Kombinatorische Optimierungsverfahren für die Flugdienstplanung

DISSERTATION

zur Erlangung des akademischen Grades
DOCTOR RERUM NATURALIUM

eingereicht an der Mathematisch-Naturwissenschaftlichen Fakulät II
Institut für Informatik, Humboldt-Universität zu Berlin

von

Dipl. Math.
THOMAS EMDEN-WEINERT
geboren am 27.11.1965 in Fulda

Präsident der Humboldt-Universität zu Berlin
Prof. Dr. Dr. h.c. Hans Meyer

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II
Prof. Dr. sc. nat. Bodo Krause

Gutachter:
1. Prof. Dr. Hans Jürgen Prömel
2. Prof. Dr. Miroslaw Malek
3. Prof. Dr. Rainer Schrader

Tag der mündlichen Prüfung: 29. Januar 1999

Zusammenfassung

Wir berichten über eine experimentelle Studie zu lokalen Suchverfahren für das Pairing-Problem bei Fluggesellschaften unter Benutzung realer Testdatensätze mit bis zu 4600 Flügen kurzer Reichweite. Unter einer Run-Cutting Formulierung vergleichen wir Verfahren wie Simulated Annealing, Modifikationen der Wasserspiegel-Algorithmen nach DUECK [Due93], eine elementare Implementation von Tabu Suche sowie einen GRASP Ansatz. Es zeigt sich, daß die Wasserspiegel-Algorithmen vergleichbare Ergebnisse wie Simulated Annealing, und um etwa 40% bessere Ergebnisse als bloße lokale Verbesserungsheuristiken liefern, während Tabu Suche und GRASP hier ungeeignet erscheinen. Die Ergebnisse von Simulated Annealing und der Wasserspiegel-Algorithmen können sich auch mit den LP-Schranken für die SET PARTITIONING Formulierung mit Packungsungleichungen und bis zu 3.9 Millionen Spalten messen. Eine Clustering-Parallelisierung des Simulated Annealings auf einem Parallelrechner GigaCluster PowerPlus weist eine sehr gute Beschleunigung auf.

Abstract

We report about an experimental study of local search optimization algorithms for the airline pairing problem using a run-cutting formulation. Computational results are reported for some real-world short-haul testproblems with up to 4600 flights per month. In particular, we evaluate the relative performance of simulated annealing, modified versions of the waterlevel algorithms proposed by DUECK [Due93], an elementary tabu search, and a GRASP approach. We find that the waterlevel algorithms compare well with simulated annealing and both improve on mere improvement heuristics by about 40%. Tabu search and GRASP do not yield competitive results here. Simulated annealing and the waterlevel algorithms also compare favourably with LP-bounds for a SET PARTITIONING formulation with knapsack constraints containing up to 3.9 million columns. A clustering parallelization of simulated annealing on a GigaCluster PowerPlus parallel computer exhibits an excellent speed-up.

Maria Böschen

** 18. 3. 1901*

† 21. 1. 1998

Danksagung

Die vorliegende Arbeit entstand im Rahmen des vom Bundesministeriums für Bildung, Wissenschaft, Forschung und Technologie (BMBF) geförderten Verbundprojekts PARALOR (Parallel Algorithms for Large Scale Operations Research Problems), Förderkennzeichen 01 IR 411 E2. Mein Dank gilt zuvörderst Herrn Professor Prömel, der das Projekt mitinitiiert und geleitet hat, für seine Unterstützung und Ermutigung bis zur Fertigstellung dieser Arbeit.

Ich danke der Lufthansa Systems Berlin GmbH, insbesondere Heike Ripphausen-Lipa und Arfst Ludwig, für die Bereitstellung der Testdaten und die Begutachtung von Lösungen, die das von uns entwickelte System produziert.

Weiter danke ich meinem Diplomanden Mark Proksch für viele interessante Diskussionen, insbesondere zu philosophischen Fragen der Softwareentwicklung, für die wertvolle Unterstützung bei technischen Problemen und für die Durchführung der Testläufe zu den Abbildungen 2.4 und 4.1, Peter Weisse für die Entwicklung der Benutzungsoberfläche des Prototypen. Peter Unruh danke ich für seine Hilfe bei der Anpassung unseres Programmsystems an die Paderborner Parallel-Simulated-Annealing Library und die Durchführung der Rechnungen zu Abbildung 4.1 auf dem dortigen Parallelrechner, Andreas Erdmann vom Zentrum für Paralleles Rechnen, Köln, für die Berechnung der LP-Schranken aus Abschnitt 3 mit CPLEX und nicht zuletzt Eva Sandig für die Anfertigung von Abbildung 1.1. Allen AlCoXlern danke ich für die angenehme Atmosphäre in der Arbeitsgruppe.

Besonders möchte ich an dieser Stelle meinen Eltern und Großeltern danken, die mir das Studium der Mathematik erst ermöglicht und mich in allen Phasen mit Anteilnahme und Zuspruch unterstützt haben, sowie meinem Bruder, der mich stets mit den letzten Neuigkeiten aus den Humour-Newsgroups bei Laune hielt und mir zusätzlichen Ansporn gab, als er nicht mehr an die Fertigstellung dieser Arbeit glaubte. Christian Auth und Corinna Berg hielten mich an, die Dinge immer wieder aus einer anderen Perspektive zu sehen und gewährten mir das eine oder andere Wochenende behaglichen Unterschlupf in Berlin, als ich schon bei der VSS in Bremen arbeitete.

Thomas Emden-Weinert
Berlin, 19. April 1998

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Einleitung

Die Globalisierung der Märkte und der zunehmende Wettbewerb zwingen Unternehmen in immer stärkerem Maße, personelle und materielle Ressourcen effizient einzusetzen und Kosten für Logistik und Transport bei gleichbleibenden Leistungsmerkmalen zu reduzieren. In der Folge haben sich die Tourenplanung im Transportwesen (das sogenannte Vehicle-Routing, siehe z.B. [DP95, Fis95, GLP97]) und die Dienstplanung für fliegendes Personal bei Fluggesellschaften bzw. für das Fahrpersonal im öffentlichen Nah- und Fernverkehr (das sogenannte Crew-Scheduling, siehe z.B. [RHP95, DDSS95, DDGS97, AHKW97, CFG+97, WR95]) zu prominenten Anwendungsfeldern der kombinatorischen oder diskreten Optimierung entwickelt. Wo Unternehmensprozesse und -daten auf den Rechner abgebildet wurden, können mathematische Optimierungsverfahren eine ökonomische Lösung für komplexe und aufgrund ihrer Größe für den Menschen kaum überschaubare Probleme finden. Die kurzfristige Verfügbarkeit von optimierten Lösungen eröffnet darüber hinaus die Möglichkeit von Fallstudien und Simulationen, mit deren Hilfe sich die Planung flexibel an veränderte Randbedingungen anpassen und die Prozeßkette, in die das Einzelproblem eingebettet ist, optimieren läßt.

Wir beschäftigen uns hier mit der Dienstplanung für fliegendes Personal bei Fluggesellschaften. Da die Aufwendungen für Crew-Personal sehr hoch sind, schlagen sich bereits geringfügige Verbesserungen bei der Planung in großen Kostenersparnissen nieder. Typisch für das Problem ist die große Anzahl von Nebenbedingungen, die sich aus gesetzlichen und tarifrechtlichen Bestimmungen und betrieblichen Vereinbarungen ergeben, sowie eine aus etlichen, einander widerstrebenden Kostenarten zusammengesetzte Zielfunktion.

Wie auch im Schienenverkehr wird das Problem traditionell in zwei Phasen gelöst. Zunächst werden die Dienstelemente, d.h. die Bedienung einzelner Flüge, zu größeren Diensten, den Pairings, die meist Rundreisen über wenige Tage darstellen, gruppiert. Diese zunächst noch anonymen Pairings werden in der Rostering-Phase dem vorhandenen Crew-Personal zugewiesen, wodurch sich die Dienstpläne der Crew-Mitglieder ergeben.

Wir untersuchen das Pairing-Problem. In der Literatur wird das Pairing-Problem, abgesehen von frühen graphentheoretischen Heuristiken, zumeist als SET COVERING oder als SET PARTITIONING Problem modelliert. Hier wird zunächst eine große Zahl günstig erscheinender Pairings erzeugt und sodann aus diesen eine gute oder gar optimale Teilmenge ausgewählt, die jeden Flug überdeckt. Wir entwickeln eine graphentheoretische Formulierung, die darauf abzielt, daß Pairings in der Regel aus Abschnitten von Flugzeug-Umläufen aufgebaut sind. Anhand realer Testdaten von mittelgroßen europäischen Fluglinien mit bis zu 4600 (Kurzstrecken-) Flügen im Monat untersuchen und vergleichen wir lokale Optimierungsverfahren (oder Metaheuristiken) wie Simulated Annealing und Varianten der Wasserspiegel-Algorithmen nach DUECK [Due93] an Hand des Problems der Flugdienstplanung. Lokale Optimierungsverfahren gehen das Pairing-Problem in einem Schritt an (ohne Unterteilung in eine Pairing-Generierungsphase und ein SET PARTITIONING), und jedes zulässige Pairing kann in der Lösung enthalten sein. Seit ihrer Erfindung 1983 haben Metaheuristiken eine große Verbreitung gefunden und sich als robuste und effektive Verfahren für Probleme des Operations Research erwiesen, für die keine problemspezifischen Algorithmen bekannt sind.

Gliederung. Die vorliegende Arbeit ist wie folgt aufgebaut. Kapitel 1 führt in das Problem Flugdienstplanung ein und beschreibt die hier verwendete „Run-Cutting“-Modellierung des Pairing-Problems. Die von uns verwendeten Verfahren der lokalen Suche sowie die Ergebnisse der vergleichenden Studie werden in Kapitel 2 vorgestellt. In Kapitel 3 vergleichen wir unsere Ergebnisse mit LP-Schranken für die SET PARTITIONING Formulierung mit Packungsungleichungen. Kapitel 4 berichtet über eine Clustering-Parallelisierung des Simulated-Annealing-Verfahrens auf einem Parallelrechner unter Benutzung der Paderborner Parallel Simulated Annealing Toolbox sowie über die Methode der voneinander unabhängigen Simulated-Annealing-Läufe. Kapitel 5 schließt mit einer Zusammenfassung der Ergebnisse und einem Ausblick. Anhang A beschreibt kurz die graphische Benutzungsoberfläche des entstandenen prototypischen Optimierungssystems.

Notation und Terminologie. Für Begriffe aus der Graphentheorie oder der Komplexitätstheorie, die hier nicht definiert werden, verweisen wir auf DIESTEL [Die97] bzw. PAPADIMITRIOU [Pap94].

Dokumentation der Meßergebnisse. Das dem Prototypen zugrundeliegende objekt-orientierte Datenmodell wurde mit Hilfe der Object-Modeling-Technique (siehe RUMBAUGH et al. [RBP+91]) entworfen. Das System wurde in der Programmiersprache C++ entwickelt, der Quellcode mit dem GNU-Projekt C++-Compiler, Version 2.7.2, ohne Compileroptionen übersetzt. Alle Laufzeiten beziehen sich auf eine SUN ULTRA SPARC 1 mit 167 MHz Taktfrequenz und 160 MB Hauptspeicher. Die verwendeten Algorithmen sind randomisiert; um aussagefähige Grafiken zur Beurteilung der Algorithmen zu erhalten, repräsentieren daher die einer Kurve zugrundeliegenden Meßpunkte – soweit nicht anders verzeichnet – die Ergebnisse von mindestens 15 Testläufen mit identischer Parametereinstellung.

Kapitel 1

Das Crew-Scheduling-Problem

1.1 Flugplanung

Eine große Fluggesellschaft, wie z.B. die Deutsche Lufthansa, bietet wöchentlich etwa 10.000 Flüge an. Diese werden mit ca. 250 Flugzeugen und 11.000 Crew-Mitgliedern geflogen. Angebotsplanung und Ressourcenallokation stellen bei diesen Größenordungen komplexe Aufgaben dar. Der Flugplanungsprozeß wird daher typischerweise in Stufen gegliedert (vergleiche z.B. Suhl [Suh95]).

- Die Netzwerkplanung beschäftigt sich mit der Frage, welche (non-stop) Flüge eine Fluggesellschaft anbieten soll. Unternehmensstrategie und Marktbedingungen bestimmen im wesentlichen die Topologie des Netzwerkes und die Frequenz, mit der ein bestimmter Flug geflogen werden soll. Doch schon die genauen Abflug- und Ankunftszeiten von Flügen müssen optimiert werden, um günstige Umsteigemöglichkeiten zu gewährleisten, die der Kundennachfrage entsprechen. Hier gibt es stochastische Modelle, um die Nachfrage, auch in Abhängigkeit von den Flugplänen konkurrierender Fluggesellschaften, zu modellieren.

- Die Flotte einer Fluggesellschaft enthält in der Regel eine Reihe unterschiedlicher Flugzeugmuster oder -typen, die sich beispielsweise im Hinblick auf die Anzahl der zur Verfügung stehenden Sitze, die Reichweite oder die Kosten pro Passagier und Flugkilometer unterscheiden. Das *Fleet Assignment* beschäftigt sich mit dem Problem, jedem Flug ein zulässiges Flugzeugmuster zuzuweisen, so daß insgesamt möglichst viel Nachfrage mitgenommen werden kann und damit der zu erwartende Ertrag maximiert wird. Im Zusammenhang damit steht das *Tail Routing*, bei dem die Flugpläne der einzelnen zur Verfügung stehenden Flugzeuge zusammengestellt werden, um alle einem Flugzeugmuster zugeteilten Flüge zu bedienen. Hier sind insbesondere Wartungsereignisse einzuplanen. Die Folge von Flügen, die ein einzelnes Flugzeug während einer Planungsperiode ableistet, heißt auch Flugzeugrotation (oder -umlauf).

- Die Flugdienst- und Flugpersonaleinsatzplanung, das *Crew-Scheduling*, weist jedem Flug gemäß seines Crew-Komplementes, das die Anzahl der von jedem Rang (Kapitän, Erster Offizier, Flugbegleiter, etc.) benötigten Crew-Mitglieder angibt, Crew-Personal zu. Die (monatlichen) Dienstpläne des Crew-Personals werden erstellt.

- Während des Betriebes schließlich müssen kurzfristige Störungen des Flugplanes und Ausfälle behandelt werden.

Es sollte klar sein, daß es hier Rückkopplungen von späteren zu früheren Planungsphasen gibt, wenn sich z.B. herausstellt, daß ein Flugplan mit den vorhandenen Ressourcen nicht bedient werden kann. Auch möchte man gern die Stellen im Flugplan identifizieren, die unverhältnismäßig hohe Kosten hervorrufen.

1.2 Crew-Scheduling

In der Flugdienst- und Flugpersonaleinsatzplanung werden die Flüge mit Crew-Personal versorgt. Die Komplexität dieser Probleme rührt daher, daß sie erstens mit einer Reihe von widerstrebenden Kostenarten behaftet und zweitens eine Vielzahl von Nebenbedingungen enthalten, die sich aus gesetzlichen und tarifrechtlichen Bestimmungen sowie Betriebsvereinbarungen ergeben und insbesondere die Länge von Flugdienstperioden beschränken und Mindestanforderungen an Ruhezeiten über kürzere und längere Perioden stellen.

Um das Problem handhabbarer zu machen, wird es traditionell in zwei Phasen gelöst. Das Flugzeug-Routing definiert eine Menge von Ketten von Flügen, die jeweils von einem bestimmten Flugzeug bedient werden. Während nun Flugzeuge nahezu ununterbrochen eingesetzt werden können (Wartungen können meist während der Abfertigung am Boden ausgeführt werden), brauchen Menschen Ruhepausen. Im öffentlichen Nahverkehr bekommen manchmal Fahrer einen Bus für die Dauer ihres Dienstes fest zugewiesen. Aufgrund der hohen Fixkosten von Flugzeugen wird die Ressource Flugzeug jedoch früher und unabhängig vom Flugpersonal geplant. Ein Dienst einer Crew stellt sich dann so dar, daß die Crew für einige Flüge an Bord eines Flugzeugs bleibt und dessen Rotation folgt, dann das Flugzeug verläßt, gegebenenfalls eine Pause oder Ruhezeit einlegt, anschließend den Dienst auf einem anderen Flugzeug fortsetzt, u.s.w. Dabei ist darauf zu achten, daß der erste Flug am Heimatort der Crew, der sogenannten Crew-Basis, beginnt und der letzte Flug dort wieder endet. Ein solcher Dienst nennt sich Pairing (oder Besatzungsumlauf). Pairing heißt allgemeiner jede Folge von Flügen (und möglicherweise anderen Dienstelementen), die räumlich und zeitlich konsistent ist und von einer Crew bedient werden kann, ohne die oben angesprochenen Nebenbedingungen zu verletzen. Typischerweise entspricht ein Pairing einem Abschnitt einer Flugzeugrotation, oder ist aus solchen Abschnitten zusammengesetzt, und stellt eine (i.a. mehrtägige) Rundreise um eine Basis dar.

In der ersten Phase des Crew-Schedulings, der Pairing-Konstruktion, werden Flüge zu solchen Pairings zusammengefaßt. Erst in einer zweiten Phase, dem Rostering (oder Crew-Assignment), wird jedem Pairing entsprechend seinem Crew-Komplement Crew-Personal zugewiesen und werden die individuellen Dienstpläne des Flugpersonals erstellt. Hier müssen insbesondere freie Tage und Urlaub des Personals berücksichtigt werden.

Dieser zweistufige Ansatz hat ein Reihe von Vorteilen. Zunächst einmal wird die Größe des Rostering-Problems reduziert, wenn man Pairings anstatt von Flügen verplant. D.h. man kann die Pairing-Konstruktion als ein Preprocessing für die Rostering-Phase ansehen. Sodann erlaubt die Trennung in Pairing und Rostering, sich in den verschiedenen Phasen auf unterschiedliche Optimierungsziele und Nebenbedingungen zu konzentrieren.

1.3 Eine graphentheoretische Formulierung des Pairing-Problems

Da Crew-Mitglieder in der Regel nur Lizenzen für wenige Flugzeugmuster halten, kann man das Pairing-Problem für jedes Flugzeugmuster getrennt behandeln. Zudem nehmen wir an, daß das Crew-Komplement für alle Flüge eines Flugzeugmusters identisch ist, was eine sinnvolle Annahme ist, solange es nur wenige Flüge mit einem erhöhten Crew-Komplement gibt: diesen Fall kann man behandeln, indem man entweder entsprechend dem erforderlichen Crew-Komplement mehrfache Kopien eines Fluges anlegt oder eine erhaltene Lösung nachträglich entsprechend modifiziert.

Der Flugplan nordamerikanischer Fluggesellschaften wiederholt sich in der Regel täglich (außer samstags und sonntags). In diesem Fall genügt es beim Pairing, das tägliche Problem zu betrachten. Wie in ANDERSSON et al. [AHKW97] dargestellt, sind die Flugpläne europäischer Fluggesellschaften jedoch oft unregelmäßiger. Häufig wird hier eine Verbindung zwischen zwei Flughäfen nur zwei- oder dreimal pro Woche angeboten. Wir behandeln daher das sogenannte voll-datierte Modell: der Planungshorizont unserer Testprobleme erstreckt sich über bis zu vier Wochen und ein Flug bezeichnet einen Flug, der an einem bestimmten Tag des Planungszeitraums stattfindet. Auf diese Weise können im Modell neben den Wochenenden auch „hot spots“, d.h. Extraflüge aus Anlaß von Messen oder zu Ferienzeiten, direkt berücksichtigt werden.

Der Digraph *conn*. Der Digraph $conn = (V, A)$ ist eine Datenstruktur, die die möglichen Verbindungen zwischen Flügen verwaltet. Als Knotenmenge V hat er die Menge der Flüge, die zu verplanen sind, und zwei Flüge ℓ_1 und ℓ_2 sind in *conn* durch eine gerichtete Kante $(\ell_1, \ell_2) \in A$ verbunden genau dann, wenn es für eine Crew möglich ist, Flug ℓ_2 nach Flug ℓ_1 zu bedienen, d.h. der Abflugsflughafen von ℓ_2 muß gleich dem Ankunftsflughafen von ℓ_1 und die Abflugszeit von ℓ_2 muß mindestens der Ankunftszeit von ℓ_1 zuzüglich einer Mindestbodenzeit (plus *min_rotation_change_time*, falls ein Rotationswechsel vorliegt) sein. Da Kanten in *conn* einer Bewegung in der Zeit entsprechen, ist dieser Graph stets azyklisch. Damit *conn* eine vertretbare Größe hat, werden Verbindungen, die zu einer Bodenzeit von mehr als *max_ground_time* führen, entfernt. Zur Abhängigkeit der Größe von *conn* von *max_ground_time* siehe die Tabellen 3.3 und 3.4. Tabelle 1.1 zeigt einige statistische Daten des Graphen *conn* für unsere drei Testinstanzen (wobei *max_ground_time* auf 36 Stunden festgesetzt wurde).

Flugzeugmuster	#Knoten	#Kanten	øGrad	max. Grad
F70	374	2327	6.2	16
DH8	1395	21311	15.3	44
X	4612	108451	23.5	66

Tabelle 1.1: Der Digraph *conn*

Lösungen des Pairing-Problems partitionieren den Graphen *conn* in knotendisjunkte Pfade, von denen jeder ein zulässiges Pairing darstellt. Die Zielfunktion ist eine Linearkombination von mehreren Kostenarten. Die Gewichte der Linearkombination widerspiegeln die relative Wichtigkeit der einzelnen Kostenarten für den Anwender sowie die relative Schwierigkeit, die jeweilige Kostenart zu minimieren. Die Gewichte wurden in Rücksprache mit einem Crew-Scheduling Experten, der die produzierten Lösungen begutachtete, eingestellt. Die Kosten aus den folgenden Kostenarten werden pro Pairing erhoben.

Kostenarten.

- *Proceedings* (Deadheads): Das Crew-Personal einer Fluggesellschaft ist im allgemeinen über mehrere Flughäfen verteilt angesiedelt. Falls der Abflugsflughafen des ersten Fluges eines Pairings (bzw. der Ankunftsflughafen des letzten Fluges) nun nicht mit der Basis übereinstimmt, muß die Crew dorthin (bzw. von dort zurück) gebracht oder, wie wir sagen, positioniert werden – sei es nun per Flugzeug (vorzugsweise mit einem Flug der eigenen Fluglinie), Bahn, Taxi oder öffentlichen Nahverkehrsmitteln. Idealerweise hätte man hier gern für jeden Flug und jede Basis Informationen über Dauer und Kosten, die Crew von der Basis zum Abflugsflughafen und vom Ankunftsflughafen zurück zur Basis zu transportieren, verfügbar. Da hierzu keine Daten vorhanden waren, haben wir einige wenige Hubs, d.h. von den zu planenden Flügen stark frequentierte Flughäfen, berechnet und die Kosten von Proceedings zwischen Hubs auf 1 und sonst auf 2 gesetzt. Aus den genannten Gründen konnte die für Proceedings aufzuwendende Zeit nicht in die Berechnung der Dienstzeit einbezogen werden.

- *Hotel*: Man kann Proceeding-Kosten sparen, indem man eine Crew auswärtig übernachten läßt. In diesem Fall (und auch, wenn eine Crew tagsüber längere Zeit an einem Ort Pause macht) sind allerdings Aufwendungen für Unterkunft und Spesen zu erbringen. Da keine Daten über Hotelpreise an den verschiedenen Flughäfen und zu den verschiedenen (Jahres-) Zeiten verfügbar waren, wurden hierfür uniforme Kosten angesetzt.

- *Xwork*: Pairings sollten dicht mit Flügen gepackt sein, um die unproduktive Dienstzeit des Flugpersonals zu minimieren. Dies gewährleistet nicht nur eine hohe Verfügbarkeit des Flugpersonals (die Ruhezeit nach einer Flugdienstperiode muß mindestens so lang wie diese selbst sein) und trägt dazu bei, Überstunden zu minimieren, es ist auch beim Crew-Personal selbst willkommen, das ungern in Flughafenlounges wartet. Aus diesem Grund bestrafen wir die Dienstzeit (inklusive Check-in und Check-out Zeiten) von Pairings, die keine Flugzeit ist.

- *Rotation_Changes*: Die Minimierung von Rotationswechseln (d.h. Wechseln des Flugzeuges) dient der Stabilität des Flugplanes in Anbetracht von Verspätungen und Ausfällen während des Betriebes. Daher wird jeder Rotationswechsel mit Bodenzeit nicht größer als *max_rotation_change_time* bestraft.

- *Shorties*: Im Hinblick auf die sich anschließende Rostering-Phase sollten Pairings gemessen an der in ihnen enthaltenen Flugzeit und Anzahl der Landungen nicht zu kurz sein, sondern möglichst einer Tagesschicht entsprechen (analog bei mehrtägigen Pairings). Der Strafkostenterm für in diesem Sinne zu kurze Pairings stellt einen Anreiz für Hub-Turns, d.h. Durchgänge durch die Basis, dar, die durch *Xwork* bestraft werden. Man könnte hier auch die Anzahl der Flugdienstperioden bestrafen, was jedoch wiederum geteilte Flugdienstperioden begünstigt.

Die Kostenarten *Hotel* und *Proceedings* sind offensichtlich widerstrebend, ebenso *Proceedings* und *Xwork*. Wir erlauben keine Proceedings innerhalb von Pairings; Barnhart, Hatay und Johnson [BHJ95] konnten allerdings zeigen, daß sich durch die zusätzliche Verwendung solcher Proceedings die Kosten noch einmal deutlich reduzieren lassen.

Nebenbedingungen. Zunächst müssen Pairings die gesetzlichen und tarifrechtlichen Bestimmungen sowie etwaige zusätzliche Betriebsvereinbarungen insbesondere zu maximalen Flugdienstzeiten und minimalen Ruhezeiten erfüllen. Hier wurden einige der wichtigsten europäischen JAR-OPS Regeln [EU95] implementiert: Ein Pairing ist eine Folge von Flugdienstperioden, die von Ruhezeiten von mindestens max{*min_rest_time*, Länge der vorherigen Flugdienstperiode} Minuten getrennt werden. Die Dauer eines Pairings ist auf *max_pairing_time* Minuten beschränkt, die enthaltene Dienstzeit auf *max_pairing_dutytime*. Die maximale Länge einer Flugdienstperiode variiert zwischen 9 und 14 Stunden je nachdem, zu welcher Tageszeit sie beginnt und wieviele Landungen sie enthält. Geteilte Flugdienstperioden dürfen eine längere Dauer haben und eine verkürzte Nachtruhe enthalten, wenn die Nachtruhe eine bestimmte Mindestlänge hat und die Dienstabschnitte vorher und nachher etwas kürzer sind. Allerdings zählt diese Übernachtungszeit als Dienstzeit und muß daher später als Ruhezeit gewährt werden.

Die zweite Nebenbedingung koppelt Pairing und Rostering. Jedes Pairing wird von einer Crew einer Crew-Basis bedient und fragt dort somit Crew-Kapazität nach. Die Crew-Kapazitäten sind in der Regel jedoch ganz unterschiedlich über die verschiedenen Basen der Fluggesellschaft verteilt, so daß es naheliegt, diese Verteilung schon beim Pairing zu berücksichtigen, um nicht dort Arbeitszeit nachzufragen, wo kein Crew-Personal mehr vorhanden ist und wohin es erst kostenaufwendig positioniert werden müßte. In unserer Modellierung gehört daher zu jedem Pairing ein Vorschlag, von welcher Basis aus dieses Pairing zu bedienen ist, um global die Proceeding- und Hotel-Kosten zu minimieren. Dies ist nicht notwendigerweise die nächste Basis (um die das Pairing möglicherweise eine Rundreise darstellt). Eine Lösung muß dann die vorhandenen Crew-Kapazitäten an den Basen an den verschiedenen Tagen des Planungszeitraums als Nebenbedingung respektieren, womit sich auch Personalengpässe durch Vorgaben wie Urlaub, freie Tage, vorab zugewiesene Pairings etc. behandeln lassen. Kleinste Maßeinheit ist ein Personentag, und ein Pairing fragt so viele Personentage nach, wieviel Tage es berührt, vergleiche Abbildung 1.1. Hier veranschaulichen die drei Rechtecke in der ersten Zeile die an den entsprechenden Tagen an der Basis ZDD vorhandene Crew-Kapazität, der mittlere Teil des Diagramms enthält in den Zeilen die zu einer Flugzeugrotation gehörigen Flüge, und ein durch die geschwungene Linie angedeutetes Pairing startet in Basis ZDD, enthält vier Flüge und kehrt zur Basis, die auch Basis des Pairings selbst ist, zurück. Die schwarz unterlegte Fläche in den Kästchen für die Crew-Kapazität am 1. und 2. März repräsentiert die von diesem Pairing verbrauchte Crew-Kapazität. Crew-Mitglieder werden auf der anderen Seite nur dann zur Crew-Kapazität hinzugerechnet, wenn sie den ganzen Tag über verfügbar und nicht zu anderweitigen Diensten (wie Training, Schulung, medizinische Checks) eingeteilt sind. Ein Nebeneffekt des Konzeptes der Basis eines Pairings ist, daß das Rostering-Problem in kleinere und unabhängige Teilprobleme an den einzelnen Basen zerfällt.

Je nachdem, ob man die Verteilung der Crew-Kapazitäten berücksichtigt oder nicht, unterscheiden sich die (jeweils optimierten) Lösungen unserer Test-Instanzen hinsichtlich der Auslastung des Crew-Personals an den Basen drastisch. Der obere Teil von Abbildung 1.2 zeigt die Crew-Auslastung einer optimierten Lösung, die ohne Beachtung der Crew-Kapazitäten an den Basen berechnet wurde. Da etwa 45% des Crew-Personals am Flughafen ZDD beheimatet sind, müßten in der Rostering-Phase viele Crew-Mitglieder positioniert werden. Die Lösung aus dem unteren Teil andererseits wurde unter Strafkosten für eine Auslastung von Crew-Kapazität über 60% an einer Basis und an einem Tag des Planungszeitraums optimiert. Sie zeigt eine nahezu gleichmäßige Auslastung der Crew-Kapazität von etwa 60% (an einigen Tagen des Planungszeitraums fanden allerdings keine Flüge statt).

Bemerkenswerterweise sind die Proceeding-Kosten der Pairing-Zerlegung aus der oberen Abbildung etwa fünfmal so hoch wie die Proceeding-Kosten der Lösung aus der unteren Abbildung (die dafür allerdings in der Rostering-Phase entsprechende Proceeding-Kosten verursachen würde). Die Simulation zeigt also, daß ein großer Anteil der Proceeding-Kosten durch eine bessere Verteilung des Crew-Personals über die Basen eingespart werden könnte.

Andere Systeme bestrafen Pairings um Basen mit wenig Crew-Personal immer stärker, bis eine befriedigende Lösung gefunden ist. Es scheint jedoch, daß dieses Vorgehen, abgesehen von dem enormen Rechenzeitbedarf, dem Planer ein großes Fingerspitzengefühl abverlangt. Hoffman and Padberg [HP93] beschreiben eine Modellierung als ganzzahliges lineares Optimierungsproblem, in die derartige Packungsungleichungen einfach zu integrieren sind. Sie lösen jedoch nur ein tageweises Pairing Problem.

Relaxierung von Nebenbedingungen. Die Nebenbedingung für die Auslastung der Crew-Kapazität wurde relaxiert, so daß eine Nachfrage nach Crew-Kapazität

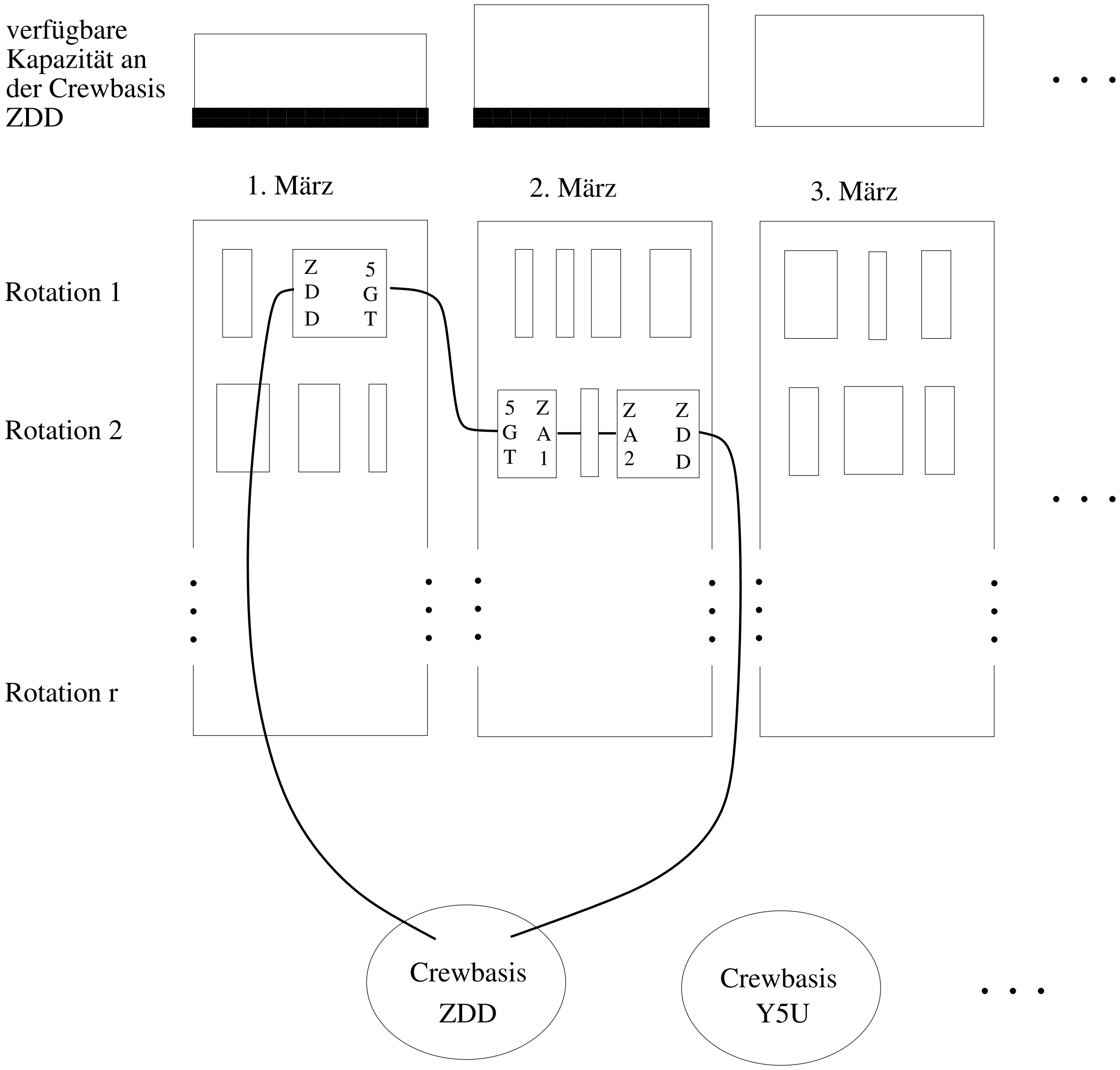

Abbildung 1.1: Die Bildung von Pairings und der Verbrauch von Crew-Kapazität

(durch die Pairings), die an einem bestimmten Tag des Planungszeitraums und an einer bestimmten Basis die dort vorhandene Crew-Kapazität überschreitet, zwar nicht verboten, aber quadratische Strafkosten nach sich zieht. Dadurch läßt sich unter Umständen eine große Verminderung in anderen Kostenarten bewirken, während eine kleine Überschreitung der Crew-Kapazität in der Regel keine nachteiligen Auswirkungen hat, da die Werte für die Crew-Kapazitäten lediglich Zielwerte sind – in unserem Fall 60% (DH8) bzw. 40% (X) der nominal vorhandenen tatsächlichen Crew-Kapazität. Dies rührt daher, daß Crew-Kapazität für andere Zwecke wie freie Tage (mindestens 2 Tage innerhalb von 10 Tagen, entspricht den Wochenenden), Reserve und Bodendienste vorgehalten werden muß (für die wir allerdings keine Daten zur Verfügung hatten). Da es auf der Seite der freien Tage, des Urlaubs und der Bodendienste wie Gesundheitschecks und Trainings eine gewisse planerische Freiheit gibt, kann an einem bestimmten Tag an einer bestimmten Basis durchaus mehr

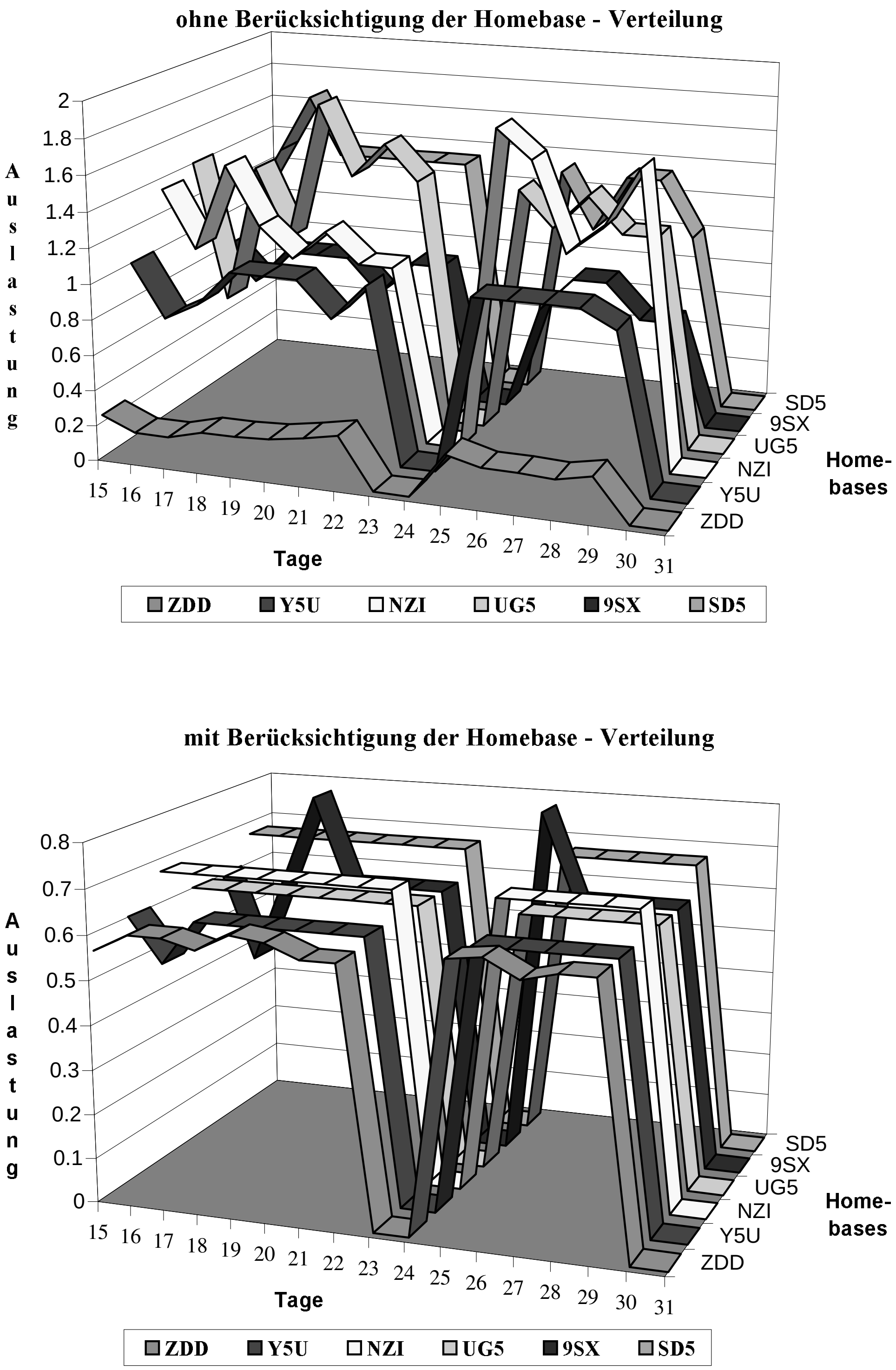

Abbildung 1.2: Crew-Auslastung an den Basen (DH8)

Crew-Personal verfügbar gemacht werden, ohne daß dies notwendig monetäre Kosten nach sich zöge. Verfahren der ganzzahligen linearen Optimierung werden jedoch solche Nebenbedingungen (sogenannte Packungsungleichungen) strikt einhalten.

Die Nebenbedingung, die die gesetzliche und tarifrechtliche Zulässigkeit betrifft, muß andererseits von einer Lösung immer strikt eingehalten werden. Hier stellt sich allerdings heraus, daß die verwendeten Metaheuristiken bessere Lösungen produzieren, wenn während der Optimierung auch unzulässige Pairings erlaubt sind, diese Nebenbedingung also auch relaxiert wird. Eine Nachbearbeitungsroutine sorgt dafür, daß die ausgegebene Lösung schließlich zulässig ist, indem sie schrittweise die Strafkosten für Unzulässigkeit hochsetzt und einige Schritte des Verfahrens STEILSTER_ABSTIEG durchführt – die von der Metaheuristik gefundene Lösung wird also gewissermaßen auf die Menge der zulässigen Lösungen projiziert. Diese Nachbearbeitungsroutine benötigt keine nennenswerte Rechenzeit. Wenn man die Strafkosten für unzulässig Pairings jedoch so niedrig setzt, daß die Metaheuristik Lösungen produziert, die „fast zulässig“ sind, kann die Lösungsqualität insgesamt um bis zu 4% verbessert werden, vergleiche PROKSCH [Pro97]. Der Grund scheint zu sein, daß dadurch die einzelnen Lösungen des Lösungsraums näher zusammenrücken: wenn (zwischenzeitlich) unzulässige Lösungen erlaubt sind, wird es einfacher, zwei Pairings zu einem langen Pairing zu verketten und dieses später an einer möglicherweise günstigeren Stelle wieder aufzuspalten. Der Effekt scheint allerdings nachzulassen, je langsamer abgekühlt wird. Bei einem zu klein gewählten Strafkostenterm verändert sich zudem die Kostenstruktur der Lösung.

Verwandtschaften und Unterschiede zu Vehicle-Routing- und Packungsproblemen. In der gewählten Formulierung stellt sich das Pairing-Problem als ein spezielles MULTI DEPOT VEHICLE ROUTING Problem (siehe z.B. DADUNA und PAIXAO [DP95], FISHER [Fis95] sowie GENDREAU, LAPORTE und POTVIN [GLP97]) dar. Bei diesem Problem sind eine Menge von Kunden mit Lieferzeiten und Servicedauer gegeben und möglichst günstige Touren für die Lieferfahrzeuge gesucht, so daß alle Kunden bedient werden. Eine Tour beginnt und endet in einem der Depots und darf eine gewisse Zeitdauer nicht überschreiten. Ihre Kosten berechnen sich aus den (zeitlichen bzw. räumlichen) Entfernungen zwischen den Kunden auf der Tour. Im Fall des Pairing-Problems entsprechen die Flüge den zu beliefernden Kunden, die Basen den Depots, die Crews den Lieferfahrzeugen und die Pairings den Touren. Die Touren (d.h. die Pairings) gehen hier allerdings i.a. über mehrere Tage und ein Lieferfahrzeug (d.h. eine Crew) fährt i.a. mehrere Touren. Die Probleme sind zudem stark geclustert, da es zu einem Flug in der Regel einen günstigsten Folgekunden (den nächsten Flug auf der Flugzeugrotation) gibt, während für andere Anschlußflüge eine erhöhte Mindestbodenzeit einzuhalten ist. Die Entfernungsmatrix ist aufgrund der Zeitstruktur und Proceeding-Kosten stark asymmetrisch, die größten Kosten fallen bei der Anbindung der Touren (d.h. der Pairings) an die Depots an.

Andererseits ähnelt das Pairing-Problem einem Packungsproblem (siehe z.B. DYCKHOFF, SCHEITHAUER und TERNO [DST97], COFFMAN und LUEKER [CL91], DOWSLAND und DOWSLAND [DD92] sowie MARTELLO und TOTH [MT90]): die Pairings sind in (Anzahl Basen) mal (Anzahl Tage des Planungshorizonts) viele

Container zu verpacken, wobei ein Pairing allerdings auf (Anzahl Tage, über die das Pairing läuft) viele, aufeinanderfolgende Container verteilt jeweils genau einen Personentag Kapazität in jedem der betroffenen Container verbraucht. Aufgrund der zeitlichen Struktur des Problems gibt es für jedes Pairing zudem nur Anzahl Basen viele Packungsmöglichkeiten. Abgesehen davon, daß die Überschreitung der Kapazität eines Containers zulässig ist und lediglich Strafkosten nach sich zieht, berechnen sich die Kosten einer Lösung des Packungsproblems schließlich Pairing-weise in Abhängigkeit von der jeweils gewählten Basis.

Komplexität. Bertossi, Carraresi und Gallo [BCG87] studieren die folgenden Vehicle-Routing Probleme. Capacitated Matching: gegeben ein azyklischer Digraph $D = (V, A)$, wobei V der Menge der zu bedienenden Kunden entspricht und die Kosten $c(i, j)$ einer Kante die Kosten der Verbindungsfahrt vom Kunden i zum Kunden j wiedergeben, berechne man eine Partition von D in höchstens k Touren (Pfade) mit minimalen Kosten, wobei für jede Tour noch die Kosten für die Anbindung an das Depot zu berücksichtigen sind. Wie sich zeigt, kann dieses Problem mit einem Minimum-Kosten-Fluß-Algorithmus gelöst werden. Der Fall jedoch, wenn die Kunden von $d \geq 2$ Depots aus bedient werden sollen und höchstens k_i Touren vom i-ten Depot aus starten dürfen, $i = 1 \ldots d$, das sogenannte Capacitated Multicommodity Matching Problem, ist – selbst ohne die durch die k_i's gegebenen Kapazitätbeschränkungen an den Depots – $\mathcal{NP}$-vollständig. Mithin ist auch unsere Formulierung des Pairing-Problems (wobei die Flüge den Kunden entsprechen) $\mathcal{NP}$-vollständig, selbst ohne die die Zulässigkeit und Crew-Kapazitäten betreffenden Nebenbedingungen.

Die Testprobleme. Tabelle 1.2 macht Angaben über die in unserer Studie verwendeten Testprobleme F70, DH8 und X. Sie stammen von mittelgroßen europäischen Fluggesellschaften und enthalten Kurz- bis Mittelstreckenflüge. Die Spalten der Tabelle führen nacheinander die Anzahl der im Testproblem enthaltenen Flüge, die Länge der Planungsperiode in Tagen, die Anzahl der Flugzeugrotationen, die die Menge der Flüge partitionieren, die Anzahl der besuchten Flughäfen, die Anzahl der Crew-Basen des verfügbaren Crew-Personals sowie die durchschnittliche Blockzeit (d.h. Dauer) der Flüge in Minuten an.

Flugzeugmuster	#Flüge	#Tage	#Rot	#Flughäfen	#Basen	øBlock
F70	374	13	3	10	3	58
DH8	1395	13	9	18	6	71
X	4612	30	23	45	4	77

Tabelle 1.2: Die Testprobleme

Da wir mit dem voll-datierten Modell rechnen, spiegelt weniger die Anzahl der Flüge als die Anzahl der Flugzeugrotationen die Schwierigkeit eines Testproblems wider.

Die Testprobleme weisen eine sogenannte Hub-and-Spoke-Struktur auf, d.h. die Mehrzahl der Flughäfen wird nur von einer kleinen Anzahl von Flughäfen, den sogenannten Hubs, aus bedient. Tabelle 1.3 zeigt für unser kleinstes Problem, F70, die Matrix, deren (i, j)-ter Eintrag die Anzahl der Flüge vom Flughafen i zum Flughafen j während des Planungszeitraums angibt.

$F70$	NZI	$Y5U$	YMG	ZDD	RGP	$5GT$	NOI	$5PU$	$SD5$	ZYM
NZI	0	0	44	36	0	0	27	16	0	0
$Y5U$	0	0	0	0	31	31	0	0	0	0
YMG	43	0	0	0	0	0	0	0	0	0
ZDD	33	0	0	0	0	0	0	0	3	1
RGP	0	31	0	0	0	0	0	0	0	0
$5GT$	0	31	0	0	0	0	0	0	0	0
NOI	27	0	0	0	0	0	0	0	0	0
$5PU$	16	0	0	0	0	0	0	0	0	0
$SD5$	3	0	0	0	0	0	0	0	0	0
ZYM	0	0	0	1	0	0	0	0	0	0

Tabelle 1.3: Die Hub-and-Spoke-Struktur

1.4 Rostering

Beim Rostering werden die sich aus den Pairings gemäß Crew-Komplement ergebenden Dienste dem Crew-Personal zugewiesen und die individuellen Dienstpläne (Roster) erstellt. Bei nordamerikanischen Fluggesellschaften ist es üblich, auch die Roster zunächst anonym zu erstellen, wobei Wirtschaftlichkeitskriterien und Gleichmäßigkeit (damit Crew-Mitglieder mit den Routen vertraut werden) eine Rolle spielen. Diese Roster (oder auch „bidlines“) können sich daher untereinander im Hinblick auf ihre Attraktivität bei den Crew-Mitgliedern und ihr Arbeitspensum erheblich unterscheiden, können aber auch unterschiedlich entlohnt werden. Die Crew-Mitglieder wählen sich dann in Reihenfolge ihres Dienstalters ihren Dienstplan selbst aus. In Europa dagegen haben alle Crew-Mitglieder in der Regel den gleichen Dienstvertrag und erhalten das gleiche Gehalt. Somit entsteht das Problem, die Dienste gerecht auf Dienstpläne zu verteilen.

Die wesentlichen Optimierungskriterien und Nebenbedingungen der Rostering-Phase sind

- Alle Flüge müssen bedient werden. Ebenso sind für jedes einzelne Crew-Mitglied verschiedene Bodendienste wie Reservedienste, Bürodienste, Trainingskurse oder medizinische Routineuntersuchungen zu planen. Der Dienstplan sollte gegenüber kleinen betrieblichen Störungen (wie z.B. Verspätungen) robust sein.

- Die Dienstpläne müssen den gesetzlichen und tarifrechtlichen Bestimmungen und Betriebsvereinbarungen insbesondere zu Arbeits- und Flugzeitbegrenzungen, minimalen Ruhezeiten und freien Tagen in Zeitintervallen unterschiedlicher Längen genügen.

- Die Dienstpläne sollen Vorbelegungen durch Urlaub, freie Tage, etc. berücksichtigen.
- Die Dienstpläne sollten hinsichtlich der Arbeits- und Flugzeitbelastung sowie hinsichtlich unattraktiver Dienste ausgeglichen sein.
- Die Dienstpläne sollten eine gewisse Gleichmäßigkeit hinsichtlich der Verteilung von Diensten und freien Tagen sowie der Verteilung von besonders schweren Diensten über den Planungszeitraum aufweisen.
- Die Dienstpläne sollten möglichst individuelle Wünsche (z.B. bezüglich bestimmter Flugziele) berücksichtigen.

Um Lizenzen für mehrere Flugzeugmuster aufrechtzuerhalten, ist es zudem erforderlich, eine gewisse Mindestzahl an Flugstunden auf jedem dieser Flugzeugtypen zu leisten. Für einige Flughäfen gibt es Landeerlaubnisse, die ähnlich zu behandeln sind.

Es gibt zwei Herangehensweisen an das Rostering-Problem. In der ersten wird für jedes Crew-Mitglied ein individueller Dienstplan erstellt. Das Problem kann als Generalized Assignment Problem (siehe z.B. Cattrysse und Van Wassenhove [CW92], Martello und Toth [MT92] sowie Osman [Osm95]) mit nichtlinearen Nebenbedingungen betrachtet werden. Wir haben die Dienste auf der einen und das Flugpersonal auf der anderen Seite und wollen die Dienste dem Flugpersonal so zuordnen, daß für kein Crew-Mitglied sein Arbeitszeitkonto überschritten wird. Dienste können einem Crew-Mitglied allerdings nicht unabhängig voneinander zugewiesen werden: es muß alle ihm zugewiesenen Dienste legal ausführen können. Bianco et al. [BBM+92] schlagen ein auf Matching-Heuristiken basierendes Verfahren vor, das eine Kombination von bezüglich des Rhythmus von freien Tagen verschiedenen Rostertypen bestimmt, um die maximale Arbeitslast eines Dienstplans zu minimieren. Individuelle Roster sind von Ryan [Rya92] auch mit Hilfe von Set Partitioning Algorithmen berechnet worden, die auf einer apriori berechneten Auswahl von Rostern, die mittels einer Tiefensuche, die für jede bereits konstruierte Dienstfolge eine beschränkte Anzahl die unproduktive Zeit minimierender Folgedienste durchtestet, konstruiert werden, arbeiten. Dieses Manko räumen Gamache et al. [GSMD94] aus, die einen Spalten-Generierungsansatz für eine verallgemeinerte Set Partitioning Formulierung vorschlagen, um die (Gesamtdauer der) nicht-bedienten Dienste zu minimieren. Das verallgemeinerte Set Partitioning Problem wird dann mit GENCOL gelöst. Die Arbeitslast scheint hier allerdings nicht so gleichmäßig über den Planungszeitraum verteilt zu werden, wie das bei Zuweisungsheuristiken in Verbindung mit lokaler Verbesserung der Fall ist. Federal Express benutzt einen Bidline-Generator, der auf Simulated Annealing basiert, um die Auswirkung der Änderung von Betriebsvereinbarungen zu simulieren, siehe Campbell, Durfee und Hines [CDH97]; Die Arbeit präsentiert jedoch keine Zahlen zu den Ergebnissen. Die Minimierung des Bedarfs an Crew-Personal kann bei all diesen Methoden durch Simulation einer geringeren Personalstärke erreicht werden.

Eine andere Methode wird angewandt, wenn – wie im öffentlichen Nahverkehr – die Dienst(schicht)folge in den Dienstplänen dicht ist, ein einzelner Fahrer von bestimmten Diensten, wie z.B. Nachtdiensten oder Wochenenddiensten, nur begrenzt viele leisten soll und die Dienstmasse als wochenweise identisch angenommen werden kann. Hier erhalten, um alle Ruhezeitbestimmungen einzuhalten und eine Gleichverteilung z.B. der Nacht- und Wochenenddienste zu erreichen, ganze Gruppen von Fahrern identische, zyklische Dienstpläne (sogenannte Turnusse), die von den Fahrern wochenweise versetzt gefahren werden, so daß an einem bestimmten Tag alle Dienste des Turnus gefahren werden. Siehe z.B. BAKER und MAGAZINE [BM77] sowie CAPRARA et al. [CFTV95, CFG+97] und ERNST, KRISHNAMOORTHY und DOWLING [EKD96] für eine Anwendung auf die Dienstplanung im Eisenbahnverkehr.

Kapitel 2

Verfahren der lokalen Suche für das Crew-Scheduling-Problem

2.1 Lokale Suche

Lokale Suche ist eine Technik, die für polynomiell lösbare Optimierungsprobleme wohlbekannt ist. Sie steht beispielsweise hinter solch klassischen (exakten) Verfahren wie dem Simplex-Algorithmus der Linearen Optimierung, Matching- und Flußalgorithmen oder dem Kantenaustausch-Algorithmus für das MINIMUM SPANNING TREE Problem. Seit Anfang der 80er Jahre Simulated Annealing als Heuristik für $\mathcal{NP}$-schwere kombinatorische Optimierungsprobleme populär wurde, erfreuen sich Verfahren der lokalen Suche und sogenannte Metaheuristiken zunehmenden Interesses, siehe AARTS und LENSTRA [AL97] und REEVES [Ree95].

Ein kombinatorisches Optimierungsproblem Π besteht aus einer Menge $D_\Pi \subset \Sigma^*$ von (Problem-) Instanzen, wobei für jede Instanz $I \in D_\Pi$ eine (endliche) Menge möglicher Lösungen $\mathcal{S}_I$ und eine Zielfunktion $c: \mathcal{S}_I \to \mathbb{Q}$ gegeben ist. Die Aufgabe besteht darin, zu gegeber Instanz $I \in D_\Pi$ eine Lösung $s^* \in \mathcal{S}_I$ zu finden, die c o.B.d.A. minimiert, d.h. die $c(s^*) \leq c(s)$ für alle $s \in \mathcal{S}_I$ erfüllt. Typischerweise ist die Größe von $\mathcal{S}_I$ exponentiell in $|I|$ (der Codierungslänge der Instanz I) und $\mathcal{S}_I$ nur implizit gegeben. Eine vollständige Enumeration des Lösungsraumes $\mathcal{S}_I$ verbietet sich in der Regel selbst schon für mäßig große Probleminstanzen aufgrund der Größe von $\mathcal{S}_I$. Es stellt sich das Problem, ein effizientes Verfahren zu entwickeln, das eine optimale oder zumindest eine gute Lösung konstruiert. Effizient heißt ein Verfahren (oder Algorithmus) dabei, wenn seine Laufzeit (genauer gesagt, die längste Laufzeit unter allen Probleminstanzen $I \in D_\Pi$ derselben Codierungslänge) polynomiell in der Eingabegröße $|I|$ wächst.

Verfahren der lokale Suche arbeiten nach dem Prinzip, ausgehend von einer zu Beginn erzeugten Lösung $s_0 \in \mathcal{S}$ der betrachteten Probleminstanz, eine Folge $s_0, s_1, s_2, \ldots, s_N$, $s_i \in \mathcal{S}$, $N \in \mathbb{N}_0$, von Lösungen zu produzieren, wobei im i-ten Schritt die Lösung s_i durch kleine, lokale Modifikationen an s_{i-1} entsteht und sich die Qualität der Lösungen (nicht notwendig monoton) verbessert. Wenn keine Verbesserung mehr erreicht wird oder die zur Verfügung stehende Zeit abgelaufen ist, bricht das Verfahren ab und gibt die beste gefundene Lösung aus. Die zulässigen lokalen Modifikationen oder Übergänge definieren einen Nachbarschaftsgraphen $\mathcal{N} = (\mathcal{S}, A)$ auf dem Lösungsraum $\mathcal{S}$. Für $s \in \mathcal{S}$ bezeichne $N(s) \subseteq \mathcal{S} \setminus \{s\}$ die Menge der Lösungen, in die die Lösung s durch lokale Modifikationen transformiert werden kann. Die Lösungen in $N(s)$ heißen auch Nachbarn von s. Dann hat $\mathcal{N}$ gerade die Kantenmenge $A = \{(s_1, s_2) : \ s_1 \in \mathcal{S}, \ s_2 \in N(s_1)\}$. Die von einem lokalen Suchverfahren konstruierte Folge $s_0, s_1, s_2, \ldots, s_N$ mit $(s_{i-1}, s_i) \in A$ für $i = 1, \ldots, N$ heißt *Trajektorie* des Verfahrens im Lösungsraum $\mathcal{S}$ unter der Startlösung s_0 und der Nachbarschaft $\mathcal{N}$. Ein lokales Optimum ist eine Lösung $s^* \in \mathcal{S}$, deren Zielfunktionswert höchstens so groß ist wie der eines jeden Nachbarn: $c(s^*) \leq c(s)$ für alle $s \in N(s^*)$, d.h. eine lokal optimale Lösung ist eine Lösung, die durch keine der möglichen Modifikationen mehr verbessert werden kann. Ziel einer lokalen Suche ist es, immerhin ein *lokales Optimum* in $\mathcal{S}$ zu finden. Die verschiedenen Verfahren der lokalen Suche unterscheiden sich im wesentlichen darin, nach welchem Kriterium die nächste Lösung unter den Nachbarn $N(s)$ der aktuellen Lösung s ausgewählt wird.

Lokale Suchverfahren zeichnen sich durch weitgehende Problemunabhängigkeit aus, da zu ihrer Implementation seitens der Anwendung lediglich

- ein Konstruktionsverfahren, welches die Startlösung $s_0 \in \mathcal{S}$ erzeugt,
- die (implizite) Definition der möglichen lokalen Modifikationen an einer Lösung, das heißt der Nachbarschaftsgraph $\mathcal{N}$ auf $\mathcal{S}$, und
- eine Funktion, die die Kosten $c(s)$ einer Lösung $s \in \mathcal{S}$ berechnet,

erforderlich sind. Wenn Nebenbedingungen relaxiert werden und als Strafkostenterm in die Kostenfunktion eingehen, ist es meist nicht schwer, eine Startlösung zu konstruieren. Wir werden jedoch am Beispiel des Crew-Scheduling-Problems sehen, daß eine wesentliche Merkmale des Problems berücksichtigende Startlösung Qualität und Laufzeit einer lokalen Suche positiv beeinflußt. Entscheidend für den Erfolg einer lokalen Suche ist allerdings die Wahl einer problemangepaßten Nachbarschaft – bestimmt sie doch,

- wie gut lokale Optima sind,
- wie stark der Lösungsraum zusammenhängt, d.h. wie einfach eine gute oder gar optimale Lösung von einer mehr oder weniger beliebigen Startlösung aus zu erreichen ist sowie
- die Zeitkomplexität eines einzelnen Übergangs der lokalen Suche, denn hierzu sind die Kosten (von möglicherweise mehreren) Nachbarn zu bestimmen und zu vergleichen. Die Kosten eines Nachbarn sollten also möglichst schnell aus den Kosten der aktuellen Lösung abgeleitet werden können.

Um Lösungen von hoher Qualität zu produzieren, kann die Verwendung einer teuer auszuwertenden Nachbarschaft notwendig sein. Der Zusammenhang des Lösungsraums wird insbesondere dadurch beeinflußt, ob und wie Nebenbedingungen relaxiert werden.

Verfahren der lokalen Suche lassen sich grob in zwei Klassen einteilen: Verfahren der lokalen Verbesserung und Metaheuristiken.

Unter einem lokalen Verbesserungsverfahren versteht man einen Algorithmus, der ausschließlich Übergänge ausführt, die die Zielfunktion verbessern. Beispiele hierfür sind die Verfahren GREEDY, das die Nachbarschaft durchsucht und den ersten Übergang, der zu einer besseren Lösung führt, ausführt, und STEILSTER_ABSTIEG, das in jedem Schritt die komplette Nachbarschaft bewertet und zu einem solchen Nachbarn übergeht, der die in diesem Schritt bestmögliche Verbesserung der Zielfunktion liefert. Offensichtlich hängt hier das gefundene lokale Optimum stark von der Lösung ab, mit der man gestartet ist.

Metaheuristiken treten dagegen mit dem Anspruch auf, unter den meist unzähligen lokalen Optima ein (mehr oder weniger) *gutes* lokales Optimum zu finden. Zu diesem Zweck werden zum einen kontrolliert auch Verschlechterungen der Kostenfunktion akzeptiert und wird zum anderen – zumindest bei vielen Verfahren – die Auswahl des nächsten Übergangs randomisiert. Dies soll die Verfahren zum einen unabhängiger von der Startlösung machen und zum anderen ermöglichen, lokale Optima auch wieder verlassen zu können. Über den Parameter(satz), der die Akzeptanz von Verschlechterungen kontrolliert, kann zudem oft die Lösungsqualität (auf Kosten der Laufzeit) auf die gewünschten Anforderungen eingestellt werden. Wir werden in dieser Arbeit anhand des Crew-Scheduling-Problems sehen, daß sich mit Hilfe von Metaheuristiken Lösungen finden lassen, die circa 40% besser als Lösungen aus Konstruktionsheuristiken kombiniert mit lokalen Verbesserungsverfahren sind.

Aufgrund der genannten Vorzüge von Metaheuristiken – die weitgehende Problemunabhängigkeit sowie die Steuerbarkeit der Lösungsqualität – empfehlen sich solche Verfahren insbesondere für Probleme, wo es aufgrund einer komplexen, aus widerstrebenden Kostenarten zusammengesetzten Zielfunktion und/oder zahlreicher Nebenbedingungen schwierig ist, problemspezifische Algorithmen zu entwickeln. Derartige Probleme treten insbesondere im Operations Research auf, wo es darum geht, Geschäftsablaufs- und Produktionsprozesse zu optimieren. Einen Überblick über eine Reihe von Anwendungen geben z.B. OSMAN und LAPORTE [OL96] und KOULAMAS, ANTONY und JAEN [KAJ94]. Verfahren der lokalen Suche erwiesen sich zudem als sehr flexibel und robust hinsichtlich Änderungen und Anpassungen der Zielfunktion oder der Nebenbedingungen. Dies ist insbesondere für Probleme aus der Praxis relevant, wo die mathematische Modellierung meist über Rücksprache mit einem Anwendungsexperten den Anforderungen angepaßt werden muß. Für mathematische Modellprobleme wie beispielsweise das TRAVELING SALESPERSON PROBLEM oder GRAPH PARTITIONING, für die es sehr leistungsfähige, problemspezifische Heuristiken gibt (in obigem Fall die klassischen lokalen Verbesserungsverfahren nach LIN-KERNIGHAN [LK73] bzw. KERNIGHAN-LIN [KL70]), sind Metaheuristiken hingegen oftmals unterlegen – zumindest auf dünnen oder geometrischen Graphen, wie sie in der Praxis eher auftreten – siehe JOHNSON und MCGEOCH [JM97] und JOHNSON et al. [JAMS89].

Selbst in Fällen, wo inzwischen leistungsfähige exakte Algorithmen entwickelt wurden, können Heuristiken gute obere Schranken beisteuern und dadurch das exakte Verfahren beschleunigen, vergleiche z.B. KOCH und MARTIN [KM96] im Fall des Steiner Tree Problems. Manchmal, wie z.B. im Falle des Set Partitioning Problems, kann die Heuristik auch dazu dienen, gute Basislösungen zu generieren, die dann von einem ganzzahligen linearen Optimierungsprogramm kombiniert werden, vergleiche z.B. Clements et al. [CCJ+97].

Die Komplexität von lokaler Suche. Theoretische Ergebnisse belegen, daß die Zeitkomplexität eines Algorithmus der lokalen Suche (im schlechtesten Fall) exponentiell sein kann. Llewellyn, Tovey und Trick [LTT89] bewiesen, daß selbst, wenn die Topologie des Nachbarschaftsgraphen $\mathcal{N}$ eine hochgradig reguläre Struktur aufweist, lokale Optima schwer zu finden sind: Ein Algorithmus, der für jede Zielfunktion $f : V(H_d) \to \mathbb{Z}$ ein lokales Optimum auf dem d-dimensionalen Hyperwürfel H_d, $d \in \mathbb{N}$, findet, benötigt im schlechtesten Fall mindestens $2^{d-1}/\sqrt{d}$ Funktionsauswertungen. Für den d-dimensionalen Gittergraphen $G_{d,k}$ auf $\{0, \ldots, k-1\}^d$ bewiesen Althöfer und Koschnick [AK93] eine analoge Schranke von $k^d/(2dk + 1)$.

Johnson, Papadimitriou und Yannakakis [JPY88] definieren eine Komplexitätsklasse $\mathcal{PLS}$ (für „polynomial-time local search"), die die Komplexität von lokalen Suchproblemen beschreibt, und wiesen unter einem in diesem Zusammenhang natürlichen Reduzierbarkeitsbegriff ein erstes Problem als $\mathcal{PLS}$-vollständig aus.

Definition 2.1.1 *Ein kombinatorisches Optimierungsproblem Π zusammen mit einer Nachbarschaft $\mathcal{N}_I$ auf der Menge der zulässigen Lösungen $\mathcal{S}_I$ für jede Instanz $I \in D_\Pi$ gehört zur Klasse $\mathcal{PLS}$, wenn*

- *es eine primale Heuristik gibt, die zu einer Instanz $I \in D_\Pi$ in polynomieller Zeit eine zulässige Lösung $s \in \mathcal{S}_I$ produziert, und*
- *eine polynomiale Verbesserungsheuristik zum Nachbarschaftsgraph $\mathcal{N}_I$ gibt, die zu einer Lösung $s \in \mathcal{S}_I$ eine Lösung $r \in N_I(s)$ ausgibt mit besserem Kostenwert als s oder aber berichtet, daß es keine solche Lösung $r \in N_I(s)$ gibt.*

Der Standardalgorithmus für Probleme in $\mathcal{PLS}$ wendet, ausgehend von der durch die primale Heuristik konstruierte Startlösung, sukzessive die Verbesserungsheuristik an, bis ein lokales Optimum erreicht ist. Für ungewichtete Optimierungsprobleme wie Max Cut und Maximum Independent Set, deren Zielfunktion nur polynomiell viele Werte annimmt, wird er offensichtlich in polynomieller Zeit terminieren. Es stellt sich die Frage, ob der Standardalgorithmus in Fällen, wo die Zielfunktion exponentiell viele Werte annehmen kann (wie bei Problemen mit binär codierten Gewichten), polynomiell ist. Wenn es allerdings für ein $\mathcal{PLS}$-vollständiges Problem einen polynomiellen Algorithmus gibt, der zu jeder Instanz eine lokal optimale Lösung konstruieren würde, so ließe sich daraus für *jedes* lokale Suchproblem in $\mathcal{PLS}$ (unter anderem also beispielsweise auch für das Problem Linear Programming) ein polynomieller Algorithmus ableiten. Erstaunlicherweise konnten aber selbst einfache lokale Suchprobleme wie z.B. Max Cut mit der 1-Opt-Nachbarschaft als $\mathcal{PLS}$-vollständig klassifiziert werden.

Eine sogenannte strikte Reduktion überträgt auch untere Schranken an die Laufzeit des Standardalgorithmus. Da die Reduktionen aus $\mathcal{PLS}$-Vollständigkeitsbeweisen zumeist strikt sind, ergeben sich die folgenden negativen Resultate.

Satz 2.1.2 *Unabhängig von der verwendeten tie-breaking-Regel hat der lokale Suchalgorithmus exponentielle Zeitkomplexität für die folgenden (gewichteten) Probleme:*
(i) GRAPH PARTITIONING *unter der* KERNIGHAN-LIN- [JPY88] *oder der* SWAP-*Nachbarschaft* [SY91]*;*
(ii) TRAVELING SALESPERSON *unter der* LIN-KERNIGHAN- [Pap92] *oder der k-*OPT-*Nachbarschaft für ein genügend großes, konstantes k* [Kre90]*;*
(iii) MAX CUT *unter der 1-*OPT-*Nachbarschaft* [SY91]*;*
(iv) SAT *unter der* FLIP-*Nachbarschaft* [Kre90]*.*

Andererseits gilt:

Satz 2.1.3 (JOHNSON, PAPADIMITRIOU, YANNAKAKIS 1988)
Wenn ein lokales Suchproblem $\Pi \in \mathcal{PLS}$ $\mathcal{NP}$*-schwer ist, so ist* $\mathcal{NP} = co - \mathcal{NP}$*.*

Aus $\mathcal{NP} = co - \mathcal{NP}$ folgt zwar noch nicht $\mathcal{P} = \mathcal{NP}$, aber aus $\mathcal{NP} \neq co - \mathcal{NP}$ folgt $\mathcal{P} \neq \mathcal{NP}$ (da $\mathcal{P}$ unter Komplementbildung abgeschlossen ist). Eine Einführung in die $\mathcal{PLS}$-Theorie gibt YANNAKAKIS [Yan97].

Allerdings verhält sich beispielsweise das wohl verbreitetste Verfahren der lokalen Suche, der auf (DANZIG [Dan49]) zurückgehende Simplex-Algorithmus der linearen Optimierung, trotz exponentieller Zeitkomplexität (siehe KLEE und MINTY [KM72]) in der Praxis gutartig: Erfahrungsgemäß ist die Anzahl der Iterationen sogar etwa linear bezogen auf die Eingabegröße, was BORGWARDT [Bor87] durch den Nachweis einer polynomiellen *durchschnittlichen* Laufzeit unter einem naheliegenden Wahrscheinlichkeitsmodell untermauert. (Für das Problem LINEAR PROGRAMMING gibt es allerdings polynomielle Algorithmen, die auf anderen Methoden beruhen, vergleiche KHACHIAN [Kha79], KARMARKAR [Kar84] und GRÖTSCHEL, LOVÁSZ und SCHRIJVER [GLS88] – auch wenn man in der Praxis oft dem Simplex-Algorithmus den Vorzug gibt.)

Die Qualität lokaler Optima. Wenn man bei $\mathcal{NP}$-schweren kombinatorischen Optimierungsproblemen auf lokale Suche zurückgreift, verzichtet man auf die Optimalität der Lösung, um dafür einen, wie oben gesehen, nicht notwendig polynomiellen, aber doch in der Praxis schnellen Algorithmus zu erhalten. Während lokale Verbesserungsverfahren wie die LIN-KERNIGHAN-Heuristik für das TRAVELING SALESPERSON Problem oder die KERNIGHAN-LIN-Heuristik für GRAPH PARTITIONING in der Praxis exzellente Lösungen produzieren, liegen keine rigorosen Resultate über ihre Qualität vor.

Zwei frühe Resultate von PAPADIMITRIOU und STEIGLITZ [PS77] beziehen sich auf das TRAVELING SALESPERSON Problem.

Satz 2.1.4 (PAPADIMITRIOU, STEIGLITZ 1977)
Kein lokaler Suchalgorithmus für das TRAVELING SALESPERSON *Problem mit Dreiecksungleichung, der pro Iterationsschritt polynomielle Zeitkomplexität hat, kann exakt sein, d.h. Optimalität garantieren, außer es ist* $\mathcal{P} = \mathcal{NP}$*.*

Der Grund dafür ist, daß das Entscheidungsproblem, ob eine vorgelegte TSP-Tour optimal ist, co-$\mathcal{NP}$-vollständig ist. Tatsächlich konstruieren sie TRAVELING SALESPERSON Probleminstanzen, die unter der 2-Opt-Nachbarschaft eine eindeutige optimale Lösung besitzen, aber exponentiell viele zweitbeste Lösungen, die zum einen beliebig schlecht und zum anderen durch den Austausch von $o(n)$ Kanten nicht verbessert werden können [PS78]. (Bei Instanzen, die die Dreiecksungleichung erfüllen, ist der relative Abstand der zweitbesten Lösung von der Optimallösung allerdings $o(1)$.)

Allgemeiner bewiesen RÖDL und TOVEY [RT87] für das MAXIMUM INDEPENDENT SET Problem, daß es eine Familie von Graphen $\{G_n : n \in \mathbb{N}\}$ gibt, die für alle Verfahren der lokalen Verbesserung, deren Nachbarschaften polynomiell beschränkt sind (d.h. $\max_{s \in \mathcal{S}} |N_I(s)|$ polynomiell beschränkt in $|I|$), schwere Instanzen sind; genauer: für alle polynomiell beschränkten Nachbarschaften besitzen fast alle Permutationen von G_n exponentiell viele strikte lokale Maxima, die nicht globale Maxima sind.

Um die Qualität von Heuristiken für $\mathcal{NP}$-vollständige Probleme zu messen, wurde das Konzept der Approximationsalgorithmen eingeführt, siehe HOCHBAUM [Hoc96] für einen Überblick. Man sagt, ein Algorithmus (o.B.d.A. für ein Minimierungsproblem) hat Güteratio $r \in \mathbb{Q}$, wenn die Kosten der von ihm produzierten Lösungen höchstens r-mal so groß wie die Kosten einer optimalen Lösung sind. Für die beiden oben erwähnten Probleme, TRAVELING SALESPERSON und MAXIMUM INDEPENDENT SET, ist allerdings für jedes konstante $r \in \mathbb{Q}$ schon das Approximationsproblem $\mathcal{NP}$-schwer, siehe SAHNI und GONZALEZ [SG76] bzw. ARORA et al. [ALM+98].

Für $\mathcal{PLS}$-Probleme ergibt sich die folgende Verallgemeinerung von Satz 2.1.4.

Satz 2.1.5 *Sei Π ein Optimierungsproblem und $\mathcal{N}$ eine Nachbarschaft auf den Lösungen seiner Instanzen, so daß das zugehörige lokale Suchproblem in $\mathcal{PLS}$ liegt. Dann gilt:*

- *Wenn Π [streng] $\mathcal{NP}$-schwer ist, so kann $\mathcal{N}$ nicht exakt sein, außer es gilt $\mathcal{NP} = co-\mathcal{NP}$ [$\mathcal{P} = \mathcal{NP}$].*

- *Wenn das Problem, Π auf einen Faktor r zu approximieren, [streng] $\mathcal{NP}$-schwer ist, so kann $\mathcal{N}$ keine Güte r garantieren, außer es gilt $\mathcal{NP} = co-\mathcal{NP}$ [$\mathcal{P} = \mathcal{NP}$].*

Einfache Beispiele polynomieller lokaler Suchalgorithmen mit Gütegarantie sind die 1-OPT-Verfahren für das ungewichtete MAX CUT Problem und für DEGREE-BOUNDED MAXIMUM INDEPENDENT SET, die Güteratio 2 bzw. $\Delta(G)$ haben. Für das EUCLIDEAN TRAVELING SALESPERSON Problem unter der 2-OPT-Nachbarschaft beweisen CHANDRA, KARLOFF und TOVEY [CKT94], daß die erwartete Anzahl von Iterationen der (greedy-mäßigen) lokalen Suche für zufällige Instanzen im Einheitsquadrat $\mathcal{O}(n^{16} \log n)$ und die erwartete Güteratio $\mathcal{O}(1)$ ist (das Euklidische TSP besitzt allerdings auch ein polynomielles Approximationsschema, siehe ARORA [Aro97]).

2.2 Eine Nachbarschaft für das Pairing-Problem

Lokale Verbesserungsverfahren. Baker et al. [BBFP79] entwerfen graphentheoretische Konstruktions- und Verbesserungsheuristiken für das Pairing-Problem. Die Konstruktionsheuristik verkettet greedy-artig Flüge mit bereits konstruierten partiellen Pairings, beginnend mit der Menge aller die Crew-Basis verlassender Flüge. Die Verbesserungsheuristik wählt einen Flughafen aus und reoptimiert die Verbindungen von Pairings, die diesen Flughafen kreuzen, durch das Lösen eines Matching-Problems. Die Nicht-Linearität, die durch Pairings entsteht, die den Flughafen mehrfach kreuzen, wird nicht thematisiert. Da das betrachtete Testproblem nur eine Basis hat, entsteht kein Problem mit Crew-Kapazitäten. Der Erfolg der Methode, so wird eingestanden, könnte allerdings in der speziellen Struktur des zugrundeliegenden Flugnetzwerks von Federal Express liegen.

Ball und Roberts [BR85] entwickeln ein zweiphasiges Verfahren, das ebenfalls Algorithmen für Matching verwendet, um den Verbindungsgraphen in Pfade zu zerlegen. Proceedings werden erst in einer Postprocessing-Routine eingefügt. Die erste Phase konstruiert eine zulässige Pfadpartition. Für jeden Knoten des Verbindungsgraphen wird zunächst die Länge des längsten gerichteten Pfades zu diesem Knoten, sein Level, berechnet. Beginnend mit den Knoten auf Level 0 werden sodann im i-ten Schritt Pairings, die an Knoten auf dem Level i enden, um Anschlußflüge verlängert, sofern diese nicht bereits in ein anderes Pairing integriert sind und sofern die Verkettung in ein zulässiges Pairing mündet. Dieses Problem läßt sich als ein bipartites gewichtetes Matching-Problem beschreiben und lösen. Die zweite Phase besteht aus einer Verbesserungsheuristik, die iterativ Matching-Probleme auf der Menge der Pairings löst. Die Kosten einer zwei Pairings verbindenden Kante repräsentieren hierbei die Verbesserung, die möglich ist, wenn man die beiden Pairings verkettet und das erhaltene Pairing an einer anderen Stelle wieder aufspaltet.

Eine Nachbarschaftsstruktur für Metaheuristiken. Eine Lösung des Crew-Scheduling-Problems ist eine Partition der Menge der Flüge in (nicht notwendig zulässige) Pairings zusammen mit einer Funktion, die jedem dieser Pairings eine Basis zuordnet. Wir definieren die Nachbarschaft einer solchen Lösung als die Menge aller im *conn*-Graphen möglichen (d.h. insbesondere zeitlich und räumlich konsistenten und das Crew-Komplement berücksichtigender) Verkettungen zweier Pairings zu einem Pairing sowie aller Aufspaltungen, d.h. Aufbrechen eines Pairings nach einem Flug zu zwei Pairings. Diese Wahl ist insofern natürlich, als Pairings in der Regel kleine Abschnitte von Flugzeugrotationen sind oder aus solchen aufgebaut sind. Wir bezeichnen dies als eine Run-Cutting-Formulierung in Anlehnung an den Prozeß des „Run-Cuttings" in der Fahrdienstplanung des öffentlichen Nahverkehrs, wo aus den Umläufen der Bus- bzw. Straßenbahnwagen Dienststücke herausgeschnitten werden, die wiederum, ggf. durch Pausen getrennt, zu Tagesschichten für die Fahrer zusammengefügt werden. Bei unseren Testproblemen haben „vernünftige Lösungen" etwa 3300 (DH8) bzw. 22.000 (X) Nachbarn. Die Laufzeit der Funktion, die die Kostendifferenz zwischen einer Lösung und einer ihrer Nachbarn berechnet, ist proportional zur Anzahl der Flüge des aufgespaltenen bzw. des verketteten Pairings und zur Anzahl der von ihm berührten Tage, also $\mathcal{O}(1)$.

Die Zuordnung zu den Basen. Um die Größe der Nachbarschaft klein zu halten, wurden keine Übergänge definiert, die dem Wechsel der Basis eines Pairings entsprechen. Vielmehr werden bei jedem Übergang (Verkettung oder Aufspaltung) für die entstehenden Pairings die lokal optimalen Basen bestimmt. Die gewählte Nachbarschaft modelliert also nicht die Zuordnung von Pairings zu Basen, sondern lediglich die Pfadzerlegung des *conn*-Graphen. Man beachte, daß dadurch der Nachbarschaftsgraph i.a. nicht mehr die Eigenschaften der Symmetrie und des starken Zusammenhangs aufweist. Wenn man ein Pairing in zwei Pairings aufspaltet, so führt die „inverse“ Operation, die die beiden entstandenen Pairings wieder miteinander verkettet, nicht notwendig wieder zur ursprünglichen Lösung, da die Basis des entstehenden Pairings neu berechnet wird. Allerdings sind die Kosten dieser Lösung höchstens so groß wie die der ursprünglichen Lösung. In der umgekehrten Richtung, wenn zunächst zwei Pairings verkettet und anschließend an derselben Stelle wieder aufgespalten werden, können sich die Kosten i.a. sogar erhöhen, weil die Kosten der Pairings, die durch die Wahl der Basis entstehen, über die Crew-Verteilungskosten gekoppelt sind und der Algorithmus zunächst das vordere Pairing lokal optimal zuordnet und dann das hintere Pairing.

Unsere Intuition war, daß sich aufgrund der vielen zufällig gewählten Übergänge bei einer Metaheuristik eine gute Basis-Zuordnung von selbst einstellt. Um dies zu überprüfen, wurde eine Variante implementiert, die regelmäßig (z.B. nach jedem Durchgang durch die Nachbarschaft) die Zuordnung der Pairings zu den Basen nachoptimiert. In dieser zwischengeschalteten Prozedur, genannt OPT_HOMEBASES, wird zunächst eine zufällige Permutation der Pairings bestimmt. In der durch die Permutation gegebenen Reihenfolge wird sodann für jedes Pairing die jeweils lokal optimale Basis bestimmt und zugewiesen. Abbildung 2.1 zeigt, daß sich die Lösungsqualität dadurch sogar tendenziell eher verschlechtert, während sich die Laufzeit geringfügig erhöht (jeder Punkt repräsentiert hier einen Lauf der Metaheuristik Simulated Annealing).

Die Bestimmung einer optimalen Zuordnung von Pairings zu Basen ähnelt einem GENERALIZED ASSIGNMENT Problem, vergleiche CATTRYSSE und VAN WASSENHOVE [CW92], MARTELLO und TOTH [MT92] und OSMAN [Osm95]. Es gelang uns jedoch nicht, den komplexitätstheoretischen Status des Problems zu bestimmen. Das oben ausgeführte heuristische Vorgehen rechtfertigt sich allerdings dadurch, daß es für ein lokales Suchverfahren von großer Bedeutung ist, bei einem Übergang die Kosten – für deren Berechnung man in unserem Fall die Basis-Zuordnung benötigt – in konstanter Zeit aktualisieren zu können. Durch eine Verkettung oder eine Aufspaltung kann sich eine optimale Basis-Zuordnung jedoch auf linear vielen Pairings ändern.

Bemerkungen.

(1) Auch die Algorithmen für ganzzahlige lineare Optimierung mit Branch&Cut-Methodik verzweigen danach, ob Flug i an Flug j angehängt wird oder nicht, anstatt nach den Entscheidungsvariablen, ob ein Pairing in die Lösung aufgenommen wird oder nicht, vergleiche z.B. JOHNSON, NEMHAUSER und SAVELSBERGH [JNS97].

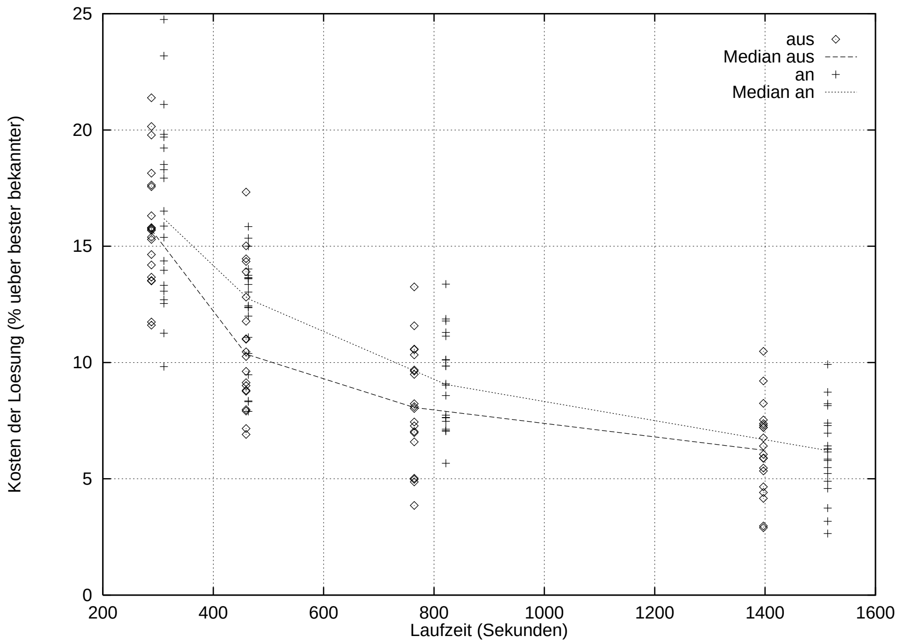

Abbildung 2.1: Der Effekt von Opt_Homebases auf Simulated Annealing (DH8)

(2) Für lokale Suchverfahren ist es von Vorteil, wenn es denn speichertechnisch vertretbar ist, ein Feld vorliegen zu haben, das alle Übergänge zu Nachbarn der aktuellen Lösung bereithält:

- Wenn man die möglichen Übergänge der aktuellen Lösung beispielsweise aus dem Verbindungsgraph berechnen wollte, müßte man die Kanten des Verbindungsgraphen, die weder einer Verkettung noch einer Aufspaltung entsprechen, herausfiltern. So muß man lediglich das Feld nach einem erfolgten Übergang lokal aktualisieren, was bei einem Annealing mit durchschnittlicher Akzeptanzrate von 1% über den gesamten Lauf, beim Verfahren des Steilsten Abstieges oder bei Tabu-Suche günstiger sein kann.
- Bei randomisierten Algorithmen kann ein Übergang zufällig und gleichverteilt ausgewählt werden.
- Man kann die (Gesamt-) Kosten eines vorgeschlagenen Übergangs speichern, um sie nicht neu berechnen zu müssen, wenn der Übergang beim ersten Mal nicht akzeptiert, aber später noch einmal vorgeschlagen wird. In unserem Fall werden diese Kostenwerte allerdings hinfällig, sobald ein Übergang akzeptiert wird, da sich die Crew-Nachfrage ändert und damit die Kosten von i.a. linear vielen anderen Übergängen. Trotzdem kann sich das lohnen, wenn die Auswertung der Kostenfunktion teuer, d.h. rechenzeitintensiv, ist. In unserem Fall ergab sich z.B. beim DH8-Datensatz ein Rechenzeitgewinn von ca. 15%, vergleiche Emden-Weinert und Proksch [EP98].

In unserem Fall werden zur Erstellung des Feldes aller Übergänge die Kanten im Verbindungsgraphen *conn* daraufhin überprüft, ob sie Verkettungen, Aufspaltungen oder nichts davon darstellen. Die Datenstruktur *conn* ermöglicht ebenso die effiziente Aktualisierung dieses Feldes nach einem erfolgten Übergang. Hierzu sind lediglich die mit den beteiligten Flügen inzidierenden Kanten im Verbindungsgraphen zu untersuchen und zugehörige Übergänge (im Falle einer Aufspaltung) dem Feld der Übergänge der aktuellen Lösung hinzuzufügen bzw. (im Falle einer Verkettung) aus dem Feld zu entfernen.

2.3 Simulated Annealing

Die Idee von Simulated Annealing geht auf METROPOLIS et al. [MRR+53] zurück, die allerdings noch nicht die Temperatur variierten. Anfang der 80er Jahre übertrugen KIRKPATRICK, GELATT und VECCHI [KGV83] sowie ČERNY [Čer85] das Konzept auf die kombinatorische Optimierung. Die Idee ist, auch Übergänge zuzulassen, die die Lösung verschlechtern, dies jedoch in Abhängigkeit davon, wie groß die Verschlechterung ist (je größer die Verschlechterung, desto unwahrscheinlicher der Übergang) und in Abhängigkeit von der Zeit: Zu Beginn des Annealings werden große Verschlechterungen eher akzeptiert, gegen Ende nur noch mit einer kleinen Wahrscheinlichkeit. Ausführliche Darstellungen des Verfahrens findet man bei VAN LAARHOVEN und AARTS [LA87] sowie AARTS und KORST [AK89]. Daß Simulated Annealing seitdem eine solche Verbreitung gefunden hat, liegt nicht zuletzt darin begründet, daß es rigorose mathematische Konvergenzbeweise gibt (einen Überblick geben hier ROMEO und SANGIOVANNI-VINCENTELLI [RS91] sowie AARTS, KORST und VAN LAARHOVEN [AKL97]). Sie beziehen sich jedoch auf Kühlschemata, die nicht von praktischer Relevanz sind.

Anwendungen. Die Simulation von MARKOV-Ketten hat sich insbesondere für kombinatorische Zählprobleme, stellvertretend sei hier POLYTOPE VOLUME genannt, als äußerst erfolgreich erwiesen, siehe JERRUM und SINCLAIR [JS97]. Andere positive Resultate betreffen den METROPOLIS Prozeß für GRAPH BISECTION, siehe JERRUM und SORKIN [JS93], oder approximatives kardinalitätsmaximales Matching (ein Problem, das allerdings in $\mathcal{P}$ liegt), siehe SASAKI und HAJEK [SH88]. Mit einem dem Simulated Annealing ähnlichen Sampling-Verfahren zeigen NOLTE und SCHRADER [NS97], daß fast alle zufälligen 3-färbbaren Graphen in sublinearer Zeit mit hoher Wahrscheinlichkeit gefärbt werden können. Es gibt allerdings einen deterministischen Algorithmus, der 3-färbbare Graphen in erwartet linearer Zeit färbt, siehe PRÖMEL und STEGER [PS92].

Auf der negativen Seite konnte gezeigt werden, daß Probleme wie MAXIMUM CLIQUE (JERRUM [Jer92]), 3-COLORING (NOLTE und SCHRADER [NS96]) und MAXIMUM CARDINALITY MATCHING (SASAKI und HAJEK [SH88]) im worst-case-Sinne nicht zugänglich für den METROPOLIS Prozeß bzw. für Simulated Annealing sind.

Einen Überblick über die zahlreichen Anwendungen von Simulated Annealing im Operations Research findet man bei KOULAMAS, ANTONY und JAEN [KAJ94] sowie OSMAN und LAPORTE [OL96].

Simulated Annealing für das Crew-Scheduling-Problem. Ein Kühlschema definiert, wie die Größe, die beim Annealing die Akzeptanzwahrscheinlichkeit für verschlechternde Übergänge steuert, die sogenannte Temperatur T, im Laufe der Zeit erniedrigt wird. Für $T = 0$ entspricht der Algorithmus einem lokalen Verbesserungsverfahren. Wir verwenden das geometrische Kühlschema nach JOHNSON et al. [JAMS89, JAMS91], da es eine recht einfache Steuerung der Laufzeit und damit der Lösungsqualität erlaubt. In unseren Studien produzierte es zudem Lösungen, die denen des adaptiven Kühlschemas nach AARTS and KORST [AK89] eher noch überlegen waren, siehe EMDEN-WEINERT und PROKSCH [EP98]. Das geometrische Kühlschema, vergleiche Abbildung 2.3, wird von den folgenden generischen Parametern kontrolliert. Die Starttemperatur T_0 wird so gewählt, daß auf dem ersten Temperaturlevel eine gewisse Rate *init_acc_ratio* von vorgeschlagenen (d.h. ausgewürfelten und bewerteten) Übergängen akzeptiert wird. T_0 hängt stark vom Problem und der Probleminstanz ab, so daß die Temperatur zunächst solange verdoppelt wird, bis die Akzeptanzrate auf einem Temperaturlevel *init_acc_ratio* erreicht hat. Die Länge eines Temperaturlevels wird auf ein Vielfaches L der (mittleren) Größe der Nachbarschaft gesetzt, um auf jedem Temperaturlevel allen möglichen Modifikationen mit einer gewissen Wahrscheinlichkeit auszuprobieren (wir haben $L = 60$ gesetzt). Im Hinblick auf die Herkunft des Verfahrens aus der statistischen Mechanik sagt man auch, das System solle auf jedem Temperaturlevel sein thermodynamisches Äquilibrium erreichen. Der Parameter α, die Temperaturreduktion, bestimmt, wie schnell die Temperatur und damit die Akzeptanzwahrscheinlichkeit für verschlechternde Übergänge erniedrigt wird; wir haben α auf $0.95^{1/i}$ für $i \in \mathbb{N}$ gesetzt. Die Parameter *min_acc_ratio* und *max_no_record* dienen dazu, den Prozeß zu terminieren. Wenn die Akzeptanzrate auf einem Temperaturlevel unter *min_acc_ratio* gefallen ist und die beste bisher gefundene Lösung auf den letzten *max_no_record* Temperaturleveln nicht verbessert werden konnte, wird die `REPEAT`-Schleife beendet.

Durch ein langsameres Kühlen kann die Qualität der Lösung um den Preis einer erhöhten Laufzeit verbessert werden, vergleiche Abbildung 2.2. Man beachte die hohe Standardabweichung von bis zu 2% in den Kosten der Lösungen, die unter derselben Parametereinstellung berechnet wurden. Für den größeren Testdatensatz X beträgt sie noch immer fast 1%.

Eine problemspezifische Startlösung. Es wird oft berichtet, daß Simulated Annealing sehr viel Rechenzeit erfordert. Beim Pairing-Problem erlaubt eine von einer problemspezifischen Heuristik, genannt LINEAR_CHAINER, konstruierte Startlösung, eine niedrigere Starttemperatur für das Annealing zu verwenden und dadurch gegenüber der Singleton-Startlösung, bei der jeder Flug ein Pairing für sich selbst darstellt (diese Lösung liegt nach Initialisierung der Datenstrukturen vor), 30% Laufzeit einzusparen – ohne bei der Lösungsqualität einen Kompromiß einzugehen. Die verwendete Konstruktionsheuristik LINEAR_CHAINER geht die Flüge in Reihenfolge ihrer Ankunftszeit durch und verkettet jeden Flug mit dem nächsten, den Ankunftsflughafen wieder verlassenden Flug genau dann, wenn dieser Flug der erste Flug eines Pairings ist und das Pairing, welches durch die Verkettung entsteht, zulässig ist. Anschließend werden für die entstandenen Pairings greedy-artig Basen ausgewählt und mittels OPT_HOMEBASES lokal optimiert. Diese Heuristik nimmt bei der Verkettung

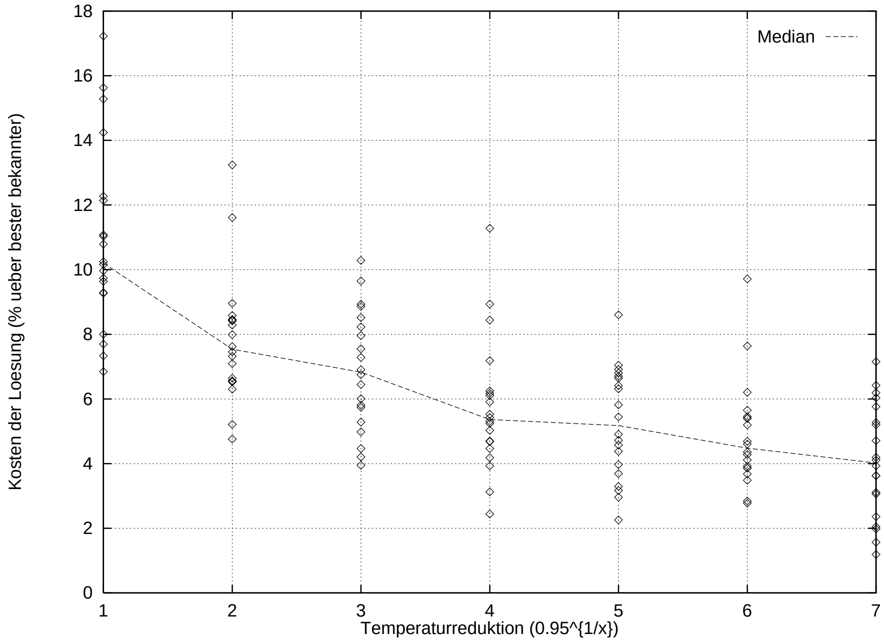

Abbildung 2.2: Die Abhängigkeit der Lösungsqualität von α (DH8)

```
PROCEDURE SIMULATED_ANNEALING;
BEGIN
  konstruiere eine Startlösung x ∈ S;
  T := T_0;
  REPEAT
    {Temperaturlevel}
    FOR i := 1 TO L DO
      wähle eine Lösung y ∈ N(x) gleichverteilt zufällig;
      Δ := c(y) − c(x);
      IF (Δ <= 0) THEN
        x := y
      ELSE {Verschlechterung}
        mit Wahrscheinlichkeit e^{−Δ/T} setze x := y;
    END; {for}
    T := α · T;
  UNTIL (kein neuer Rekord seit einer langen Zeit);
  return (beste gefundene Lösung);
END;
```

Abbildung 2.3: Der Algorithmus SIMULATED-ANNEALING der Flüge zu Pairings keine Rücksicht auf die Proceeding-Kosten; diese Kosten werden jedoch bereits im ersten Temperaturlevel des Annealings drastisch reduziert. Abbildung 2.4 zeigt den Vergleich dreier Startlösungen für das Simulated Annealing: die Singleton-Lösung, die von LINEAR_CHAINER konstruierte Lösung sowie die durch die Verbesserungsheuristik STEILSTER_ABSTIEG bearbeitete Singleton-Lösung.

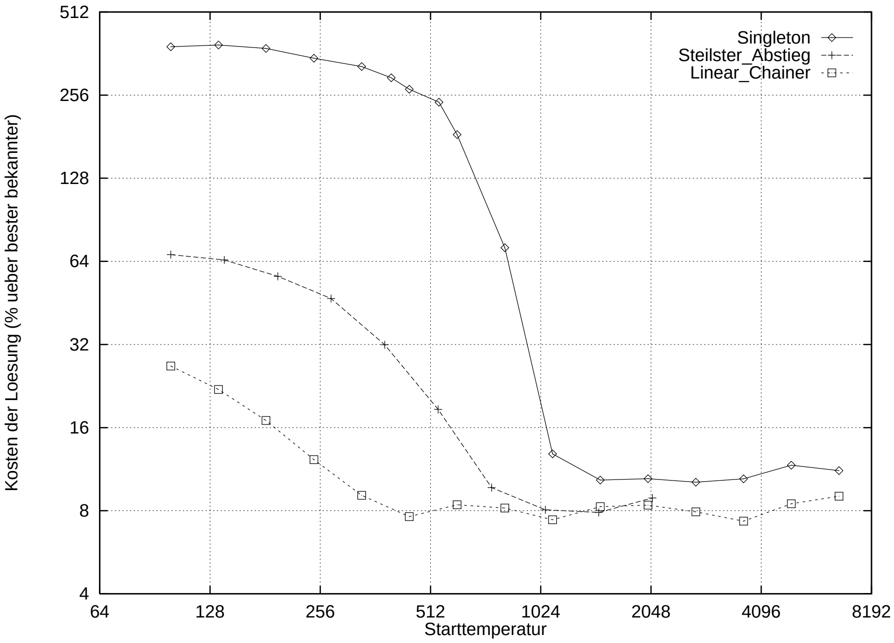

Abbildung 2.4: Die Wahl der Startlösung und der Starttemperatur (DH8)

Abgetragen ist der Median der Kosten der vom Annealing produzierten Lösungen in Abhängigkeit von der Starttemperatur. Beide Achsen sind logarithmisch geteilt. Für alle drei Startlösungen gibt es eine Starttemperatur, oberhalb derer sich die Qualität der Lösungen des Annealings nicht mehr verbessert. Mit kleineren Starttemperaturen degeneriert das Annealing immer mehr zu einem lokalen Verbesserungsverfahren und die Lösungsqualität verschlechtert sich drastisch. Man beachte, daß die Laufzeit des Annealings in etwa proportional zum Logarithmus der Starttemperatur ist. Die besten Resultate werden mit der von LINEAR_CHAINER konstruierten Startlösung und einer Startakzeptanzrate von etwa 5% erzielt, ein niedriger Wert verglichen mit der Standardeinstellung von 40% bei JOHNSON et al. [JAMS89, JAMS91] für zufällige Startlösungen. JOHNSON et al. berichten über ähnliche Erfolge mit einer Konstruktionsheuristik beim GRAPH PARTITIONING Problem und geometrischen Instanzen. Bei zufälligen Probleminstanzen scheint eine gute Startlösung dagegen nichts zu nützen. Interessanterweise werden in unserem Fall, ausgehend von der von LINEAR_CHAINER konstruierten Startlösung, auch bei höheren Starttemperaturen bessere Lösungen produziert als bei der Singleton-Startlösung.

Während die für LINEAR_CHAINER erforderliche Rechenzeit vernachlässigbar ist, liegt die Laufzeit der Verbesserungsheuristik STEILSTER_ABSTIEG im Minutenbereich. Wie die Grafik zeigt, benötigt das Annealing mit der von STEILSTER_ABSTIEG produzierten Lösung als Startlösung zudem eine höhere Starttemperatur (wohl, um dem schlechten lokalen Optimum wieder zu entfliehen). Es lohnt also nicht, eine Startlösung für das Annealing mit einer lokalen Verbesserungsheuristik, die auf derselben Nachbarschaft operiert, aufzubereiten.

2.4 Wasserspiegel-Algorithmen

Ein neues Akzeptanz-Kriterium für lokale Suche hat DUECK [Due93] vorgeschlagen. Anstatt die Akzeptanz einer Nachbarlösung von der Differenz in der Kostenfunktion zur aktuellen Lösung abhängig zu machen, gibt es eine globale Schranke, den sogenannten Wasserspiegel, an die Kosten einer akzeptablen Lösung. Wann immer die Kosten einer vorgeschlagenen Nachbarlösung nicht größer als der Wasserspiegel sind, wird der Übergang akzeptiert. Im Laufe des Algorithmus wird der Wasserspiegel dann langsam gesenkt, wie während einer Trockenzeit nach einer Sintflut. Die Trajektorie einer solchen lokalen Suche im Lösungsraum kann dann mit dem Weg eines Fisches im Wasser verglichen werden.

Es gibt zwei Varianten, den Wasserspiegel abzusenken. RECORD-TO-RECORD-TRAVEL, vergleiche Abbildung 2.6, setzt den Wasserspiegel jedesmal, wenn ein neuer Rekord (eine neue beste Lösung) gefunden wird, auf einen Faktor *margin* mal den Kosten der neuen Rekordlösung. GREAT_DELUGE, vergleiche Abbildung 2.11, reduziert den Wasserspiegel jedesmal, wenn ein Übergang akzeptiert wird, um einen festen Anteil *cutoff* des Abstands der neuen aktuellen Lösung zum Wasserspiegel.

Die Erfahrungen mit Wasserspiegel-Algorithmen sind recht begrenzt. JOHNSON and MCGEOCH [JM97] berichten über vielversprechende Ergebnisse mit einer Variante des GREAT_DELUGE Algorithmus für das TRAVELING SALESPERSON Problem. SINCLAIR [Sin93] vergleicht verschiedene Metaheuristiken für ein spezielles QUADRATIC ASSIGNMENT Problem. Eine Bewertung der präsentierten Ergebnisse ist jedoch schwierig, da keine Kostenwerte für die von den verschiedenen Methoden produzierten Lösungen auf der Basis gleicher Laufzeit gegeben werden. Genauso wenig gibt es Informationen über die Laufzeiten bei Lösungen mit vergleichbaren Kosten. Schließlich ist nicht klar, ob die Algorithmen von derselben Startlösung aus starten; Tabu Suche, der erklärte Gewinner, benutzt, wie berichtet wird, eine von einer problemspezifischen Heuristik produzierte Startlösung.

Record-to-Record-Travel. Mittels des Parameters *margin* kann hier die Lösungsqualität der von RECORD-TO-RECORD-TRAVEL produzierten Lösungen ähnlich wie mittels α bei SIMULATED_ANNEALING beeinflußt werden, siehe Abbildung 2.5. Je größer der Wert *margin*, um so mehr Spielraum bleibt der lokalen Suche, da alle Lösungen mit Kosten zwischen *record* und $margin \cdot record$ akzeptiert werden, wobei *record* die Kosten der besten bisher gefundenen Lösung bezeichnet. Je mehr Spielraum wiederum, um so länger dauert die Suche und um so besser sind schließlich die Lösungen.

Zu Beginn setzen wir den Wasserspiegel etwas höher ($1.02 \cdot record$ anstatt von $margin \cdot record$), um die Suche nicht zu stark von der Startlösung abhängig zu

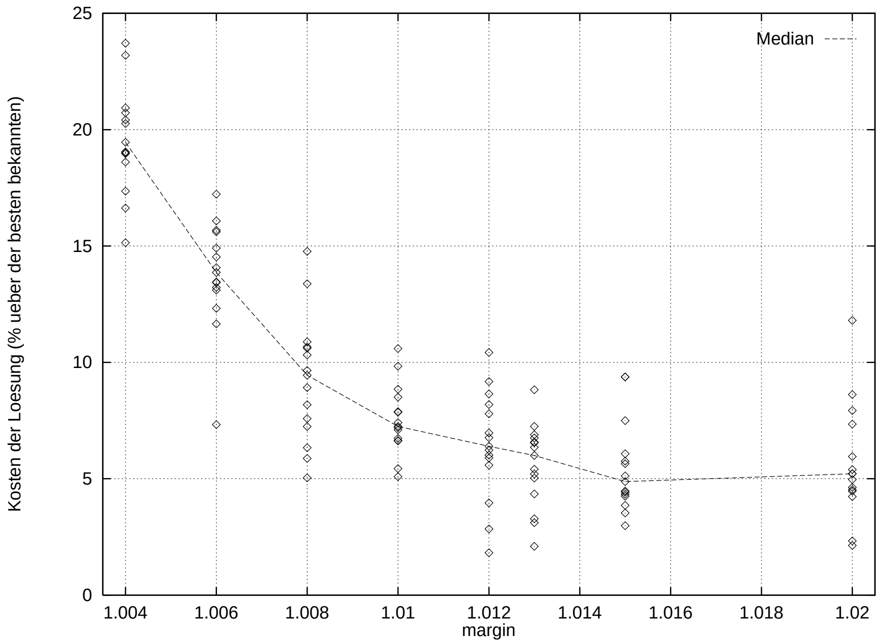

Abbildung 2.5: Die Abhängigkeit der Lösungsqualität von *margin* (DH8)

```
PROCEDURE Record_to_Record_Travel;
BEGIN
  konstruiere eine Startlösung x ∈ S;
  record := c(x);
  waterlevel := 1.02 · record;
  REPEAT
    wähle eine Nachbarlösung y ∈ N(x) gleichverteilt zufällig;
    IF (c(y) < waterlevel) THEN BEGIN
      x := y;
      IF (c(x) < record) THEN BEGIN
        record := c(x);
        waterlevel := margin · record;
      END;
    END;
  UNTIL (kein neuer Rekord seit einer langen Zeit);
  return (beste gefundene Lösung);
END;
```

Abbildung 2.6: Der Algorithmus Record_to_Record_Travel machen. Ein zu hoher Wert wiederum kann den Vorteil einer guten Startlösung zunichte machen bzw. unnötig Rechenzeit erfordern. Tatsächlich verwenden wir die folgende Modifikation des Algorithmus, die die Qualität der produzierten Lösungen konsistenter macht. Nach ($L = 60$) $\times$ (durchschnittliche Größe der Nachbarschaft) vorgeschlagenen Übergängen wird der Wasserspiegel um 0.2% gesenkt, um der (sonst relativ unproduktiven) Schlußphase des Prozesses, in der kaum mehr Rekorde gefunden werden, den Charakter einer lokalen Verbesserung zu geben.

Wir studierten auch eine Variante, die einen absoluten anstatt eines relativen Wertes für den Spielraum für die lokale Suche verwendet, d.h. der Wasserspiegel ist hier gegeben durch die Kosten der Rekordlösung plus *margin*. Auf den ersten Blick erscheint ein relativer Wert vielleicht günstiger weil problemunabhängig, doch unsere Erfahrung war, daß eine größere Probleminstanz tatsächlich einen kleineren Spielraum benötigt (der lediglich groß genug sein muß, um den lokalen Umbau einer Lösung zu gestatten). Wie die Abbildungen 2.7 und 2.8 zeigen, liefern die beiden Vorgehensweisen vergleichbare Resultate.

Johnson et al. [JAMS89, JAMS91] schlugen vor, den nächsten Nachbarn nicht gleichverteilt zufällig aus der Menge aller Nachbarn auszuwählen, sondern eine zufällige Permutation der Nachbarschaft zu bestimmen und die Nachbarn gemäß der Permutation nach und nach vorzuschlagen, bis einer der Nachbarn akzeptiert wird, in welchem Fall für die Nachbarschaft der neuen Lösung eine neue Permutation gewürfelt wird. Die Idee ist hier, insbesondere gegen Ende der Annealings, wenn sehr viele Übergänge verworfen werden, einen guten Nachbarn nicht zu „verpassen“. Während für die Crew-Scheduling-Anwendung diese Variante die Qualität des Annealings eher verschlechtert (weil Nachbarn erst wiederholt vorgeschlagen werden, nachdem die komplette Nachbarschaft bewertet und kein Nachbar akzeptiert wurde, und somit das Speichern der Kosten eines verworfenen Übergangs praktisch keinen Effekt mehr hat, siehe Emden-Weinert und Proksch [EP98]), zeigte dieses Vorgehen bei Record_to_Record_Travel einen deutlich positiven Effekt, zumindest bei der Berechnung von Lösungen hoher Qualität, siehe Abbildung 2.9. Ein Übergang beim Algorithmus Record_to_Record_Travel wird, wenn er einmal verworfen wird, auch später nie mehr akzeptiert werden, da das Akzeptanzkriterium nicht randomisiert ist, so daß es überflüssig ist, Übergänge wiederholt vorzuschlagen. Da wir die Kosten eines Übergangs speichern, falls er nicht ausgeführt wird, schlägt sich das wiederholte Vorschlagen von Übergängen nicht in der Laufzeit nieder. Unter Verwendung einer Permutation der Nachbarschaft erhöht sich aber die Akzeptanzrate, und das Verfahren bricht später ab. Dies erklärt unserer Meinung nach, warum sich die Laufzeiten bei sonst gleicher Parametereinstellung nicht signifikant unterscheiden, und die Verbesserung der Lösungsqualität. Man beachte, daß unter Verwendung von Zeitstempeln sowohl die Berechnung der i-ten Stelle einer Permutation wie auch die Initialisierung einer Permutation in konstanter Zeit möglich ist.

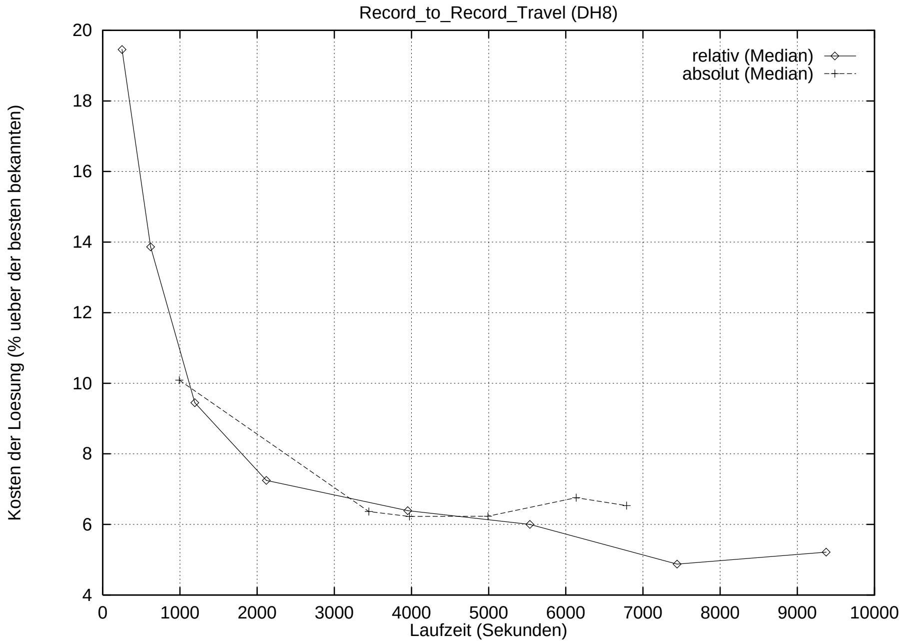

Abbildung 2.7: Relativer vs. absoluter Spielraum (DH8)

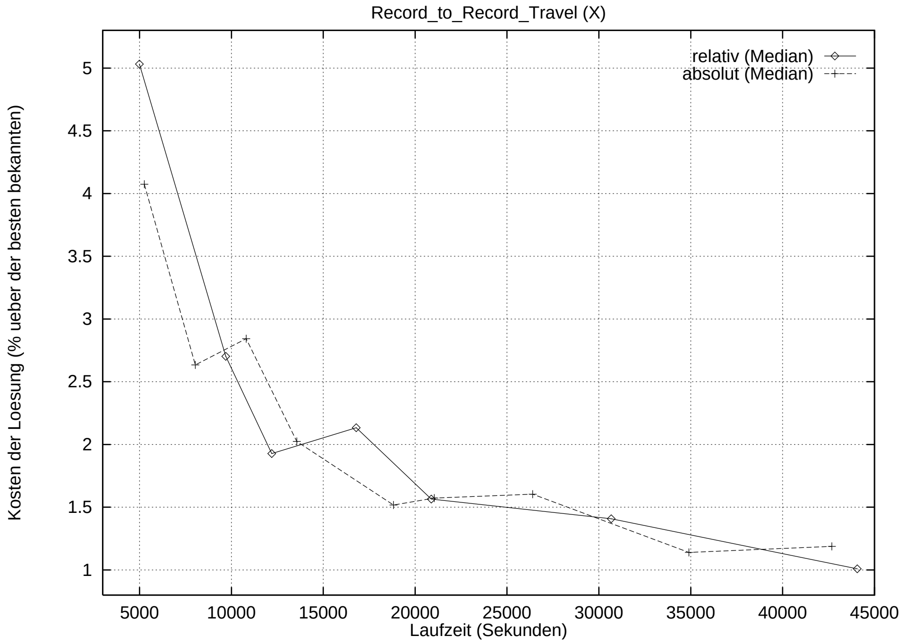

Abbildung 2.8: Relativer vs. absoluter Spielraum (X)

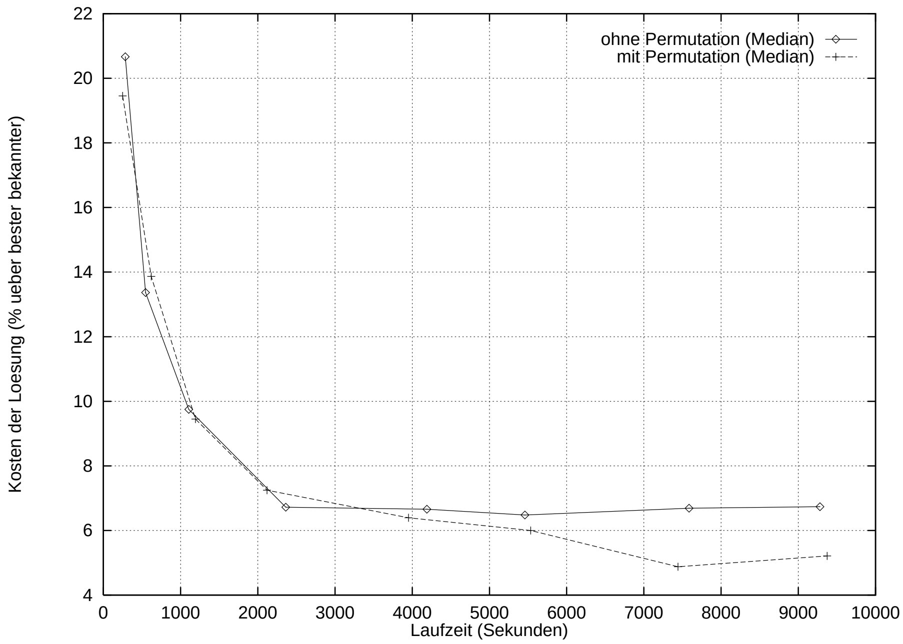

Abbildung 2.9: Permutation der Nachbarschaft (DH8)

Great Deluge. Beim Algorithmus GREAT_DELUGE wird die Höhe des Spielraums und damit die Lösungsqualität – zum Preis einer erhöhten Laufzeit – über den Parameter *cutoff* eingestellt, vergleiche Abbildung 2.10. Je kleiner *cutoff*, um so langsamer senkt sich der Wasserspiegel, um so mehr Zeit bleibt der lokalen Suche und um so besser ist die Lösungsqualität. Wir initialisieren den Wasserspiegel mit demselben Wert wie bei RECORD_TO_RECORD_TRAVEL und terminieren die Suche, wenn nach einem Durchgang durch die Nachbarschaft kein Übergang mehr akzeptiert wurde oder nach einer bestimmten Anzahl von Durchgängen durch die Nachbarschaft kein neuer Rekord mehr gefunden werden konnte. Die Nachbarn einer Lösung werden gemäß einer Permutation vorgeschlagen.

In DUECKs Version des Algorithmus GREAT_DELUGE wird der Wasserspiegel jedesmal, wenn ein Nachbar akzeptiert wird, um das Maximum eines festen Wertes und des Wertes, den wir verwenden, reduziert. Wir konnten auf diese Weise keine Lösungen hoher Qualität finden, weil der Prozeß gegen Ende zu schnell einfror.

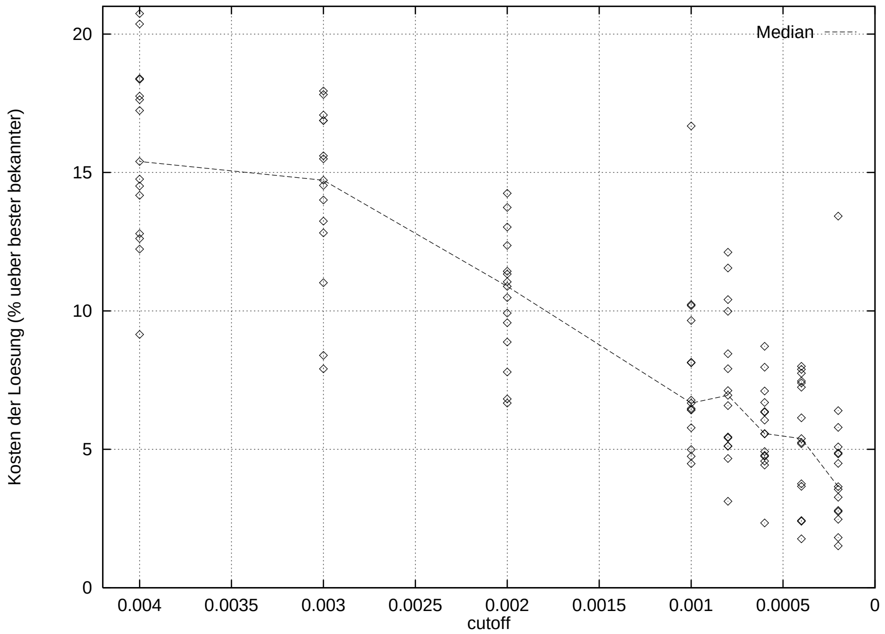

Abbildung 2.10: Die Abhängigkeit der Lösungsqualität von *cutoff* (DH8)

```
PROCEDURE GREAT_DELUGE;

BEGIN
  konstruiere eine Startlösung x ∈ S;
  waterlevel := 1.02 · c(x);
  REPEAT
    wähle eine Nachbarlösung y ∈ N(x) gleichverteilt zufällig;
    IF (c(y) < waterlevel) THEN BEGIN
      x := y;
      waterlevel := waterlevel − cutoff · (waterlevel − c(y));
    END;
  UNTIL (kein neuer Rekord seit einer langen Zeit);
  return (beste gefundene Lösung);
END;
```

Abbildung 2.11: Der Algorithmus GREAT_DELUGE

2.5 Tabu-Suche

So, wie die Idee des Simulated Annealings der Statistischen Mechanik entstammt, finden sich die Wurzeln der Tabu-Suche, begründet durch GLOVER [Glo89, Glo90], in der Künstlichen Intelligenz. Die grundlegende Idee hinter der Tabu-Suche ist, das Verfahren STEILSTER_ABSTIEG, das in jedem Schritt zu einem Nachbarn mit der bestmöglichen Kostendifferenz übergeht, über das lokale Optimum hinaus fortzusetzen und Zykeln durch ein langfristiges Gedächtnis zu verhindern zu versuchen. Idealerweise würde man sich hier bereits besuchte lokale Optimal merken wollen, weil das allerdings zu speicher- und rechenzeitintensiv ist, verbietet man lediglich die inversen Übergänge zu den letzten *tabu_list_length* vielen durchgeführten Übergängen, vergleiche Abbildung 2.12. Für einen Übergang μ bezeichne hierbei $\mu(x)$ den Nachbarn von x, zu dem μ führt, und μ^{-1} den zu μ inversen Übergang, der von $\mu(x)$ wieder zurück zu x führt.

```
PROCEDURE TABU_SUCHE;

BEGIN
  konstruiere eine Startlösung x;
  REPEAT
    berechne einen Übergang μ aus der Menge aller
        von x aus möglichen Übergänge M_x,
        so daß μ nicht tabu ist und c(μ(x)) minimal;
    x := μ(x);
    setze μ^{-1} tabu für die nächsten tabu_list_length Übergänge;
  UNTIL (kein neuer Rekord seit einer langen Zeit);
  return (beste gefundene Lösung);
END;
```

Abbildung 2.12: Der Algorithmus TABU_SUCHE

Unsere (elemementare) Implementation von Tabu-Suche enthält auch das folgende simple Aspirationskriterium: Wann immer ein Übergang zu einer neuen Rekordlösung führt, wird ggf. sein Tabustatus ignoriert. Die Bestimmung des besten, nicht verbotenen Übergangs macht offensichtlich in jedem Schritt die Bewertung der gesamten Nachbarschaft der aktuellen Lösung erforderlich, was in unserem Fall recht zeitaufwendig ist, da wir nach einem Übergang die Kosten von Nachbarn nicht inkrementell aktualisieren können. Eine Technik, dieses Problem zu umgehen, ist, stets nur mit einer Teilmenge der Nachbarschaft zu rechnen. Da es für unsere Anwendung schwierig erscheint, vielversprechende Übergänge auszuwählen, und weil wir um eines fairen Vergleichs willen den Tabu-Suche-Algorithmus nicht mit zusätzlichem problemspezifischen Wissen ausstatten wollten, wurde dieser Ansatz nicht weiter verfolgt. Dasselbe gilt für sogenannte Diversifikations- und Intensivierungstechniken, die die Suche in bestimmten Bereichen des Lösungsraums zyklisch verstärken oder beschneiden. Da Tabu-Suche vergleichsweise wenige Übergänge ausführt, reoptimieren wir allerdings mittels OPT_HOMEBASES (siehe Abschnitt 2.2) alle zehn vollzogene, d.h. akzeptierte Übergänge die Zuweisung von Basen. Aus diesem Grund ist unsere Tabu-Suche randomisiert, und die Lösungskosten weisen eine Variation auf.

Der einzige Parameter, der bei TABU_SUCHE eingestellt werden muß, ist *tabu_list_length*. Wie die Abbildungen 2.13 und 2.14 zeigen, muß dieser Parameter mit Bedacht gewählt werden, und die beste Wahl hängt von der Problemgröße ab. Es gibt bei TABU_SUCHE keinen Parameter, der es ermöglichen würde, mit mehr Laufzeit einen Qualitätsgewinn zu erkaufen. Die besten Werte für *tabu_list_length* korrespondieren in unserem Fall allerdings mit den längsten Laufzeiten. Die Laufzeit ist minimal für $tabu\_list\_length = 0$, erreicht ein Maximum und fällt dann wieder für $tabu\_list\_length \to \infty$. Man beachte, um wieviel TABU_SUCHE bei geeigneter Wahl von *tabu_list_length* das Verfahren des steilsten Abstiegs (d.h. TABU_SUCHE mit $tabu\_list\_length = 0$) übertrifft.

Eine ausführliche Darstellung von Tabu-Suche ist bei GLOVER, TAILLARD und DE WERRA [GTW93] und GLOVER und LAGUNA [GL97] zu finden.

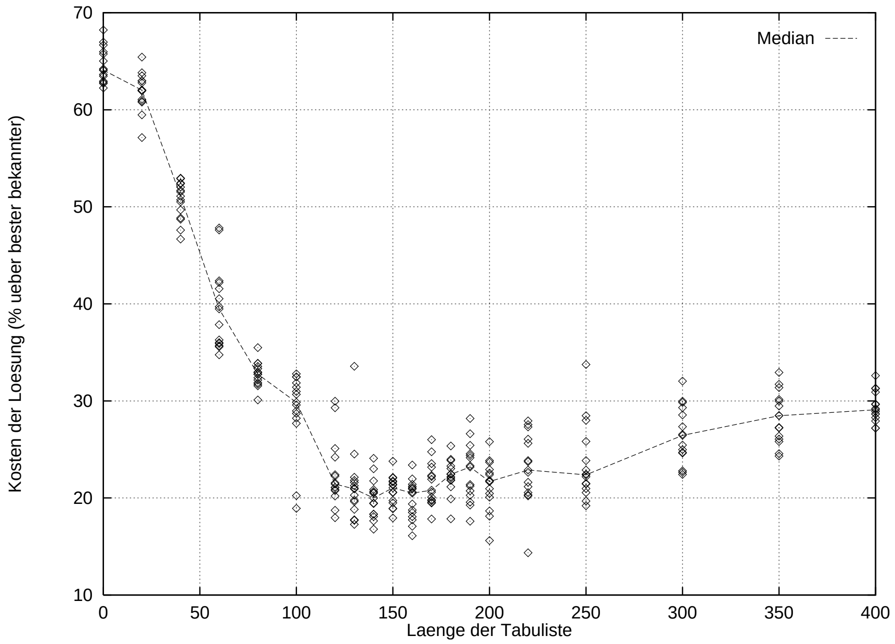

Abbildung 2.13: Lösungsqualität in Abhängigkeit von der Länge der Tabuliste (DH8)

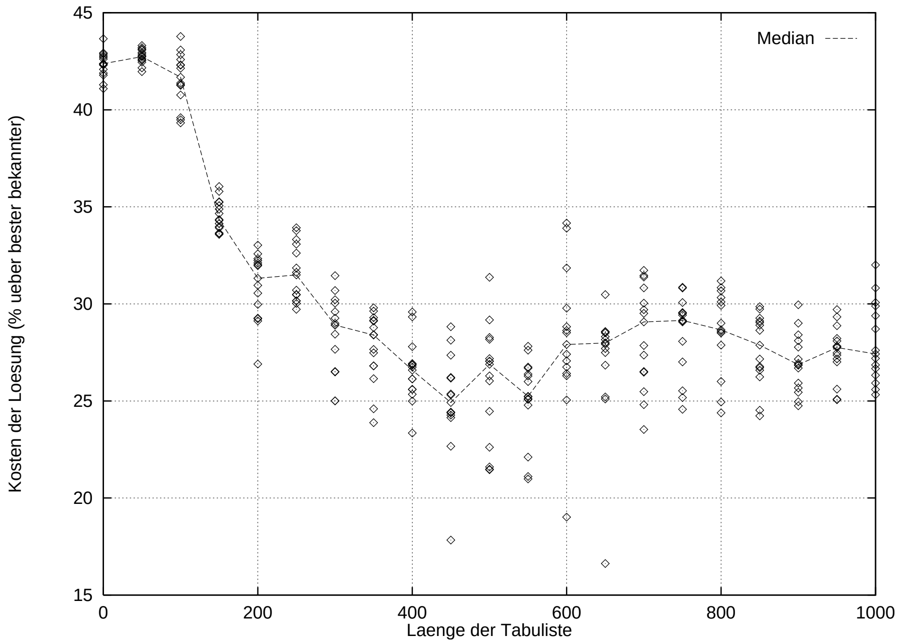

Abbildung 2.14: Lösungsqualität in Abhängigkeit von der Länge der Tabuliste (X)

2.6 Greedy Randomized Adaptive Search

Greedy-Randomized-Adaptive-Search Prozeduren (GRASP) sind von Feo und Resende [FR95] systematisch studiert worden. Die Idee ist hier, daß für einige Probleme, wo schnelle Konstruktions- und Verbesserungsheuristiken verfügbar sind, die Rechenzeit besser genutzt ist, wenn man, ausgehend von zufälligen Startlösungen, viele unabhängige Läufe der Verbesserungsheuristik macht als einen langen Lauf einer Metaheuristik. Johnson et al. [JAMS91] berichten, daß diese Technik für das Number Partitioning Problem, ein Problem mit „sehr tiefen“ lokalen Optima, Simulated Annealing übertrifft [JAMS91], während für das Traveling Salesperson Problem Simulated Annealing auf der Basis gleicher Laufzeit vor 2-Opt und bei großen zufälligen Euklidischen Instanzen auch vor 3-Opt liegt [JM97].

Für die Crew-Scheduling-Anwendung verwenden wir die Linear_Chainer Lösung als Startlösung, die aufgrund der nachfolgenden Optimierung der Basen mittels Opt_Homebases randomisiert ist. Auf diese Lösung wird ein Greedy-Verfahren angewendet, das in jedem Schritt unter allen verbessernden Übergängen zufällig einen Übergang mit Wahrscheinlichkeit proportional zur seiner Verbesserung auswählt, vergleiche Abbildung 2.15.

```
PROCEDURE GRASP;
BEGIN
  konstruiere eine zufällige Startlösung x;
  WHILE (x nicht lokal optimal)
  {
    wähle einen x verbessernden Übergang μ
    unter allen verbessernden Übergängen zufällig
    mit Wahrscheinlichkeit proportional zur Verbesserung;
    x := μ(x);
  }
  return (x);
END;
```

Abbildung 2.15: Der Algorithmus GRASP

	Minimum	Median	Maximum	Standardabweichung
DH8	65.1	67.5	70.3	0.8%
X	40.5	43.2	45.3	0.9%

Tabelle 2.1: Ergebnisse von GRASP

Für die Crew-Scheduling-Anwendung mit verschiedenen Nebenbedingungen und widerstrebenden Kostenarten war die Qualität der Greedy-Lösungen jedoch zu schlecht, ihre Varianz zu klein und ihre Laufzeit zu hoch, um konkurrenzfähige Ergebnisse erzielen zu können. Allerdings ist die Startlösung möglicherweise auch nicht „zufällig genug“, enthält sie doch stets dieselbe Pfad-Zerlegung des *conn*-Graphen. Tabelle 2.1 gibt die Lösungskosten in Prozent über der besten bekannten Lösung für eine Menge von jeweils 15 Testläufen an.

2.7 Eine vergleichende Studie

Wir wollen nun die in den vorhergehenden Abschnitten beschriebenen Metaheuristiken an Hand des Pairing-Problems vergleichen. Metaheuristiken liefern unter Umständen in Abhängigkeit von der aufgewendeten Laufzeit Lösungen unterschiedlicher Qualität. Ein Vergleich der Laufzeiten kann daher nur auf der Basis gleicher Qualität, ein Vergleich der Qualität nur auf der Basis gleicher Laufzeiten gezogen werden. Da beide Größen schwer einzustellen sind, haben wir für jede der Metaheuristiken eine ganze Versuchsreihe aufgenommen und in einer Graphik die Lösungsqualität gegen die Laufzeit aufgetragen. Die Abbildungen 2.16 für das Testproblem DH8 und 2.17 für das Testproblem X zeigen die Graphen im Vergleich. Jeder Punkt repräsentiert mindestens 15 Testläufe, wobei die Abszisse die durchschnittliche Laufzeit, die Ordinate den Median der Lösungskosten angibt. Alle Verfahren benutzen die LINEAR_CHAINER-Startlösung. Für einen Vergleich mit anderen Anwendungen bemerken wir, daß die Metaheuristiken etwa 6000 (DH8) bzw. 6600 (X) Iterationen (d.h. vorgeschlagene Übergänge) pro Sekunde ausführen. Während für DH8 SIMULATED ANNEALING dem RECORD-TO-RECORD-TRAVEL und dem Verfahren GREAT_DELUGE ein wenig überlegen erscheint, liefern diese Verfahren für X vergleichbare Ergebnisse. Man beachte hierbei, daß die Standardabweichung bei DH8 bis zu 2% und bei X noch immer bis zu 1% der Lösungskosten beträgt.

Abbildung 2.18 zeigt ein ähnliches Verhalten von SIMULATED_ANNEALING, RECORD-TO-RECORD-TRAVEL, und GREAT_DELUGE für den Fall, daß die Optimierung ohne Berücksichtigung der Verteilung des Crew-Personals über die Basen vorgenommen wird. Auch scheinen die Laufzeiten, um Lösungen ähnlicher Qualität zu erreichen, vergleichbar zu sein. Man beachte allerdings, daß die Qualität der Lösungen in beiden Fällen auf die beste gefundene Lösung bezogen ist und das Optimum nicht bekannt ist.

Der Erfolg der Wasserspiegel-Algorithmen, deren algorithmische Idee vergleichbar simpel ist, zeigt, daß die Intuition, daß beim Absenken des Wasserspiegels die Lösung schnell in einem schlechten lokalen Optimum eingefangen wird, falsch ist. Angesichts der hohen Dimension des Problems führt die Veranschaulichung der Kostenfunktion über der Lösungsmenge als dreidimensionale Berglandschaft in die Irre.

Die Vorteile der Wasserspiegel-Algorithmen gegenüber dem Simulated Annealing sind, daß nur ein Parameter einzustellen ist und daß die Auswertung der Exponentialfunktion bei der Entscheidung über die Akzeptanz eines vorgeschlagenen Übergangs entfällt (wobei die dafür notwendige Zeit bei Problemen des Operations Research in aller Regel von der Zeit, die zur Auswertung der Kostenfunktion notwendig ist, bei weitem übertroffen werden dürfte). Beim Simulated Annealing wiederum ist bei Verwendung des geometrischen Kühlschemas die Laufzeit besser zu kontrollieren.

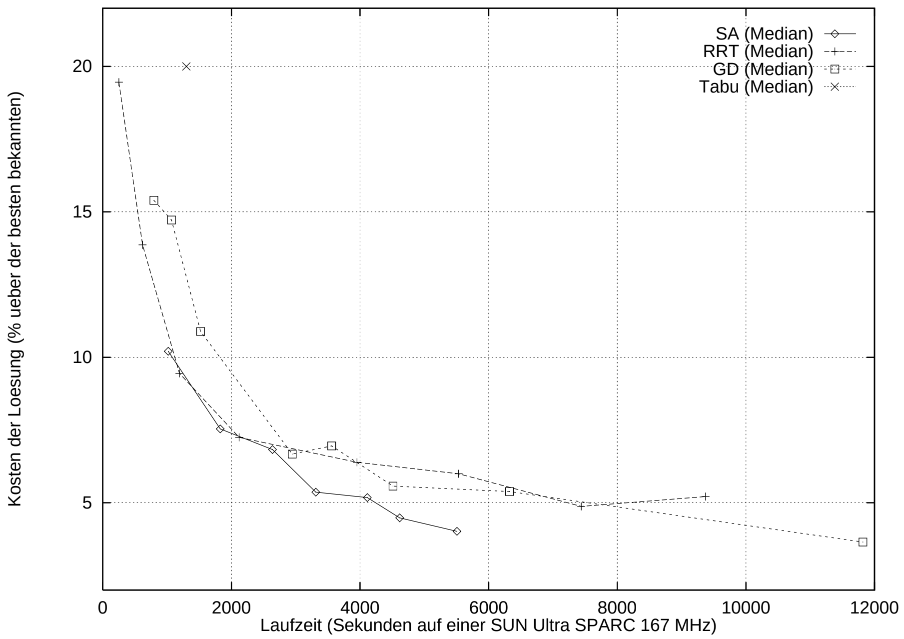

Abbildung 2.16: Vergleich der Metaheuristiken (DH8)

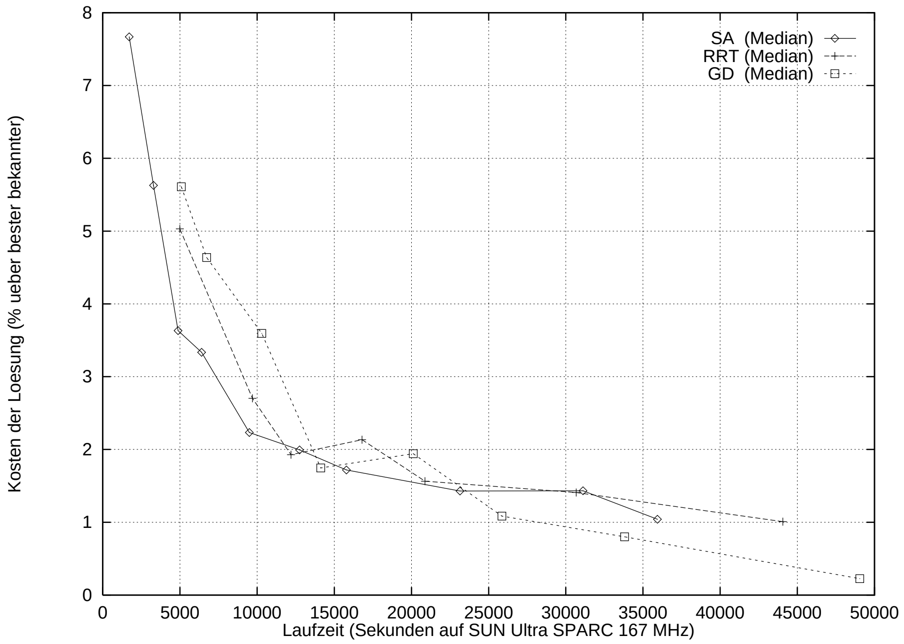

Abbildung 2.17: Vergleich der Metaheuristiken (X)

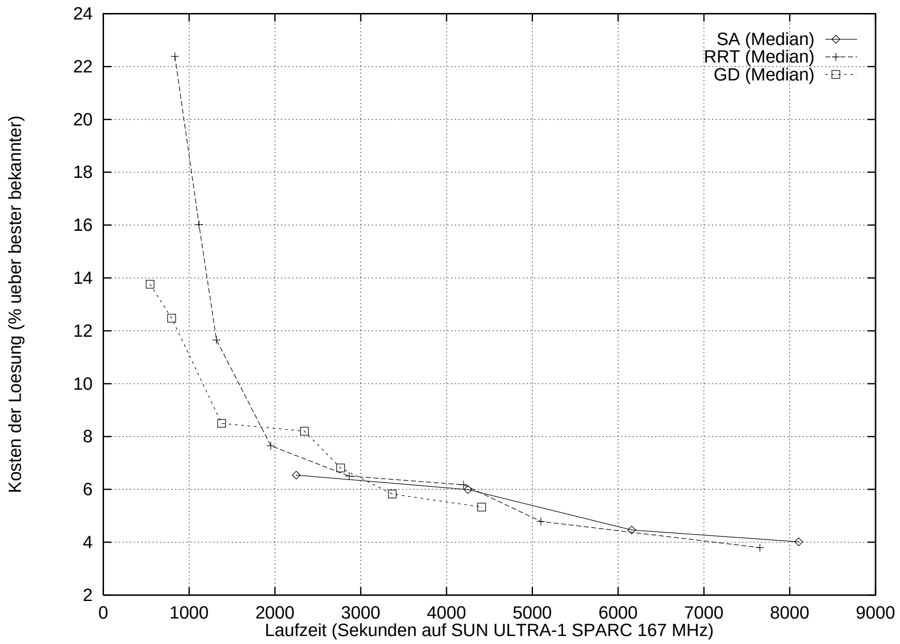

Abbildung 2.18: Vergleich der Metaheuristiken (DH8) - ohne Beachtung der Crew-Kapazitäten)

Die wesentliche Beobachtung ist jedoch, daß die Qualität der Lösungen, wie sie Simulated Annealing oder Wasserspiegel-Algorithmen gleichermaßen liefern, die Qualität von Verbesserungsheuristiken wie Greedy (die jede gefundene Verbesserung akzeptiert) oder Steilster_Abstieg (die in jedem Schritt einen besten Übergang berechnet und ausführt) bei weitem übertreffen. Unter Verwendung einer problemspezifischen Startlösung, wie sie in unserem Fall die Linear_Chainer Heuristik liefert, sind diese Heuristiken zwar schnell (etwa 17 Sekunden für DH8 und 260 Sekunden für X), doch die Lösungskosten liegen etwa 60% über den Kosten der besten von uns gefundenen Lösung, was für die Crew-Scheduling-Anwendung inakzeptabel ist.

GRASP erwies sich für unser Problem als vergleichbar mit Steilster_Abstieg, während Tabu_Suche zwar deutlich besser abschnitt als Steilster_Abstieg, jedoch nicht mit Simulated Annealing und den Wasserspiegel-Algorithmen konkurrieren kann. (Für den X-Datensatz liegt Tabu-Suche bei einer Laufzeit von 7704 Sekunden (Länge der Tabuliste = 450) noch immer 25 % über der Bestlösung, so daß wir den Wert nicht in die Grafik aufgenommen haben.) Wir vermuten jedoch, daß Tabu-Suche bei einer Nachbarschaft mit einer glatteren Kostenfunktion, wie beispielsweise die Nachbarschaft für das Pairing-Problem, die auch zusammengesetzte Übergänge wie eine Verkettung, gefolgt von der bestmöglichen Aufspaltung, oder einer Aufspaltung, gefolgt von den bestmöglichen Verkettungen, enthielte, deutlich bessere Ergebnisse erzielen würde – zum Preis einer deutlich höheren Laufzeit (und eines im Fall des Pairing-Problems deutlich höheren Programmieraufwandes). Da Tabu-Suche sich stets (deterministisch) für einen Übergang, der nicht tabu ist und die größte Verbesserung der Kostenfunktion aufweist, entscheidet, ist sie möglicherweise für eine Kostenfunktion, die – wie in unserem Fall aufgrund der Proceeding-Kosten und der Relaxierung der Zulässigkeit – auf benachbarten Lösungen stark differiert, weniger geeignet als randomisierte Verfahren.

2.8 Kombinierte Verfahren

Es liegt nahe, die Ideen der oben beschriebenen Metaheuristiken zu kombinieren zu versuchen. Für das Pairing-Problem liefern kombinierte Ansätze allerdings, wie sich zeigt, schwächere Ergebnisse als das reine Simulated-Annealing-Verfahren.

Zunächst untersuchten wir eine Kombination aus Simulated Annealing und Record-to-Record-Travel, d.h. während des Annealings werden Lösungen, deren Kosten eine globale Schranke, deren Wert von den Kosten der besten bis dahin gefundenen Lösung abhängt, überschreiten, ignoriert. Abbildung 2.19 zeigt, daß dies die Standardversion des Annealings nicht verbessert, vgl. Abbildung 2.2, Temperaturreduktion 0.95.

Außerdem kombinierten wir Simulated Annealing mit Tabu-Suche, indem wir inverse Übergänge von ausgeführten Übergängen für die nächsten $k \in \mathbb{N}$ Übergänge verbaten. Abbildung 2.20 zeigt, daß dies ebenfalls das gewöhnliche Annealing nicht verbessert, vgl. Abbildung 2.2, Temperaturreduktion $0.95^{1/4}$.

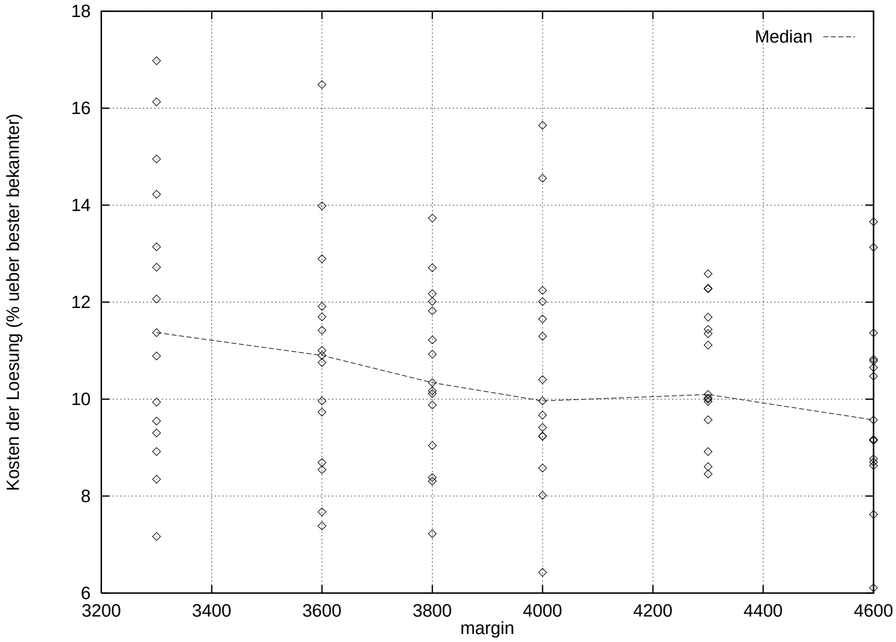

Abbildung 2.19: Record-to-Record Simulated Annealing (DH8)

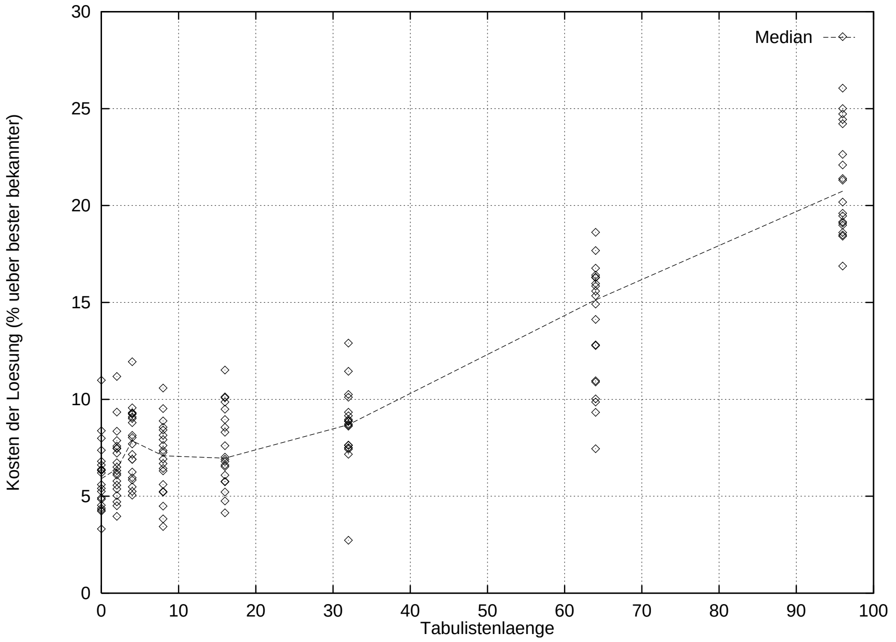

Abbildung 2.20: Tabu Simulated Annealing (DH8)

Kapitel 3

Ein Vergleich mit der SET PARTITIONING Modellierung

3.1 Die SET PARTITIONING Modellierung

Im Operations Research und der mathematischen Optimierung werden kombinatorische Optimierungsprobleme häufig als ganzzahlige lineare Programme modelliert. Durch die Relaxierung von Ganzzahligkeitsbedingungen erhält man untere Schranken, die die Berechnung exakter oder approximativer Lösungen steuern, siehe z.B. SCHRIJVER [Sch86], NEMHAUSER und WOLSEY [NW88], JOHNSON, NEMHAUSER und SAVELSBERGH [JNS97] für eine Einführung.

Das Pairing-Problem kann als sogenanntes SET PARTITIONING Problem formuliert werden. Für eine bestimmte Probleminstanz sei die Menge aller zulässigen Pairings gegeben. Die Anzahl dieser für die Dienstplanung zur Auswahl stehenden Pairings sei m. Die Kosten des j-ten Pairings, $1 \leq j \leq m$, seien durch $c_j \in \mathbb{Q}$ gegeben. Sei $A = (a_{ij})_{ij}$ eine $n \times m$ Matrix, wobei n die Anzahl der zu verplanenden Flüge bezeichnet, mit $a_{ij} = 1$ genau dann, wenn Pairing j, $1 \leq j \leq m$, den Flug i, $1 \leq i \leq n$, enthält und $a_{ij} = 0$ sonst. Ziel ist es, eine günstigste Überdeckung der Flüge mit Pairings zu konstruieren, die jeden Flug genau einmal enthält:

$$\begin{array}{rrcll} & \min \sum_{j=1}^m c_j x_j & & & \\ s.t. & \sum_{j=1}^m a_{ij} x_j & = & 1 & \text{für alle Flüge } i \in \{1,..,n\} \\ & x_j \in \{0,1\} & & & \text{für alle Pairings } j \in \{1,..,m\}. \end{array} \tag{SP}$$

Das Problem SET PARTITIONING ist $\mathcal{NP}$-vollständig (KARP [Kar72]), so daß es keinen polynomiellen exakten Algorithmus besitzt, außer es gilt $\mathcal{P} = \mathcal{NP}$. Zudem besitzen selbst kleine Probleme schon eine exorbitante Anzahl zulässiger Pairings, so daß ein gängiger Ansatz darin besteht, eine kleine Menge „mutmaßlich guter" Pairings apriori zu bestimmen und nur darauf zu optimieren. Die Erfahrung zeigt allerdings, daß zu einer guten Lösung in der Regel auch einige wenige schlechte Pairings gehören, siehe z.B. Rushmeier, Hoffman und Padberg [RHP95].

Rubin [Rub73] entwirft heuristische Verfahren, um Set Partitioning Probleme zu lösen. Darauf aufbauend entwickeln Gershkoff [Ger89] und Anbil et al. [AGPT91] das System TRIP, das bei American Airlines und Continental Airlines eingesetzt wird. Hierbei wird ein iteratives Verbesserungsverfahren eingesetzt, das aus der aktuellen Lösung eine Teilmenge der Menge von Flügen bestimmt, für diese Teilmenge zulässige Pairings generiert und das zugehörige Set Partitioning Problem mit Hilfe der Optimization Software Library (OSL) von IBM löst. Diese Teillösung wird, wenn es die aktuelle Lösung verbessert, in diese integriert. Es stellt sich heraus, daß die Qualität des Verfahrens stark von der Größe der gelösten Teilprobleme abhängt, siehe Anbil, Tanga und Johnson [ATJ92] und Chu, Gelman und Johnson [CGJ97]. Die Optima der LP-Relaxationen sind jedoch mit zunehmender Problemgröße immer weniger ganzzahlig, so daß die Größe des Branch-and-Bound-Baumes drastisch zunimmt.

United Airlines zykelt zwischen einem Pairing-Generator und einem Schnittebenen-Optimierer für das ganzzahlig-lineare Set Partitioning Programm, siehe Graves et al. [GMG+93].

Um die Auslastung des Crew-Personals an den Basen zu balancieren, benutzt Gershkoff [Ger89] eine Gewichtung der Pairings. Diese Technik scheint jedoch viel Fingerspitzengefühl seitens des System-Anwenders und viel Rechenzeit zu erfordern. Hoffman und Padberg [HP93, HP95] andererseits präsentieren einen Branch&Cut-Ansatz, in den zusätzliche Packungsungleichungen im linearen Programm (SP) einfach zu integrieren sind. Sie haben die Form

$$a^0 \;\leq\; \sum_{j\in B} a_j x_j \;\leq\; a^1,$$

wobei $B \subseteq \{1,..,m\}$ eine Menge von Pairings bezeichnet, die Crews an einer Basis zugeordnet ist, $a_j > 0$ für $j \in B$ und $0 < a^0 < a^1$ (koordinatenweise). Sie berechnen optimale Lösungen für tägliche Probleme mit Packungsungleichungen mit bis zu 179 Flügen und 85000 Spalten. Branch&Cut ist eine Kombination von Branch&Bound, das sukzessiv binäre Variablen fixiert, und dem Schnittebenen-Verfahren, das die LP-Relaxation verbessert, wobei nun an jedem Knoten des Branch&Bound-Baumes (Facetten definierende) Schnittebenen generiert und zum LP hinzugefügt werden.

Spaltengenerierung ist eine Technik, um eine optimale ganzzahlige Lösung der LP-Relaxierung der Set Partitioning Modellierung zu berechnen, ohne alle möglichen, für die Probleminstanz zulässigen Pairings generieren zu müssen. Zu diesem Zweck werden, unter Verwendung der dualen Kosten der Flüge aus der aktuellen LP-Lösung, nach und nach neue Spalten, d.h. Pairings, generiert. Die hierfür entwickelte Subroutine löst ein Kürzeste-Wege-Problem auf einem speziellen Netzwerk, in dem alle Pfade, die an einer Basis beginnen und enden und deren Länge die maximale Dauer einer Flugdienstperiode und eines Pairings berücksichtigen, Pairings entsprechen. Einen heuristischen Ansatz, der beim Generierungsprozeß eine partielle Enumeration einsetzt, verfolgen Crainic und Rousseau [CR87]. Lavoie, Minoux und Odier [LMO88] benutzen ein Verfahren, das die reduzierten Kosten der generierten Pairings approximiert.

Spaltengenerierung verbietet sich, wenn nur schlechte ganzzahlige Lösungen bekannt sind, weil jede Spalte mit reduzierten Kosten höchstens gleich der Ganzzahligkeitslücke einbezogen werden muß. Für Langstrecken-Probleme, bei denen die Dauer eines Pairings eine gute Approximation seiner Kosten darstellt, kann sie allerdings recht effektiv sein, wie Barnhart et al. [BJAH94] berichten. Die Autoren verwenden ein allgemeines Optimierungspaket für gemischt-ganzzahlige lineare Programme und erreichen Laufzeiten von 2.5 bis 8 Stunden auf einer IBM RS/6000-540 für Probleme mit 656 und 833 Flügen.

Desaulniers et al. [DDGS97] präsentieren eine einheitliche Set Partitioning Formulierung für Crew-Scheduling, Rostering und andere Probleme und stellen einen Spaltengenerierungsansatz vor.

Eine Einführung in die Branch&Price-Technik, die es erlaubt, Spalten an jedem Knoten des Branch&Bound-Baumes zu generieren, geben Barnhart et al. [BJN+96].

Eine ganzzahlige, nicht-lineare Mehrgüternetzwerkfluß-Modellierung mit Ressourcen-Variablen wird bei Air France gelöst, siehe Desaulniers et al. [DDD+97].

Die Set Covering Modellierung. Mit der Set Partitioning Modellierung eng verwandt ist die Set Covering Modellierung. Auch hier wird apriori eine Menge von Pairings bestimmt und der Optimierung zur Verfügung gestellt. Es wird aber von der Forderung, daß die in einer Lösung enthaltenen Pairings disjunkt sein sollen, also kein Flug von mehr als einem Pairing überdeckt werden soll, abgesehen. Gefordert ist hier lediglich, daß alle Flüge von der ausgewählten Menge Pairings überdeckt werden:

$$\begin{array}{rll} & \min \sum_{j=1}^{m} c_j x_j & \\ s.t. & \sum_{j=1}^{m} a_{ij} x_j \ \geq \ 1 & \text{für alle Flüge } i \in \{1,..,n\} \\ & x_j \in \{0,1\} & \text{für alle Pairings } j \in \{1,..,m\}. \end{array} \tag{SC}$$

Jede Partition ist offenbar insbesondere auch eine Überdeckung, so daß das Optimum des ganzzahligen linearen Programms (SC) höchstens so groß wie das Optimum von (SP) ist.

Die Set Covering Formulierung ist insofern problematisch, daß nicht berücksichtigt wird, daß die Kosten eines Pairings differieren können jenachdem, ob die Crew einen in ihm enthaltenen Flug bedient oder lediglich als Proceeding, d.h. zur Repositionierung, nutzt. Im zweiten Fall fällt zusätzliche Arbeitszeit, die keine Flugdienstzeit ist, aber als Flugdienstzeit in den Kosten des Pairings enthalten ist, an. Zumindest bei stark ausgelasteten Flügen wäre zudem auch der Wegfall des Ertrages aus den benötigten Sitzplätzen zu berücksichtigen. Andererseits ist es bei anderen Modellierungen ein nicht-triviales Problem, auch nur die zu verplanenden Flüge für die Repositionierung von Crews innerhalb von Pairings zu nutzen.

Baker und Fisher [BF81, BBF85] benutzen Chvátals Set Covering Heuristik [Chv79] mit Güteratio $1 + \ln \Delta$, wobei Δ die Anzahl der Flüge eines längsten Pairings bezeichnet, um eine Überdeckung der Flüge mit Pairings zu berechnen.

Wedelin [Wed95] entwickelt einen Approximationsalgorithmus für Set Covering Probleme mit Nebenbedingungen und setzt einen heuristischen Pairing-Generator ein. Dieses Verfahren kommt im System CARMEN zum Einsatz, das von den Fluggesellschaften Alitalia, KLM, Lufthansa und SAS verwendet wird, siehe [AHKW97]. Von der Berechnung von Optimallösungen der LP-Relaxierung mit Hilfe von Spaltengenerierung berichten Lavoie, Minoux und Odier [LMO88]. Das System HASTUS, vertrieben durch GIRO Enterprises Inc. (Montreal), enthält ein automatisches Dienstplanungswerkzeug CREW-OPT für den öffentlichen Nahverkehr, das eine Set Covering Formulierung und Spaltengenerierung benutzt, siehe Desrochers and Soumis [DS89, DGSS90].

Exakte Algorithmen gibt es für Set Covering Probleme mit bis zu mehreren hundert Zeilen (d.h. Trips oder Nebenbedingungen) und ein paar tausend Spalten (d.h. Pairings oder Variablen), siehe z.B. [BJ92, BC96, FK90]. Für größere Instanzen muß man auf Heuristiken zurückgreifen. Die effizientesten Verfahren basieren auf Lagrange-Relaxierungen, siehe z.B. [LL94, CNS95, CFT96]. Einen Literaturüberblick zum Set Covering Problem geben Ceria, Nobili und Sassano [CNS97].

3.2 Die Optima der LP-Relaxierung

In Kapitel 2 konnten wir die Qualität der von den lokalen Suchverfahren berechneten Lösungen zu unseren Testinstanzen nur an Hand der Kosten der jeweils besten jemals gefundenen Lösungen beurteilen. Lösungen aus der Planungsabteilung der Fluggesellschaft, von der die Daten stammen, waren nicht verfügbar. Über die Set Partitioning Modellierung besteht eine weitere Vergleichsmöglichkeit. In diesem Abschnitt stellen wir für unsere Testprobleme Instanzen für die Modellierung als Set Partitioning Problem mit Packungsungleichungen auf und berechnen die Optima der LP-Relaxierungen. Die Kosten der ganzzahligen Optimallösungen dieser Probleme, die ein speziell für diese Problemklasse entwickelter Algorithmus für ganzzahlige lineare Optimierung bestimmen könnte, können dann höchstens größer sein.

Entscheidend für die Güte der Set Partitioning Modellierung ist die Anzahl und die Auswahl der Pairings, die, in den Spalten der Matrix A codiert, für die Optimierung zur Verfügung stehen. Hierzu werden Pairings als Pfade im Graphen *conn* generiert, indem, jeweils für jeden Flug als Ausgangspunkt, eine Tiefensuche von diesem Flug aus in *conn* gestartet wird, die sowohl in ihrer Tiefe (d.h. die Anzahl der im Pairing enthaltenen Flüge) als auch in ihrer Breite (d.h. die Zahl der möglichen Anschlußflüge, die von jedem Flug aus durchprobiert wurden) beschränkt ist. Da die Zahl der möglichen Pairings rasch mit der Breite und der Tiefe der beschränkten Tiefensuche wächst, werden nicht alle erzeugten Pfade als Pairings gespeichert. Es werden nur zulässige Pairings mit höchstens einem Rotationswechsel, mit Start- oder Endflughafen eine Basis und mit einer Basis, die bezüglich der Hotel- und Proceedingskosten eine der beiden günstigsten Basen ist, gespeichert.

Die Packungsungleichungen haben die folgende Gestalt. Bezeichne $\{1,..,H\}$ die Menge der Basen, $\{1,..,D\}$ die Menge der Tage des Planungszeitraums, n die Anzahl der zu bedienenden Flüge sowie m die Anzahl der generierten Pairings. $b_{h,d} \in \mathbb{N}$

bezeichne die an einer Basis $h \in \{1,..,H\}$ an einem Tag $d \in \{1,..,D\}$ zur Verfügung stehende Crew-Kapazität (die wir auf 40% (X) bzw. 60% (DH8) der nominal angegebenen Crew-Kapazität gesetzt haben). Es sei $\chi_{j,h,d} = 1$ genau dann, wenn das Pairing j von der Basis h aus zu bedienen ist und den Tag d berührt, und $\chi_{j,h,d} = 0$ sonst. Die $H \cdot D$ Packungsungleichungen lauten damit

$$\sum_{j=1}^{m} \chi_{j,h,d} x_j \; \leq \; b_{h,d} \qquad \text{für alle } h \in \{1,..,H\},\ d \in \{1,..,D\}.$$

Man beachte, daß in unserer Run-Cutting-Formulierung die Auslastung des Crew-Personals relaxiert wurde und die von den lokalen Suchverfahren berechneten Lösungen daher die Packungsungleichungen nicht notwendig exakt einhalten. Die Kosten dieser Kostenart liegen allerdings für DH8 wie für X im Median lediglich bei 5000-6000.

Die Tabellen 3.1 und 3.2 geben jeweils für die Testinstanzen DH8 und X für verschiedene Werte für die Tiefe und Breite der Tiefensuche die Zahl der generierten Pairings, die dafür benötigte Rechenzeit (in Sekunden auf einer Sun Ultra-SPARC 167MHz), die Größe der MPS-Ausgabedatei in MB, die Kostenwerte der Optima der LP-Relaxation und die hierfür von CPLEX benötigte Rechenzeit an.

Breite	Tiefe	#Pairings	Gen-Zeit	Größe	LP-Opt	LP-Zeit
2	8	440462	54	66	306963	449
2	10	1266078	187	226	306437	3612
3	6	659956	69	87	306240	786
3	8	3944084	578	659	294122	16606
3	10	$>$ 12000000				

Tabelle 3.1: Pairing-Erzeugung für Testdatensatz DH8

Breite	Tiefe	#Pairings	Gen-Zeit	Größe	LP-Opt	LP-Zeit
2	6	465312	56	59	776624	332
2	8	1474960	206	235	736621	698

Tabelle 3.2: Pairing-Erzeugung für Testdatensatz X

Man beachte, daß diese Probleminstanzen im Hinblick auf die Anzahl der Zeilen und auf einen Tag bezogen ungefähr so groß sind wie die größten Hoffmann / Padberg Probleminstanzen für das Pairing (die nur die Flüge eines Standardtages enthalten). Obige Instanzen enthalten aber mehr Spalten als die meisten jener Instanzen. Andererseits weisen unsere Instanzen nur zum Teil eine sich täglich wiederholende Struktur auf, sind also vermutlich schwerer als die bloße Aggregation eines täglichen Problems über die Dauer des Planungszeitraums. Hoffmann und

Padbergs Probleminstanzen haben zudem eine Ganzzahligkeitslücke von 0-12 %. Borndörfer [Bor97] löst erheblich größere Set Partitioning Instanzen mit bis zu 1000 Zeilen, allerdings nicht optimal; das LP-Optimum liegt hier bis zu 26 % unterhalb der Kosten der gefundenen ganzzahligen Lösungen.

Die Kosten der besten Lösungen, die wir mit unseren Verfahren gefunden haben, betragen andererseits 786594 für den X-Datensatz und 280423 für den DH8-Datensatz. Zumindest für DH8 scheinen unsere Verfahren also ausgezeichnete Ergebnisse zu produzieren. Angesichts der raschen Zunahme der Anzahl der Pairings mit der Breite und Tiefe der Tiefensuche bei unseren Probleminstanzen scheint es hier Set Partitioning Lösern nur mit Spaltengenerierung möglich zu sein, eine vergleichbare Lösungsqualität zu erreichen. Im Falle des Testdatensatzes X scheint es jedoch durchaus möglich, daß ein Set Partitioning Löser – je nach Ganzzahligkeitslücke – schon mit wenigen Pairings bessere Resultate liefert, insbesondere angesichts der Tatsache, daß das Optimum der LP-Relaxation fast ganzzahlig ist. Hier scheint LP-Plunging, also auf heuristischem Wege zu versuchen, aus der LP-Lösung eine ganzzahlige Lösung abzuleiten, vielversprechend.

Warum es zu diesem unterschiedlichen Verhalten bei den beiden Testdatensätze kommt, darüber läßt sich nur spekulieren. Wir vermuten, daß es damit zusammenhängt, daß für den X-Datensatz die Nebenbedingung bezüglich der Verteilung der Crew-Kapazitäten über die Basen viel leichter einzuhalten ist. Gute Lösungen zum X-Datensatz, die ohne Berücksichtigung dieser Nebenbedingung erzeugt wurden, weisen immerhin halb so große Proceeding-Kosten auf, während die Proceeding-Kosten beim Datensatz DH8 auf etwa ein Fünftel fallen, wenn man diese Nebenbedingung außer acht läßt. Hier ist das mit Abstand meiste Crew-Personal an einem Flughafen beheimatet, der nur der am viertstärksten frequentierte Flughafen ist.

Um zu belegen, wie stark die Qualität der Lösungen unter der Run-Cutting-Formulierung von max_ground_time, der maximalen Bodenzeit zwischen zwei Flügen, die im Graphen *conn* durch eine Kante verbunden sind, abhängt, dokumentieren die Tabellen 3.3 und 3.4 jeweils für die Probleminstanzen DH8 und X und für verschiedene Werte von max_ground_time (in Stunden) die Größe, den durchschnittlichen Knotengrad sowie den maximalen (Außen-)Grad von *conn*, den Median der Kosten von 20 Simulated-Annealing-Läufen sowie Laufzeit und Temperaturlevel des Annealings.

Man beachte, daß die Größe der Nachbarschaft für die lokale Suche in etwa proportional zur Größe von *conn* ist. Es ist daher interessant, zu beobachten, daß die Laufzeit nicht proportional zur Größe der Nachbarschaft zunimmt. Wie die Anzahl der Temperaturlevel andeutet, ist dies darauf zurückzuführen, daß der Schwellenwert für die Akzeptanzrate, ab der die Lösung als eingefroren betrachtet wird, konstant gehalten wurde, während bei langen Bodenzeiten viele schlechte Übergänge hinzukommen. Auf der anderen Seite weisen die Simulated-Annealing-Läufe mit kleiner max_ground_time sehr viele unproduktive Temperaturlevel, in denen keine Verbesserung der besten Lösung mehr gefunden wird, auf. Der Sprung in den Kostenwerten bei $max\_ground\_time = 16$ ist auf die dann möglichen auswärtigen Übernachtungen zurückzuführen.

Stunden	#Kanten	øGrad	max Grad	Kosten	Laufzeit	#Level
4	2424	1.7	7	299998	698	162
6	3636	2.6	8	307249	687	157
8	4918	3.5	12	308296	712	145
10	6081	4.4	15	311080	805	140
16	9500	6.8	23	299139	955	134
24	14122	10.1	26	300642	1074	108
32	19472	14.0	38	298488	1106	93
40	23302	16.7	49	297126	1182	84

Tabelle 3.3: Der Einfluß von *max_ground_time* (DH8)

Stunden	#Kanten	øGrad	max Grad	Kosten	Laufzeit	#Level
4	9916	2.2	9	996516	2254	123
8	20336	4.4	18	1003874	2934	118
16	43115	9.3	35	812758	5819	108
24	71702	15.5	41	811581	9487	97
32	97809	21.2	59	816205	11790	93
40	119042	25.8	73	816764	14231	86

Tabelle 3.4: Der Einfluß von *max_ground_time* (X)

Kapitel 4

Paralleles Simulated Annealing

4.1 Einleitung

Mit Hilfe des gleichzeitigen Einsatzes mehrerer Prozessoren oder Rechner möchte man die Laufzeiten von zeitintensiven Berechnungen verkürzen. In der Theorie, d.h. im Modell einer PRAM (parallel random access machine), bei der alle Prozessoren in konstanter Zeit auf einen gemeinsamen Hauptspeicher zugreifen können, läßt sich dadurch sogar eine exponentielle Beschleunigung erzielen: etliche Probleme werden mit Hilfe von polynomiell vielen Prozessoren in polylogarithmischer Zeit lösbar, siehe z.B. Karp und Ramachandran [KR90] für einen Überblick. Viele Probleme sind allerdings $\mathcal{P}$-vollständig (siehe z.B. Greenlaw, Hoover und Ruzzo [GHR95] für eine Zusammenstellung) und können daher nur dann in polylogarithmischer Zeit gelöst werden, wenn $\mathcal{P} = \mathcal{NC}$ gilt, eine Möglichkeit, die wie $\mathcal{P} = \mathcal{NP}$ für unwahrscheinlich gehalten wird. So kann beispielsweise eine inklusionsmaximale stabile Menge in einem Graphen in polylogarithmischer Zeit berechnet werden (der Algorithmus ist allerdings erheblich komplexer als der triviale sequentielle Greedy-Algorithmus, siehe Luby [Lub86]), ein lokales Optimum für Max Cut unter der 1-Opt-Nachbarschaft zu bestimmen ist allerdings $\mathcal{P}$-vollständig (vergleiche Schäffer und Yannakakis [SY91]).

In der Praxis lassen sich jedoch zur Zeit ohnehin nur Parallelrechner mit bis zu 16 Prozessoren bei gemeinsamem Hauptspeicher realisieren. Stattdessen ist der Speicher auf die einzelnen Prozessoren verteilt, und die Prozessoren sind gemäß einer günstigen Topologie (meist das zweidimensionale Gitternetz, der dreidimensionale Torus oder der Hyperwürfel) durch leistungsfähige Kommunikationsleitungen verbunden, über die die Kommunikation zwischen Prozessoren geroutet wird. Kommunikation zwischen Prozessoren ist hier allerdings noch immer um Größenordnungen langsamer als eine elementare numerische Berechnung auf einem Prozessor. Eine Einführung in parallele Rechnerarchitekturen gibt Leighton [Lei92].

Auch ist die Entwicklung von Programmen für Parallelrechner nicht-trivial, da neuartige Probleme wie die Gewährleistung und Optimierung der Kommunikation zwischen Prozessoren auftreten. Solche Anstrengungen werden daher in der Regel nur für zeitkritische Anwendungen unternommen. Die Entwicklung von leistungsfähigen Kommunikations-Bibliotheken wie PVM oder MPI und spezieller Programmiersprachen verkürzt jedoch mittlerweile die Entwicklungszeiten.

Wir werden in diesem Kapitel zwei Methoden zur Parallelisierung von Simulated Annealing und experimentelle Ergebnisse dazu für das Pairing-Problem beschreiben. Einen Überblick über Methoden zur Parallelisierung des Simulated Annealings findet man in Azencott [Aze92]. Verhoeven und Aarts [VA95] geben allgemeiner eine Klassifizierung von parallelen lokalen Suchverfahren danach, ob

- die Parallelisierung problemspezifisch oder problemunabhängig ist,
- eine oder mehrere Trajektorien verfolgen werden und
- der Algorithmus synchrone oder asynchrone Kommunikation verwendet.

4.2 Die Clustering-Methode

Der gebräuchlichste Weg, um Probleme parallel zu lösen, ist divide-et-impera: man zerlegt das Problem in kleine, voneinander weitgehend unabhängige Teilprobleme, löst die Teilprobleme parallel (d.h. zur gleichen Zeit auf verschiedenen Prozessoren) und fügt die erhaltenen Lösungen der Teilprobleme anschließend zu einer Lösung des ursprünglichen Problems zusammen. Im Falle des Pairing-Problems könnte man sich z.B. vorstellen, den Planungszeitraum in disjunkte Intervalle aufzuteilen. Bei kombinatorischen Optimierungsproblemen ist diese Methode jedoch in der Regel nicht sinnvoll, weil durch die enge Kopplung der Teilprobleme entweder ein hoher Kommunikationsaufwand erforderlich wird oder die Qualität der zusammengesetzten Lösung unbefriedigend ist. Andererseits sind die Knoten heutiger Parallelrechner oder gar eines ATM-Netzwerks von Workstations gegenüber der Kommunikationsgeschwindigkeit zu schnell, um eine befriedigende Beschleunigung auf der Ebene einer problemspezifischen Parallelisierung wie beispielsweise einer Aufteilung der Berechnung der Kostenfunktion auf mehrere Prozessoren zu erreichen. Daher geschieht die Parallelisierung des inhärent sequentiellen Simulated-Annealing-Verfahrens bei der sogenannten Clustering-Methode nach Aarts et al. [ABHL86] (siehe auch Diekmann, Lüling und Simon [DLS93]) auf algorithmischer Ebene. Hier werden zu Beginn, wenn die Akzeptanzrate hoch ist, auf den zur Verfügung stehenden Prozessoren unabhängige Markov-Ketten parallel simuliert. Wenn die Akzeptanzrate sinkt, werden von Zeit zu Zeit Gruppen (z.B. Paare) von Markov-Ketten gebündelt. Hierzu wird in jeder Gruppe genau eine der Markov-Ketten ausgewählt (z.B. gemäß der Boltzmann-Verteilung) und von den zur Gruppe gehörigen Prozessoren gemeinsam weitersimuliert, wobei die Nachbarschaft der aktuellen Lösung der Markov-Kette auf die einzelnen beteiligten Prozessoren aufgeteilt und ein akzeptierter Übergang untereinander kommuniziert wird. Durch die Clustering-Schritte, bei denen die Lösung, mit der weitergerechnet werden soll, aus mehreren ausgewählt wird, verringert sich die Wahrscheinlichkeit, in einem schlechten lokalen Optimum stecken zu bleiben. Da bei Simulated Annealing Qualität mit Laufzeit zu bezahlen ist, beschleunigt sich also durch die Clustering-Schritte das Verfahren bei hohen Temperaturen. Bei tiefen Temperaturen hingegen, wenn die Akzeptanzrate niedrig ist, durchsuchen mehrere Prozessoren gemeinsam in der Nachbarschaft einer Lösung nach einem Übergang, der akzeptiert werden kann, was einer Parallelverarbeitung einer Programmschleife nahekommt.

Das Verfahren wurde auf einem Parallelrechner GigaCluster PowerPlus (GCPP) der Firma Parsytec mit 32 Rechnerknoten, die entlang der Gitternetzlinien eines 8x4-Gitters miteinander kommunizieren, unter Verwendung der Paderborner Parallel Simulated Annealing Toolbox (siehe Kliewer und Unruh [KU98]) getestet. Jeder Knoten des GCPP setzt sich aus zwei Prozessoren vom Typ Motorola PowerPC 601 mit einer Taktfrequenz von 80 MHz und vier Kommunikationseinheiten zusammen. Als Kommunikationsbibliothek wurde die MPICH-Implementierung von MPI (siehe z.B. Gropp, Lusk und Skjellum [GLS94] und Snir et al. [SOH+95]) in der Version 1.0.7 verwendet. Tabelle 4.1 gibt die Ergebnisse auf einem bis 16 Prozessoren für den Testdatensatz DH8 wieder.

#Proz	Zeit	Beschleunigung	Effizienz	Qualität	σ
1	5612.5	1.00	1.00	7.6	0.018
2	3840.8	1.46	0.73	6.2	0.009
4	2325.2	2.41	0.60	7.1	0.017
8	1182.4	4.75	0.59	7.8	0.019
16	595.6	9.42	0.59	7.9	0.011

Tabelle 4.1: Paralleles Simulated Annealing durch Clustering

Jeder Tabelleneintrag ist das Mittel über die Ergebnisse von mindestens 10 Testläufen. Um vergleichbare Resultate zu erhalten, wurde bei mehreren Prozessoren die Laufzeit so angepaßt, daß die Lösungsqualität (gemessen in % über der besten bekannten Lösung) über die Zeilen in etwa die gleiche ist. Bezeichne $T(n)$ die Laufzeit eines parallelen Algorithmus auf n Prozessoren. Die Effizienz(funktion) eines parallelen Algorithmus mißt, wie gut sich die mit mehreren Prozessoren zur Verfügung stehende zusätzliche Rechenleistung ausnutzen läßt (und nicht z.B. durch die notwendige Kommunikation wieder verloren geht), und ist definiert als

$$\mathrm{Eff}(n) := \frac{T(1)}{T(n)}/n.$$

Die hier gemessene Effizienz von mindestens fast 60% bis hin zu 16 Prozessoren kann als außerordentlich gut bezeichnet werden.

4.3 Die MIR-Parallelisierung

Ein anderer, von Lee [Lee95] studierter Ansatz, Simulated Annealing zu parallelisieren (siehe auch Azencott [Aze92]), nutzt die Varianz in den Lösungskosten eines randomisierten Algorithmus aus. (Für den DH8-Datensatz hatten wir oben eine Standardabweichung von etwa 2% beobachtet.) Statt einiger weniger langer (voneinander unabhängiger) Simulated-Annealing-Läufe (in der Praxis wird man sich vermutlich nicht auf einen Lauf beschränken, wenn hochqualitative Lösungen gesucht sind), werden mehrere kürzere durchgeführt und die beste gefundene Lösung ausgegeben. Dieses Verfahren ist nicht nur deshalb besonders ansprechend, weil es nahezu ohne Kommunikation auskommt. Daß Simulated Annealing bei sehr langen Laufzeiten kaum noch eine Verbesserung der Lösungsqualität liefert, führt im Bereich sehr hoher Qualitätsanforderungen zu einer super-linearen Beschleunigung. Für Abbildung 4.1 wurden Tests mit Gruppen von jeweils $k = 1, 2, 3, 4$ unabhängigen Simulated-Annealing-Läufen sowie mit verschiedenen Werten für die Temperaturreduktion durchgeführt. Die Abzisse ist der Durchschnitt der Summe der Laufzeiten der unabhängigen Simulated-Annealing-Läufe, die Ordinate gibt den Median der Kosten der besten Lösung unter den k Läufen wieder. Wie man sieht, erreicht man beispielsweise eine Qualität von knapp unter 4% über der besten jemals gefundenen Lösung genauso gut mit einem langen wie mit zwei halb so langen Läufen. (Die Rechnungen wurden allerdings auf einem anderen Rechner und unter anderen Compiler-Optionen durchgeführt, siehe PROKSCH [Pro97]. Für Abbildung 4.1 wurden die Laufzeiten entsprechend skaliert.)

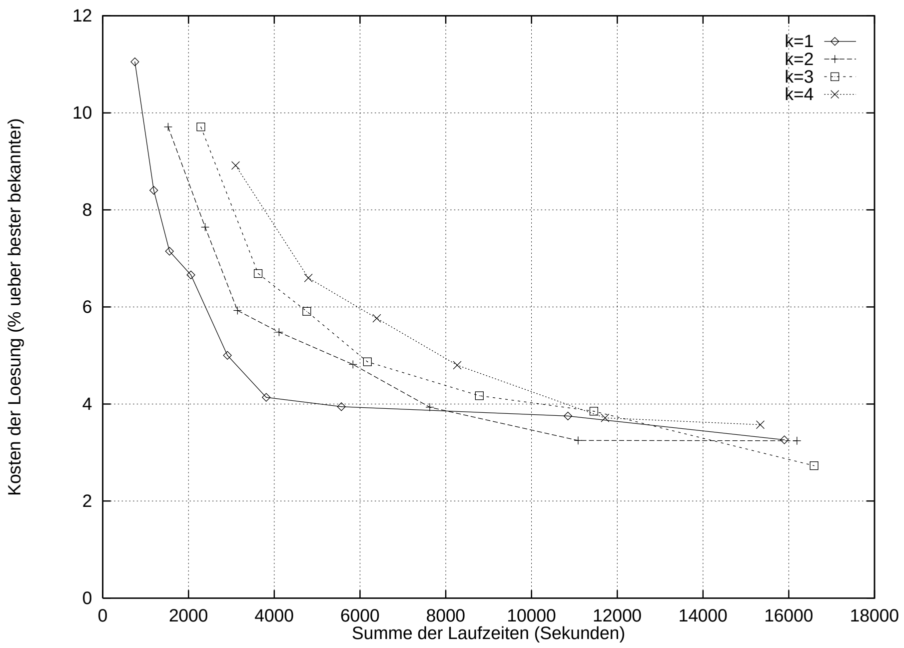

Abbildung 4.1: Unabhängige Simulated-Annealing-Läufe (DH8)

Kapitel 5

Zusammenfassung und Ausblick

Für das Pairing-Problem in der Flugdienstplanung wurde die Run-Cutting-Modellierung formuliert, die darauf abzielt, daß Pairings aus Abschnitten von Flugzeug-Umläufen aufgebaut sind. Es ist eine graphentheoretische Modellierung, die besonders für Algorithmen der lokalen Suche geeignet ist. Da das Crew-Personal gewöhnlich über mehrere Crew-Basen verteilt ist, ist jedem Pairing eine Crew-Basis zugeordnet, von der aus es am besten zu bedienen ist, um die Proceeding-Kosten global zu minimieren. Auf diese Weise zerfällt das Rostering-Problem in kleinere, unabhängige Probleme an den Basen. Zur Lösung des Pairing-Problems wurden – unseres Wissens nach erstmalig – verschiedene Metaheuristiken studiert und verglichen. Es zeigt sich, daß diese Verfahren

- äußerst flexibel handhabbar sind, was Änderungen der Zielfunktion oder der Nebenbedingungen angeht – dies ist in der Praxis, wo die Zielfunktion häufig erst in Rücksprache mit einem Anwendungsexperten sukzessive festgelegt werden muß, von großem Wert –,

- eine Optimierung ermöglichen in Situationen, wo es aufgrund einer aus widerstrebenden Kostenarten zusammengesetzten Zielfunktion und komplexen Nebenbedingungen schwierig erscheint, problemspezifische Algorithmen zu entwerfen, und

- gegenüber bloßen Verbesserungsheuristiken, wie beispielsweise dem Verfahren des steilsten Abstiegs, um 40% bessere Ergebnisse liefern.

Wie Rechnungen mit realen Testdatensätzen mit bis zu 4600 Flügen belegen, sind geeignete Modifikationen der Wasserspiegel-Verfahren, die in der Literatur noch nicht systematisch untersucht wurden, hinsichtlich der Qualität der produzieren Lösungen mit Simulated Annealing vergleichbar. Eine elementare Implementierung von Tabu-Suche und ein GRASP-Ansatz liefern unbefriedigende Resultate. Eine Kombination der Grundprinzipien einzelner Metaheuristiken erscheint wenig geeignet, die Leistungsfähigkeit der Algorithmen weiter zu steigern. Die Ergebnisse von Simulated Annealing und der Wasserspiegel-Algorithmen können sich auch mit den LP-Schranken für die SET PARTITIONING Formulierung mit Packungsungleichungen und bis zu 3.9 Millionen Spalten messen. Eine Clustering-Parallelisierung des

Simulated Annealings auf einem Parallelrechner GigaCluster PowerPlus weist eine sehr gute Beschleunigung auf. Die MIR-Parallelisierung verspricht für hohe Qualitätsanforderungen eine perfekte Beschleunigung.

Verbesserungsmöglichkeiten für den Prototypen. Für einen praktischen Einsatz des Systems sehen wir u.a. folgende Verbesserungsmöglichkeiten hinsichtlich der Problemmodellierung.

- Proceedingkosten sollten aus einem Flugplan, der mindestens alle Flüge der betrachteten Fluggesellschaft, möglichst aber auch Flüge anderer Fluggesellschaften, enthält, berechnet werden, so daß man auch Angaben über die sich aus den Proceedings ergebende zusätzliche Dienstzeit zur Verfügung hätte. In dem Fall, daß nur Flüge der eigenen Fluggesellschaft zur Repositionierung verwendet werden und somit keine nennenswerten Unterschiede in den Kosten für das Flugticket zu berücksichtigen sind, können allerdings für jeden zu verplanenden Flug die kürzesten Proceedings von und zu allen Basen im voraus berechnet werden, so daß diese Verbesserung kaum zusätzliche Laufzeit erforderte. Sonst müssen die Proceeding-Kosten zur Laufzeit berechnet werden, da ein späterer Flug zurück zur Basis, wenn die maximale Dauer einer Flugdienstperiode es zuläßt, günstiger sein kann. Eine spannende Frage wäre die Einbeziehung von Proceedings innerhalb von Pairings.

- Die Kostenfunktion sollte noch an einigen Stellen verfeinert werden. So würde man beispielsweise gerne die Dienstzeit, die keine Flugzeit ist, innerhalb von Flugdienstperioden und zwischen Flugdienstperioden unterscheiden. Dies ermöglichte es, die Länge der erforderlichen Ruhezeit nach Abschluß des Pairings, die lediglich von der letzten Flugdienstperiode abhängt, zu bewerten.

- Die Modellierung kann auf nicht-konstantes Crew-Komplement erweitert werden, indem für jeden Flug entsprechend des erforderlichen Crew-Komplements eine Anzahl Kopien angelegt wird.

- Die Modellierung kann auf den Fall erweitert werden, daß zwei Flugzeugmuster mit demselben Personal zu bedienen sind (Mehrfachlizenzen). Hierzu werden drei Crew-Kapazitätsfunktionen auf den Basen definiert: je eine für das Personal, das nur eine Lizenz führt, und eine dritte für das Personal, das beide Lizenzen führt. Entsprechend sind drei Crew-Nachfragefunktionen für eine gegebene Pairing-Zerlegung zu verwalten. Bei der Berechnung der Kosten wird zunächst die Crew-Nachfrage nach Personal mit nur einer Lizenz, die dessen Kapazität überschreitet, auf die Crew-Nachfrage nach Doppellizenzpersonal aufgeschlagen und dann die Überschreitung der Kapazität des Doppellizenzpersonals gemessen.

- Ein weiteres Problem stellen die Randbedingungen dar, die sich aus den Flügen außerhalb des Planungszeitraums ergeben, denn das Pairing-Problem ist eigentlich ein kontinuierliches und die Grenzen des Planungszeitraumes sind mehr oder weniger willkürlich. Barnhart et al. [BJAH94] beispielsweise vergeben „credits", also ein Guthaben, an Pairings, die Flüge enthalten, die sich an den Planungszeitraum anschließen. Wir schlagen vor, dies über die Proceeding-Kosten zu steuern, indem sich an den Planungszeitraum anschließende Flüge als Proceedings genutzt werden können, ohne im Sinne eines Preises für das Flugticket Proceeding-Kosten hervorzurufen (die Flugzeit muß selbstverständlich in die Betrachtung der Zulässigkeit des Pairings einfließen). Bereits im vorherigen Planungszeitraum verplante Flüge am Beginn des aktuellen Planungszeitraums werden durch eine reduzierte Crew-Personalkapazität berücksichtigt.

Metaheuristik und lokale Verbesserung in Kombination. Die spezifischen Eigenschaften eines Problems fließen über Kostenfunktion und Nebenbedingungen, aber auch über die gewählte Nachbarschaft in die lokale Suche ein. Eine Nachbarschaft, die die Charakteristika eines Problems widerspiegelt, ist daher von entscheidender Bedeutung für die Leistungsfähigkeit eines lokalen Suchverfahrens. Bei der Stundenplanung für Schulen und Universitäten, ein anderes kombinatorisches Optimierungsproblem mit gegenläufigen Optimierungszielen, berichten THOMPSON und DOWSLAND [TD96], daß der erhöhte Aufwand, eine komplexere Nachbarschaft auszuwerten, durch die bessere Qualität der Lösung mehr als ausgeglichen wird. Für unser Problem könnte man z.B. Übergänge anbieten, die aus einer Aufspaltung, gefolgt von guten oder gar den beiden bestmöglichen Verkettungen der entstandenen Teile, oder einer Verkettung, gefolgt von einer guten bzw. der bestmöglichen Aufspaltung des entstehenden Pairings, bestehen. Da es allerdings sehr viele solcher zusammengesetzter Übergänge gibt, entschlossen wir uns, ohne die Nachbarschaft zu ändern, in regelmäßigen Abständen (beim Annealing z.B. nach jedem Temperaturlevel) eine Routine einzustreuen, die derartige zusammengesetzte Übergänge sucht und im Fall einer Verbesserung ausführt. Es zeigt sich, daß sich hierdurch die Lösungsqualität des Annealings ohne nennenswerte Laufzeiteinbußen je nach dem, wieviel Laufzeit dem Annealing zur Verfügung steht, um bis zu 2% verbessern läßt, vergleiche EMDEN-WEINERT und PROKSCH [EP98]. Für große Laufzeiten scheint der Effekt jedoch nachzulassen. Eine weitere Möglichkeit bestände darin, der Menge der einfachen Übergänge lediglich eine Auswahl von vielversprechenden zusammengesetzten Übergängen hinzuzufügen und mithin mit Übergängen verschiedenen Typs zu arbeiten.

Metaheuristik und ganzzahlige lineare Optimierung. Eine weitere vielversprechende Forschungsrichtung wäre, lokale Suche und das Lösen des SET PARTITIONING LPs mit Packungsungleichungen zu verbinden. Die Rolle des LPs wäre, mittels LP-Plungings gute Startlösungen für die lokale Suche zu konstruieren, während die lokale Suche, möglicherweise an den dualen Variablen orientiert, neue Pairings für das nächste LP generiert.

Rostering. Schließlich wäre es interessant, die Leistungsfähigkeit der verwendeten Metaheuristiken für das Rostering Problem in der Formulierung eines GENERALIZED ASSIGNMENT Problems mit nicht-linearen Nebenbedinungen zu untersuchen.

Die Auswahl der Metaheuristik. Die verschiedenen betrachteten Metaheuristiken eignen sich, wie hier anhand des Pairing-Problems gesehen, unterschiedlich gut für ein bestimmtes Problem. Eine bestimmte Metaheuristik wiederum eignet sich für manche Probleme besser, für andere schlechter. Für das VEHICLE ROUTING Problem beispielsweise hat sich mittlerweile Tabu-Suche als der Favorit herauskristallisiert, siehe z.B. GENDREAU, LAPORTE und POTVIN [GLP97]. Die Frage, die sich stellt, ist, ob man generelle Aussagen darüber treffen kann, für welche Art von Problemen eine bestimmte Metaheuristik anderen Metaheuristiken vorzuziehen ist. Eine These, die sich aus dem Vergleich der Metaheuristiken für die Flugdienstplanung ableiten läßt und die es an anderen Problemen zu verifizieren bzw. zu falsifizieren gälte, lautet:

> Für ein Problem zusammen mit einer bestimmten Nachbarschaft auf den Lösungen des Problems, dessen Kostenfunktion auf benachbarten Lösungen stark differiert (gemessen an der Veränderung der Kostenfunktion der aktuellen Lösung über längere Zeiträume), eignen sich Simulated Annealing und Wasserspiegel-Algorithmen eher als Tabu-Suche.

Anhang A

Die graphische Benutzungsoberfläche

Die auf Basis von Tcl-Tk (im Rahmen des Projekts PARALOR) entwickelte graphisch-interaktive Benutzungsoberfläche des prototypischen Planungs- und Decision-Support-Systems erlaubt neben dem Einlesen von Probleminstanzen und dem Sichern von Lösungen, menügesteuert Parameter z.B. der Kostenfunktion und der Nebenbedingungen einzustellen oder das Optimierungsverfahren auszuwählen. Die Lösung selbst wird in Form eines GANTT- oder Balkendiagramms dargestellt. Waagerecht ist die Zeit abgetragen (eine Spalte entspricht einem Tag), senkrecht die Basen. Die Flüge sind durch kleine Rechtecke dargestellt, wobei die zu einem Pairing gehörigen Flüge durch einen schmalen waagerechten Balken verbunden sind. Das dreibuchstabige Akronym am Anfang einer Zeile gibt an, von welcher Basis aus die Pairings dieser Zeile vorzugsweise bedient werden sollten. Durch Klicken auf ein Pairing erhält man in einem gesonderten Fenster nähere Informationen über dieses Pairing. Während sich das GANTT-Diagramm gut dazu eignet, zu beurteilen, wie dicht die Flüge in den Pairings gepackt sind, visualisiert es weniger die Dimension Ort (d.h. die Flughäfen). Wenn ein Pairing ein Proceeding benötigt, ist deshalb der erste bzw. letzte Flug entsprechend farblich abgehoben. Unterhalb des GANTT-Diagramms gibt ein doppeltes Histogramm Auskunft über die Auslastung einer zuvor markierten Basis: die untere Kurve spiegelt den von der angezeigten Lösung dort induzierten Bedarf an Personentagen, die obere die dort zur Verfügung stehende Kapazität an Personentagen pro Kalendertag wider. Für den Planer ist hierdurch eine Überlastung des Crew-Personals an einer Basis und einem Kalendertag sofort ersichtlich. Ein Informationsfeld schließlich zeigt die Kosten der Lösung (aufgeschlüsselt nach Kostenarten) sowie statistische Angaben zur Lösung wie das Verhältnis der Dienstzeit, die keine Flugzeit ist, zur Flugzeit oder das Verhältnis der Anzahl der Proceedings zur Anzahl der Hotelübernachtungen an.

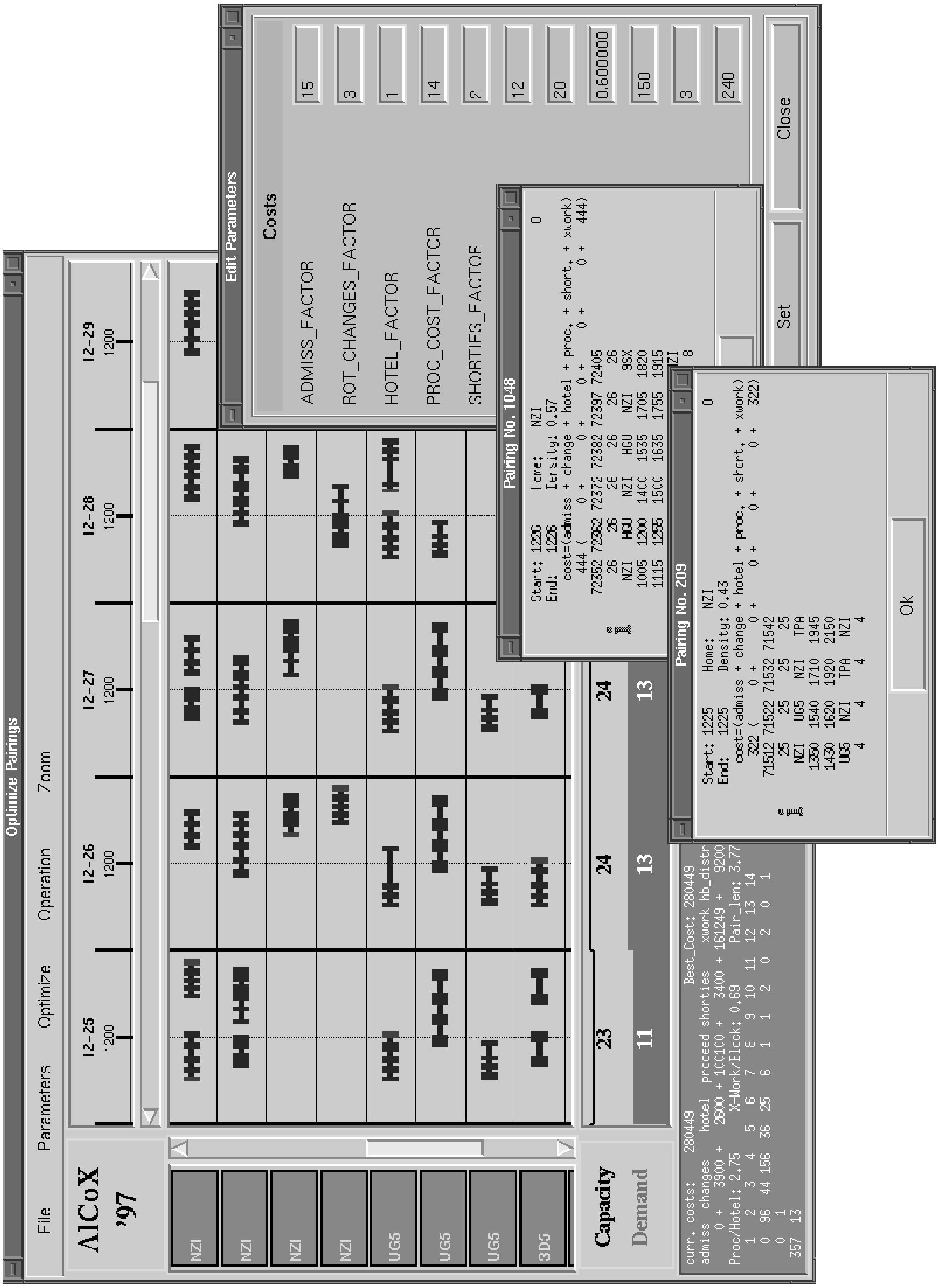

Abbildung A.1: Die graphische Benutzungsoberfläche

Literaturverzeichnis

[ABHL86] AARTS, E.H.L., DE BONT, F.M.J., HABERS, E.H.A., VAN LAARHOVEN, P.J.M., *Parallel implementations of the statistical cooling algorithm,* Integration 4, 1986, 209-238.

[AFST69] ARABEYRE, J.P., FEARNLEY, J., STEIGER, F.C., TEATHER, W., *The airline crew scheduling problem: a survey,* Transport. Sci. 3, 1969, 140-163.

[AGPT91] ANBIL, R., GELMAN, E., PATTY, B., TANGA, R., *Recent advances in crew-pairing optimization at American Airlines,* Interfaces 21, 1991, 62-74.

[AHKW97] ANDERSSON, E., HOUSOS, E., KOHL, N., WEDELIN, D., *Crew pairing optimization,* in: OR in airline industry, Hrsg. Yu, G., Kluwer Acad. Publ., Boston, 1997, in Vorbereitung.

[AK89] AARTS, E., KORST, J., *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing,* Wiley & Sons, Chichester, 1989.

[AK93] ALTHÖFER, I., KOSCHNICK, K.-U., *On the deterministic complexity of searching local maxima,* Disc. Appl. Math. 43, 1993, 111-113.

[AKL97] AARTS, E.H.L., KORST, J.H.M., VAN LAARHOVEN, P.J.M., *Simulated annealing,* in: [AL97], 91-120.

[AL97] AARTS, E.H.L., LENSTRA, J.K., (Hrsg.), *Local search in combinatorial optimization,* Wiley, Chichester, 1997.

[ALM+98] ARORA, S., LUND, C., MOTWANI, R., SUDAN, M., SZEGEDY, M., *Proof verification and the hardness of approximation problems*, Electronic Colloquium on Computational Complexity, Report TR98-008, 1998, vorläufige Version: Proc. 33rd IEEE FoCS, 1992, 14-23.

[Aro97] ARORA, S., *Polynomial time approximation schemes for Euclidean TSP and other geometric problems,* Technischer Bericht, Princeton University, Juni 1997, zur Veröffentlichung eingereicht (vorläufige Version: Proc. 37th IEEE FoCS, 1997).

[ATJ92] ANBIL, R., TANGA, R., JOHNSON, E.L., *A global approach to crew-pairing optimization,* IBM Systems J. 31, 1992, 71-78.

[Aze92] AZENCOTT, R., (Hrsg.), *Simulated annealing – parallelization techniques,* Wiley & Sons, New York, 1992.

[Bac97] BACHEM, A., et al., *Verbundprojekt PARALOR: Parallele Verfahren zur Wegoptimierung, in Flugplanung und Logistik,* Zentrum für Paralleles Rechnen, Köln, Technischer Bericht 97-268, 1997.

[BBF85] BAKER, E.K., BODIN, L.D., FISHER, M., *The development of a heuristic set covering based system for air crew scheduling,* Transp. Policy Decision Making 3, 1985, 95-110.

[BBFP79] BAKER, E.K., BODIN, L.D., FINNEGAN, W.F., PONDER, R.J. *Efficient heuristic solutions to an airline crew scheduling problem,* AIIE Trans. 11, 1979, 79-85.

[BBM+92] BIANCO, L., BIELLI, M., MINGOZZI, A., RICCIARDELLI, S., SPADONI, M., *A heuristic procedure for the crew rostering problem,* Europ. J. Oper. Res. 58, 1992, 272-283.

[BC96] BALAS, E., CARRERA, M.C., *A dynamic subgradient-based branch-and-bound procedure for set covering,* Oper. Res. 44, 1996, 875-890.

[BCG87] BERTOSSI, A. A., CARRARESI, P., GALLO, G., *On some matching problems arising in vehicle scheduling models*, Networks 17, 1987, 271-281.

[BF81] BAKER, E., FISHER, M., *Computational results for very large air crew scheduling problems,* OMEGA 9, 1981, 613-618.

[BGAB83] BODIN, L., GOLDEN, B., ASSAD, A., BALL, M., *Routing and scheduling of vehicles and crews - the state of the art*, Comput. & Oper. Res. 10, 1983, 63-211.

[BGL+92] 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, 1992, 885-897.

[BH90] BARUTT, J., HULL, T., *Airline crew scheduling - supercomputers and algorithms*, 30th Ann. Symp. AGIFORS 1990, 351-358; und: SIAM News 23 (6), November 1990.

[BHJ95] BARNHART, C., HATAY, L., JOHNSON, E.L., *Deadhead selection for the long-haul crew pairing problem,* Oper. Res. 43, 1995, 491-499.

[BJ92] BEASLEY, J.E., JORNSTEN, K., *Enhancing an algorithm for the set covering problems,* Europ. J. Oper. Res. 58, 1992, 293-300.

[BJAH94] BARNHART, C., JOHNSON, E.L., ANBIL, R., HATAY, L., *A column generation technique for the long-haul crew assignment problem,* in: Optimization in industry 2, Hrsg. Ciriani, T.A., Leachman, R.C., Wiley & Sons, 1994, 7-24.

[BJN+96] BARNHART, C., JOHNSON, E.L., NEMHAUSER, G.L., SAVELSBERGH, M.W.P., VANCE, P.H.: *Branch-and-Price: column generation for solving huge integer programs,* Georgia Institute of Technology, Atlanta, Technischer Bericht, 1996, erhältlich über das WWW an der URL „http://www.eng.auburn.edu/~pvance/“, erscheint in: Operations Research;

siehe auch die gekürzte Version in: Mathematical programming – state of the art 1994, Hrsg. Birge, J.R., Murty, K.G., Univ. of Michigan, 1994, 186-207.

[BM77] BAKER, K.R., MAGAZINE, M.J., *Workforce scheduling with cyclic demands and day-off constraints,* Manag. Sci. 24, 1977, 161-167.

[BMRS90] BIANCO, L., MINGOZZI, A., RICCIARDELLI, S., SPANDONI, M., *New exact and heuristic algorithms for the crew scheduling problem,* in: System modelling and optimization, Proc. 14th IFIP Conf., Lecture Notes Control Inf. Sci. 143, Springer, Berlin, 1990, 446-455.

[Bor87] BORGWARDT, K.H., *The simplex method. A probabilistic analysis*, Springer, 1987.

[Bor97] BORNDÖRFER, R., *Aspects of set packing, partitioning, and covering,* Dissertation, Technische Universität Berlin, 1998; erhältlich an der URL „http://www.zib.de/borndoerfer/“.

[BR85] BALL, M., ROBERTS, A., *A graph partitioning approach to airline crew scheduling,* Transport. Sci. 19, 1985, 107-126.

[CCJ+97] CLEMENTS, D.P., CRAWFORD, J.M., JOSLIN, D.E., NEMHAUSER, G.L., PUTTLITZ, M.E., SAVELSBERGH, M.W.P., *Heuristic optimization: a hybrid AI/OR approach,* in: Proceedings Constraint-Directed Scheduling - CP'97, 1997.

[CDH97] CAMPBELL, K.W., DURFEE, R.B., HINES, G.S., *FedEx generates bid lines using simulated annealing,* Interfaces 27, 1997, 1-16.

[Čer85] ČERNY, V., *Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm,* J. Optim. Theo. Appl. 45, 1985, 41-51.

[CFG+97] CAPRARA, A., FISCHETTI, M., GUIDA, P.L., TOTH, P., VIGO, D., *Solution of large-scale railway crew planning problems: the Italian experience,* Università degli Studi di Bologna, Dipartimento di Elettronica, Informatica e Sistemistica, Operations Research Group, Technischer Bericht OR-97-9, 1997, zur Veröffentlichung eingereicht.

[CFT96] CAPRARA, A., FISCHETTI, M., TOTH, P., *A heuristic method for the set covering problem,* Università degli Studi di Bologna, Dipartimento di Elettronica, Informatica e Sistemistica, Operations Research Group, Technischer Bericht OR-95-8, 1995, zur Veröffentlichung eingereicht; vorläufige Version in: Proc. 5th IPCO, Hrsg. Cunningham, W.H., McCormick, S.T, Queyranne, M., Springer LNCS 1084, 1996, 72-81.

[CFTV95] CAPRARA, A., FISCHETTI, M., TOTH, P., VIGO, D., *Modeling and solving the crew rostering problem,* Università degli Studi di Bologna, Dipartimento di Elettronica, Informatica e Sistemistica, Operations Research Group, Technischer Bericht OR-95-6(R), 1995, erscheint in Oper. Res.

[CGJ97] CHU, H.D., GELMAN, E., JOHNSON, E.L., *Solving large scale crew scheduling problems,* Europ. J. Oper. Res. 97, 1997, 260-268.

[Chv79] CHVÁTAL, V., *A greedy heuristic for the set covering problem,* Math. Oper. Res. 4, 1979, 233-235.

[CKT94] CHANDRA, B., KARLOFF, H., TOVEY, C.A., *New results on the old k-opt algorithm for the TSP,* in: Proceedings of the 5th annual ACM-SIAM symposium on discrete algorithms, ACM-SIAM, 1994, 150-159; erscheint in SIAM J. Comput.

[CL91] COFFMAN, E.G., LUEKER, G.S., *Probabilistic analysis of packing and partitioning algorithms,* Wiley & Sons, New York, 1991.

[CNS95] CERIA, S., NOBILI, P., SASSANO, A., *A Lagrangian-based heuristic for large-scale set covering problems,* IASI-CNR, Roma, Technischer Bericht # 406, 1995; erscheint in Math. Progr. B.

[CNS97] CERIA, S., NOBILI, P., SASSANO, A., *Set covering problem,* in: [DMM97], 415-428.

[CR87] CRAINIC, T.G., ROUSSEAU, J.-M., *The column generation principle and the airline crew scheduling problem,* INFOR 25, 1987, 136-151.

[CW92] CATTRYSSE, D.G., VAN WASSENHOVE, L.N., *A survey of algorithms for the generalized assignment problem,* Europ. J. Oper. Res. 60, 1992, 260-272.

[DAG92] *Manteltarifvertrag Nr. 4 für das Bordpersonal,* Deutsche Angestellten-Gewerkschaft, Hamburg, 1992.

[Dan49] DANZIG, G.B., *Linear programming and extensions*, Econometrics 17, 1949, 200-211; Princeton Univ. Press, Princeton, NJ, 1963.

[DBP95] DADUNA, J.R., BRANCO, I., PAIXAO, J.M.P., (Hrsg.), *Computer aided transit scheduling '93,* Springer LNEMS 430, 1995.

[DD92] DOWSLAND, K.A., DOWSLAND, W.B., *Packing problems,* Europ. J. Oper. Res. 56, 1992, 2-14.

[DDD+97] DESAULNIERS, G., DESROSIERS, J., DUMAS, Y., MARC, S., RIOUX, B., SOLOMON, M.M., SOUMIS, F., *Crew pairing at Air France,* Europ. J. Oper. Res. 97, 1997, 245-259.

[DDGS97] DESAULNIERS, G., DESROSIERS, J., GAMACHE, M., SOUMIS, F., *Crew scheduling in air transportation,* Technical report G-97-26, GERAD Montréal, May 1997.

[DDSS95] DESROSIERS, J., DUMAS, Y., SOLOMON, M.M., SOUMIS, F. *Time constrained routing and scheduling,* in: Network routing, Hrsg. Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L., Handbooks in OR & MS Vol. 8, Elsevier, Amsterdam, 1995, 35-139.

[DGSS90] DESROCHERS, M., GILBERT, J., SAUVÉ, M., SOUMIS, F., *CREW-OPT: subproblem modeling in a column generation approach to urban crew scheduling,* in: [DR92], 395-406.

[Die97] DIESTEL, R., *Graph theory,* Springer, Heidelberg, 1997 (Übersetzung der deutschen Ausgabe von 1996).

[DLM+95] DESROSIERS, J., LASRY, A., MCINNIS, D., SOLOMON, M.M., SOUMIS, F., *ALTITUDE: the airline operations management system at Air Transat,* Technical report G-95-23, GERAD, Montréal, 1995.

[DLS93] DIEKMANN, R., LÜLING, R., SIMON, J., *Problem independent distributed simulated annealing and its applications,* in: Applied simulated annealing, Hrsg. Vidal, R.V.V., Springer LNEMS 396, 1993, 17-44.

[DMM97] DELL'AMICO, M., MAFFIOLI, F., MARTELLO, S., (Hrsg.): *Annotated bibliographies in combinatorial optimization,* J. Wiley and Sons, Chichester, 1997.

[DP95] DADUNA, J.R., PAIXAO, J.M.P., *Vehicle scheduling for public mass transit – an overview,* in: [DBP95], 76-90.

[DR92] DESROCHERS, M., ROUSSEAU, J.-M., (Hrsg.), *Computer aided transit scheduling '90,* Springer LNEMS 386, 1992.

[DS89] DESROCHERS, M., SOUMIS, F., *A column generation approach to the urban transit crew scheduling problem,* Transport. Sci. 23, 1989, 1-13.

[DST97] DYCKHOFF, H., SCHEITHAUER, G., TERNO, J., *Cutting and packing,* in: [DMM97], 393-413.

[Due93] DUECK, G., *New optimization heuristics: the great deluge algorithm and the record-to-record travel,* J. Computat. Phys. 104, 1993, 86-92.

[EKD96] ERNST, A.T., KRISHNAMOORTHY, M., DOWLING, D., *Train driver rostering using simulated annealing,* Technical report, CSIRO, Australia, März 1996; zur Veröffentlichung eingereicht bei Ann. Oper. Res.

[EP98] EMDEN-WEINERT, TH., PROKSCH, M., *Best practice simulated annealing for the airline crew scheduling problem,* Technischer Bericht, Institut für Informatik, Humboldt-Universität zu Berlin, März 1998, erscheint in: Journal of Heuristics.

[EU95] *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.

[Fis95] FISHER, M., *Vehicle routing,* in: Network routing, Hrsg. Ball, B.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L., Handbooks in OR & MS Vol. 8, Elsevier, Amsterdam, 1995, 1-33.

[FK90] FISHER, M.L., KEDIA, P., *Optimal solution of set covering / partitioning problems using dual heuristics,* Management Sci. 36, 1990, 674-688.

[FR95] FEO, T.A., RESENDE, M.G.C. *Greedy randomized adaptive search procedures,* J. Global Opt. 6, 1995, 109-133.

[Ger89] GERSHKOFF, I., *Optimizing flight crew schedules*, Interfaces 19 (4), 1989, 29-43.

[GHR95] GREENLAW, R., HOOVER, H.J., RUZZO, W.L., *Limits to parallel computation: P-completeness theory,* Oxford University Press, 1995.

[GL97] GLOVER, F., LAGUNA, M., *Tabu search,* Kluwer Acad. Publ., 1997.

[Glo89] GLOVER, F., *Tabu search – part I,* ORSA J. Comput. 1, 1989, 190-206.

[Glo90] GLOVER, F., *Tabu search – part II,* ORSA J. Comput. 2, 1990, 4-32.

[GLP97] GENDREAU, M., LAPORTE, G., POTVIN, J.-Y., *Vehicle routing: modern heuristics,* in: [AL97], 311-336.

[GLS88] GRÖTSCHEL, M., LOVÁSZ, L., SCHRIJVER, A., *Geometric algorithms and combinatorial optimization,* Springer, Berlin, 1988, 2. verbesserte Auflage 1993.

[GLS94] GROPP, W., LUSK, E., SKJELLUM, A., *Using MPI – Portable parallel programming with the message-passing interface,* MIT Press, 1994.

[GMG+93] GRAVES, G.W., MCBRIDE, R.D., GERSHKOFF, I., ANDERSON, D., MAHIDHARA, D., *Flight crew scheduling*, Management Science 39, 1993, 736-745.

[GSMD94] GAMACHE, M., SOUMIS, F., MARQUIS, G., DESROSIERS, J., *A column generation approach for large scale aircrew rostering problems,* Cahier du GERAD G-94-20, 1994.

[GTW93] GLOVER, F., TAILLARD, E., DE WERRA, D., *A user's guide to tabu search,* in: Tabu search, Hrsg. Glover, F., De Werra, D., Ann. Oper. Res. 41, 1993, 3-28.

[Hoc96] HOCHBAUM, D.S., (Hrsg.), *Approximation algorithms for NP-hard problems,* PWS Publishers, Boston, 1996.

[HP93] HOFFMAN, K., PADBERG, M., *Solving airline crew scheduling problems by branch-and-cut*, Management Science 39, 1993, 657-682.

[HP95] HOFFMAN, K., PADBERG, M., *Bestimmung optimaler Einsatzpläne für Flugpersonal,* in: Mathematik in der Praxis, Hrsg. Bachem, A., Jünger, M., Schrader, R., Springer, 1995, 509-532.

[JAMS89] JOHNSON, D.S., ARAGON, C.R., MCGEOCH, L.A., SCHEVON, C., *Optimization by simulated annealing: an experimental evaluation; Part I: graph partitioning,* Oper. Res. 37, 1989, 865-892.

[JAMS91] JOHNSON, D.S., ARAGON, C.R., MCGEOCH, L.A., SCHEVON, C., *Optimization by simulated annealing: an experimental evaluation; Part II: graph coloring and number partitioning,* Oper. Res. 39, 1991, 378-406.

[Jer92] JERRUM, M., *Large cliques elude the metropolis process,* Random Struct. Alg. 3, 1992, 347-359.

[JM97] JOHNSON, D.S., MCGEOCH, L.A., *The traveling salesman problem: a case study,* in: [AL97], 215-310.

[JNS97] JOHNSON, E.L., NEMHAUSER, G.L., SAVELSBERGH, M.W.P., *Progress in integer programming: an exposition,* erscheint in INFORMS J. Comput., 1997.

[JPY88] JOHNSON, D.S., PAPADIMITRIOU, C.H., YANNAKAKIS, M., *How easy is local search?,* J. Comput. System Sci. 37, 1988, 79-100.

[JS93] JERRUM, M., SORKIN, G.B., *Simulated annealing for graph bisection,* Proc. 34th IEEE FOCS, 1993, 94-103.

[JS97] JERRUM, M., SINCLAIR, A., *The Markov chain Monte Carlo method: an approach to approximate counting and integration,* in: Approximation algorithms for NP-hard problems, Hrsg. Hochbaum, D.S., PWS Publishers, Boston, 1997, 482-520.

[KAJ94] KOULAMAS, C., ANTONY, S., JAEN, R., *A survey of simulated annealing applications to operations research problems,* Omega Int. J. Mgmt. Sci. 22, 1994, 41-56.

[Kar72] KARP, R.M. *Reducibility among combinatorial problems,* in: Complexity of computer computations, Hrsg. Miller, R.E., Thatcher, J.W., Plenum Press, 1972, 85-103.

[Kar84] KARMARKAR, N., *A new polynomial time algorithm for linear programming,* Combin. 4, 1984, 373-395.

[KGV83] KIRKPATRICK, S., GELATT, C.D., VECCHI, M.P., *Optimization by simulated annealing,* Science 220, 1983, 671-680.

[Kha79] KHACHIAN, L.G., *A polynomial algorithm for linear programming*, Soviet Math. Doklady, 20, 1979, 191-194.

[KL70] KERNIGHAN, B.W., LIN, S., *An efficient heuristic procedure for partitioning graphs,* Bell Systems Tech. J. 49, 1970, 291-307.

[KM72] KLEE, V., MINTY, G.J., *How good is the simplex algorithm?*, in: Inequalities III, Hrsg. Shisha, O., Academic Press, New York, 1972, 159-175.

[KM96] KOCH, T., MARTIN, A., *Solving Steiner tree problems in graphs to optimality,* Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 96-42, 1996.

[KR90] KARP, R., RAMACHANDRAN, V., *Parallel algorithms for shared-memory machines*, in: Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity, Hrsg. Van Leeuwen, J., Elsevier Science Publ., 1990, 871-941.

[Kre90] KRENTEL, M.W., *On finding and verifying locally optimal solutions,* SIAM J. Comput. 19, 1990, 742-749.

[KU98] KLIEWER, G., UNRUH, P., *Parallele Simulated Annealing Bibliothek: Entwicklung und Einsatz in der Flugplanoptimierung,* Diplomarbeit, Fachbereich Mathematik / Informatik, Universität-Gesamthochschule Paderborn, 1998.

[LA87] VAN LAARHOVEN, P.J.M., AARTS, E.H.L., *Simulated annealing: theory and applications,* Kluwer Acad. Publ., 1987.

[LBA94] *Zweite Durchführungsverordnung zur Betriebsordnung für Luftfahrtgerät (2. DV LuftBO),* Luftfahrt-Bundesamt, 2/94.

[Lee95] LEE, F. H. A., *Parallel simulated annealing on a message-passing multicomputer,* Dissertation, Utah State University, 1995.

[Lei92] LEIGHTON, F.T., *Introduction to parallel algorithms and architectures: arrays . trees . hypercubes,* Morgan Kaufmann, San Mateo CA, 1992.

[LK73] LIN, S., KERNIGHAN, B.W., *An effective heuristic for the traveling salesman problem,* Oper. Res. 21, 1973, 498-516.

[LL94] LORENA, L.A.N., LOPES, F.B., *A surrogate heuristic for set covering problems,* Europ. J. Oper. Res. 79, 1994, 138-150.

[LMO88] 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, 1988, 45-58.

[LTT89] LLEWELLYN, D.C., TOVEY, C., TRICK, M., *Local optimization on graphs,* Discrete Applied Math. 23, 1989, 157-178.

[Lub86] LUBY, M., *A simple algorithm for the maximal independent set problem,* SIAM J. Comp. 15, 1986, 1036-1053.

[MRR+53] METROPOLIS, N., ROSENBLUTH, A.W., ROSENBLUTH, M.N, TELLER, A.H., TELLER, E., *Equation of state calculations by fasts computing machines,* J. Chem. Phys. 21, 1953, 1087-1092.

[MT90] MARTELLO, S., TOTH, P., *Knapsack problems – algorithms and computer implementations,* Wiley & Sons, Chichester, 1990.

[MT92] MARTELLO, S., TOTH, P., *Generalized assignment problems,* in: Algorithms and computation – ISAAC'92, Hrsg. Ibaraki, T., et al., Springer LNCS 650, 1992, 351-369.

[NS96] NOLTE, A., SCHRADER, R., *Simulated annealing and its problems to color graphs,* in: Algorithms - ESA'96, Hrsg. Diaz, J., Serna, M., Springer LNCS 1136, 1996, 138-151.

[NS97] NOLTE, A., SCHRADER, R., *Coloring in sublinear time,* in: Proc. 5th Annual European Symposium on Algorithms – ESA'97, Hrsg. Burkard, R., Woeginger, G., Springer LNCS 1284, Berlin, 1997.

[NW88] NEMHAUSER, G.L., WOLSEY, L.A., *Integer and combinatorial optimization,* Wiley, New York, 1988.

[OL96] OSMAN, I.H, LAPORTE, G., *Metaheuristics: a bibliography,* Ann. Oper. Res. 63, 1996, 513-628; als PDF-Datei erhältlich an der URL „http://www.informatik.hu-berlin.de/~weinert/SA.html".

[Osm95] OSMAN, I.H., *Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches,* OR Spektrum 17, 1995, 211-225.

[Pap92] PAPADIMITRIOU, C.H., *The complexity of the Lin-Kernighan heuristic for the traveling salesman problem,* SIAM J. Comput. 21, 1992, 450-465.

[Pap94] PAPADIMITRIOU, C.H., *Computational complexity,* Addison-Wesley, Reading MA, 1994.

[Pro97] PROKSCH, M., *Simulated Annealing und seine Anwendung auf das Crew Scheduling Problem,* Diplomarbeit, Institut für Informatik, Humboldt-Universität zu Berlin, 1997.

[PS77] PAPADIMITRIOU, C.H., STEIGLITZ, K., *On the complexity of local search for the traveling salesman problem*, SIAM J. Comput. 6, 1977, 76-83.

[PS78] PAPADIMITRIOU, C.H., STEIGLITZ, K., *Some examples of difficult traveling salesman problems*, Oper. Res. 26, 1978, 434-443.

[PS92] PRÖMEL, H.J., STEGER, A., *Coloring clique-free graphs in linear expected time*, Random Struct. Alg. 3, 1992, 375-402.

[Ree95] REEVES, C.R., (Hrsg.), *Modern heuristic techniques for combinatorial problems,* McGraw-Hill, London, 1995.

[RBP+91] RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., LORENSEN, W., *Object-oriented modeling and design,* Prentice-Hall Intern., Englewood Cliffs, 1991.

[RHP95] RUSHMEIER, R.A., HOFFMAN, K.L., PADBERG, M., *Recent advances in exact optimization of airline scheduling problems*, Technischer Bericht, Juli 1995, erhältlich über das WWW an der URL „http://bass.gmu.edu/~khoffman/hoffman.html".

[RS91] ROMEO, F., SANGIOVANNI-VINCENTELLI, A., *A theoretical framework for simulated annealing,* Algorithmica 6, 1991, 302-345.

[RT87] RÖDL, V., TOVEY, C., *Multiple optima in local search,* J. Alg. 8, 1987, 250-259.

[Rub73] RUBIN, J., *A technique for the solution of massive set covering problem, with application to airline crew scheduling,* Transport. Sci. 7, 1973, 34-48.

[Rya92] RYAN, D.M., *The solution of massive generalized set partitioning problems in aircrew rostering,* J. Opl. Res. Soc. 43, 1992, 459-467.

[Sch86] SCHRIJVER, A., *Theory of linear and integer programming,* Wiley & Sons, Chichester, 1986.

[SG76] SAHNI, S., GONZALEZ, T., *P-complete approximation problems,* J. ACM 23, 1976, 555-565.

[SH88] SASAKI, G.H., HAJEK, B., *The time complexity of maximum matching by simulated annealing,* J. ACM 35, 1988, 387-403.

[Sin93] SINCLAIR, M., *Comparison of the performance of modern heuristics for combinatorial optimization on real data,* Computers Ops. Res. 20, 1993, 687-695.

[SOH+95] SNIR, M., OTTO, S.W., HUSS-LEDERMAN, S., WALKER, D.W., DONGARRA, J., *MPI – The complete reference,* MIT Press, 1995.

[Suh95] SUHL, L., *Computer-aided scheduling - an airline perspective,* Gabler Edition Wissenschaft, Wiesbaden, 1995.

[SY91] SCHÄFFER, A.A., YANNAKAKIS, M., *Simple local search problems that are hard to solve,* SIAM J. Comput. 20, 1991, 56-87.

[TD96] THOMPSON, J.M., DOWSLAND, K.A., *Variants of simulated annealing for the examination timetabling problem,* Ann. Oper. Res. 63, 1996, 105-128.

[VA95] VERHOEVEN, M.G.A., AARTS, E.H.L., *Parallel local search,* J. Heuristics 1, 1995, 43-65.

[Wed95] WEDELIN, D., *An algorithm for 0-1 programming with an application to airline crew scheduling,* in: Mathematics of industrial systems I, Hrsg. Burkard, R.E., Ibaraki, T., Queyranne, M., Ann. Oper. Res. 57, 1995, 283-301.

[WR95] WREN, A., ROUSSEAU, J.-M., *Bus driver scheduling - an overview,* in: [DBP95], 173-187.

[Yan97] YANNAKAKIS, M., *Computational complexity,* in: [AL97], 19-55.