degree Ordering* und das *Nested Dissection Ordering* Im Laufe des Projektes konnte basierend

	n	$\|A\|$	$\|L\|$	2	4	8	16	32
GRID255X255	65.025	258.572	2.504.361	10,48	5,33	2,9	1,74	1,0
GRID511X511	261.121	1.041.420	12.021.665		49,1	24,7	12,59	6,44
MAT02HBF	46.949	116.476	6.623.665	71,41	35,81	18,12	9,84	5,62
MAT03HBF	73.752	183.547	12.328.847		103,29	52,47	26,64	13,57

caption$n$: Spalten/Zeilen, $|A|$: Einträge $\neq 0$ in A, $|L|$: Einträge $\neq 0$ in L

auf der *Multifrontal-Methode* [23] ein effizienter, skalierbarer, paralleler Algorithmus zur Berechnung von L entwickelt werden. Tabelle 4.1 zeigt die Ausführungszeiten (in sec.) auf einem Parsytec CC System mit 32 Knoten. Bei den ersten beiden Matrizen handelt es sich um regelmäßige Gitterprobleme, die letzten beiden wurden uns von der Firma Inpro zur Verfügung gestellt. Da selbst für große A die Faktormatrix L in wenigen Sekunden bestimmt werden kann, muß auch das Ordering in sehr kurzer Zeit berechenbar sein. Ein Vergleich verschiedener Nested Dissection Verfahren hat gezeigt, daß mit Hilfe von *Multilevel Ansätzen* sehr gute Orderings in kurzer Zeit generiert werden können [28].

4.2 Parallele Branch-and-Bound Library

Zur optimalen Lösung kombinatorischer Optimierungsprobleme werden häufig Branch-and-Bound Verfahren eingesetzt. In der Regel können nur kleine Probleminstanzen optimal gelöst werden. Nur durch die Parallelisierung effizienter sequentieller Branch-and-Bound Algorithmen ist es möglich, auch mittlere bis größere Instanzen optimal zu lösen. Durch die im Rahmen des Projektes entwickelte parallele Branch-and-Bound Library wird eine automatische Portierung beliebiger sequentieller Branch-and-Bound Algorithmen auf verschiedene Parallelrechnersysteme ermöglicht. Entscheidend für die parallele Performance

eines sequentiell effizienten Algorithmus ist im wesentlichen die Verteilung der Last im parallelen System [31]. Hier werden von der Bibliothek verschiedene Lastverteilungsverfahren *(Diffusion, Dimension Exchange)* zur Verfügung gestellt. Unter diesen kann der Benutzer auswählen, oder eigene Strategien entwerfen, so daß die speziellen Bedürfnisse seiner Anwendung optimal reflektiert werden. Die Library dient zur Bestimmung optimaler

Lösungen im Bereich Crew Scheduling, Fleet Assignment und der dreidimensionalen Stauraumoptimierung [26]. Darüberhinaus wird die Library von mehreren externen Anwendern benutzt. So konnte beispielsweise ein an der DIKU Kopenhagen entwickelter, effizienter Branch-and-Bound Algorithmus zur Lösung des *Quadratic-Assignment–Problems* parallelsiert werden. Auf diese Weise war es erstmals möglich, eine Optimallösung von *Nuggent 24* zu bestimmen.

4.3 Parallele Simulated Annealing Library

Das Ziel der parallelen Simulated Annealing Library ist die Bereitstellung einer universellen Schnittstelle zur Anbindung unterschiedlicher Ausprägungen der Simulated Annealing Me-

thode (darunter verschiedene Abkühlungs- und insbesondere Parallelisierungsstrategien) an eine gegebene sequentielle iterative Verbesserungsheuristik. Nur die problemspezifische Funktionalität muß vom Benutzer implementiert werden. Diese Funktionalität umfaßt das Einlesen der Problemdaten, das Erzeugen einer Initiallösung, die Veränderung und die Bewertung der jeweils aktuellen Lösung, sowie die Ausgabe der optimierten Lösung. Die Parallelisierung der Verbesserungsheuristik geschieht ausschließlich auf der Ebene des Simulated Annealing Algorithmus in der Library. Die zur Verfügung stehenden Abkühlungs-

strategien sind vor allem selbstadaptive Verfahren, die aber auch mittels Parametern auf die jeweilige Optimierungsaufgabe exakt angepaßt werden können. Darüberhinaus wurden zwei Parallelisierungsvarianten implementiert:

1. Ein *Clustering Algorithmus* [2], bei dem alle zur Verfügung stehenden Prozessoren dynamisch in Cluster unterteilt werden und jeder Cluster eine Teilkette der Zustände berechnet. Der Zeitpunkt, an dem die Vergrößerung der Anzahl der Prozessoren im Cluster aus Gründen der Effizienz sinnvoll ist, wird problem- und plattformunabhängig anhand von Zeitmessungen und statistischen Voraussagen bestimmt, wodurch eine optimale Anpassung an die Kommunikationsgeschwindigkeit des benutzten Parallelrechners und den Kommunikationsaufwand des jeweiligen Problems ermöglicht wird. Um den Kommunikations-Overhead zusätzlich zu verringern, werden während der Berechnung der Teilkette nur die Veränderungen der aktuellen Lösung übermittelt.
2. Ein Ansatz der *Multiple Independent Runs* [22], bei dem die Suche nach einer guten Lösung in zwei Phasen geschieht. In der ersten Phase werden die Eigenschaften des jeweiligen Problems durch mehrere kurze Annealing-Läufe ermittelt. Daraus wird die notwendige Länge und die Anzahl der voneinander unabhängigen Läufe prognostiziert. Diese werden in der zweiten Phase auf den zur Verfügung stehenden Prozessoren ausgeführt. Anschließend wird die beste gefundene Lösung ausgewählt. Ein Kommunikations-Overhead entsteht nur in der ersten Phase.

Die Library wurde unter Benutzung des MPI-Standards realisiert, wodurch eine hohe Plattformunabhängigkeit gewährleistet ist.

Literatur

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

[2] E. Aarts, F. de Bont, E. Haberts, P. van Laarhoven, *Parallel Implementations of the Statistical Cooling Algorithm*, INTEGRATION, the VLSI journal 4(1986), 209–238.

[3] A. Bachem, W. Hochstättler, M. Malich, *The simulated trading heuristic for solving vehicle routing problems*, Discrete Applied Mathematics, 65(1-3):47–72, 1996.

[4] B.E. Bengtsson, *Packing rectangular pieces – A heuristic approach*, The Computer Journal, 25:353–357, 1982.

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

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

[7] W.C. Chiang, R.A. Russell, *Simmulated annealing metaheuristic for the vehicle routing problem with time windows*, Annals of Operations Research, 63:3–27, 1996.

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

[9] J. Desrosiers, Y. Dumas, M.M. Solomon, F. Soumis, *Time constrained routing and scheduling*, in: Network routing, Hrsg. M.O. Ball et al., Handbooks in OR & MS Vol. 8, Elsevier, Amsterdam (1995), 35–139.

[10] R. Diekmann, R. Lüling, J. Simon, *Problem independent distributed simulated annealing and its applications*, in: Applied simulated annealing, Hrsg. R.V.V. Vidal, Springer LNEMS 396 (1993), 17–44.

[11] M. Dorigo, V. Maniezzo, *Parallel Genetic Algorithms: Theory & Application*, Parallel Genetic Algorithms: Introduction and Overview of Current Research, 1993.

[12] *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.

[13] J.A. George, D.F. Robinson, *A heuristic for packing boxes into a container*, Computers and Operations Research 7, 1980, 147–156.

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

[15] F. Glover, E. Taillard, D. de Werra, *A user's guide to tabu search*, Annals of Operations Research, 41 (1993), 3–28.

[16] D.E. Goldberg, *Genetic Algorithms in search, optimization and machine learning*, Addison-Wesley, 1989.

[17] V.S. Gordon, D. Whitley, *Serial and Parallel Genetic Algorithms as Functions Optimizers*, Proc. 5th International Conference on Genetic Algorithms, 1993.

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

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

[20] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, *Optimization by simulated annealing: an experimental evaluation; Part I: graph partitioning*, Oper. Res. 37 (1989), 865–892.

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

[22] F.H. Lee, *Parallel Simulated Annealing on a Message Passing Multi-Computer*, PhD Thesis, Utah State University, 1995.

[23] J.W.-H. Liu, *The Multifrontal Method for Sparse Matrix Solution: Theory and Practice*, SIAM Review, 34 (1992), 82–109.

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

[25] J. Schepers, M. Wottawa, *Warehouse Optimization: Packing-algorithms for goods with complex geometries*, Proc. of SOR'95, 125–131.

[26] J. Schepers, *An Exact Algorithm for General Orthogonal n-dimensional Knapsack Problems*, Proc. of SOR'96.

[27] J. Schulze, T. Fahle, *A parallel algorithm for the vehicle routing problem with time window constraints*, eingereicht für: Annals of Operations Research, Recent Advances in Combinatorial Optimization: Theory & Practice.

[28] J. Schulze, R. Diekmann, R. Preis, *Comparing Nested Dissection Orderings for Parallel Sparse Matrix Factorization*, Proc. of PDPTA '95, CSREA 96-1103, 280-289.

[29] M.M. Solomon, *Algorithms for the vehicle routing and scheduling problem with time window constraints*, Operations Research, 35 (1987), 254–265.

[30] L. Suhl, *Computer-aided scheduling – an airline perspective*, Gabler Edition Wissenschaft, Wiesbaden, 1995.

[31] S. Tschöke, C. Xu, *Performace Evaluation of Load Distribution Strategies in Parallel Branch and Bound Computations*, Proc. of SPDP '95.

[32] V.V Vazirani, *Parallel Graph Matching*, in J.H. Reif (Ed.), *Synthesis of Parallel Algorithms*, Morgan Kaufmann, 1993.

[33] M. Wottawa, *Struktur und algorithmische Behandlung von praxisorientierten dreidimensionalen Packungsproblemen.* Dissertation, Mathematisches Institut, Universität zu Köln, 1996.