Universität Bonn
Institut für Informatik III
Prof. Dr. A.B. Cremers

Diplomarbeit

zum Thema

Theorie und Praxis der automatischen Stundenplanerstellung

Vorgelegt von
Martin Löhnertz

WS 1998/99

Hiermit versichere ich, die Arbeit selbständig durchgeführt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht zu haben.

Danksagung

Die vorliegende Arbeit wurde betreut durch Prof. Dr. A.B. Cremers sowie Dipl. Math. C. Hundack und Dipl. Inf. J. Lüssem, denen ich für ihre Unterstützung danke. Mein besonderer Dank gilt Prof. Dr. M. Karpinski, der sich die Zeit für eine längere Diskussion zum Thema Approximationsalgorithmen nahm. Ebenso danke ich den Lehrern, die bereit waren, mir Testdaten zu überlassen bzw. an der Befragung teilzunehmen.

Martin Löhnertz Bonn, 1.2. 1999

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Einführung . . . 1
1.2 Ziele der Arbeit und Überblick . . . 3

2 Problemstellungen **5**
2.1 Terminologie . . . 5
2.2 STP . . . 7
2.3 Daily School Timetabling . . . 8
2.4 UTP . . . 9
2.5 Das Kurswahlproblem . . . 10
2.6 Prüfungsverteilung . . . 10
2.7 Raumverteilung . . . 11
2.8 Nachbargebiete . . . 12
2.9 Transformationen . . . 12
2.10 Zusammenfassung . . . 13

3 Komplexität **15**
3.1 Terminologie . . . 15
3.2 Polynomiell lösbare Fälle . . . 16
3.3 Realistische Problemstellungen sind $\mathcal{NP}$-vollständig . . . 16
3.4 Approximierbarkeit . . . 17
3.5 Randomisierte Algorithmen/Heuristiken . . . 21
3.6 Die Komplexität von Teilproblemen . . . 21
3.7 Der „Normalfall" . . . 23
3.8 Zusammenfassung . . . 24

4 Lösungsansätze **25**
4.1 Direkte Heuristiken . . . 26
4.2 Expertensysteme . . . 27
4.3 Graphentheoriebasierte Verfahren . . . 27
4.4 LP / ILP / SDP . . . 38
4.5 Constraintbasierte Verfahren . . . 41
4.6 Genetische Algorithmen . . . 43
4.7 Simulated Annealing . . . 44
4.8 Negotiation . . . 46
4.9 Weitere Ansätze . . . 46
4.10 Unergiebige Ansätze . . . 47
4.11 Zusammenfassung . . . 52

5 Programmarchitektur und Algorithmen **55**
5.1 Problemstellung . . . 55
5.2 Modell . . . 56
5.3 Gesamtaufbau . . . 59
5.4 Wahl der Algorithmen . . . 60

6 Verfahren 1: Tabusuche **63**
6.1 Allgemeine Betrachtungen zum Lösungsraum . . . 63
6.2 Tabuliste . . . 64
6.3 Tabusuche . . . 65
6.4 Kodierung . . . 66
6.5 Wahl der Nachbarschaft . . . 66
6.6 Hybridisierung . . . 69
6.7 Implementierung dieses Verfahrens . . . 76
6.8 Variationen des Graphenalgorithmus . . . 78
6.9 Zusammenfassung . . . 79

7 Verfahren 2: Handels - Heuristik **81**
7.1 Grundlagen . . . 81
7.2 Das Prinzip des „Simulated Trading" . . . 83
7.3 Erweiterung auf eine „individuell ausgeglichene" Methode . . . 84
7.4 Spieltheorie . . . 84
7.5 Ablauf der Auktion und Strategien . . . 87
7.6 Erste Versuche einer Implementierung . . . 91
7.7 Zusammenfassung . . . 93

8 Implementierung **95**
8.1 Datenstrukturen . . . 95
8.2 Editor . . . 96
8.3 Die interaktive Unterstützung . . . 103
8.4 Besonderheiten . . . 104
8.5 Sonstiges . . . 109
8.6 Qualitätsprüfung und -management . . . 110
8.7 Zusammenfassung . . . 111

9 Evaluation **113**
9.1 Testdaten . . . 113
9.2 Ergebnisse . . . 113
9.3 Weitere Anmerkungen . . . 116

10 Ausblick **117**
10.1 Bemerkungen zur betrachteten Problemstellung . . . 118
10.2 Erweiterungsmöglichkeiten . . . 119
10.3 Perspektiven . . . 120

Literaturverzeichnis **122**

A Beispielausdruck **131**

Kapitel 1

Einleitung

1.1 Einführung

Die automatische Stundenplanerstellung bildet trotz jahrzehntelanger intensiver Bemühungen eines der letzten Gebiete der kombinatorischen Optimierung, in denen die menschliche Expertise unverzichtbar geblieben ist. Die bisher vollzogene formale Modellbildung reicht noch bei weitem nicht an die Aufgabenstellungen der Praxis heran, obwohl die Problemstellung intuitiv klar und einfach ist und zahlreiche wissenschaftliche Arbeiten anzugeben vermögen, wie ein guter Plan prinzipiell aussehen sollte. Und selbst diese noch zu kurz greifenden Untersuchungen zeigen mit wenigen Ausnahmen, daß mit einer einfachen Lösbarkeit des Problems nicht zu rechnen ist.
Verwundern muß aber in diesem Zusammenhang, daß das Problem - wenn auch meist mit sehr hohem Zeitaufwand - von Personen ohne kombinatorische Vorbildung häufig zufriedenstellend und sogar besser gelöst wird, als dies mit den bestehenden wissenschaftlichen Ansätzen möglich ist. Dies mag daran liegen, daß man sich bei formalen Ansätzen stets um ein gewisses Maß an Allgemeinheit bemüht, während die mit der Planung Beauftragten spezifische und damit vereinfachte Lösungen anstreben können. Aber die Regelmäßigkeit mit der dies geschieht, läßt dennoch vermuten, daß es auch unabhängig von konkreten Problemstellungen bessere Verfahren geben müßte.

Man sieht sich also mit dem sehr interessanten Effekt konfrontiert, daß, obwohl der Rechner Widersprüche, Abweichungen etc. wesentlich sicherer, schneller und zuverlässiger erkennen kann als ein menschlicher Stundenplaner, dieser ihm im Auflösen dieser Konflikte immer noch überlegen ist. Vor diesem Hintergrund, der dem Computer vorläufig nur oder zumindest die Rolle des Helfers zubilligt, erscheint es sinnvoll, sich zunächst mit der Lösbarkeit zunehmend umfangreicherer Teilprobleme zu beschäftigen. Hierdurch kann einerseits dem Planer eine immer bessere Unterstützung gegeben werden und andererseits kann jedes weitere gelöste Teilproblem vielleicht dazu beisteuern, der Lösung der eigentlichen Aufgabe näher zu kommen.

Die Menge dieser Teilprobleme erweist sich dabei als so groß und mannigfaltig, daß ihre Betrachtung einem Streifzug durch die meisten Methoden der kombinatorischen Optimierung gleichkommt. Hier zeigt sich rasch, daß der weitaus überwiegende Teil der Probleme mindestens $\mathcal{NP}$-hart ist. Insoweit scheint die Frage nach der Existenz guter Algorithmen negativ beantwortet und in den letzten Jahrzehnten konnten hier, abgesehen von der Identifizierung außergewöhnlich spezieller Fälle, kaum Fortschritte verzeichnet werden. Diese Beurteilung scheint erlaubt, wird doch im Rahmen des

von der GMD betreuten Projekts „INTEGA“ noch heute der Algorithmus von 1978 verwandt, da es, nach Selbstauskunft der GMD im Internet Ende 1998, seither keine Veränderungen gegeben habe. Die Fortschritte, die in letzter Zeit im Bereich der Approximationsalgorithmen erzielt wurden, haben sich bisher in diesem Gebiet noch nicht bemerkbar gemacht.
Parallel entwickelten sich entsprechende Heuristiken, teilweise auch unter Einfluß der Methoden der künstlichen Intelligenz, weiter, wobei es allerdings bis heute keinerlei gesicherte Aussagen zur Leistungsfähigkeit gibt, mit denen sich zeigen ließe, daß sie den älteren Verfahren überlegen sind. Da hier insbesondere die neueren Methoden eher allgemeine Optimierungsheuristiken für beliebige kombinatorische Probleme sind, hat die automatische Stundenplanerstellung mit wenigen Ausnahmen (s. Kapitel 4.3.5) keine eigenen tieferen Resultate vorzuweisen, sondern bedient sich, soweit diese anwendbar sind, der Ergebnisse anderer Theorien. Mit Ausnahme einiger neuerer Anwendungen der Theorie von Markovketten können aber auch hier zumeist nur die grundlegenden Erkenntnisse verwendet werden.
Das Fehlen einer Abgrenzung der Stundenplanproblematik gegenüber allgemeineren Problemstellungen, wie z.B. dem Färben beliebiger Graphen, führte und führt zudem dazu, daß jeder Fortschritt auf diesem Gebiet sofort wichtige Erkenntnisse für die gesamte Optimierungstheorie mit sich bringen würde. Methodisch wäre es daher in diesem Kontext vorzuziehen, zunächst eine Lösung der theoretischen Probleme anzugehen. Die Langwierigkeit dieses Vorgehens hat jedoch in der Wechselwirkung von hohem Praxisdruck und unzureichender theoretischer Grundlagen zahlreiche „Rezepte“ entstehen lassen, die mehr dem „gesunden Menschenverstand“ folgen, als einem theoretisch abgesicherten Prinzip.

Eine andere, aber mindestens ebenso wichtige Dimension des Problemes liegt in der bisher unzureichenden Kommunikation zwischen Mensch und Algorithmus. So kann es vorkommen, daß gemäß gegebener Zielfunktionen tatsächlich optimale Pläne für die praktische Anwendung inakzeptabel sind, weil die Unzulänglichkeiten des Planes hierbei vom Benutzer erst erkannt werden können, wenn ihm dieser fertig präsentiert wird. Häufig ist es so, daß ein zufriedenstellender Plan erst durch mehrfaches Nachbessern der Parameter und durch einen Arbeitseinsatz seitens des Benutzers, der dem Aufwand nahekommt, den er ohne Unterstützung hätte erbringen müssen, erzielt werden kann. Da aber bisher noch keine Möglichkeit gefunden wurde, die Wünsche des Anwenders von vorneherein korrekt zu erfassen, ist man hier auf hochgradig interaktive Systeme angewiesen.
Aus diesem Grunde darf sich die Untersuchung des Problems, sofern man sich nicht auf einen vollständig theoretischen Standpunkt zurückziehen will, nicht auf die Untersuchung eines einzelnen isolierten Verfahrens mit statischen Parametern beschränken. Erst eine Einbindung in ein vollständiges System und der Vergleich mit anderen Vorgehensweisen im Rahmen dieses größeren Zusammenhangs vermag die Vor- und Nachteile einer konkreten Methode sichtbar werden zu lassen. Dies bringt es mit sich, daß derartige Unterfangen eine Mindestgröße aufweisen, die deutlich über der üblicher prototypischer Implementierungen liegen. Programme zur Unterstützung der manuellen Planung mit der Möglichkeit zum Datenexport wie „MIMOSA“ [Mim98] können hier keine Abhilfe schaffen, es sei denn, man entschiede sich, den Paradigmen, die den zur Verfügung gestellten Daten inhärent sind, vollständig zu folgen.

Haben auch die meisten Menschen ihre früheren (Schul-) Stundenpläne als eine profane Regelung in Erinnerung, die angab, ob gerade Geschichte oder Biologie zu lernen sei, so verbarg und verbirgt sich dahinter doch die Arbeit von Wochen oder aber die von Tagen und eines rechnerbasierten Hilfsmittels. Insbesondere der schulische Stundenplan, dessen Struktur einerseits zahlreichen ministerialen und arbeitsrechtlichen Regelungen unterworfen ist, andererseits aber auch pädagogische Absichten unterstützen und für das Lehrpersonal erträglich sein soll, erweist sich als eine der großen Herausforderungen auf diesem Gebiet. Nahezu alle Nebenbedingungen, die bei anderen Planungsaufgaben auftreten, finden sich im Bereich der Bildungsanstalten wieder. Daher bietet sich dieses, fast prototypisch zu nennende Problem für eine Untersuchung besonders an.

1.2 Ziele der Arbeit und Überblick

Die vorliegende Arbeit stellt die erste ausführlichere Untersuchung zum Gebiet der automatischen Stundenplanerstellung am Institut für Informatik dar und dient in erster Linie dazu, neben einem umfassenden Überblick den Rahmen der Anwendbarkeit vorhandener Theorien innerhalb dieses Gebietes zu sondieren. Zweites Ziel ist es, durch die Erstellung eines Programms zur automatischen Stundenplanerstellung erste praktische Erfahrungen mit Implementierungen zu gewinnen, die über die Ansprüche eines Prototyps zum Algorithmentest hinausgehen, und in diesem Rahmen die Funktionsfähigkeit verschiedener Lösungsmethoden zu untersuchen.

Angestrebt wurde hierbei ein Programm, das sowohl den Anforderungen einer Realschule, als auch denen einer Fachhochschule gerecht werden würde. Da bei ersteren der Klassenunterricht überwiegt, während bei letzteren verstärkt Kursunterricht auftritt, umfassen die Anforderungen damit auch die Ansprüche von Gymnasien und Gesamtschulen. Die somit erfolgte Konkretisierung der Anforderungen ist Garant dafür, daß der dringend notwendige Praxisbezug bei der Betrachtung der Verfahren nicht vernachlässigt werden kann, während die Unterschiedlichkeit der Voraussetzungen dennoch eine ausreichende Allgemeinheit sichert.

Die Arbeit gliedert sich in drei Teile, wobei der erste Stand und Möglichkeiten der Entwicklung referiert, während die beiden anderen sich mit den im Rahmen dieser Arbeit entwickelten Verfahren befassen:

Der erste Teil gibt in Kapitel 2 zunächst einen Überblick über die verschiedenen Probleme, die unter dem Begriff der automatischen Stundenplanerstellung zusammengefaßt werden und führt den benötigten Begriffsapparat ein. Des weiteren wird aufgezeigt, daß einige der zunächst verschieden scheinenden Problemstellungen übereinstimmen.
In Kapitel 3 wird die Komplexität des Problems untersucht, wobei insbesondere die Nichtapproximierbarkeit unter den bisher formulierbaren Nebenbedingungen betrachtet wird. Die hier feststellbaren Eigenschaften lassen die Tatsache, daß alle Schulen über einen funktionsfähigen Stundenplan verfügen, erstaunlich erscheinen, was darauf hindeutet, daß es weitere noch nicht identifizierte Nebenbedingungen geben muß (Abschnitt 3.7 S. 23). Es wird ebenfalls kurz auf die Komplexität verschiedener Teilprobleme eingegangen.
In Kapitel 4 folgt dann eine Übersicht über die bisher eingesetzten Lösungsverfahren, wobei hier ein kleiner Schwerpunkt bei graphentheoretischen Ansätzen liegt, da auf einen relativ neuen Satz eingegangen wird, der bisher im „Timetabling“ noch keine Beachtung gefunden zu haben scheint (Abschnitt 4.3.4 S. 33). Außerdem wird hier für eine große Zahl erfolgversprechend scheinender Verfahren die meist einen möglichen

Einsatz ausschließende Grenze ihrer Anwendbarkeit nachgewiesen.

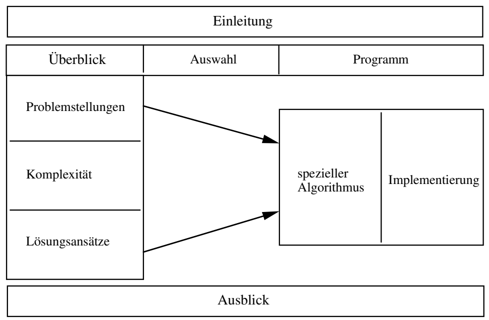
Der zweite Teil beschreibt die in dieser Arbeit näher untersuchten Lösungsverfahren. In Kapitel 5 wird zunächst ein Überblick über das Gesamtkonzept gegeben und das verwendete Modell, sowie die betrachteten Zielfunktionen werden diskutiert. Ebenso wird hier aufgezeigt, wie sich die im Rest dieses Teils untersuchten Verfahren in die Struktur des Gesamtprogramms einfügen.

Das erste Verfahren (Kapitel 6) folgt dem aktuell meistversuchten Ansatz, der Tabusuche, wobei diese zusätzlich mit einem komplexen graphentheoretischen Ansatz hybridisiert wurde (Abschnitt 6.6 S. 69). Hier wurde basierend auf einer genauen Betrachtung des Lösungsraums eine neue Form der Tabuliste eingesetzt.
Das zweite Lösungsverfahren basiert auf einer neuen Methode, die unter anderem durch das Verfahren von Bachem [BHM96] inspiriert wurde. Hier werden in Kapitel 7 ausgehend von dem bei Bachem beschriebenen heuristischen Verfahren, das auf der Simulation einer Handelssituation beruht, die Möglichkeiten einer theoretischen Fundierung untersucht und eine erste Testimplementierung vorgenommen.

Der dritte Teil geht auf die Struktur und auf die wesentlichen Algorithmen des im Rahmen der Arbeit erstellten Programms ein.
Hierbei werden in Kapitel 8 sowohl die verwendeten Datenstrukturen, als auch die Entscheidungen, die zur Festlegung der Semantik einiger Operationen notwendig waren, dokumentiert. Da benutzerfreundliche Dialoge und Eingabemöglichkeiten vorgesehen wurden und somit auch Konsistenzprüfungen und Eingabeinterpretationen notwendig waren, erreichen Teile dieses „Rahmenprogramms“ durchaus auch die Komplexität, die man eigentlich nur in den Lösungsverfahren vermuten würde (z.B. Abschnitt 8.4.2 S. 105). Des weiteren wurde dem oben angesprochenen Paradigma hoher Interaktivität folgend eine Beratungsfunktion zur Ergänzung der interaktiven Veränderungsmöglichkeiten geschaffen, die den Benutzer je nach Situation mehrere Schritte vor dem Auftreten einer Kollision auf diese hinweisen kann (Abschnitt 8.3 S. 103).

Die Arbeit endet mit einer Evaluation der durchgeführten Testläufe in Kapitel 9 und mit einem kurzen Ausblick in Kapitel 10.

Insgesamt stellt sich das ganze also wie folgt dar:



Kapitel 2

Übersicht über das Spektrum der Problemstellungen

2.1 Terminologie

Obwohl das Gebiet als solches schon relativ alt ist [Lew61], konnte sich bis heute keine einheitliche Terminologie durchsetzen. Infolgedessen mußten innerhalb dieser Arbeit eigene Definitionen vorgenommen werden, wobei zu großen Teilen SCHAERF [Sch95] gefolgt wurde.

2.1.1 „Scheduling“, „Roostering“

Im allgemeinen wird die automatische Stundenplanerstellung als Teilgebiet des „Scheduling“ betrachtet. Wie WREN [Wre95] hierzu ausführt, hat sich diese Relation aber im Laufe der Zeit verschoben, insbesondere, da sich der Schwerpunkt im „Scheduling“ langsam zugunsten speziellerer Fragestellungen verändert hat. In [Wre95] finden sich folgende Definitionsversuche:

Scheduling „ist das Lösen praktischer Probleme bestehend aus der von Nebenbedingungen beeinflußten Zuordnung von Ressourcen zu Objekten, die in der Raumzeit[1] angeordnet werden“, wobei (laut WREN) die Wahl der Methoden keine Rolle spielt. Häufig tritt dabei auch ein Optimierungsaspekt hinzu.

Timetabling ist „Scheduling“ mit dem Versuch, die Nebenbedingungen so gut wie nur möglich zu erfüllen.

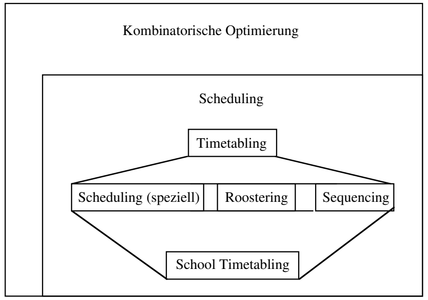
Scheduling (speziell) ist „Scheduling“ zur Minimierung einer Teilmenge der verwendeten Ressourcen.

Roostering ist die Ressourcenzuteilung in einem ansonsten fertigen Plan.

Sequencing beschränkt sich auf die Angabe einer Reihenfolge, ohne konkrete Zeitpunkte oder Orte festzulegen.

Somit ergibt sich das folgende Bild, wobei auffällig ist, daß es keinen eigentlichen Unterschied zwischen „Scheduling“ und „Timetabling“ mehr gibt.

[1] nicht physikalisch zu verstehen



Obwohl sich das „Timetabling“ als Gebiet vom speziellen „Scheduling“ schon soweit entfernt hat, daß einige Autoren bereits explizit darauf hinweisen, wenn sie Methoden aus diesem Gebiet anwenden [SM97], sind doch einige grundlegende Ideen aus diesem Themenkomplex übernommen worden. Beschränkt man sich zunächst auf die „herkömmliche“ Aufgabe des „Scheduling“, Aktionen, die bestimmte Ressourcen benötigen, zeitlich anzuordnen, so können folgende Adjektive dieses Gebiets zur genaueren Klassifikation der automatischen Stundenplanerstellung verwendet werden:

preemptiv ↔ nicht preemptiv: Preemptiv bedeutet, daß die Aktionen unterbrochen werden können.

statisch ↔ dynamisch: Als dynamisch bezeichnet man die Variante des Problems, bei der einzelne Aktionen verplant werden müssen, bevor alle dem Algorithmus bekannt sind. Der dynamische Fall tritt z.B. beim Prozessor-Scheduling auf. Diese Variante fällt somit in den Bereich der „Online“-Algorithmen.

Demnach wäre Stundenplanerstellung nichtpreemptives, statisches Scheduling.

2.1.2 Das „open shop scheduling model“ (OSS)

Das „Open Shop Scheduling Modell“ (z.B. [dW96]) unterscheidet sich vom schulischen Problem in erster Linie durch den verwendeten Kontext: Eine Menge von Prozessoren $\{p_1, \ldots, p_n\}$ soll eine Menge von Aufgaben $\{a_1, \ldots, a_m\}$ bearbeiten. Jede Aufgabe besteht aus n Einzelschritten $\{t_{j1}, \ldots, t_{jn}\}$ $j = 1, \ldots, m$, wobei Einzelschritt $t_{\bullet i}$ auf Prozessor p_i ausgeführt werden muß. Jeder Prozessor kann nur einen Einzelschritt gleichzeitig behandeln und von jeder Aufgabe kann sich nur ein Einzelschritt in Bearbeitung befinden.

DEWERRA [dW96] unterscheidet weiter zwischen dem preemptiven OSS (POSS) und dem nichtpreemptiven (NOSS). Bei ersterem können die Einzelschritte zu beliebigen Zeitpunkten unterbrochen und später fortgesetzt werden. Dem schulischen Problem entspräche ein diskretisiertes POSS, bei dem die Einzelschritte in jeweils nicht unterbrechbare kleinere Stücke gleicher Größe aufgeteilt werden können.

Im Rahmen des sogenannten „shop scheduling“ gibt es noch zwei weitere Modelle, „flow shop“ (FSS) und „job shop“ (JSS). Beim „job shop“ ist eine Reihenfolge der einzelnen Operationen einer Aufgabe vorgegeben, beim „flow-shop“ haben alle Aufgaben die gleiche Zahl von Einzelschritten und die Reihenfolgen sind ebenfalls für alle identisch.

2.1.3 „Eigene“ Bezeichnungen

Die folgenden Bezeichnungen entsprechen größtenteils dem an Schulen üblichen Sprachgebrauch.

- „Zeitstunde“: Im Gegensatz zur Schulstunde ein Zeitrahmen, zu dem eine solche stattfinden kann. Hier wird angenommen, daß sich verschiedene Zeitstunden nicht überlappen, d.h. daß ein Objekt dadurch, daß es in Stunde a verfügbar ist, nicht a priori in Stunde b nicht zur Verfügung steht.
- „Differenzierungsstunde“: Unter differenziertem Unterricht soll hier eine Unterrichtseinheit verstanden werden, an der verschiedene Lehrer und Klassen gleichzeitig teilnehmen. So werden z.B. häufig die Schüler einer Jahrgangsstufe in zwei nach Geschlechtern getrennten Gruppen in Sport unterrichtet.
- „Springstunde“: Als Springstunde bezeichnet man eine Zeitstunde, zu der ein Objekt (Lehrer, Klasse) keinen Unterricht hat, vor der und nach der dieses aber beschäftigt ist. Insbesondere können diese auch für Lehrer-Klassen-Paare definiert werden.
- „Relative Nebenbedingung“: Eine Anforderung an eine Zuordnung, die durch eine andere Zuordnung beeinflußt wird. So kann z.B. eine Springstunde nur dann auftreten, wenn eine andere Stunde vorhanden ist, relativ zu der sie springt.
- „Harte Bedingungen“: Harte Nebenbedingungen sind solche, die unabdingbar sind, wie z.B. diejenige, daß keine Person zur selben Zeit an verschiedenen Orten sein kann. Analog sind „weiche“ Nebenbedingungen erwünscht aber nicht notwendig. Eine Lösung, die die harten Nebenbedingungen erfüllt, heißt **geeignet** und die zugehörige Aufgabe das „Such“-Problem. Eine geeignete Lösung, die die „weichen“ Nebenbedingungen bestmöglich erfüllt, heißt **optimal** und man spricht entsprechend vom Optimierungsproblem.
- „Flexible Ressource“: Eine flexible Ressource ist eine, die zu einem gewissen Grad und kontextabhängig ausgetauscht werden kann. So z.B. ist ein Klassenraum eine flexible Ressource, da zumeist in einen anderen Raum ausgewichen werden kann. Wird dieser aber explizit angefordert, so wird er zur statischen Ressource. Lehrer und Klassen treten zumeist als solche auf.

2.2 Schulische Stundenpläne (School-Timetabling/„STP“)

Informelle Beschreibung:
Das School-Timetabling-Problem (STP) ist das klassische Problem des Gebietes und in seiner Grundform mit dem oben dargestellten Spezialfall des POSS (s. Abschnitt 2.1.2) identisch. Gegeben eine Menge von Lehrern und eine von Klassen, sowie eine Vorgabe, wieviele Stunden ein Lehrer eine Klasse unterrichten soll, ist eine Zuordnung von Unterrichtsstunden zu Schulstunden zu finden.

Darstellung als ILP[2]:
Gesucht ist eine Lösung des Problems: Gegeben $l, k, t \in \mathbb{N}$ und $r_{a,b} \in \mathbb{N}^{l\times k}$ finde

[2] Ganzzahliges (integer) lineares Programm: Wenn nicht anders angegeben, sind stets alle Variablen aus $\{0,1\}$

$x_{a,b,c} \in \{0,1\}^{l\times k\times t}$ mit

$$\begin{aligned}
\sum_{i=0}^{k} x_{a,i,c} &\leq 1 \ \forall a, c\\
\sum_{i=0}^{l} x_{i,b,c} &\leq 1 \ \forall b, c\\
\sum_{i=0}^{t} x_{a,b,i} &= r_{a,b} \ \forall a, b
\end{aligned}$$

D.h. jeder Lehrer (a) und jede Klasse (b) haben pro Zeitstunde (c) maximal eine Schulstunde und erfüllen dennoch in der Summe ihr Deputat ($r_{a,b}$).

Nebenbedingungen (z.B. [Cos94]):

- Unverfügbarkeiten von Lehrern und Klassen
- Minimierung der Springstundenzahl
- keine Klasse darf dasselbe Fach zweimal am Tag haben (mit Unterbrechung)
- keine Klasse darf ein Fach am selben Tag länger als drei Stunden haben
- die einzelnen Stunden eines Faches müssen über die Woche verteilt sein
- die Fächer müssen gleichmäßig über die Woche verteilt liegen.
- Gleichzeitigkeitsanforderungen
- optimale Fachraumausnutzung
- Lehrer mit halber Stelle haben mindestens zwei Tage frei
- örtliche Gegebenheiten: Der Übergang von einem Raum zum anderen muß z.B. innerhalb einer Pause möglich sein
- ungünstige Fachabfolgen: Z.B. drei Fremdsprachenkurse hintereinander sollten vermieden werden
- Klassenlehrer sollten jeden Tag eine Stunde in ihrer Klasse unterrichten
- konkrete Frei- und Vorbereitungszeitsanforderungen der Lehrer

2.3 Daily School Timetabling

Informelle Beschreibung:
Hier wird darauf verzichtet, die einzelnen Termine explizit festzulegen. Es wird nur bestimmt, an welchem Tag die jeweilige Veranstaltung stattfinden soll. Als zusätzliche Eingabe ist bekannt, wieviele Stunden jeder Lehrer bzw. jede Klasse an jedem Tag zur Verfügung steht. Ausgehend von diesen Tagesplänen kann dann wiederum versucht werden, einen Stundenplan zu erstellen, wobei die Gleichmäßigkeit der Belastung erhalten bleibt.

Darstellung als ILP:
Gegeben $l, k, t \in \mathbb{R}$,$r_{a,b} \in \mathbb{N}^{l \times k}$ sowie $T \in \mathbb{N}^l$ und $C \in \mathbb{N}^k$ finde $x_{a,b,c} \in \mathbb{N}^{l \times k \times t}$ mit

$$\begin{aligned}
\sum_{i=0}^{k} x_{a,i,c} &\leq T_a \ \forall a, c \\
\sum_{i=0}^{l} x_{i,b,c} &\leq C_b \ \forall b, c \\
\sum_{i=0}^{t} x_{a,b,i} &= r_{a,b} \ \forall a, b
\end{aligned}$$

Nebenbedingungen
Die klassische Nebenbedingung dieses Gebietes ist die, daß alle Lehrer bzw. Klassen an allen Tagen ungefähr gleich stark beansprucht werden, sofern diese Bedingung stärker ist als die Wahl der C_b, T_a. Des weiteren sollen auch die Lehrer-Klassen-Kombinationen in der Woche gleichmäßig verteilt sein [dW85].

2.4 Universitäre Veranstaltungsplanung (University-Timetabling/„UTP")

Informelle Beschreibung:
Als „University Timetabling" bezeichnet man ein Problem, das auch der automatischen Stundenplanerstellung in der gymnasialen Oberstufe entspricht. Gegeben eine Menge von Kursen K und eine Menge von Curricula $C \subseteq \mathbb{P}(K)$, finde ein „Scheduling" für K, so daß keine zwei Kurse eines Curriculums zur selben Zeit stattfinden. Ersetzt man „Curriculum" durch Ressource und „Kurs" durch Schulstunde, so erhält man eine realistischere Modellierung des School-Timetabling Problems, da somit Räume als Ressource zulässig sind. Auch wenn es so erscheinen mag, ist dies keine Verzerrung der Interpretation, da unter einem Curriculum stets eine Gruppe von Studenten (Ressourcen) zu verstehen ist, die diesem folgt.

Darstellung als ILP:
Gegeben: Kurse und Curricula wie oben, sowie eine feste Zahl von Zeitstunden T. Gesucht $x_{ij} \in \{0,1\}^{t \times |K|}$ mit

$$\begin{aligned}
\sum_{i=1}^{t} x_{ij} &= 1 \ \forall j \in K \\
\sum_{l \in A} x_{il} &\leq 1 \ \forall i \in T, A \in C
\end{aligned}$$

Nebenbedingungen:

- Die Raumgröße muß der Größe des Auditoriums angemessen sein
- Unverfügbarkeiten von Dozenten und Räumen

- Keine Überlappungen mit Veranstaltungen desselben Kurses
- keine Überlastung der Studenten eines Kurses
- Maximierung der Kombinationsmöglichkeiten
- Einschränkungen durch die Lage der Universitätsgebäude

2.5 Das Kurswahlproblem

Informelle Beschreibung:
Im Bereich des Kursunterrichts (z.B. gymnasiale Oberstufe) ist es zumeist so, daß die Schüler sich zunächst ihr individuelles Programm auswählen können, danach versucht wird, einen Stundenplan zu erzeugen, der diesem möglichst entspricht und dann Schüler, in deren Stundenplan Konflikte auftreten, umwählen müssen. Ziel ist hier dann auch nicht, möglichst viele Stunden zu verteilen, sondern möglichst wenige Konflikte auszulösen (Allerdings besteht hier häufig ein linearer Zusammenhang; vgl. Abschnitt 4.10.6.). Zumeist werden hier a priori ganze Kurse parallel gelegt, wobei allerdings fraglich ist, ob dies optimal sein muß.

Darstellung als ILP:
Es tritt wiederum eine Variante des „UTP" auf, diesmal allerdings nicht als reines Suchproblem, sondern als Optimierungsaufgabe. Hier ist allerdings zu beachten, daß ein ausfallender Kurs ebenfalls als „Kollision" gewertet werden muß. Hierbei seien I die Schüler und F die Fächer. r_{if} sei eine 0, 1-Matrix, die aussagt, ob Schüler i an Kurs f teilnimmt und w_f sei die Zahl der Stunden von Kurs f. Die x geben die Stunden der Schüler und der Vektor der y die Zeitpunkte der Kurse an.

$$\begin{aligned}
\max \sum_{I,F,T} & x_{ift} \\
x_{ift} &\leq y_{f,t} \ \forall i \in I, f \in F, t \in T \\
\sum_{T} y_{f,t} &\leq w_f \ \forall f \in F \\
\sum_{f \in F} x_{ift} &\leq 1 \forall i \in I, t \in T \\
x_{ift} &\leq r_{if} \forall i \in I, t \in T, f \in F
\end{aligned}$$

Nebenbedingungen:
Dieses Problem erscheint für theoretische Ansätze besonders interessant, da gar keine Nebenbedingungen auftreten. Die Minimierung der Umwahlzahlen hat Vorrang vor allen anderen Überlegungen und eine Optimierung der Verteilung wird zumeist gemäß der Problemstellung in Abschnitt 3.6.3 vorgenommen.

2.6 Prüfungsverteilung (Examination-Timetabling)

Informelle Beschreibung:
Der wesentliche Unterschied zum herkömmlichen „Timetabling" besteht darin, daß die Zahl der Zeitstunden nicht im voraus bekannt ist, sondern minimiert werden soll. Insofern unterscheidet sich dieses Problem vom Färben eines Graphen nur dadurch, daß sämtliche inklusionsmaximalen Cliquen bekannt sind.

Darstellung als ILP:
Die Darstellung des Grundproblems als ILP ist mit der des UTP identisch; lediglich ist hier unabhängig von den Nebenbedingungen das Ziel vorgegeben, die Zahl der Zeitstunden zu minimieren. Ist ein Zeitrahmen für die Prüfungen vorgegeben, entfällt auch dieser Unterschied.

Nebenbedingungen:

- kein Student hat mehr als zwei Prüfungen in einer Woche
- kein Student hat mehrere Prüfungen an aufeinanderfolgenden (oder identischen) Tagen
- alle Prüfer haben an allen Tagen ungefähr gleichviele Prüfungen

2.7 Raumverteilung (Room Allocation)

Informelle Beschreibung:
Obwohl das Problem der Raumzuteilung ein Teilproblem von STP bzw. UTP ist, konnte es sich als eigenständiges Gebiet etablieren, da insbesondere im universitären Bereich die Veranstaltungszeiten von jedem Dozenten einzeln und häufig ohne Koordination mit seinen Kollegen beschlossen werden, und somit der Verwaltung die Aufgabe zufällt, für die notwendigen Räumlichkeiten zu sorgen. Ebenfalls von Bedeutung ist, daß bereits dieses Teilproblem als in einigen Fälle $\mathcal{NP}$-hart bewiesen wurde [CT92]. Die einfachste Variante ist in $\mathcal{P}$: Gegeben Mengen von Veranstaltungen, die jeweils gleichzeitig stattfinden sollen, finde eine Zuordnung von Veranstaltungen zu Räumen. $\mathcal{NP}$-hart wird das Problem, sobald man verlangt, daß bestimmte Veranstaltungen stets denselben Raum benutzen sollen (s. Kapitel 3).

Darstellung als ILP:
Für den einfachen Fall ergibt sich, da die Zeitstunden getrennt betrachtet werden, eine orthogonale Summe von Matching-Polytopen, die nach von Neumann [Pul95] ganzzahlige Ecken haben. Es genügt, ein einzelnes zu beschreiben, wobei hier bereits angenommen sein soll, daß durch Hinzufügen von Hilfsobjekten die Zahl von Stunden und Räumen gleich ist, so daß wahlweise die i oder j die Räume bzw. Stunden repräsentieren sollen und die $c_{i,j}$ angeben, wie sehr ein bestimmter Raum für eine bestimmte Kursstunde geeignet ist.

$$\max \sum_{i,j} c_{i,j} x_{i,j}$$

$$\begin{aligned} \sum_i x_{i,j} &= 1 \ \forall j \\ \sum_j x_{i,j} &= 1 \ \forall i \end{aligned}$$

Nebenbedingungen:

- bestimmte Veranstaltungen sollen stets im selben Raum stattfinden

- Distanzprobleme: Es muß jedem Teilnehmer möglich sein, seinen jeweils nächsten Raum in der Pause zu erreichen (relative Nebenbedingung!)
- eine fortlaufende Veranstaltung sollte die Räume zwischenzeitlich nicht wechseln müssen.

2.8 Nachbargebiete

„Employee Timetabling“ Dieses Problem unterscheidet sich von der automatischen Stundenplanerstellung nur dadurch, daß es noch wesentlich kompliziertere Nebenbedingungen gibt, die zumeist durch gewerkschaftliche Regelungen entstanden sind. Die bekanntesten Beispiele sind Personalplanungen bei Bus und Bahn, aber auch die Wachpläne in Krankenhäusern[MK98] und die Verteilung von Gebetszeiten[CO97] zählen hierzu.

Sportveranstaltungen Dieses Problem ist dem „Examination-Timetabling“ sehr ähnlich, zeichnet sich aber in der Praxis durch eine große Zahl von Symmetrien aus.

„Flight Scheduling“ Das Flight Scheduling Problem besteht in der Zuordnung von Flugzeugen zu Flugstrecken und Zeiten. Die Bedingungen unterscheiden sich wesentlich von denen des Timetablings, da z.B. die Geographie eine wesentlich größere Rolle spielt. Als zusätzliche Einschränkungen treten verschiedene Flugzeugtypen, Wartungsintervalle etc. auf. Ziel ist es, die Zahl der Leerflüge zu minimieren.

„Crew Roostering“ Das „Crew Pairing“ ist der zweite Schritt bei der Flugplanung. Jeder Mannschaft wird eine Flugstrecke zugeteilt, die bei demselben Flughafen (einer „Crew-Base“) startet und endet, d.h. der Graph der Flugbewegungen muß in disjunkte Kreise zerlegt werden. Soweit es aus der betrachteten umfangreichen Literatur ersichtlich ist, versucht niemand diese beiden Probleme simultan zu lösen.

„Processor Scheduling“ Dies ist normalerweise ein dynamischer und preemptiver Fall, der von der „Timetabling“-Thematik weitgehend abweicht und häufig mit „Online“- Algorithmen behandelt wird. Stellt man aber Echtzeitanforderungen und muß entschieden werden, ob ein neuer Task noch angenommen werden kann, so ist das Problem bei Annahme eines minimalen Quotas für die Prozessorzuteilung mit dem „Timetabling“ im wesentlichen identisch. Im Bereich des Compilerbaus kann auch die statische Variante auftreten, wenn z.B. mehrere unabhängige Ausführungseinheiten in einem Computer gegeben sind, die von dem zu erstellenden Programm exklusiv genutzt werden können. Dies ähnelt, je nach Parallelisierbarkeit des Codes, aber eher dem „job-shop“ Problem. Etwas ähnliches tritt auch bei der Registerverteilung auf; die hier verwendeten Graphen sind aber, sofern es keine verschiedenen Registertypen gibt, perfekt (s.u.).

2.9 Transformationen

Da die Probleme recht ähnlich sind, können diese jeweils durch die Wahl geeigneter Nebenbedingungen ineinander überführt werden. Die konkrete Form der Überführung wäre mit Mitteln der Graphentheorie leichter zu beschreiben; es wurde aber darauf verzichtet, damit keine Abhängigkeit von einer speziellen Form der Modellierung entsteht:

Parallel abzuhaltende Stunden im STP und das UTP:
Das UTP kann leicht als STP mit der zusätzlichen Nebenbedingung formuliert werden, daß bestimmte Stunden gleichzeitig stattfinden müssen. Hierzu erzeugt man für jedes Curriculum eine Klasse und soviele Lehrer, wie dieses Kurse hat. Für jeden Kurs in jedem Curriculum fügt man eine Stunde dieser Klasse mit einem der Lehrer hinzu und verlangt, daß alle zu einem Kurs gehörigen Stunden gleichzeitig stattfinden müssen. Offenbar ergibt somit ein Stundenplan für dieses spezielle STP eine Lösung des zugehörigen UTPs.

Festlegungen und Abwesenheitszeiten:
Sind einzelne Termine vor Begin der Verteilung festgelegt worden, so kann man dies, sofern nicht zusätzlich die Lehrerverteilung erstellt werden soll, mit Hilfe von Abwesenheitszeiten modellieren, indem man die betroffenen Objekte als für diese Zeit abwesend deklariert und die entsprechenden Stunden nach der übrigen Erstellung einfügt.

Nicht-Gleichzeitigkeits-Bedingungen beim STP und das UTP:
Eine Differenzierung (z.B. bestehend aus zwei Lehrern und einer Klasse) könnte man auch auf dem Wege modellieren, daß man eine normale Stunde für einen Lehrer und die Klasse einplant und von dieser verlangt, daß sie zu einem anderen Zeitpunkt stattfindet als sämtliche Stunden des anderen Lehrers. Nimmt man aber eine Instanz des UTP und wählt ein STP, das für jeden Kurs des UTP einen Lehrer, eine Klasse und eine von diesen abzuhaltende Schulstunde enthält, und verlangt, daß zwei dieser Stunden genau dann nicht gleichzeitig abzuhalten sind, wenn die entsprechenden Kurse im selben Curriculum auftreten, so bildet eine Lösung dieses STP offenbar eine des UTPs.

2.10 Zusammenfassung

Diese Übersicht folgt im wesentlichen den Artikeln [Sch95] und [dW85]. Andere Übersichten zum Thema (z.B. [SS79], [Jun86]) wurden von diesen beiden bereits berücksichtigt.
Trotz der Vielfalt der Anwendungen sind nahezu alle Probleme, sofern man sie mit halbwegs realistischen Nebenbedingungen sieht, zum „University Timetabling“ mit einer geeigneten Zielfunktion äquivalent. Da selten alle Nebenbedingungen der dargestellten Probleme erfüllt werden können, werden diese zumeist in die Zielfunktion der entsprechenden Modelle integriert. Primäres Ziel dabei ist, seltener in der Theorie aber in der Praxis, die Gerechtigkeit in bezug auf die beteiligten Personen. Die Optimierung des Gesamtplans ist demgegenüber untergeordnet. Diese in konkrete Bedingungen zu fassen ist schwer, meistens unmöglich, da sie dem Benutzer der entsprechenden Programme nicht intellektuell bewußt sind.[3]

[3] Tatsächlich besteht nach eigener Erfahrung ein großer Unterschied darin, ob er die betroffenen Personen kennt, oder mit einem anonymisierten Plan arbeitet.

Kapitel 3

Komplexität des Problems

3.1 Terminologie

Da überwiegend Reduktionen auf graphentheoretische Probleme vorgenommen werden, bedient sich dieser Abschnitt in verstärktem Maße dem zugehörigen Begriffsapparat. Die hier definierten Ausdrücke werden auch in Kapitel 4 und in Abschnitt 6 verwandt, weshalb die Beschreibung etwas umfangreicher ausfällt, als eigentlich notwendig. Die hier eingeführte Terminologie folgt im wesentlichen der von DIESTEL [Die96], wobei im folgenden kurz auf einige weniger geläufige Begriffe eingegangen wird.

Linegraph: Zu einem Graphen (Hypergraphen) G ist $L(G) = (E, K)$ mit $E = E(V)$ und $K = \{(e, e') \in E \times E | e \cap e' \neq \emptyset\}$; d.h. die Ecken von $L(G)$ repräsentieren die Kanten von G, und zwei Ecken von $L(G)$ sind genau dann verbunden, wenn die entsprechenden Kanten einen Knoten gemeinsam haben.

Cliquengraph: Gegeben ein Graph (Hypergraph) G repräsentiert der Cliquengraph $CL(G)$ jede Clique von G durch einen Knoten, und zwei „Cliquen“ sind genau dann miteinander verbunden, wenn sie einen Knoten in G gemeinsam haben.

Listenfärbung: Zu jedem zu färbenden Objekt wird eine Liste von k Farben assoziiert, die für die Färbung verwendet werden dürfen. Die minimale Listengröße[1], die unabhängig von der Wahl der Farben in den Listen eine Färbung zuläßt, wird als Listenfärbbarkeitszahl bezeichnet. Diese ist größer-gleich der Färbbarkeitszahl und im Fall des $K_{3,3}$, der 2-färbbar aber nicht 2-listenfärbbar ist, erkennt man, daß Ungleichheit vorliegen kann.

Zudem seien die folgenden Symbole vereinbart:

$\chi(G)$	Knotenfärbbarkeitszahl
$\chi'(G)$	Kantenfärbbarkeitszahl
$\mathrm{ch}(G)$	Knoten-Listenfärbbarkeitszahl
$\mathrm{ch}'(G)$	Kanten-Listenfärbbarkeitszahl
$\Delta(G)$	max. Knotengrad
$\omega(G)$	max. Größe einer Clique

Definition 3.1 (Approximationsalgorithmus) *[Vyg] Ein asymptotischer k-Faktor Approximationsalgorithmus ist ein polynomieller Algorithmus A, so daß es eine Kon-*

[1] wobei alle Knoten Listen der gleichen Größe erhalten

stante c gibt, mit der auf allen Instanzen I der zugehörigen Problemklasse

$$\frac{1}{k} OPT(I) - c \leq A(I) \leq k OPT(I) + c$$

gilt.

Die zusätzliche konstante Abweichung bewahrt an dieser Stelle u.a. vor Problemen mit $OPT(I) = 0$.

Definition 3.2 (Approximationsschema) *[AKK98] Ein polynomielles Approximationsschema ist ein in der Größe der Eingabe polynomieller Algorithmus, der für jedes $\epsilon > 0$ eine Approximationsgüte von mindestens $1 + \epsilon$ erreicht.*

3.2 Polynomiell lösbare Fälle

Wie in Satz 4.3.3 gezeigt wird, liegt das STP ohne Nebenbedingungen in $\mathcal{P}$. Gemäß Kapitel 4.3.5 können zudem einige Gleichgewichtsbedingungen berücksichtigt werden. Aus der engen Verbindung zur Graphentheorie ergeben sich zudem einige Fälle, die gut lösbar sind, da es für sie Darstellungen als LPs gibt, für die die zugehörigen Polyeder ganzzahlige Ecken haben. Diese sind jedoch mit Ausnahme der ersten drei Fälle so eingeschränkt, daß eine praktische Anwendung nicht abzusehen ist. Daher soll über diese, die zumeist von DEWERRA ([dW97] [dW96]) gefunden wurden, lediglich eine tabellarische Übersicht gegeben werden. Auf die ersten drei Fälle wird in Abschnitt 4.10 nochmals näher eingegangen

Bezeichnung	**Beschreibung**
Perfekte Graphen	für jeden ind. Teilgraph gilt $\chi(H) = \omega(H)$
Total unimodulare Graphen	Jede Submatrix der Adjazenzmatrix hat Determinante $0, 1, -1$
Balancierte Graphen	Linegraphen von Bäumen
PROS-perfekte Graphen	Graphen, die unabhängig von gewissen Parametern ein halbpreemptives JSS erlauben
Candies, Crowns, Caterpillars	Spezialfälle von PROS-Perfekten Graphen
Ice-perfekte Graphen	Graphen, die eine Färbung erlauben, bei der die einzelnen Farben, die an einem Knoten anliegen, aufeinanderfolgen
BOC-Graphen	bipartiter Graph, der keine drei geraden Ketten enthält, die sich in ihren Endpunkten treffen

3.3 Realistische Problemstellungen sind $\mathcal{NP}$-vollständig

In [EIS76] wurde erstmals ein recht allgemeines Timetabling-Problem als $\mathcal{NP}$-vollständig nachgewiesen. Dennoch ist dieser Beweis nicht vollständig zufriedenstellend, da das Konstrukt ein ausgesprochen unrealistisches Stundenplanproblem verwendet. Zahlreiche Reduktionen $\mathcal{NP}$-vollständiger Probleme für verschiedene (einfache) Erweiterungen des STP finden sich in [CK95a] und [dWMP93], aber auch hier werden überwiegend unrealistische Fälle verwendet. Da die verwendeten Methoden - auch wegen der Art der Konstrukte - nicht instruktiv bezüglich eines möglichen Lösungsverfahrens erscheinen, wird an dieser Stelle auf eine umfassende Darstellung verzichtet.

Die $\mathcal{NP}$-Vollständigkeit des in dieser Arbeit untersuchten Problems geht zudem auch aus der im nächsten Kapitel beschriebenen Reduktion des Graphenfärbungsproblems hervor.

3.4 Approximierbarkeit

3.4.1 Fälle mit unbeschränktem Grad

Minimierung der Zeitstundenzahl
Bei zahlreichen $\mathcal{NP}$-vollständigen Problemen kann sowohl die Approximation bis auf konstante Abweichung, als auch bis auf einen gewissen konstanten Faktor als ebenfalls $\mathcal{NP}$-vollständig nachgewiesen werden. Es zeigt sich, daß bei uneingeschränkten Formen von Prüfungsverteilungen und beim „University-Timetabling“ noch wesentlich schlechtere Verhältnisse vorliegen, da das Problem des Graphenfärbens darauf L-reduziert werden kann.

Definition 3.3 (L-Reduktion) *[Vyg] Gegeben zwei Kombinatorische Optimierungsprobleme A,B, ist eine L-Reduktion ein Paar von in polynomieller Zeit berechenbare Funktionen f, g und eine Konstante γ, von denen f jede Instanz I von A auf eine von B abbildet und g jede Lösung einer Instanz aus dem Bild von f mit Kosten c' in eine Lösung des Ausgangsproblems mit Kosten c zurückwandelt, wobei*

$$\frac{|OPT(I) - c|}{OPT(I)} \leq \gamma \frac{|OPT(f(I)) - c'|}{OPT(f(I))}.$$

Das heißt, wenn es einen Approximationsalgorithmus für B gibt, gibt es auch einen für A, wobei dieser höchstens um einen konstanten Faktor schlechter ist. Im folgenden soll gezeigt werden, daß - wenn man keine speziellen Voraussetzungen anwendet - ein approximativer Algorithmus für das „Timetabling“-Problem verwendet werden könnte, um das Färben eines Graphen zu approximieren.

Satz 3.1 (Lund,Yannakakis 1993) *[LY94] Es gibt ein $0 < \epsilon < 1$ so daß, wenn es einen polynomiellen Algorithmus, der jeden Graphen mit $c \leq \chi(G)|V(G)|^\epsilon$ Farben färben kann, gibt, dies $\mathcal{P} = \mathcal{NP}$ impliziert. Dies gilt ebenfalls, wenn $c \leq r\chi(G) + d$ für festes r, d garantiert werden kann.*

Bemerkung 3.1 *Gibt man jedem Knoten eine andere Farbe, so kann jeder Graph G mit $|V(G)| \leq |V(G)|\chi(G)$ Farben gefärbt werden.*

In [Hal93] wird - allerdings basierend auf anderen Voraussetzungen - die Vermutung geäußert, daß keine bessere Approximation als $\mathcal{O}(n/\log^c n)$ für ein festes c möglich ist. Die Reduktionen auf die beiden genannten „Timetabling“ Probleme erweisen sich als triviale Umformungen, die, da jeweils eins zu eins Beziehungen zwischen den zu färbenden Objekten bestehen, offensichtlich L-Reduktionen sind: Für jeden Knoten wird ein Termin vorgesehen und für jede Kante eine Person, die an beiden Terminen teilnimmt.

Es kann natürlich dennoch approximative Algorithmen für bestimmte Spezialfälle geben. Die Anforderungen an einen solchen wären relativ zu der obigen Aussage aber

recht hoch, da z.B. herkömmliche Schul-Stundenpläne bei 40 zur Verfügung stehenden Wochenstunden Klassen bis zu 30 Unterrichtsstunden in der Woche haben können, so daß mit $\chi(G) \geq \Delta(G) = 30$ mindestens ein $\frac{4}{3}$- Approximationsalgorithmus gefunden werden müßte, wobei es unwahrscheinlich ist, daß tatsächlich eine Färbung mit 30 Farben existiert.

Vorgegebene Zeitstundenzahl
Man kann die Anforderung an einen „Timetabling“-Algorithmus also zu präzisieren versuchen und fragen, ob es einen Algorithmus gibt, der, gegeben eine Zahl k von Farben, einen Graphen mit diesen färbt und nur einen konstanten Anteil der Knoten ungefärbt läßt[2]. Aber auch hierfür ergibt sich, daß der allgemeine Fall unlösbar ist:

Lemma 3.1 *[Hoc97] Hat man einen Algorithmus, der, gegeben ein Graph G, mit $\chi(G) = k$ stets $c|V(G)|$ der Knoten mit k Farben korrekt färbt, so kann man diesen Algorithmus verwenden, um den Graphen mit $k \log |V(G)|$ Farben zu färben.*

Beweis:
Man wendet den Algorithmus mit stets neuen k Farben iteriert auf die jeweils noch nicht gefärbten Knoten an. Dies muß man höchstens $\lceil -\log_{1-c} |V(G)| \rceil$ mal anwenden, bis weniger als 1 Knoten übrig ist.

Für jedes ϵ gibt es aber ein n_0, so daß $\chi \log(n) \leq \chi n^\epsilon \; \forall n \geq n_0$ ist wobei anzumerken ist, daß es zu jedem k beliebig große Graphen G mit $\chi(G) = k$ gibt.

Um diesen Fall zu betrachten, kann des weiteren folgende Konstruktion herangezogen werden ([Tuz97],[dW95])[3]: Jeder Knoten des Graphen werde durch eine Clique der Größe k ersetzt, wobei mit jedem Knoten der Clique eine Farbe assoziiert wird. Die neuen Knoten seien jeweils durch das Paar aus alter Knotenbezeichnung und Farbe markiert. Dann verbindet man alle Knoten „gleicher assoziierter Farbe“ miteinander, deren Originale im Ausgangsgraphen verbunden waren. Die Kantenmenge des neuen Graphen ergäbe sich somit also als

$$\{((x, a), (y, b))|x = y \vee (a = b \wedge (x, y) \in E(G))\}$$

Offenbar korrespondiert eine unabhängige Menge im neuen Graphen zu einer korrekten, aber nicht notwendig vollständigen Färbung des alten, wenn man ein Element dieser Menge so interpretiert, daß der Originalknoten eines ausgewählten Knotens die diesem assoziierte Farbe erhält. Die Wohldefiniertheit wird hierbei durch die Cliquen und die Korrektheit durch die übrigen Kanten gewährleistet.
Diese Konstruktion erlaubt es zudem, einige relative Nebenbedingungen zu modellieren, da man nun dafür sorgen kann, daß Zuordnungen einander ausschließen. So z.B. kann man jede Kombination (s, x), bei der s eine Schwimmstunde für die entsprechende Klasse darstellt, mit allen Stunden (p, y) verbinden, bei denen p eine Mathematikstunde derselben Klasse und y eine Stunde ist, die am selben Tag aber später als x liegt. Damit wäre dann sichergestellt, daß niemals Mathematik auf Schwimmen folgen kann.
Umgekehrt kann man natürlich das Problem der maximalen unabhängigen Teilmenge (MIS) als das Problem ansehen, einen Graphen mit $k = 1$ Farbe so zu färben, daß möglichst wenige Knoten übrigbleiben. [Hoc97] folgend ist dieses Problem im nicht gradbeschränkten Fall konsistent damit genauso „hart“ wie das des Graphenfärbens,

[2] Die Möglichkeit der Garantie einer konstanten Zahl kann man ausschließen, indem man den Graphen mehrfach reproduziert.

[3] Eine analoge Konstruktion läßt sich im Bereich der Listenfärbbarkeit einsetzen.

da auch für das MIS eine untere Approximierbarkeitsschranke des selben Typs angegeben werden kann [Has96].
Ein Vorteil, den man gegenüber dem allgemeinen Graphenfärbungsproblem hat, ist, daß, da der Graph aus einem anderen Problem konstruiert wurde, die Cliquenzahl bekannt ist, deren Ermittlung normalerweise ebenfalls hart ist[4]. Hieraus läßt sich aber für einen exakten Algorithmus anscheinend kein Vorteil gewinnen, da, obwohl z.B. in einem planaren Graphen alle Cliquen in $\mathcal{O}(n^4)$ aufgezählt werden können, die Entscheidung ob dieser 3- oder 4-färbbar ist, $\mathcal{NP}$-vollständig ist.[5] Ob dies die Approximation erleichtert, scheint bisher noch nicht untersucht worden zu sein. Zudem ist mit der Lovaszschen ϑ-Funktion [Knu94] stets ein Wert bekannt, der zwischen chromatischer Zahl und Cliquenzahl liegt, so daß somit eine bessere untere Schranke für χ immer gegeben ist.

Insgesamt läßt sich also sagen, daß das Problem, wenn man keine Anforderungen an die Größe der Differenzierungen stellt, nicht mit konstanter Qualität approximierbar ist.

3.4.2 Beschränkter Grad

Minimierung der Zeitstundenzahl
Für den Fall, daß man die Größe der Differenzierungsgruppen als beschränkt annimmt, funktioniert die obige Reduktion des Knotenfärbens nicht mehr, da das „Timetabling“ dann zu Fällen mit begrenztem Knotengrad korrespondiert. Die umgekehrte Transformation ist natürlich auch möglich, was dazu führt, daß der Satz von Brooks, der $\chi = \Delta$ zeigt[6] [Die96], anwendbar wird. Dies liefert aber dennoch keinen Algorithmus für die automatische Stundenplanerstellung, da der Knotengrad hier im Schnitt ca. dreimal größer ist, als die angestrebte Farbenzahl. Selbst dann wäre er nur um 1 besser, als die von jedem Greedy-Algorithmus garantierbare Schranke von $\Delta + 1$.
Relaxiert man die Ansprüche weiter und verlangt, daß nur ein bestimmter Anteil der Kanten beide Enden in der jeweils gleichen Farbklasse hat, so gelangt man zum Begriff der sogenannten „Semifärbungen“, wobei bei diesen ein Fehleranteil von maximal $\frac{1}{4}$ vorgesehen ist. Hier läßt sich mit hoher Wahrscheinlichkeit und unter Einsatz semidefiniter Programmierung eine Semi-Färbung mit $\mathcal{O}(\Delta^{1-2/k})$ Farben erzielen [Hoc97], was selbst, wenn man als Konstante nur 1 zuläßt, mit $\Delta = 60$ und $k = 30$ zu einer Farbenzahl von 52 führen würde, wobei wegen der relativ großen Zahl von Kanten noch immer jeder Knoten mit einer ungültigen Kante inzident sein könnte. Nach dem Schubfachprinzip müßte, um einen Anteil von p gesetzten Stunden zu garantieren, die Approximationsgüte bezüglich der Kanten $\frac{p|V|}{2|E|} \geq \frac{p|V|}{|V|\Delta} = \frac{p}{\Delta}$ betragen.

Vorgegebene Zeitstundenzahl
Wählt man die selbe Reduktion auf das „MIS“-Problem wie im letzten Abschnitt[7], so liegt auch hier der Fall mit beschränktem Grad vor, da die Größe jeder Clique durch k und die Zahl der übrigen Kanten durch den (beschränkten) Grad des Originalknotens gegeben ist. Für Graphen mit $\Delta \leq 5$ zeigen Berman und Karpinski [BK98], daß sich eine Approximationsgüte von $\frac{68}{67}$ nicht unterschreiten läßt. Dies wäre zwar noch hinreichend, aber die erreichbare Güte scheint mit wachsendem Grad strikt größer zu

[4] MIS im komplementären Graphen
[5] Für dieses Beispiel danke ich Dr. Vygen vom Institut für Diskrete Mathematik.
[6] Sofern der Graph nicht vollständig oder ein Kreis ungerader Länge ist.
[7] ohne Hinzunahme relativer Bedingungen

werden.[8]
Betrachtet man die entsprechenden Approximationsversuche für das Problem der maximalen unabhängigen Knotenmenge in gradbeschränkten Graphen, so findet sich beispielsweise in [BF96] eine Approximationsgüte in der Größenordnung von $\frac{\Delta}{5}$, was in diesem Fall bedeuten würde, daß nur ca. $\frac{5}{60} = 1/12$ der Knoten gefärbt würde ,wobei die hier eingesetzten Zahlwerte, die voraussetzen, daß es nur Differenzierungen der Größe 3 mit je 20 Stunden pro Objekt gibt, meiner Ansicht nach mehr als optimistisch sind. Durch die umgekehrte Reduktion und durch die Anwendung des Satzes von Brooks erhielte man $\frac{1}{\Delta}$. Unter der Voraussetzung, daß für den zu färbenden Graphen $\omega = \chi = 30$ gilt, erhielte man durch iteriertes Färben mit χ Kanten also $30 \log_{11/12} n$, was nur für $n \leq 2$ zu weniger als 38 Farben führt.
Das Problem mit beschränktem Grad ist somit zwar konstant approximierbar, aber die Qualität ist für die praktische Anwendung nicht ausreichend. Polynomielle Approximationsschemata existieren für diesen Fall - zumindest in dieser Allgemeinheit - ebenfalls nicht, was diese Variante des Problems in die Komplexitätsklasse „APX-PB“, die laut [KMSV94] mit dem Abschluß der formalen Klasse „MAX SNP“ übereinstimmt, einordnet.

3.4.3 Gewichtete Fälle

Da man stets eine Gewichtung mit den Werten $\{0, 1\}$ wählen kann, sind die gewichteten Problemstellungen mindestens so schwer wie die ungewichteten. Über die exakte Komplexität und Approximierbarkeit des allgemeinen Falles gibt die Literatur soweit mir bekannt keinerlei Auskunft. Das Problem liegt natürlich in $\mathcal{NP}$, da es möglich ist, all diese Kriterien in polynomieller Zeit zu überprüfen. Läßt man die die relativen Nebenbedingungen gänzlich außer Acht, so gelangt man zum „GOCCP“, über dessen allgemeine Approximierbarkeit nichts bekannt ist, das aber auf speziellen Graphenklassen gelöst werden kann (s. Abschnitt 4.10.1).
Verwendet man hier - eine vorgegebene Farbenzahl vorausgesetzt - erneut die obige Konstruktion zum Übergang auf das MIS-Problem an, so gibt es in [Hoc97] ein anwendbaren Approximationsalgorithmus, der eine minimale unabhängige Kantenmenge mit einem Gewicht findet, das vom optimalen nur um $\frac{2}{\Delta+1}$ abweicht. Dieses Verfahren besteht im wesentlichen in der Ausnutzung einer vorangegangenen Zerlegung des Graphen in Farbklassen[9]. Könnte das obige Suchproblem besser gelöst werden und eine Färbung mit l Farben gefunden werden, so kann dies sogar auf $\frac{2}{l}$ verbessert werden, da man aus einer Färbung des Ausgangsgraphen stets eine Färbung des neuen (MIS-) Graphen erhalten kann, indem man l injektive Abbildungen $\phi_i\{1, \ldots, k\} \to \{1, \ldots, l\}$ mit $\phi_i(a) \neq \phi_j(a)\ \forall i, j, a$ wählt, was z.B. durch zyklisches Permutieren geht. Dann ordnet man dem Knoten (a, b) die Farbe $\phi_a(b)$ zu. In praktischen Anwendungen ist es natürlich dennoch nicht sinnvoll, diese Färbung, die, wenn man die kleinsten $l - k$ der Farbklassen als nichtgefärbt deklariert, immerhin $\frac{k}{l}$ approximativ war, zugunsten einer gewichteten $\frac{2}{l}$-Approximation „aufzugeben“.

[8] Zumindest kann sie nicht fallen, da die Problemmenge stets erweitert wird.
[9] Was mittels eines „Greedy“-Algorithmus stets mit $\Delta + 1$ Farben möglich ist.

3.5 Randomisierte Algorithmen/Heuristiken

Auch die besten randomisierten Algorithmen können weder in der Zahl der Zeitstunden, noch der nicht gesetzten Schulstunden eine Abweichung um einen konstanten Faktor versprechen (s. [Hoc97]). Die Ergebnisse, die hier im Bereich des „Schedulings“ erzielt wurden, liefern Grenzen, die im Bereich der doppelten Zahl der tatsächlich zur Verfügung stehenden Zeitstunden liegen, da sie im übertragenen Sinne zumeist auch Schulstunden beliebiger Länge verwalten könnten. Zudem zeigen die obigen Ergebnisse, daß eventuell existierende Algorithmen nicht derandomisiert werden können, da die Nichtexistenz[10] des somit zu erreichenden deterministischen Algorithmus dies ausschließt. Da aber für die meisten für dieses Gebiet vorliegenden Verfahren Derandomisierungsschemata gegeben sind, die die Laufzeit deutlich aber nur polynomiell verschlechtern, würde ihre Anwendbarkeit sofort $\mathcal{P} = \mathcal{NP}$ implizieren. Die beweisbare Performanz (nichtderandomisierbarer) randomisierter „lokaler“ Suchverfahren, wie z.B. Simulated Annealing“ (vgl. Kapitel 4.7), liegt zumeist in der Größe des Lösungsraums ([Hoc97],[ECF97]), was sie vom theoretischen Standpunkt her einem einfachen Enumerieren aller Varianten äquivalent werden läßt, wobei die Bedeutung der Tatsache, daß sie tatsächlich eine Lösung finden können [Aar89], nicht herabgesetzt werden soll. Dennoch ist es so, daß diese Verfahren, auch wenn keine Schranken gezeigt werden können und nur - überwiegend nicht vergleichbare - numerische Ergebnisse vorliegen, auf den Problemfällen der Realität bessere Lösungen finden, als die oben angesprochenen Verfahren, da sie sämtliche Nebenbedingungen berücksichtigen können, während das „GOCCP“ als allgemeinstes theoretisches Modell weder relative Nebenbedingungen noch einen Belastungsausgleich über nichtlineare Zielfunktionen unterstützt.

3.6 Die Komplexität von Teilproblemen

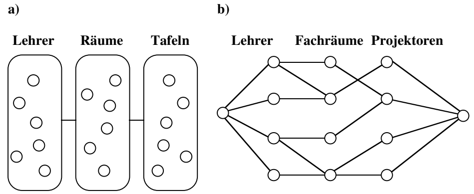
Insbesondere im Rahmen eines interaktiven Systems ist die Komplexität von Teilproblemen interessant, da seitens des Benutzers häufig verlangt wird, lokale Widersprüche aufzuheben. Im folgenden wird daher immer ein Teil des Problems bzw. ein bestimmter Parameter als konstant angenommen.

3.6.1 Festhalten des Zeitpunkts

Geht es lediglich darum, die zu einem bestimmten Zeitpunkt angeforderten Ressourcen zu verteilen, so kann dies durch einfaches Matching geschehen: Siehe Abschnitt 8.4.3. Abhängigkeiten zwischen flexiblen Ressourcen - wie z.B. Bevorzugung von benachbarten Klassenräumen - können aber bereits nicht mehr in polynomieller Zeit aufgelöst werden. Es gibt auch keinen approximativen Algorithmus, da dieses Problem durch Komplementbildung in das der maximalen unabhängigen Teilmenge überführt werden kann. Sei ein Graph gegeben, in dem eine Clique gegebener Größe u zu finden ist: Assoziiert man jeden Knoten mit einer flexiblen Ressource und legt die Vorlieben so fest, daß zwei Ressourcen einander bevorzugen wenn die entsprechenden Kanten verbunden sind, so korrespondiert offenbar eine entsprechende Clique im Graphen zu einer optimalen Ressourcenzuteilung von u Objekten.
Kann man die Objekte aber nach Wichtigkeit der Wünsche so aufteilen, daß die jewei-

[10] $\mathcal{P} \neq \mathcal{NP}$ vorausgesetzt

ligen Gruppen innerhalb keine Abhängigkeiten aufweisen, so besteht auf jeder Ebene nur das Problem, die u bevorzugtesten Elemente auszuwählen (a). Sind diese zusätzlich nur von einigen Elementen der höheren Ebene abhängig, läßt sich das Ganze durch Netzwerkflüsse behandeln (b)[11]. Das oben angesprochene Matching wird dann ein Spezialfall dieses Verfahrens mit zwei Ebenen:

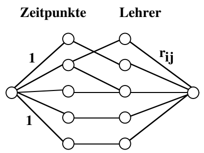


Allerdings zeigt dieses Teilproblem ein Verhalten, das dem von „2SAT“ bzw. von „Max2SAT“ ähnelt: Ist eine Menge von Stunden gegeben, der Ressourcen zugeteilt werden sollen, bei der aber nicht gesichert ist, daß wirklich jede Stunde mit Räumen o.ä. versorgt werden kann, so ist das Problem, eine optimale Lösung zu finden, $\mathcal{NP}$-hart und nicht konstant approximierbar. Denn: Gegeben ein Graph, in dem eine maximale unabhängige Menge gefunden werden soll, assoziiere jeden Knoten mit einer Stunde, füge für jede Kante einen gesonderten Raum hinzu, und stelle die Anforderung, daß die beiden inzidenten „Stunden“ genau diesen Raum brauchen. Offenbar korrespondiert dann eine maximale Zahl verteilter Stunden zu einer maximalen unabhängigen Knotenmenge.
Während das Entscheidungsproblem also in $\mathcal{P}$ liegt, ist das Optimierungsproblem sehr schwer.[12]

Eine ebenfalls $\mathcal{NP}$-harte Variante ergibt sich, wenn man verlangt, daß zu allen Zeiten die gleichen Ressourcen, z.B. Räume, verwandt werden sollen. Hier erhält man wieder ein allgemeines Graphenfärbungsproblem, indem man die Räume als Farben und die Veranstaltungen als Knoten interpretiert, die genau dann verbunden sind, wenn sich diese zu irgendeinem Zeitpunkt überschneiden.

3.6.2 Festhalten eines Objekts

Wie [dW85] zeigt, kann auch unter der Annahme von Abwesenheitszeiten der Plan für eine einzelne Klasse optimal durch Netzwerkflüsse gelöst werden.



Das gleiche kann natürlich auch für Lehrer, Räume etc. durchgeführt werden.

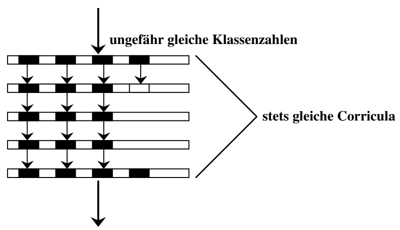
[11] Sind die Graphen zwischen den Ebenen vollständig, geht b) in a) über.
[12] Im Gegensatz zu „Max2SAT“, das konstant approximierbar ist.

3.6.3 Festhalten der „Blockungen“

Hat man die Schulstunden den Zeitstunden zugeordnet, so läßt das Permutieren der Zeitstunden die Gültigkeit der harten Nebenbedingungen unberührt. Betrachtet man hier zunächst nur die indirekten Nebenbedingungen, so erhält man in der einfachsten Form - nur Relationen zwischen aufeinanderfolgenden Stunden - eine Variante des Problems des Handlungsreisenden. Will man z.B. den Wochenplan einer Klasse unter dem Aspekt modifizieren, daß die zurückzulegende Wegstrecke minimiert wird, so erhält man eventuell noch eine euklidische Metrik, so daß Approximationsalgorithmen wie der von Arora [Vyg] anwendbar bleiben. Kommen dazu noch Regelungen wie die, daß auf eine Schwimmstunde z.B. nicht Mathematik folgen sollte, so müssen die „Entfernungen“ in diesem Problem noch nicht einmal mehr der Dreiecksungleichung genügen, was dazu führt, daß noch nicht einmal eine Approximation bis auf einen konstanten Faktor möglich ist [Jun90]. Es können zudem natürlich auch mehrere Stunden übergreifende Relationen auftreten.

3.7 Der „Normalfall“

All diese Überlegungen zeigen, daß es nach dem derzeitigen Stand der Dinge aus informatischer Sicht eigentlich nahezu unmöglich sein müßte, einen geeigneten Stundenplan zu erzeugen. Dennoch kann jede Schule einen solchen vorweisen. Dies legt nahe, daß es in der Realität irgendwelche noch nicht erfaßte Randbedingungen gibt, die das Problem wesentlich vereinfachen. Untersuchungen dieser Art gibt es z.B. für das „Vehicle Routing Problem“[13]. Dort konnte gezeigt werden, daß die Eckpunkte des zugehörigen Polyeders der möglichen Lösungen überdurchschnittlich häufig ganzzahlig sind [RF88]. Bei der automatische Stundenplanerstellung ist auffällig, daß die Aufgabenstellung zumeist aus der nach [CK95b] ebenfalls nur mit exponentiellem Aufwand zu bestimmenden Lehrerverteilung hervorgeht, wenn auch, wie auch von diesen Autoren betont, deren Integration in das Gesamtproblem eleganter wäre.
Hinzu kommt, daß in der schulischen Praxis ein Stundenplan kein isoliertes Problem, sondern eine Problemabfolge ist: Es ist relativ einfach, zu einem gegebenen Stundenplan für die Klassen Lehrer so (neu) einzustellen, daß der Plan erfüllt werden kann: Es genügt bei ähnlichen Curricula, z.B. alle Klassen die Fächer in derselben Reihenfolge durchnehmen zu lassen, diese aber jeweils um 1 gegeneinander zu verschieben. D.h. sozusagen, der erste Stundenplan, den eine Schule benötigt, ist einfach zu erstellen. Beim Übergang von einem Jahr zum nächsten ändern sich strukturell meist nur wenige Dinge, da die Jahrgangsgrößen in Klassenzahlen selten um mehr als ± 1 schwanken und nur ein Jahrgang ausgewechselt wird. Zudem bleiben die Curricula zumeist gleich.



[13] TSP mit mehreren Händlern

Die Frage hier wäre also, ob die Lösbarkeit des letztjährigen Stundenplanproblems hinreichend ist für die Existenz eines neuen Stundenplans. Beim Hinzunehmen der Gymnasialen Oberstufe verschwindet dieser Effekt allerdings teilweise, da hier die individuellen Wahlen der Schüler den Plan in starkem Maße verändern können.
Auch scheint es so, daß die Prüfungsverteilung (vgl. Kapitel 2.6), bei der solche Regelmäßigkeiten noch seltener auftreten, in der Praxis schwieriger ist.

3.8 Zusammenfassung

Ergäbe die obige Reduktion des Knotenfärbens stets „sinnvolle“ Stundenpläne, so wäre der Fall nahezu hoffnungslos. Bei der beschriebenen Konstruktion tritt aber für jede Kante ein neues Objekt auf, d.h. die Zahl der Objekte wäre bis zu „quadratisch“ in der Zahl der Schulstunden. In der Realität ist es aber so, daß diese nur ungefähr ein Zehntel der Schulstundenzahl beträgt. Allerdings ist es bisher nicht gelungen, daraus einen Vorteil zu gewinnen, da die meisten Sätze zu diesen Problemen hier keinen Unterschied vorsehen, so daß eine Anwendbarkeit stets $\mathcal{P} = \mathcal{NP}$ nach sich ziehen würde.
Solange es nicht gelingt, weitere, dem Sachkontext inhärente Nebenbedingungen zu identifizieren, die das Problem einfacher werden lassen, muß das Problem als nicht lösbar, ja nicht einmal mit konstantem Faktor randomisiert approximierbar angesehen werden. Dies äußert sich auch in der großen Zahl „alchemistisch“ anmutender Lösungsversuche, bei denen einige nicht theoretisch, sondern rein experimentell gefundene Parametersätze als Errungenschaften angesehen werden.
Die aufgrund der Verschiedenheit der Problemstellungen fehlende Vergleichbarkeit dieser Ergebnisse laßt auch einen Versuch der Parameterbestimmung im größeren Rahmen nicht zu, vgl. unten und [CK95a], die den besten Einblick in die „Härte“ des Problems bieten. Beiträge über die Approximierbarkeit des „Timetabling“-Problems sind bisher nicht erschienen. Untersuchungen hierzu wurden jedoch bereits in [Sch95] als lohnendes Ziel erwähnt.
Die Lösbarkeit einiger Teilprobleme stellt zumindest die prinzipielle Anwendbarkeit informatischer Prinzipien im Bereich der automatische Stundenplanerstellung sicher.

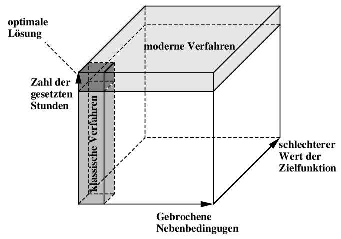
Kapitel 4

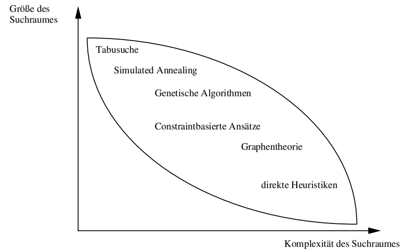
Lösungsansätze

Wie bei den meisten Optimierungsproblemen mit Nebenbedingungen gibt es auch bei der automatischen Stundenplanerstellung die Möglichkeit, die einzelnen Bedingungen nahezu beliebig auf Zielfunktion und Nebenbedingungen zu verteilen. Hierbei schränkt eine Betonung der Nebenbedingungen den Suchraum stark ein und verkompliziert die Operationen, die nötig sind, um von einer Lösung zur anderen zu wechseln, während deren Einbeziehung in die Zielfunktion durch „Barrier“- und „Penalty-Funktionen“ den Suchraum einfacher gestaltet und auch dann eine annähernd gute Lösung liefert, wenn das Suchproblem nur schwer lösbar ist. Im Gegenzug aber kann die Zahl der möglichen Varianten nochmals wesentlich ansteigen.

Letzteres Vorgehen bietet sich bei $\mathcal{NP}$-leichten Problemen natürlich an, da das Überprüfen der Qualität der Lösung entsprechend in $\mathcal{P}$ liegt.[1] Der fortlaufenden Beschleunigung der Rechner folgend zeigt sich daher in jüngerer Zeit die Tendenz, zu größeren Suchräumen und allgemeinen Suchverfahren überzugehen. Die nachfolgende Aufzählung ist mit Ausnahme des letzten Punktes primär chronologisch aufgebaut, was jedoch immanent auch einer Ordnung nach der zunehmenden Größe des Suchraumes entspricht. Im Rahmen der gängigen Terminologie ([CK93]) werden dabei jene Nebenbedingungen, die während des Optimierunsprozesses nicht verletzt werden, als Invarianten bezeichnet. CATONI [Cat98] gibt eine Variante an, wie verschiedene Metaheuristiken im „klassischen“ Suchraum operieren können.

[1]In einigen Randgebieten allerdings sind zur Berechnung der Qualität selbst z.B. Simulationen erforderlich [MR98].





4.1 Direkte Heuristiken

Die direkten Heuristiken sind zumeist unmittelbar aus den Erfahrungen der „menschlichen“ Stundenplaner abgeleitet, wobei jedoch kaum mehr als allgemeine Vorgehensreihenfolgen gewonnen werden konnten. Die Verfahren basieren letztendlich alle auf einem „Greedy“-Verfahren mit „Backtracking“ und wählen lediglich jeweils einen anderen Weg durch den Suchbaum, um einer Lösung möglichst nahe zu kommen, bevor zum erstenmal ein Schritt rückgängig gemacht werden muß.

Das folgende Lemma mag zwar auf den ersten Blick erfreulich erscheinen, aber das Bestimmen der als existent bewiesenen Reihenfolge ist offenbar wieder $\mathcal{NP}$-vollständig:

Lemma 4.1 *Es gibt eine Stundenreihenfolge, mit der als Vorgabe der „Greedy“-Algorithmus nur die minimale Zahl von Timeslots benötigt.*

Beweis:
Es sei ein Stundenplan, der nur die minimal notwendige Stundenzahl benötigt, gegeben. Die gesuchte Reihenfolge ergibt sich, indem man die einzelnen Timeslots in irgendeiner Reihenfolge - und auch intern jeweils beliebig - ausliest. Offenbar kann der „Greedy“-Algorithmus nur Stunden in früher aufgeführte Zeitstunden verschieben, was die weitere Verteilung aber nicht stört.

Daher nun zu heuristisch inspirierten Wahlen von Reihenfolgen:

- Klassenreihenfolge: Dieses wohl einfachste Verfahren besteht darin, jeweils den Stundenplan für eine ganze Klasse festzulegen und dann zur nächsten überzugehen. Tritt eine Überlappung auf, werden entsprechend viele Schritte rückgängig gemacht. Der Vorteil gegenüber einer Lehrerreihenfolge besteht darin, daß Klassenspringstunden unerwünschter sind als Lehrerspringstunden.
- „Scarcest Ressource first“: Hier wird die Reihenfolge durch das Verhältnis von „Angebot“ und „Nachfrage“ nach den jeweiligen Ressourcen geregelt. Stunden, die stark nachgefragte Ressourcen verwenden, werden in der Hoffnung zuerst verplant, daß das Einsetzen der folgenden Stunden trotz der wachsenden Möglichkeiten einer Überschneidung erleichtert wird. Da sich „Ressource“ hier durchaus

auch auf Gruppen und Kombinationen von Ressourcen beziehen kann, ist die Bestimmung der „seltensten“ zumeist nicht trivial, sondern erfordert das Wissen eines erfahrenen menschlichen Stundenplaners. In Programmen wird zumeist so vorgegangen, daß mehrere Planungsversuche unternommen werden und jeweils diejenigen Objekte, die die größten Schwierigkeiten bereiteten, im nächsten Durchlauf als die „seltensten“ angenommen werden.

- Blockungen: Diese heute zumeist nur noch für die Gymnasiale Oberstufe eingesetzte Methode besteht darin, nichkollidierende Veranstaltungen zu einer Gruppe zusammenzufassen und diese dann als eine neue Veranstaltung zu behandeln. Vorteile dieses Vorgehens sind eine Reduktion der zu verplanenden Elemente und ein sehr regelmäßig strukturierter Stundenplan. Es existieren Programme, die ausschließlich zum Erstellen solcher Blöcke dienen, ohne daß konkrete Schulstunden betrachtet werden.
- Teile und Herrsche: Fast zwangsläufig wird häufig versucht, das Problem in mehrere Teilprobleme zu zerlegen. Dies ist insbesondere möglich, wenn man ohnehin einige Objekte „vorbehaltlos“ bevorzugen möchte. Z.B. wird häufig der Plan für die gymnasiale Oberstufe ohne Rücksicht auf den noch zu erstellenden Mittelstufenplan festgelegt.

Die Beschreibung einer solchen Heuristik, wie sie z.B. in dem von der GMD angebotenen System INTEGA verwendet wird, findet sich in [Jun86]. Sämtliche kommerziell angebotenen Stundenplanprogramme beruhen auf mehr oder minder ausgefeilten Variationen der obigen Prinzipien. Diese konnten sich laut Stüber [Stü97] in praktischen Vergleichen stets gegenüber „wissenschaftlich fundierten“ Ansätzen durchsetzen. Ein großer Vorteil besteht des weiteren darin, daß diese Verfahren von den Anwendern verstanden und somit eher akzeptiert werden.

4.2 Wissensbasierte Verfahren / Expertensysteme

In diese Verfahren werden andere Methoden - zumeist verschiedene direkte Heuristiken (vgl. oben) - im Rahmen von Regeln modelliert und je nach Gegebenheit angewandt. Dieser zunächst relativ unspektakulär klingende Ansatz wird dadurch wirksam, daß durch wiederholte Anwendung derselben Verfahren deren Parameter neu bestimmt werden können, der Algorithmus also gewissermaßen „aus Erfahrung“ lernt. Dies wird insbesondere bei der „scarcest Ressource first“ Heuristik erkennbar, bei der der nächste Lauf mit den jeweils beim letzten Durchgang nicht verteilten Stunden begonnen wird [GPU98]. In diesem Sinne stellen diese Verfahren einen indirekten Ansatz dar, der dazu führt, daß sie stets so gut sind wie der beste integrierte Algorithmus, vorausgesetzt, daß dieser auch gewählt wird.

4.3 Graphentheoriebasierte Verfahren

Wie auch die Heuristiken bewahren diese Verfahren die Invariante, daß kein Objekt zur selben Zeit an zwei verschiedenen Orten sein kann und versuchen, die Zahl der gesetzten Stunden zu erhöhen.

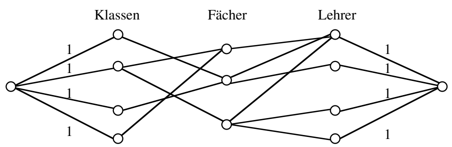
4.3.1 Modellierungsansätze

Aus der Theorie der $\mathcal{NP}$-Vollständigkeit ergibt sich, daß jedes $\mathcal{NP}$-schwere Problem dazu verwendet werden kann, das „Timetabling“ zu modellieren. Daher werden im folgenden ausschließlich Beispiele angegeben, die in tatsächlichen Lösungsversuchen angewandt wurden bzw. bei denen die Modellierung besonders einfach ist.

Knotenfärbungen:
Jeder Termin, der angesetzt wurde, wird als Knoten eines Graphen repräsentiert. Zwei Knoten werden genau dann durch eine (ungerichtete) Kante verbunden, wenn die beiden zugehörigen Termine kollidieren. Offenbar sind dann die Mengen von Terminen, die gleichzeitig stattfinden können, genau die unabhängigen Knotenmengen. Eine Knotenfärbung dieses Graphen ergibt somit eine gültige Lösung, wenn man jeder Farbklasse einem Zeitpunkt zuordnet.

Kantenfärbungen:
Hier werden die einzelnen Ressourcen als Knoten und die Termine als (Hyper-)Kanten modelliert. Zwei Termine kollidieren genau dann, wenn sie einen gemeinsamen Knoten haben. Eine Färbung der Kanten induziert somit wieder einen gültigen Stundenplan. Diese Darstellung und die des Knotenfärbens sind durch Cliquen bzw. Linegraphbildung ineinander überführbar.

Netzwerkflüsse:
Das Problem, einen Fluß in einem Netzwerk[2] zu finden, liegt, wenn keine weiteren Anforderungen gestellt werden, in $\mathcal{P}$. Daher läßt es sich nur für Spezialfälle wie den folgenden anwenden: Chahal und DeWerra [CdW89] beschreiben ein „Timetabling“-Problem, bei dem es egal ist, welcher konkrete Lehrer in einer Klasse ein Fach gibt, solange nur die Belastungszahlen von Klassen und Lehrern relativ zu den jeweiligen Fächern eingehalten wird. Hier kann die Bestimmung eines maximalen Matchings, wie sie in Satz 4.3.3 verwendet wird, als Flußproblem interpretiert und mit einer weiteren Menge von Knoten versehen werden, so daß der Graph dann tripartit wird:



Hierbei repräsentiert die innere Knotenreihe die Fächer und jeder Fluß kann als eine Menge von parallel zu haltenden Stunden interpretiert werden. Die Möglichkeit einer guten Lösung - ohne zusätzliche Bedingungen - ergibt sich aus Kapitel 4.3.3. Zusatzbedingungen - wie z.B. Raumzahlen - lassen sich einbinden, indem man die Fachknoten aufspaltet und zwischen den beiden Duplikaten eine mit der Raumzahl kapazitätsbeschränkte Kante einfügt; hierbei geht die Garantie der Existenz einer Lösung natürlich verloren.

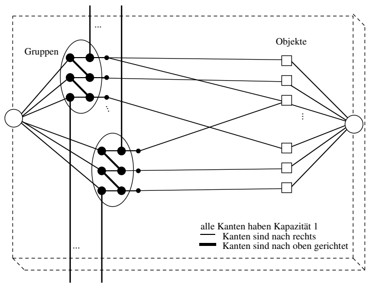
Die ersten beiden Ansätze haben den Nachteil, daß sie keine Modellierung dynamischer Ressourcen erlauben. Eine Konstruktion für einen Spezialfall im Bereich der Knotenfärbung ergibt sich z.B. [JT95] folgend: Will man bei einer k- Färbung erreichen, daß aus einer gegebenen Menge M von weniger als k Knoten ein Knoten eine andere Farbe erhält als alle anderen, kann man einen K_k hinzufügen und jeden Kno-

[2] ggf. mit Kapazitäten, Anforderungen und Kosten

ten aus M mit mindestens einem Knoten davon verbinden, so daß jeder Knoten des K_k davon überdeckt wird. Offenbar ist eine monochromatische Färbung von M somit nicht mehr möglich.
Es bleibt aber zu zeigen, daß stets eine Färbung gefunden werden kann. Hierzu seien (nach geeigneter Umnumerierung) $\{1, \ldots, l\}, l \leq k$, die Farben von M. $\phi(f)$ bezeichne ihre Vielfachheit. Ist $k = l$, so genügt es , eine beliebige fixpunktfreie Permutation der Farben für die Knoten aus K_k zu wählen. Ansonsten seien die Knoten des K_n der Farbgröße der zugehörigen Knoten aus M nach sortiert als $k_1, \ldots, k_k$ Nun gibt man k_i für $i < k$ jeweils die Farbe $i+1$. Da $\phi(f) \geq 1$ ist, ergibt sich somit leicht per Induktion, daß die Knoten 1 bis $k-1$ korrekt gefärbt werden. Da es nach Voraussetzung mehr als eine Farbe gibt, kann der zu Knoten k_n adjazente Knoten aus M nicht die Farbe 1 haben, so daß k_n diese erhalten kann.

4.3.2 „Integer Multicommodity“ Flüsse

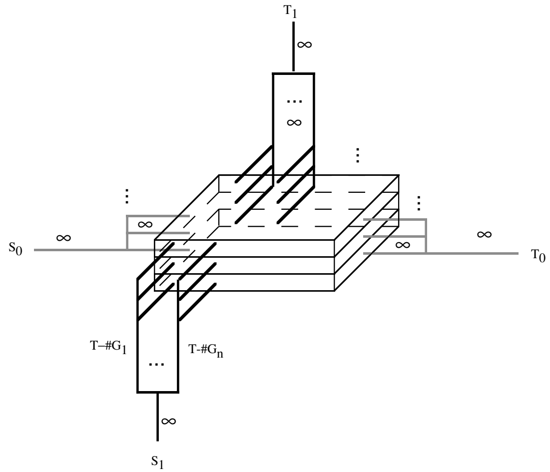
Zur Bearbeitung des Gesamtproblems fehlt in klassischen Netzwerkflüssen eine Art „Schalter“, da der benötigte „Schaltfluß“ stets auch die dem zu steuernden Fluß zugedachten Kanten verwenden kann. Die gewünschte Erweiterung des Flußmodells stellen die sogenannten „Integer Multicommodity-Flüsse“ dar. Zugrunde liegt hier die Vorstellung von mehreren - in diesem Fall zwei - Flüssen, die sich dieselben Netzwerkkapazitäten teilen.[3] Die folgenden zwei Abbildungen beschreiben eine einfache Reduktionsmöglichkeit.



Die Ovale repräsentieren Gruppen von Ressourcenanforderungen, die je eine Unterrichtseinheit bilden: Z.B. Lehrer A + Klasse C + 1 Biologieraum. Die Knoten auf der rechten Seite repräsentieren die verfügbaren Objekte und eine Kante verbindet eine Anforderung mit allen Objekten, die diese erfüllen können. Durch die Anforderungen laufen „quer“ die Kanten eines zweiten Flusses. Der zweite Fluß kann diese Kanten nicht verlassen, da er sonst seine Senke nicht erreichen kann.

Die dargestellten „Platten“ werden wie folgt zusammengesetzt:

[3] Interessanterweise wird deren $\mathcal{NP}$-Vollständigkeit erstmals im selben Paper gezeigt, wie die des „Timetabling“ [EIS76], ohne daß dabei auf die Reduktion eingegangen wird.



Für jede Zeitstunde wird eine solche Platte eingefügt. Die Kanten des zweiten Flusses werden gruppenweise verbunden und der Zufluß zu jeder Gruppe erhält als Kapazität die Zahl der Zeitstunden, die diese Gruppe nicht benötigt. Die Summe dieser Werte über alle Gruppen wird in der Senke angefordert. Die „Verteilerknoten“ des anderen Flusses werden jeweils mit Quelle und Senke verbunden, wobei an dieser Senke die Gesamtzahl aller je benötigten Objekte angefordert wird.
Wenn es einen gültigen Stundenplan gibt, gibt es offensichtlich einen Multi-Commodity-Fluß, der das gegebene Netzwerk befriedigt. Ist umgekehrt ein Fluß gegeben, so müssen alle nicht vom zweiten Fluß blockierten Anforderungen erfüllt werden. Dies ist aber nur möglich, wenn die entsprechende Anzahl von Stunden der jeweiligen Gruppen mit Ressourcen versehen werden kann. [4]

Hierbei bezeichne T die Zahl der Timeslices und $\#G_i$ die Zahl der Stunden, die Gruppe i (Lehrer, Klassen, Räume, s. Kapitel 5.3.1) benötigt.

4.3.3 Der Algorithmus nach König

Dieser und der nächste Abschnitt beschreiben auf Kantenfärbungen beruhende Ansätze zur Lösung von Spezialfällen des STP. Der älteste und wohl bekannteste Algorithmus beruht auf der Tatsache, daß für bipartite Graphen Maximalgrad (Δ) und Kantenfärbbarkeitszahl (χ') übereinstimmen. Diese bereits 1916 von KÖNIG durch Ergänzung zu einem regulären Graphen bewiesene Tatsache soll hier nochmals auf anderem Wege hergeleitet werden:

4.3.3.1 Einige Grundlagen aus der Matchingtheorie

Als Matching in einem Graphen bezeichnet man eine unabhängige Kantenmenge, d.h. eine Menge von Kanten, die keinen Knoten gemeinsam haben.

[4] Es ist kein Problem, daß Fluß 1 Kanten von Fluß 2 benutzen kann, da er auf diese Weise nur Kanten erreicht, die auf kürzerem Wege von seiner Quelle aus ohnehin erreicht werden und Kapazität 1 haben.

Wesentliches Element bei der Erzeugung von Matchings sind alternierende Pfade, d.h. Pfade in Graphen, bei denen abwechselnd zum Matching gehörige und nicht zum Matching gehörige Kanten auftreten. Solange ein alternierender Pfad existiert, der mit je einem nicht vom Matching überdeckten Knoten beginnt und endet, kann das Matching weiter vergrößert werden, indem man auf diesem Pfad jeweils Matchingkanten durch Nicht-Matchingkanten ersetzt und umgekehrt. Daß dieses Verfahren hinreichend ist, um maximale Matchings zu erstellen, ist Aussage des Satzes von Berge.

Satz 4.1 (Berge(1957)) *[Blu98] Ein Matching ist genau dann maximal, wenn es keine vergrößernden alternierenden (augmentierenden) Wege gibt.*

Beweis:
Betrachtet man die symmetrische Differenz zwischen dem gegebenen und einem maximalen Matching, so hat jeder Knoten einen Grad kleiner-gleich zwei. Damit besteht der Differenzgraph aus alternierenden Wegen und Kreisen. Da in jedem Kreis genausoviele Kanten des gegebenen, wie des maximalen Matchings auftreten, muß eine überzählige Kante des maximalen Matchings in einem Weg auftreten, der somit ein augmentierender sein muß.

Dieser Satz besagt insbesondere, daß man mit einem beliebigen Matching beginnen und dieses dann durch Finden alternierender Pfade zu einem maximalen Matching vergrößern kann. Dies liefert das sehr nützliche

Korollar 4.1 *Gegeben sei ein Graph G und ein Matching M auf G. Dann gibt es ein maximales Matching M' auf G, daß alle Knoten berührt, die von M überdeckt werden.*

Beweis:
Man konstruiere ein maximales Matching M' aus M durch beständiges Alternieren über augmentierende Pfade. Bei jeder Augmentierung bleiben alle vom ursprünglichen Matching übberdeckten Knoten überdeckt, da an jedem Knoten auch stets eine „Nicht-Matching-Kante" innerhalb des Pfades anliegt.

Zum Abschluß dieses Kapitels sei noch einer der wichtigsten Sätze über bipartite Graphen, d.h. Graphen, die sich in zwei innerhalb nicht verbundene Komponenten zerlegen lassen, erwähnt. Im Bereich der automatische Stundenplanerstellung und somit auch im Rest dieser Arbeit werden fast ausschließlich Graphen dieses Typs verwendet.

Satz 4.2 (Hall(1935)) *[Jun90] Ein bipartiter Graph $G = (A\dot{\cup}B, E)$ $(|A| = |B| \wedge E \subseteq (A \times B))$ hat ein perfektes Matching genau dann, wenn für alle $A' \subseteq A : |N(A')| \geq |A'|$ gilt.*

Der Beweis findet sich in nahezu jedem Buch über Graphentheorie und wird daher und auch wegen seines Umfangs und der Größe des benötigten Begriffsapparates hier nicht wiedergegeben.

4.3.3.2 Anwendung im „Timetabling"

Zunächst sollen hier einige unmittelbare Korollare aus Satz 4.2 gezogen werden. Das folgende, eigentlich allgemeinere Ergebnis fällt zumeist als Nebenprodukt des Beweises des Satzes von Hall ab, kann aber auch aus diesem recht einfach bewiesen werden.

Korollar 4.2 *Ein bipartiter Graph (s.o.) hat genau dann ein Matching, das eine gegebene Teilmenge A' von A überdeckt, wenn für alle $A'' \subseteq A'$ gilt, daß $|N(A'')| \geq |A'|$.*

Beweis:
Die Notwendigkeit ist wie im Satz von Hall trivial. Man verbinde nun alle Knoten in $A \backslash A'$ mit allen Knoten aus B. Der so entstandene Graph G' erfüllt die Voraussetzungen des Satzes von Hall: Enthält eine Teilmenge einen Knoten aus $A \backslash A'$ so gilt die Behauptung trivial, ansonsten folgt sie aus der Voraussetzung dieses Korollars. G' besitzt somit ein Matching, das $V(G')$ und somit auch A' überdeckt. Betrachtet man nun die Kanten, die in A' enden, so bilden sie offenbar auch in G ein Matching.

Mit Hilfe des Schubfachprinzips erhält man:

Korollar 4.3 *Jeder bipartite Graph besitzt ein Matching M_1, das alle Knoten maximalen Grades in A überdeckt.*

Mit Hilfe dieser Aussage kommt man schließlich zum folgenden Korollar:

Korollar 4.4 *Jeder bipartite Graph besitzt ein Matching M, das alle Knoten maximalen Grades überdeckt.*

Beweis [Huc]:
Offenbar gibt es mit Korollar 4.3 auch ein Matching, daß alle Knoten maximalen Grades in B überdeckt (M_2). Die Frage ist ausgehend von M_1, ob es eines gibt, daß die verbleibenden Knoten (eventuell ehemals) maximalen Grades in B überdeckt. Hierzu sei zunächst die Menge der Knoten maximalen Grades in A mit A' bezeichnet (analog B'). Es seien aus M_1 diejenigen Kanten E_1 gewählt, die von A' nach B' führen. Diese bilden offenbar ein Matching in dem von $B' \cup A$ induzierten Teilgraphen. Gemäß Korollar 4.1 und dem obigen Korollar kann man dieses zu einem Matching M_B erweitern, daß ganz B' und $A \cap V(E_1)$ überdeckt. Damit ist $A' \backslash V(M_B) \subset A' \backslash V(E_1)$. Für jeden Knoten $a \in A' \backslash V(E_1)$ gibt es aber aus M_1 eine Kante nach $B \backslash B'$. Nimmt man diese dann zu M_B hinzu, wenn a nicht bereits von M_B überdeckt wurde, erhält man ein Matching der gewünschten Art.

Somit erhält man unmittelbar den

Satz 4.3 (König (1916)) *Für jeden bipartiten Graphen gilt $\chi' = \Delta$.*

Beweis:
Man wähle als Farbklasse stets die Kanten eines Matchings das alle Knoten maximalen Grades überdeckt. Offenbar sinkt jedesmal der Maximalgrad im Graphen der noch nicht gefärbten Kanten um 1. Damit sind nach Δ Schritten alle Kanten gefärbt. Als maximales Ergebnis dieses Sachverhalts für einen Algorithmus zur automatischen Stundenplanerstellung erhält man somit das folgende Lemma:

Lemma 4.2 *Ein Algorithmus der sukzessive Timeslots zu Stunden zuordnet, kommt genau dann mit k Timeslots aus, wenn bei jeder Wahl zumindestens die Lehrer und Klassen ausgewählt werden, deren verbleibendes Stundendeputat mit der Zahl der verbleibenden Timeslots übereinstimmt.*

Die zu diesen Lehrern bzw. Klassen assoziierten Knoten werden im folgenden als „kritisch" bezeichnet. Dieser Satz ermöglicht es, den Algorithmus von König als Teilroutine anderer Verfahren zu verwenden. Siehe hierzu Kapitel 6.

4.3.4 Der Satz von Galvin

Lange Zeit gab es gegenüber den Verfahren von KÖNIG keinerlei Verbesserungen [dW85]. Mit dem Beweis der Vizingschen Listenfärbungsvermutung für bipartite Graphen durch Galvin ergibt sich für allgemeine Vorgaben- und Unverfügbarkeitszahlen ein wohl für das „School-Timetabling“ endgültiges Ergebnis, sofern man keine spezifischen Forderungen an die Unverfügbarkeiten stellt.[5]

Satz 4.4 (Galvin 1995) *Für einen bipartiten Graphen $G = (A\dot{\cup}B, E)$ gilt $\chi'(G) = \mathrm{ch}'(G)$.*

Beweis:
[Die96] Der Beweis verläuft in zwei Schritten. Zunächst wird gezeigt, daß jeder gerichtete Graph, dessen sämtliche induzierten Teilgraphen einen sog. Kern haben, und der für alle $v \in V$ $d^+(v) < |S_v|$ erfüllt, mit den S_v listenfärbbar ist. Dann wird bewiesen, daß jeder Linegraph eines bipartiten Graphen so orientiert werden kann, daß diese Bedingungen für Listen der Größe $\chi'(G)$ gelten. Da in bipartiten Graphen $\chi = \Delta$ gilt, folgt somit die Δ-Kanten-Listenfärbbarkeit bipartiter Graphen und daraus wegen $\chi'(G) \le \mathrm{ch}'(G)$ die Aussage des Satzes.

Definition 4.1 (Kern) *Eine Teilmenge U eines gerichteten Graphen $G = (V, E)$ ist ein Kern dieses Graphen, wenn U unabhängig ist und $\forall v \in V\backslash U\ \exists e = (v, u) \in E$ mit $u \in U$.*

Das Wesen der Methode besteht nun darin, die Bedingung $d^+(v) < |S_v|$ zu erhalten, indem man jedesmal, wenn $|S_v|$ verkleinert wird, sicherstellt, daß auch $d^+(v)$ reduziert wird.

Lemma 4.3 *Jeder Graph, den man so orientieren kann, daß $d^+(v) < |S_v|$ für alle v ist und jeder induzierte Teilgraph einen Kern besitzt, ist mit den Listen S_v färbbar.*

Beweis:
Der Beweis erfolgt durch Induktion. Der leere Graph kann offenbar mit 0 Farben gefärbt werden. Sei nun $H = (V, E)$ mit obigen Eigenschaften gegeben. Sei eine Farbe α beliebig gewählt und sei D derjenige Teilgraph von H, der von den Knoten v mit $\alpha \in S_v$ erzeugt wird. Dann besitzt D nach Voraussetzung einen Kern U. Man färbe nun alle Knoten in U mit α und entferne α aus allen Listen in $D\backslash U$. Da alle Knoten, deren Liste verkleinert wird, mit U verbunden waren, sinkt deren Ausgangsgrad ebenfalls um eins und $H\backslash U$ erfüllt die oben gestellten Bedingungen. Nach Induktion ist $H\backslash U$ mit den Farbmengen $S_v\backslash\alpha$ färbbar, und somit gilt die Behauptung.

Lemma 4.4 *Jeder Linegraph eines bipartiten Graph $G = (X\dot{\cup}Y, E)$ kann so orientiert werden, daß $d^+(v) < |S_v| = \chi'(G)$ für alle v ist und jeder induzierte Teilgraph einen Kern besitzt.*

Beweis: Im folgenden wird weiterhin vom Graphen G die Rede sein. Da das eigentliche Objekt der Betrachtungen der zugehörige Linegraph ist, werden nun Kanten zwischen Kanten auftreten.

[5]Erstaunlicherweise scheint dieser Satz im „Timetabling“-Bereich noch keine Beachtung gefunden zu haben, vgl. z.B. [dW96].

Es sei eine beliebige (herkömmliche) Färbung von G mit den Farben $1, \dots, \chi'(G)$ gegeben. Treffen sich zwei Kanten in X, so sei die Kante zwischen ihnen von der mit der größeren Farbe zu der mit der kleineren Farbe gerichtet. Treffen sie sich in Y, sei die Orientierung umgekehrt.

Zunächst ist zu zeigen, daß $d^+(v) < |S_v|\ \forall v \in X \cup Y$. Sei dazu e_0 fest gewählt. Alle Nachbarn von e_0, die Nachfolger von e_0 bezüglich der Kanten des Linegraphen sind, treffen e_0 in einem Punkt in X bzw. in einem Punkt von Y. Die Kanten in X haben alle kleinere und paarweise verschiedene Farben als e_0; die Kanten in Y haben größere und paarweise verschiedene Werte. Damit kann es maximal $\chi' - 1$ dieser Kanten geben. Zu e_0 parallele Kanten sind entweder in X oder in Y Nachfolger von e_0. *Damit gefährdet ihre Existenz die Gültigkeit der Aussage nicht.*

Als zweites ist zu zeigen, daß jeder Teilgraph einen Kern besitzt. Die Aussage gilt offenbar für leere Teilgraphen. Somit kann die Aussage wie folgt durch Induktion bewiesen werden: Es sei gezeigt, daß die Aussage für Graphen mit n Kanten gelte. Sei nun ein beliebiger Teilgraph D mit $n+1$ Kanten gewählt. Für $x \in X$ sei E_x die Menge der mit x inzidenten Kanten und e_x diejenige Kante aus E_x mit minimaler Farbe. Sei $U := \bigcup_{x \in X \cap V(D)} \{e_x\}$. Offenbar hat jede Kante in D einen Nachfolger in U, da jede Kante einen Knoten aus X trifft. *Dies gilt ebenfalls für parallele Kanten. Insbesondere wird von zwei parallelen Kanten immer nur eine in U sein.* Ist U also unabhängig, so ist U ein Kern. Es sei also angenommen, daß U nicht unabhängig ist, sich also zwei Kanten e, e' mit o.B.d.A. $c(e') > c(e)$ in Y treffen. Man entfernt nun e und erhält einen Teilgraph D' mit n Kanten, der nach Voraussetzung einen Kern U' besitzt. Ist e' in U', so ist wegen der Existenz der Kante (e, e') U' auch ein Kern von D.

Ansonsten gibt es eine Kante $e'' \in U'$, die von e' aus erreicht wird. Träfen sich e'' und e' in X, so wäre $c(e'') < c(e')$, was der ursprünglichen Wahl von e' widerspricht. Somit treffen sie sich in Y, was $c(e'') > c(e')$ impliziert, woraus wegen $c(e') > c(e)$ $c(e'') > c(e)$ folgt, was wiederum die Existenz der Kante (e, e'') erzwingt. Somit ist U' ein Kern von D.

Wiederum ergibt sich kein Konflikt bei mehrfachen Kanten.

Korollar 4.5 *Der obige Satz gilt auch für bipartite Multigraphen.*

Dies ergibt sich unmittelbar aus den Anmerkungen.

Korollar 4.6 *Für bipartite Multigraphen gilt $\mathrm{ch}' = \Delta$.*

Dies folgt aus dem vorigen Korollar und Satz 4.3.3.

Lemma 4.5 *Gegeben ein bipartiter Multigraph, die Farben $1, \dots, k$ und zu jeder Kante eine Liste V_e von „verbotenen“ Farben. Gilt $k \ge \Delta(G)$ und $|V_e| \le k - \Delta(G)\ \forall e$, so gibt es eine Färbung der Kanten von G mit den gegebenen Farben, ohne daß eine Kante eine für sie „verbotene“ Farbe erhält.*

Beweis: Jeder Kante e sei $\{1, \dots, k\}\backslash V_k$ als Farbliste zugewiesen. Diese Liste ist n.V. größer als $\Delta(G)$. Somit existiert nach obigem Korollar eine gültige Listenfärbung.

Damit kann also im klassischen STP jeder Kante eine Menge von verbotenen Timeslots zugeordnet werden, die der Größe nach durch die Differenz der Zahl der möglichen

Stunden und der maximalen Stundenzahl einer Klasse (bzw.eines Lehrers) beschränkt ist. Bei normalerweise 39 verfügbaren Schulstunden und 33 Klassen-Wochenstunden entspräche dies also 6 Stunden. [Tuz97] erwähnt, daß sich dieses Ergebnis noch soweit verschärfen läßt, daß jede Liste nur soviele Farben enthalten muß, wie der größere Maximalgrad der inzidenten Knoten. Dies bringt im „Timetabling“ aber keinen Vorteil, da nahezu alle Klassen stets zu fast allen Zeitstunden Unterricht haben, der obige Wert sich also nie sonderlich vom Maximalgrad des Graphen unterscheidet.

Der Beweis des Satzes von Galvin ist so konstruktiv, daß sich ein Algorithmus unmittelbar ableiten läßt (vgl. [Sli96]):

Algorithmus 4.1

1. Färbe den Graphen mit $\Delta(G)$ Farben z.B. mit dem Algorithmus aus Abschnitt 4.3.3.
2. Richte die Kanten des Linegraphen wie im Beweis beschrieben.
3. Für jede Farbe $\alpha \in \{1, \dots, k\}$:
   (a) Sei D der von den Kanten e mit $\alpha \in S_e$ erzeugte Teilgraph des Linegraphen.
   (b) Setze $D' = D$.
   (c) Wähle U aus D' wie oben beschrieben.
   (d) Solange U nicht unabhängig ist,
      - wähle zwei adjazente Kanten in U und entferne aus D' diejenige mit der kleineren Farbe,
      - bestimme U erneut aus D'.
   (e) Färbe die Kanten aus U mit α, entferne sie aus D und entferne α aus allen Farblisten.

Die Korrektheit des Algorithmus ergibt sich unmittelbar aus den vorangehenden Sätzen und Korollaren.

Lemma 4.6 *Obiger Algorithmus kann mit einer Laufzeit von $\mathcal{O}(kn^3)$ implementiert werden.*

Beweis: Sei o.B.d.A $|X| = |Y| = n$ und m die Zahl der Kanten. Die Listen seien durch eine Farben-Kanten Matrix repräsentiert. Der Algorithmus nach König bestimmt k ungewichtete Matchings in jeweils bipartiten Graphen, was in $\mathcal{O}(nm)$[Blu98][6] zu bewerkstelligen ist. Nach jeder Bestimmung eines Matchings können alle Kanten im Linegraphen, die Kanten des letzten Matchings (die die bisher größte Farbe erhalten) mit Kanten kleinerer Farbe verbinden (und umgekehrt), sofort erzeugt werden. Da jede Kante maximal mit $2\Delta(G)$ Kanten verbunden ist, läßt sich der Aufwand dafür mit diesem Wert abschätzen. Da für das jeweils nächste Matching die jeweils „nächstgrößte“ Farbe gewählt wird, können die zu einem Knoten gehörigen Kanten und die Farben-Kanten-Matrix gleichzeitig der Farbgröße nach sortiert werden. Somit erhält

[6] Im gradbeschränkten Fall läßt sich dies laut [CL92] auf $\mathcal{O}(\Delta n^{3/2})$ verbessern.

man für die Schritte 1 und 2, die hier gleichzeitig behandelt werden, eine Laufzeit von $\mathcal{O}(k(nm + 2\Delta(G)))$, wobei $\Delta(G) \leq n$ gilt.

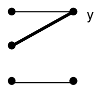
Die nun folgende Schleife wird k mal ausgeführt. Der Teilgraph kann über die Matrix in $\mathcal{O}(m)$ bestimmt werden und aufgrund der Sortierung kann gleichzeitig U daraus erzeugt werden, indem bei jedem vorangegangenen Entfernen nicht zum Teilgraphen gehörender Knoten aus Y ein Zeiger, der stets auf die Kante mit minimaler Farbe zeigt, auf das nächstgrößere, nicht gelöschte gesetzt wird. Dies setzt voraus, daß die Kanten-Farben-Matrix entsprechend ihrer Sortierung abgearbeitet wird. Das abschließende Auslesen von U geht in $\mathcal{O}(n)$.

Der Test auf Unabhängigkeit benötigt $\mathcal{O}(n)$ Schritte, da sich die entsprechenden Kanten nur in Y schneiden können und es somit ausreicht, die bereits überdeckten Knoten in Y zu markieren und ein nicht unabhängiges Paar daran zu erkennen, daß einer nochmals markiert werden soll. Das Entfernen einer Kante und das Neuberechnen von U sind wiederum aufgrund der Sortierung in $\mathcal{O}(1)$ möglich, da beim Entfernen einer Kante nur jeweils die Kante mit der nächstgrößeren Farbe am selben Knoten aus X diese ersetzt. Es müssen höchstens m Kanten entfernt werden, um Unabhängigkeit zu erzielen. Damit ergibt sich eine Gesamtlaufzeit von

$$\mathcal{O}(knm + k(m + n + m(2n + 1))) \leq \mathcal{O}(knm) \leq \mathcal{O}(kn^3)$$

Im „Timetabling“-Kontext ist zumeist $\mathcal{O}(m) \leq \mathcal{O}(n)$, da Lehrer und Klassen eine beschränkte Stundenzahl haben, und der Algorithmus hätte hier somit eine Laufzeit von $\mathcal{O}(kn^2)$, was der des Algorithmus nach König für diese Fälle entspricht.

Der Algorithmus läßt drei Freiheitsgrade für weitere Verbesserungen. Zum einen könnte überprüft werden, ob es möglich ist, jeweils andere Kerne zu wählen oder die Reihenfolge der Farbwahl zu variieren, zum anderen ist es möglich, die Ausgangsfärbung analog zu Abschnitt 6.6 zu modifizieren. Hier tritt allerdings das Problem auf, daß obiger Algorithmus bereits gültige Ausgangsfärbungen nicht erhält, wie das folgende Beispiel zeigt. Dabei seien für alle Punkte beide Farben zulässig; die dicken Kanten tragen Farbe 2:



Die oben dargestellte Färbung ist offensichtlich gültig. Der Algorithmus würde nun zunächst alle Kanten in U aufnehmen, dann aber die obere Kante aus U entfernen, da sie die kleinere Farbe an y hat. Damit würden zwei Kanten, die in der gültigen Ausgangsfärbung verschiedene Farben hatten, nun die gleiche Farbe erhalten. Dies ist auch nicht weiter verwunderlich, da die Werte der Ausgangsfärbung durch eine andere Wahl der Reihenfolge im zweiten Schritt des Algorithmus teilweise außer Kraft gesetzt werden können.

Dennoch läßt sich durch Modifikation dieser Färbung etwas erreichen: Im Bereich der automatische Stundenplanerstellung ist es häufig so, daß nur einige Objekte von Bedingungen betroffen sind. Daher ergibt sich in geringfügiger Abweichung von der Bestimmung der Listenfärbbarkeitszahl die Frage, wie klein man irgendeine bestimmte Liste wählen kann. Hier nutzt der Satz von Galvin die Kraft seiner beiden Lemmata nicht voll aus, da $\chi' \leq \mathrm{ch}'$ unmittelbar klar ist, und sich die Frage nach unterschiedlich großen Listen beim reinen Beweis der Viezingschen Vermutung nicht stellte. Durch striktere Interpretation kommt man somit zu dem:

Satz 4.5 (Galvin,umformuliert) *Gegeben ein bipartiter Graph G mit $k = \Delta(G)$ und eine beliebige k-Färbung F. Sei H der wie im Beweis von Lemma 4.4 gemäß der gegebenen Färbung orientiert. Dann ist für jede Wahl von Listen S mit $|S_e| > \delta^+_H(e)$ G aus diesen Listen kantenfärbbar.*

Die Frage ist also, unter welchen Umständen man gezielt den Ausgangsgrad eines Knotens des Linegraphen H reduzieren kann. Dies ist offensichtlich nicht möglich, wenn beide Knoten, zu denen die entsprechende Kante inzident ist, den Grad $\Delta(G)$ aufweisen. Nimmt man aber an, daß einer (o.B.d.A der in Y) geringeren Grad hat, so kann man durch geeignete Wahl der Farbe dieser Kante e in F erreichen, daß der Grad in H entsprechend sinkt, z.B. indem man e in F die kleinstmögliche Farbe gibt. Damit hat e um eins geringeren Ausgangsgrad in H, als der zugehörige Knoten in Y Nachbarn, da alle in X e treffenden Kanten somit Vorgänger von e in H sind. Gibt man dann der nächsten Kante die zweitkleinste Farbe, so ist in X ein Nachfolger möglich, die Zahl derer in Y reduziert sich aber um 1. Insgesamt erhält man

Lemma 4.7 *Treten an einem Knoten in $v \in Y$ in F nur die kleinsten $j \geq \delta$ Farben auf, so reicht es, den zugehörigen Kanten eine Liste der Größe j zuzuordnen.*

Beweis:
Hat die Kante e mit der l. größten Farbe an $e \cap Y$ die Farbe p, so ist $p \leq j - (l-1)$, da die übrigen $l-1$ Kanten Farben aus dem Intervall $(p, j]$ erhalten müssen. Andererseits können nur noch diese $l-1$ der Kanten, die ebenfalls an diesem Knoten in Y anliegen, Nachfolger von e in Y sein. In X kann es maximal $p-1$ Nachfolger geben, da es nur entsprechend viele kleinere Farben gibt. Somit hat e maximal $j-(l-1)-1+(l-1) = j-1$ Nachfolger. Somit ist $d^+_H(e) < j$

Man könnte nun also versuchen, die Tatsache, daß nur diese j Farben auftreten damit zu erzwingen, daß man den entsprechenden Kanten in G bei dieser Vorfärbung Listen der Größe $j_{e\cap V}$ zuordnet, die nur die ersten $j_{e\cap V}$ Farben enthalten. Was zunächst wie ein reiner Zirkelschluß aussieht, liefert dann aber letztlich doch den

Satz 4.6 *Sei ein bipartiter Graph G gegeben und jeder Kante e eine Liste der Größe j_i zugeordnet, wobei $i = e \cap Y$. Dann ist der Linegraph von G genau dann mit allen beliebigen Listen dieser Größe färbbar, wenn er es auch für den Fall ist, daß jede Liste die kleinstmöglichen Farben enthält.*

Beweis:
„$\Rightarrow$“ ist trivial, da der Spezialfall auch zu „allen“ Fällen gehört.
„$\Leftarrow$“ ergibt sich, indem man Lemma 4.7 auf jeden Knoten in Y anwendet.

Das Ganze gilt symmetrisch auch für die andere Komponente, wobei hier die j größten Farben verwendet werden müßten. Dieser Satz bestätigt somit die intuitiv klare Tatsache, daß der schlechtestmögliche Fall der ist, wenn alle Lehrer (Klassen) gleichzeitig abwesend sind.

Korollar 4.7 *Das Problem zu entscheiden, ob ein bipartiter Graph mit einer Zuordnung von Listengrößen zu Knoten einer Komponente gemäß aller Listen der jeweiligen Größen gefärbt werden kann, liegt in $\mathcal{NP}$.*

Beweis
Offenbar stellt eine Färbung, in der nur die kleinsten $|L_i|$ Farben in jeder Liste L_i auftreten, einen geeigneten (polynomiell überprüfbaren) Beweiskandidaten dar.

Bemerkung 4.1 *Das Problem zu entscheiden, ob ein bipartiter Graph mit einer beliebigen Zuordnung von Listengrößen zu Kanten gemäß aller Listen der jeweiligen Größen gefärbt werden kann, ist Π_2-vollständig ([Rei90] [Tuz97]).*

4.3.5 „equitable colorings“

Häufig wird verlangt, daß sowohl Lehrer als auch Klassen an jedem Tag gleichstark beansprucht werden. Weisen alle Tage gleichviele Stunden auf, so kann dies mit Hilfe von „equitable Colorings“ ([dW96]) erreicht werden. Eine Kantenfärbung wird als „equitable“ (balanciert) bezeichnet, wenn für jeden Knoten die Zahl der inzidenten Kanten jeder Farbe ungefähr gleich ist. Genauer: Ist $\phi(f, i)$ die Häufigkeit des Auftretens von Farbe f an Knoten i, so gilt für alle Knoten i und für alle Paare von Farben (f_1, f_2): $-1 \leq \phi(f_1, i) - \phi(f_2, i) \leq 1$.
Eine solche Färbung mit k Farben kann gefunden werden, indem jeder Knoten v in $\lceil \delta(v)/k \rceil$ Hilfsknoten vom Grad k, mit möglicherweise einer Ausnahme, aufgespalten wird und auf dem so gebildeten Graphen eine normale Kantenfärbung vorgenommen wird. Identifiziert man danach alle Hilfsknoten wieder mit dem Original, so erhält man eine Färbung des gewünschten Typs, da mindestens $\lfloor \delta(k)/k \rfloor$ mal und höchstens $\lceil \delta(k)/k \rceil$ mal die gleiche Farbe an einem Knoten vorliegt.

Zusätzlich kann erreicht werden, daß die einzelnen Lehrer-Klassen-Kombinationen gleichmäßiger in der Woche verteilt liegen (vgl. [dWM95]). Ist $\max_{v \in V} \lceil \frac{\delta(v)}{k} \rceil$ kleiner gleich der Zahl der Zeitstunden an einem Tag l, so kann das „equitable“ Scheduling zu einem normalen Stundenplan erweitert werden, indem der Algorithmus nach König auf die jeweils von den Kanten einer Farbe induzierten Teilgraphen angewandt wird, da deren Maximalgrad dann kleiner als l ist.

4.4 Lineare Programmierung / Ganzzahlige Lineare Programmierung / Semidefinite Programme

4.4.1 Lineare Programmierung und Ganzzahlige Lineare Programmierung

Obwohl Lineare Programmierung und Ganzzahlige Lineare Programmierung zum üblichen Methodenrepertoire der kombinatorischen Optimierung gehören, konnten sie sich erst in den letzten Jahren im „Timetabling“-Bereich etablieren [Bau97]. Die Größe der Problemstellung machte ein Herangehen mit „Branch & Bound“-Verfahren unmöglich und ist auf in Schulen verfügbarer Hardware auch heute noch nicht praktikabel. Positive Ergebnisse bezüglich der Gutartigkeit der jeweiligen Polytope, wie sie z.B. beim „Vehicle Routing“ [RF88] gegeben sind, liegen noch nicht vor.
Dafür kann man - entsprechende Flexibilität bei der jeweiligen Institution voraussetzend - nichtganzzahlige Lösungen dennoch anwenden, indem man entsprechend dem kleinsten gemeinsamen Vielfachen der Nenner der Ergebnisse viele Wochen wählt und dann ganze Stunden entsprechend den jeweiligen Bruchteilen verteilt. Dieses Verfahren

wird aber in der Regel nur bis zu einer Größe des Hauptnenners von 2 eingesetzt.

Lineare Programme dienen daher primär als Lieferant unterer Schranken für „Branch & Bound“ Verfahren. Zudem sind lineare Zielfunktionen für das „Timetabling“ zumeist nicht ausreichend. So ist es z.B. unbefriedigend, wenn 10 Lehrer keine ungünstigen Stunden haben, einer aber deren 10. Diesem kann abgeholfen werden, indem man annimmt, daß es für jedes (lineare) Kriterium K_i eine von einem Parameter abhängige Funktion $\phi_i(u)$ gibt, so daß für alle u (z.B. $\in \mathbb{N}_{\geq u_0}$) gilt, daß jede Lösung mit $K_i(u) \leq \phi_i(u)\ \forall i$ befriedigender ist, als jede, bei der eine der Ungleichungen gebrochen wird. Das kleinste u, das noch eine solche Lösung zuläßt, könnte mit binärer Suche dann relativ schnell gefunden werden. Im „Timetabling“ ließe sich eine solche Funktion zumeist sehr leicht angeben, da z.B. eine gleiche Nachmittagsbelastung der Klassen o.ä. zu den normalen Anforderungen zählt.
Wollte man aber versuchen, diese Methode zur weiteren Optimierung einzusetzen, so stünde man vor dem Problem, eine Teilmenge der K_i auswählen zu müssen, für die der Wert von ϕ_i ab sofort anders gewählt werden soll. Diese Wahl dürfte aber wieder einen exponentiellen Suchaufwand verlangen. Schränkt man die Suche auf ganzzahlige Probleme ein, so müßte dann allerdings der entsprechende ILP-Algorithmus für jeden Wert von u erneut gestartet werden.

Eine Darstellung der Probleme als ILPs wurde bei ihrer Einführung bereits gegeben. Lineare Programme mit mehreren zehntausend Variablen - bei einer mittleren Schule ca. 30000 - können beim heutigen Stand der Technik selbst auf Microcomputern schnell gelöst werden. Relative gewichtete Nebenbedingungen können allerdings nicht modelliert werden. Bei ILPs liegt die Grenze im allgemeinen noch etwas niedriger [MK98], d.h. im Bereich einiger tausend Variablen und es gibt bereits Untersuchungen zu Problemen geringer Größe, bei denen sich ILP z.B. „Simulated Annealing“ überlegen zeigte [BGH97].

Bemerkung 4.2 *Setzt man beim STP/UTP eine der gegebenen Modellierungen ein, so kann die Springstundenzahl eines Lehrers nicht durch eine lineare Funktion beschrieben werden.*

Offenbar bilden Pläne, in denen der Lehrer jeweils nur eine Stunde hat, eine Basis des $\{0,1\}^{|Stundenzahl|}$, über dem diese Funktion modelliert werden müßte. Bei je einer Stunde liegt aber keine Springstunde vor. Des weiteren liegt keine Springstunde vor, wenn der Lehrer gar keinen Unterricht hat. Somit wäre eine entsprechende affin-lineare Funktion auf den Vektoren einer Basis und auf dem Nullpunkt Null und somit identisch gleich Null.
Dennoch könnte man die Springstundenminimierung in der Form $\min a : a \in [0,1]$ $a+1+m-r-l \geq 0$ hinzufügen, wobei die entsprechenden Stunden in der Reihenfolge rml im Plan auftreten. a kann somit nur dann Null sein, wenn m keine Springstunde ist.

4.4.2 Semidefinite Programme

Einfache quadratische konvexe Probleme lassen sich mit Hilfe semidefiniter Programmierung darstellen. Diese wiederum kann durch pfadverfolgende Methoden [VB96] oder der Karmakar-Methode für lineare Programme ähnelnde Verfahren [Ali95] in polynomieller Zeit ausgewertet werden. Sie wird insbesondere in Approximationsalgorithmen für Graphenfärbungsprobleme [Hoc97] und der Relaxierung von ILP-Problemen

eingesetzt.
Ein semidefinites Programm ist gegeben durch eine Menge von Variablen $(x_i)_{i\in I}$, eine lineare Zielfunktion $c(x) = \sum_{i\in I} c_i x_i$ und durch eine Menge symmetrischer[7] Matrizen S_i. Das Optimierungsproblem hat die Form

$$\begin{aligned}
& && \text{minimiere } c^t x \\
& && \text{unter der Nebenbedingung, daß} \\
H &:= && \sum_{i\in I} x_i S_i + S_0 \text{ positiv semidefinit ist, d.h.} \\
a^t H a &\geq 0 && \forall a \in \mathbb{R}^I
\end{aligned}$$

Dies kann z.B. dazu verwendet werden, eine gleichmäßige Belastung der Lehrer zu gewährleisten: Wäre z.B. durch l_i ein Maß für die Belastung eines Lehrers gegeben, so könnte der Ausdruck

$$\sum_{i<j} (l_i - l_j)^2 \tag{4.1}$$

minimiert werden. Hierzu muß zunächst das LP, das die gültige Menge beschreibt, in ein Teil des SDP überführt werden. Jedes LP aus Kapitel 2 kann hier eingesetzt werden: Formt man zunächst die Nebenbedingungen zu $Ax - b \geq 0$ für alle i um und hat A k Zeilen, so kann man durch Wahl der oberen $k \times k$ Einträge als

$$S_i := \begin{pmatrix} a_{i1} & 0 & \dots & 0 & \dots \\ 0 & \ddots & \ddots & \vdots & \\ \vdots & \ddots & \ddots & 0 & \\ 0 & \vdots & 0 & a_{ik} & \\ \vdots & & & & \ddots \end{pmatrix} \tag{4.2}$$

und

$$S_0 = \begin{pmatrix} -b_1 & 0 & \dots & 0 & \dots \\ 0 & \ddots & \ddots & \vdots & \\ \vdots & \ddots & \ddots & 0 & \\ 0 & \vdots & 0 & -b_k & \\ \vdots & & & & \ddots \end{pmatrix} \tag{4.3}$$

wählen, was die entsprechenden Bedingungen garantiert.
Um die obige Optimierungsaufgabe einzubinden, seien zunächst die Hilfsvariablen $x_{ij} := (l_i - l_j)$ und t hinzugefügt. Diese Eigenschaft der x_{ij} ist linear und kann daher analog zum obigen Vorgehen sichergestellt werden. Nun fügt man weitere Matrizen S_{ij} und S_t zu der Menge hinzu, so daß sich als deren Summe eine auf der Diagonalen liegende Teilmatrix der Form

$$\begin{pmatrix} t & \mathcal{X} \\ \mathcal{X}^t & 1 \end{pmatrix} \quad \mathcal{X} = \begin{pmatrix} x_{11} \\ x_{12} \\ \dots \end{pmatrix} \tag{4.4}$$

[7] Notwendig, damit die Menge konvex bleibt.

ergibt. Wie man [VB96] entnehmen kann, sorgt dies dafür, daß $t - x^t x \geq 0$ gilt.[8] Strebt man nun die Minimierung von t an, so wird offensichtlich auch 4.1 minimiert.

Bemerkung 4.3 *Mit rein linearen Bedingungen könnte man ähnliches durch* $\min t : l_i - l_j \leq t \wedge l_j - l_i \leq t \; \forall i < j$ *versuchen. Hierbei würde allerdings eine große Abweichung in einer Variable die Minimierung der übrigen verhindern.*

Bemerkung 4.4 *Interpretiert man die entstehenden Ergebnisse aus Sicht der Ganzzahligen Linearen Programmierung, so bringt, was die reine Beschreibung angeht, der Schritt von LPs zu beliebigen konvexen Problemen für beschränkte Polyeder (Lösungsmengen) keinen Vorteil, da sich - gegeben eine solche konvexe Menge - zu jedem ganzzahligen Punkt außerhalb dieser Menge aber innerhalb eines ebenfalls beschränkbaren umgebenden Polyeders eine trennende Hyperebene finden läßt. Da innerhalb einer beschränkten Menge aber nur endlich viele ganzzahlige Punkte liegen können, bräuchte man also nur endlich viele Hyperebenen hinzuzunehmen, um einen Polyeder zu erhalten, der dieselben ganzzahligen Punkte enthält.* [9]

Da die Ränder der Lösungsmenge nicht mehr von konvexen $(|I| - 1)$-dimensionalen Polyedern gebildet werden, wie dies bei den LPs zumeist der Fall ist, muß die Lösung nicht mehr zwangsläufig in einer Ecke liegen, und somit können hier selbst bei ganzzahligen Ecken nichtganzzahlige Lösungen auftreten.

4.5 Constraintbasierte Verfahren

4.5.1 Prinzip

Diese Methode besteht darin, eine Menge von Variablen anzunehmen, für diese „Domains" (Wertbereiche) und „Constraints" (Relationen) zu definieren und dann mit Hilfe von Propagationsverfahren, z.B. durch Intervallarithmetik[SM97] [10] die Domains mehr und mehr einzuschränken, bzw. neue Constraints zu erzeugen. Im Falle der automatischen Stundenplanerstellung ist die Ausgangsdarstellung zumeist mit der für ILPs identisch. Zum Lösen der Aufgaben werden entweder fertige Pakete zur Constraintpropagierung [WZ97] oder Variationen des Verfahrens von GOTTLIEB ([Got63]) eingesetzt. Da dies nach dem Algorithmus nach KÖNIG, der nicht speziell im Hinblick auf das Timetabling entstand, der erste „echte" Algorithmus der automatischen Stundenplanerstellung war, soll er hier kurz beschrieben werden:

4.5.2 Das Verfahren von Gotlieb

Grundlegend ist es die gleiche Konstruktion, wie sie auch in Abschnitt 4.3.3 verwendet wird. Zusätzlich werden zwei $\{0,1\}$-Vektoren eingeführt, die angeben, zu welchen Zeiten die Lehrer bzw. Klassen verfügbar sind.[11] Seien mit r_{ij} die Anforderungsmatrix, mit s_{ij}^k die Verteilung in der $k$ten Stunde und mit t bzw c die oben genannten Vektoren bezeichnet. $A^{ij} = t_i \wedge c_j$ sei als Schreibweise für die Zeiten vereinbart zu denen sowohl Lehrer i, als auch Klasse j zur Verfügung stehen. Die Mengen der Klassen bzw. Lehrer seien $\mathcal{C}$ bzw. $\mathcal{T}$. Die Zeitstunden seien mit $\mathcal{Z}$ bezeichnet. Man erhält damit folgende Constraints (bei [Got63] noch „Restraints" genannt):

$$\begin{aligned} \sum_{j \in C} \sum_{\mathcal{Z}} A^{ij} &\geq \sum_{j \in C} r_{ij} \; \forall i \forall C \subset \mathcal{C} \\ \sum_{i \in T} \sum_{\mathcal{Z}} A^{ij} &\geq \sum_{i \in T} r_{ij} \; \forall j \forall T \subset \mathcal{T} \end{aligned}$$

In moderner Sprechweise: Die Hall-Bedingungen für ein Matching zwischen den geforderten Stunden und den Zeitpunkten zu denen jeweils eine solche Stunde stattfinden kann, müssen erfüllt sein.
Fälle, in denen in einer der obigen Ungleichungen Gleichheit gilt, bezeichnet GOTLIEB als „tight" und nur in diesen ist nach seiner Methode eine „Restraint-Propagierung" möglich: Da bei einer „tight" Kombination alle Stunden dafür verwendet werden müssen, können diese für andere Lehrer (Klassen) gesperrt werden. Hiermit sinken bei verschiedenen anderen Gleichungen die linken Seiten, was zum Beweis der Unlösbarkeit oder zu neuen „tight" Kombinationen führen kann. Ist keins davon der Fall, werden solange Stunden beliebig festgelegt, bis wieder eine solche Menge auftritt. Als zusätzliche Bedingung wird noch angeführt, daß entsprechend auch ein Matching von den „tight" Zeitpunkten zurück zu den Lehrern bzw. Klassen möglich sein muß. GOTLIEB[Got63] gibt allerdings weder eine Reihenfolge der Variablen noch Überlegungen dazu an, wie zeitaufwendig das Verfahren ist. LIONS [Lio66] reduziert den von ihm auch nicht näher untersuchten Aufwand für das Überprüfen der Bedingungen durch Einführung des „Ungarischen Algorithmus" (vgl. Kapitel 6), wobei hier aber einfaches bipartites Matching als dessen Sonderfall schneller wäre.

4.5.3 Anwendbarkeit

Da die meisten Standardpakete den Raum der verbleibenden möglichen Lösungen nahezu vollständig durchsuchen und zur Eingrenzung nur aus den Constraints, nicht aber aus dem Kontext gewonnene Heuristiken verwenden, dürften sich diese Verfahren durch eine stärkere Abkehr von den fertigen Paketen hin zu „Timetabling"-spezifischen Lösungen verbessern.

Insbesondere ist es wahrscheinlich nicht hinreichend, sich bei der Untersuchung auf eine Klassifikation der Constrainttypen zu beschränken, da, wie [KS96] zeigt, bei der reinen Vorgabe von Typen und ohne Einschränkung ihrer Anwendung die Constraint-Satisfaction-Probleme (CSP) entweder trivial - was das „Timetabling" nicht ist - oder MAX-SNP-hart sind.
Modifikationen werden zumeist hinsichtlich der Reihenfolge der Auswertung der verschiedenen Implikationen und hinsichtlich der Auswahl der fest zu setzenden Variablen vorgenommen. Letzteres ist notwendig, da zu Beginn nahezu keine Folgerungen möglich sind und deren Anzahl erst mit zunehmender Festlegung immer weiter wächst.

[8] Obwohl der Schluß zwischen Vorzeichen der Teildeterminanten auf die Determiniertheit i.a. nur für positive Determiniertheit gilt.

[9] Es kann allerdings im Sinne der Problemstellung „hart" sein, die entsprechenden Hyperebenen zu ermitteln [GLS88].

[10] Oder auch durch Resolution; dann wird das Ganze als Logisches Verfahren bezeichnet.

[11] Damit wird genau das Problem betrachtet, das EVEN et al. [EIS76] 13 Jahre später als $\mathcal{NP}$-hart beweist.

Es ist aber wohl so, daß die Bestimmung einer optimalen Reihenfolge, ebenso wie beim „Greedy-Algorithmus“ (vgl. oben) selbst NP-hart ist. MARTE [Mar98] berichtet sogar, daß diese Verfahren keine Lösungen mehr liefern, wenn die Zahl der Nebenbedingungen gesenkt wird. In derselben Arbeit finden sich einige statistisch orientierte Heuristiken zur Wahl der Reihenfolge.

Da diese Methode keine Mechanismen für Optimierungsvorgänge vorsieht, wird sie häufig in Kombination mit anderen Techniken z.B. als „Reparaturmechanismus“ verwendet. Abgesehen von den oben angedeuteten Propagationsversuchen finden sich unter dem Begriff „Constraintbasiert“ aber auch andere Verfahren, bei denen dieses aber nur als Bestandteil der Terminologie verwendet wird, da gemäß einer im KI-Bereich verbreiteten Klassifizierung die automatische Stundenplanerstellung unter die „Constraint Satisfaction“-Probleme fällt.

4.6 Genetische Algorithmen / Evolutionsstrategien

Die natürliche Evolution war das Vorbild für dieses Verfahren. Ausgehend von einer Modellierung verschiedener Stundenpläne als Lebewesen (Chromosomen, Bitstränge u.s.w.) werden drei Typen von Operationen auf diese angewandt:

Selektion: Aus der Menge der Pläne wird ein bestimmter Prozentsatz ausgewählt, der sich durch besondere Qualität auszeichnet. Alle anderen werden entfernt.

Mutation: Ein Plan wird geringfügig verändert (ggf. vorher kopiert).

Kreuzung: Aus zwei Stundenplänen wird ein neuer generiert.

Läßt man die „Kreuzung“ weg, so stellt dieses Verfahren eine parallelisierte Variante der in Abschnitt 4.7 beschriebenen Verfahren dar, wobei der Mutationsoperator die „Nachbarschaft“ und die „Selektion“ das Akzeptanzkriterium bildet.

Es hat zahlreiche Versuche gegeben, GAs auf das „Timetabling“-Problem anzuwenden. Zur Codierung werden zumeist nicht klassische Bitstrings, sondern ganzzahlige Repräsentationen verwendet [CR97], auf denen dann Tauschoperationen durchgeführt werden. Die Ergebnisse sind, wie z.Zt. im „Timetabling“-Bereich noch häufig üblich, nicht vergleichbar. Es gibt seit Mitte 1997 Bestrebungen, ein einheitliches Datenformat für Probleme im Kontext der Automatischen Stundenplanerstellung zu schaffen [BKP97]; diese scheinen aber noch keine Wirkung zu zeigen.

Wie von ROSS, HART und CORNE [RHC97] beobachtet, zeigen sich bei „Timetabling“ und Graphenfärbungsproblemen Schwierigkeiten bei der Anwendung dieser Verfahren. Ein Grund hierfür dürfte in der starken Trennung zwischen harten und weichen Constraints und in dem Widerspruch zwischen der großen Symmetrie der harten Constraints und deren Fehlen bei den weichen Nebenbedingungen liegen: Die Befriedigung der harten Bedingungen, insbesondere dabei der, daß kein Objekt zur selben Zeit an verschiedenen Orten sein kann, ist unter Permutationen der Zeitstunden invariant. D.h. ausgehend davon, daß es nur eine gute Lösung für das „harte Suchproblem“ gibt, tritt dies bei einer die schwachen Nebenbedingungen ebenfalls beachtenden Darstellung und bei 40 zur Verfügung stehenden Wochenstunden 40! mal auf. Gerade wenn das Verfahren also schnell zu guten Lösungen konvergiert, wird dies also dazu führen, daß die selbe Lösung in jeweils anderer Erscheinungsform mehrfach auftritt. Eine Kreuzung kann also entweder nur zu einer weiteren Repräsentation desselben „harten“ Typs oder

zu einem ungültigen Plan führen, da die einkopierten Gene bei dem Zielchromosom an anderer Stelle vollkommen gleichartig auftreten.
Allgemeine Untersuchungen zur Funktion genetischer Algorithmen postulieren, daß sich bestimmte „Schemata“, d.h. bestimmte Abschnitte des genetischen Codes anderen gegenüber mit exponentiell wachsender Wahrscheinlichkeit durchsetzen, sofern sie diesen von der Zielfunktion her überlegen sind [Mic92]. Bei der oben angedeuteten großen Zahl guter Lösungen dürfte es aber zu jeder ausreichend kurzen Sequenz - die Wahrscheinlichkeit des Auftretens einer bestimmten langen Sequenz nimmt mit ihrer Länge exponentiell ab - eine Lösung geben, deren Teil sie ist. Die somit erfolgende Nivellierung der „Schemata“ dürfte die Qualität des Verfahrens bis in die Nähe rein randomisierten Suchens reduzieren [RHC97]. Versuche, eine spezialisierte Form „Genetischer Codes“ für Probleme dieses Typs zu schaffen, wurden unternommen, zeigten sich aber den herkömmlichen Verfahren unterlegen [EdHH98]. Außerdem kann keine wesentlich andere die verschiedenen Permutationen zusammenfassende Kodierung gewählt werden, da ansonsten eine Berücksichtigung der „weichen“ Bedingungen nicht mehr möglich ist. Beobachtet wurde dieser Effekt z.B. von [MR97], die angeben, daß nach einer gewissen Zeit die Population aus lauter gleichguten Spezies bestehe.

Der größte Teil der Literatur zu genetischen Algorithmen im Timetabling (s. z.B. [CR97]) befaßt sich neben der Bestimmung von Zielfunktionen mit der Konstruktion von „lokalen Reparaturverfahren“, die ein durch Kreuzung entstandenes ungültiges Chromosom wieder im Sinne einiger Nebenbedingungen gültig machen sollen. Aufgrund der hohen Laufzeitanforderungen - es sollen zumeist mehrere tausend Kreuzungen vorgenommen werden - kommen hier allerdings nur die einfachsten Heuristiken in Frage, wie z.B. das Suchen einer beliebigen freien Stelle etc.

4.7 „Lokale“ Suchverfahren: Simulated Annealing und Threshold Accepting

Das Wesen beider Methoden besteht darin, von einem gegebenen Punkt im Lösungsraum jeweils zu einem in irgendeinem Sinne „benachbarten“ überzugehen und diese Veränderung unter bestimmten Bedingungen zu akzeptieren.

Es ist klar, daß die Geschwindigkeit und Leistungsfähigkeit wesentlich von dieser Nachbarschaftsmenge abhängt: Ist $M \subseteq L$ die Menge der akzeptablen Lösungen im Lösungsraum L, so würde ein Algorithmus für den die Nachbarschaft jedes Punktes aus M besteht, nach einem Schritt erfolgreich anhalten, während ein anderer, bei dem sämtliche Nachbarschaften in $L\backslash M$ liegen, nie anhält.

Da sich ersteres nicht erreichen läßt, ohne M zu kennen, achtet man zumeist darauf, letzteres zu vermeiden, indem man sicherstellt, daß jeder Punkt von jedem anderen in einer endlichen Zahl von Schritten erreicht werden kann. Diese Bedingung ist auch stets Voraussetzung bei Untersuchungen zur Qualität dieser Verfahren [Aar89].

Zur Zeit haben sich die folgenden Varianten etabliert:

Simulated Annealing: In Annäherung an den physikalischen Vorgang des „Annealings“, des Abkühlens eines Kristalls, wird jeder bessere Zustand stets und jeder schlechtere mit einer Wahrscheinlichkeit von $e^{-\left(\frac{f_{neu}-f_{alt}}{c}\right)}$ akzeptiert[12], wobei c eine rückläufige „Temperatur“ angibt. Auch wenn für das „Simulated Annealing“ inzwischen auf Markovketten beruhende Analysen zum Laufzeitverhalten vorliegen [Aar89], so sind diese noch mit Vorsicht zu genießen, da sich Begriffe wie „polynomieller Algorithmus ohne Garantie für die Abweichung“ o.ä. finden lassen. Ebenfalls von Bedeutung ist die Geschwindigkeit, mit der der Temperatur-Parameter fällt. ELMOHAMED [ECF97] z.B. schlägt gelegentliches „Wiederaufwärmen“ vor, wobei in weiterer Analogie zu Abkühlungsprozessen und ihrer Analyse durch die statistische Physik die sogenannte „Phase Transition“-Temperatur wieder erreicht werden soll, die den empirisch beobachtbaren Übergangsbereich zwischen der Möglichkeit zu großräumigen Veränderungen und der lokaler Optimierung darstellt [HHW96].[13] CATONI [Cat98] gibt durch Einbeziehung einer nicht bestimmten Invarianten ein Verfahren an, durch das sich die Performanz verschiedener Übergangsmatrizen vergleichen läßt, ohne auf ein „sicheres“ Konvergenzverhalten zurückgreifen zu müssen. Genauere Untersuchungen zum Konvergenzverhalten dieses Verfahrens finden sich nur für den „Shop-Scheduling“-Bereich, der naturgemäß eine höhere Regularität aufweist als die automatische Stundenplanerstellung [SAW98]. CATONI bietet hierbei unter den betrachteten Arbeiten das solideste theoretische Fundament, ist aber auch darauf angewiesen, die Parameter, die in seine Theorie einfließen, experimentell zu bestimmen, wobei er sich auf das Lösen einfacher Puzzle geringer Größe beschränkt, so daß eine unmittelbare Anwendbarkeit im „Timetabling“-Bereich auch hier nicht gegeben ist und kein direkter Vorteil gegenüber dem vollständig versuchsgesteuerten Ermitteln der Parameter zu erkennen ist.

Metropolis-Prozesse sind eigentlich die Vorgänger des heute weiter verbreiteten Simulated Annealing und verwenden eine fest gewählte Temperatur. Diese Übergangsmethoden sind zumeist leichter zu analysieren, da die auftretenden Markovketten homogen bleiben.

Threshold Accepting: Hier wird ein schlechterer Punkt akzeptiert, wenn die absolute Abweichung einen fest vorgegebenen Wert nicht überschreitet. Dieses Verfahren hat sich laut DUECK [DS90] für einige Probleme dem „Simulated Annealing“ selbst in deterministischer Ausführung als überlegen erwiesen.

Zur Bestimmung des Startpunktes wird zumeist eines der anderen Verfahren verwendet, so daß das resultierende Ergebnis mindestens so gut ist, wie das von diesem ermittelte. Wiederholt man - nichtdeterministisch - die Startpunktbestimmung mehrfach (Diversifikation), so hat man die Chance verschiedene lokale Minima zu erreichen.

In der Praxis wird die Menge der zu untersuchenden Punkte zumeist dadurch eingeschränkt, daß bei der Wahl des nächsten Punktes verschiedene Invarianten eingehalten werden, wie z.B. keine Überlappungen für Lehrer. In den meisten Fällen ist die Nachbarschaft durch diese Eigenschaft bereits vollständig charakterisiert und die Geschwindigkeit der Konvergenz wird dementsprechend nur über Testläufe ermittelt. Dennoch liegt der Schwerpunkt im wesentlichen in der Zielfunktion, die laut [Sch96]

[12] Beim Minimierungsproblem.

[13] Mit dem Begriff „Phase Transition“ wird häufig auch ein entsprechender Übergangsbereich bei Problemklassen assoziiert, was auch so interpretiert werden kann, daß mit wechselnder Temperatur das Problem die Klassen „wechselt“ [Cor97].

bis zu $\frac{1}{5}$ des Programmcodes ausmachen kann. In Kapitel 6 wird der Versuch unternommen, einen ähnlichen Effekt durch Hybridisierung mit einem Graphenalgorithmus zu erzielen.
Die sog. „Tabusuche“ wird auch zu den lokalen Suchverfahren gezählt. Eine genauere Beschreibung findet sich in Abschnitt 6.3, da dieses Verfahren auch im Rahmen dieser Arbeit angewandt wurde.

4.8 Negotiation

Im KI-Bereich wird „Negotiation“ häufig verwendet, um „Meinungsverschiedenheiten“ zwischen Agenten zu schlichten. In diesem Sinne setzen auch [RW97] ein Agentennetzwerk für die automatische Stundenplanerstellung ein. Das dort beschriebene System besteht aus einem Zentralagenten, sowie mehreren externen Agenten, die jeweils einen Raum repräsentieren. Der Zentralagent behält die Übersicht in bezug auf Lehrer- und Klassen-Konflikte, die übrigen beachten lediglich ihren Raum. Eine Handelsrunde sieht nun so aus, daß der Zentralagent einen bestimmten Kurs ausschreibt, die einzelnen „Räume“ dafür Angebote abgegeben können und der Raum mit dem niedrigsten Preis den Zuschlag erhält. In einer zweiten Runde können, wenn ein Termin nicht verteilt wurde, die Agenten aufgefordert werden, festzustellen, ob durch lokale Veränderung der sie jeweils betreffenden Stunden dieser Widerspruch aufgelöst werden kann. Der Vorteil dieses Systems ist wohl in der einfachen Parallelisierbarkeit zu sehen. In Kapitel 7 wird ein ähnliches, aber nicht physikalisch verteiltes Verfahren untersucht.

4.9 Weitere Ansätze

„Neuronale Netze“: Wie in Kapitel 3.4 gezeigt wurde, kann das Suchproblem durch das Problem der minimalen unabhängigen Knotenmenge in einem Graphen modelliert werden. Dies ist einer Bearbeitung mit neuronalen Netzen unmittelbar zugänglich, indem das Netz isomorph zum Graphen aufgebaut wird und sämtliche Neuronen alle ihrer Nachbarn hemmen; für komplexere Fälle siehe z.B. [MM96]. Zur schnelleren Ermittlung eines möglichen stabilen Zustands des Netzes wird häufig das sogenannte „mean field annealing“ eingesetzt, das aber [ECF97] gemäß nur für kleinere Problemstellungen geeignet ist. Beschreibungen derartiger Versuche finden sich in [GPS92].

„Beam Search“ ist ein „Backtracking“ Verfahren, wobei jeweils eine Menge möglicher Lösungsansätze , der „Beam“, simultan untersucht wird ([CK93],[CK95b]). Die jeweils besten Kandidaten verbleiben im „Beam“. Das Verfahren stellt somit eine Kombination von Backtracking und Genetischen Algorithmen dar.

„Memetische Algorithmen“ basieren auf der Vorstellung daß Informationen von Person zu Person weitergegeben werden [BNW95]. In der konkreten Implementierung für das „Timetabling“-Problem wurde von obigen Autoren so vorgegangen, daß einem genetischen Algorithmus ein Abstiegsverfahren als Filterroutine zugeordnet wurde, so daß stets nur lokale Minima zur Weitervermehrung zur Verfügung stehen - also so, als ob vor Weitergabe der Information eine Person diese noch optimieren würde. Es bleibt allerdings fraglich, ob dieses Vorgehen die Funktion des „Mutations“-Operators nicht untergräbt.

„Goal Programming“: Einige Autoren [WW98] verwenden diesen Begriff zur Beschreibung eines Optimierungsproblems, bei dem Bedingungen zwischen Constraints und Zielfunktion ausgetauscht werden können. Dahinter verbirgt sich zumeist normale lineare Programmierung.

4.10 Unergiebige Ansätze

Eines der wohl häufigsten Probleme bei dem Versuch, einen Algorithmus für ein Problem zu finden, besteht darin, daß sehr viel Zeit damit verbracht wird, bereits begangene, aber nicht fruchtbare Wege erneut zu untersuchen. Da aber solche Fehlschläge verständlicherweise kaum publiziert werden,[14] läßt sich dies zumeist nicht verhindern. Im folgenden werden einige Versuche aufgeführt, das „Timetabling“-Problem entweder als lösbare Teilklasse des Problems der Graphenfärbung zu identifizieren oder zumindest Teilaspekte davon zu modellieren, und es wird aufgezeigt, woran diese Versuche jeweils scheitern.

4.10.1 Das „Optimum Cost Chromatic Partition Problem“

Als schon eher an der Realität der automatischen Stundenplanerstellung entsprechendes Modell bietet sich das „OCCP“ an, bei dem jeder Farbe ein Preis c_f zugeordnet wird, und es die Aufgabe ist, den Graphen so zu färben, daß die Summe über die so entstandenen Kosten aller Kanten minimal wird. Dies entspricht insofern einem annehmbaren Ansatz, als es zumeist möglich ist, die Schulstunden vollständig nach ihrer Beliebtheit zu ordnen. Eine Erweiterung stellt das „General Optimum Cost Chromatic Partition Problem“ dar, das dem „Timetabling“ bis auf die relativen Constraints entspricht; hier kann jeder Kanten-Farben-Paarung ein Gewicht zugewiesen werden. Wie Jansen [Jan97] zeigt, sind beide Probleme für fast alle Arten von Graphen und für sehr eingeschränkte Farbzahlen NP-hart. Effiziente Algorithmen existieren nur für Bäume und für Graphen mit „beschränkter Baumweite“ (treewidth). Auch DeWerra [dW95] zeigt einige auf Bäumen lösbare „Timetabling“-Probleme mit diversen Nebenbedingungen auf.

4.10.2 Bekannt gutartige Formen von Hypergraphen

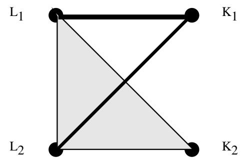
Da sich, wie oben dargestellt, „University Timetabling“ und „Examination Timetabling“ auf das Färben von Hypergraphen zurückführen lassen, stellt sich natürlich die Frage, ob bei diesen nicht eine der Teilklassen der Hypergraphen vorliegt, für die Färbungsalgorithmen bzw. Matchingalgorithmen (z.B. [CCKV96]) bekannt sind.

Eine Übersicht über die wichtigsten besonderen Klassen von Hypergraphen findet sich in [Dou95][15]. Wie aus dem dort gezeigten Diagramm hervorgeht, besitzen fast alle dieser Graphen die sog. Helly-Eigenschaft, nach der Mengen paarweise überschneidender Kanten einen gemeinsamen Knoten besitzen. Dazu sei folgendes Beispiel betrachtet: Gegeben seien zwei Lehrer und zwei Klassen. Es sollen folgende Unterrichtsstunden

[14] In der Topologie gibt es ein Buch, daß zwar keine Fehlschläge, aber immerhin die entsprechenden Gegenbeispiele aufzählt [SS78].

[15] Total unimodulare Graphen sind auch unimodular.

abgehalten werden: (L_1, K_1), (L_2, K_1), (L_1, L_2, K_2). Der zugehörige Hypergraph ist in dem folgenden Diagramm wiedergegeben und besitzt diese Eigenschaft offenbar nicht:



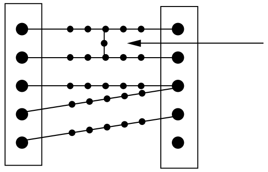
Ebenso lassen sich Beispiele konstruieren, so daß die entsprechenden Graphen nicht perfekt sind.

4.10.3 Andere Ergebnisse zu Kantenfärbungen und Matchings in Hypergraphen

Verwendbare Sätze über die Kantenfärbbarkeit von Hypergraphen gibt es in erster Linie für den Fall, daß eine Färbung zu einer Färbung des korrespondierenden vollständigen Graphen erweitert werden soll (z.B. [Dou95],[HH93],[Bro95]). Diese lassen sich hier nicht anwenden, weil die Farbenzahl dieser vollständigen Graphen wesentlich zu hoch liegt. Zudem bringt die Einführung echter Differenzierungsstunden mit verschiedenen Lehrern und Klassen mit sich, daß die Graphen nicht mehr durch die Trennung nach Objekttypen partitioniert werden können. Die Möglichkeit einer Färbung scheint bei größeren Graphen immer eher gegeben zu sein: Wie Pippenger und Spencer [PS89] zeigen, gibt es für jede Größe einer Differenzierung k und jede Abweichungsanforderung δ ein n_0 und ein δ', so daß für alle Problemstellungen mit mehr als n_0 Objekten, bei denen die minimale und die maximale Stundenzahl D zweier Objekte keinen größeren Quotienten als $1 - \delta'$ aufweisen und der Quotient aus der maximalen Zahl von Stunden, an der zwei identische Objekte beteiligt sind und D $(=: \psi)$ kleiner als δ' bleibt, die maximale Zahl der benötigten Stunden durch $(1 - \delta)D$ beschränkt ist. Hierbei tritt allerdings das Problem auf, daß ψ im „Timetabling“-Kontext nicht beliebig klein werden kann, da die maximale Zahl der Stunden in einem Kurs - in der Regel 5 - eine untere Grenze bildet.
Eine Erweiterung des Satzes von Hall auf Differenzierungen, die nur einen Lehrer, aber mehrere Klassen enthalten können, findet sich in [Hax95]. Allerdings läßt sich dies weder auf eine Färbung erweitern, noch zu einem effizienten Algorithmus umgestalten, da das Kriterium selbst wieder auf kleineren Matchings beruht.

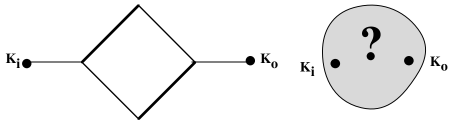
4.10.4 „PathMatching“

Die polynomielle Lösbarkeit einer gemeinsamen Verallgemeinerung der Algorithmen zum maximalen gewichteten Matching und zu Matroidschnitten, die laut Vygen [Vyg] zu den schwersten noch polynomiell lösbaren Verfahren zählen, ist das von Cunningham und Geelen [CG97] untersuchte Path-Matching. Gegeben zwei Matroide (d.h. durch den „Greedy“ -Algorithmus optimal lösbare Probleme), deren Elemente durch unabhängige Knotenmengen repräsentiert werden und ein diese verbindender Graph G, ist es Ziel, knotendisjunkte Pfade zwischen jeweils zwei Elementen zweier Basen (d.h. maximal unabhängiger Mengen) zu finden und die verbleibenden Knoten von G durch ein maximales Matching zu überdecken. Es sieht zunächst so aus, als könne man dieses Problem dazu verwenden, „Timetabling“-Probleme mit exakten Ressourcenvorgaben (d.h. = statt $\leq$) zu lösen[16], indem man, ausgehend von dem bipartiten Graphen wie er im Satz von König auftritt, die Knoten als die Elemente trivialer Matroide, in denen alle Teilmengen unabhängig sind, interpretiert und jede Kante durch einen Pfad ungerader Länge ersetzt. Verbindet man nun die beiden jeweils mittleren Knoten der Pfade mit einem weiteren Punkt, so würde eine Einbeziehung dieses Knotens in das Matching dafür sorgen, daß nur einer der beiden Pfade existieren kann. Analog könnte man n dieser mittleren Knoten mit m weiteren zu einem vollständigen bipartiten Graphen ($K_{n,m}$) kombinieren, so daß nur $n - m$ dieser Pfade erlaubt wären.



G

Ungünstigerweise gibt es aber - ähnlich zu den Netzwerkflüssen - keine Möglichkeit, die Pfade daran zu hindern, diese „Steuerkanten“ zu benutzen. Es ist natürlich möglich, eine Kante für einen Pfad undurchquerbar zu gestalten,



aber da beide Endkanten nie zu einem Matching gehören können, könnte man die Kante ohne Implikationen weglassen. Soll also „Information“ durch eine derartige Konstruktion gelangen, so müssen zwei Knoten auftreten, die nicht von allen perfekten Matchings überdeckt werden, sich aber stets gleich verhalten. Dies ist dazu äquivalent, daß es sowohl ein perfektes Matching gibt, das beide überdeckt, als auch eines, das beide nicht überdeckt. Wenn einer nicht überdeckt ist, hat der verbleibende Graph eine ungerade Knotenzahl,was dazu führt, daß ein weiterer Knoten nicht überdeckt ist. Da aber nur K_i und K_o Verbindung zum übrigen Graphen haben, ist ein perfektes Matching nur möglich, wenn dies der jeweils andere ist. Ein analoges Argument gilt für den umgekehrten Fall. Aber leider gilt:

Lemma 4.8 *Wenn in einem Graphen G mit zwei ausgezeichneten Knoten K_i, K_o, G und $G \backslash \{K_i, K_o\}$ perfekte Matchings besitzen, gibt es einen Pfad P, so daß $G \backslash P$ ein perfektes Matching hat.*

Beweis:
Man nehme die symmetrische Differenz S zwischen einem perfekten Matching mit den beiden Knoten (M_1) und einem anderen Matching ohne beide Knoten (M_2). Diese besteht, wie in Abschnitt 4.3.3.1 gezeigt, aus Pfaden und Kreisen. Da jedes der beiden Matchings ganz $G \backslash \{K_i, K_o\}$ überdeckt, hat jeder Knoten darin in der symmetrischen Differenz den Grad 2. Nur K_i und K_o haben den Grad 1 und ein Pfad hat somit seinen Endpunkt in ihnen. Da diese beiden aber die einzigen mit Grad 1 sind, muß es sich um denselben Pfad handeln. Weil zu jedem Knoten auf P eine Kante aus M_1 inzident ist, die ganz in P liegt, gibt es keine Matchingkanten von P nach $G \backslash P$. D.h. $G \backslash P$ besitzt das perfekte Matching $M_1 \cap E(G \backslash P)$.

[16] Was nicht die Lösbarkeit von $\leq$ impliziert.

Analog läßt sich kein Fall konstruieren, bei dem eine Überdeckung von K_i äquivalent ist zu einer Nichtüberdeckung von K_o, da sich durch das Hinzufügen eines Knotens K_o' und der Kante (K_o, K_o') dann ein Graph des oben geforderten Typs ergäbe.

Diese Überlegungen schließen natürlich eine generelle Anwendbarkeit des „PathMatching“ nicht aus, insbesondere, da es auch einen gewichteten Fall gibt; es wird lediglich die Unmöglichkeit dieser intuitiven Konstruktion gezeigt. Bei genauerer Betrachtung war ein Funktionieren dieses Vorgehens auch unwahrscheinlich, da es auch das Modellieren des STPs mit Abwesenheitszeiten erlaubt hätte, was - wie gesagt- nach [EIS76] $\mathcal{NP}$-hart ist.

4.10.5 Approximationsalgorithmen für Multicommodity-Flow Probleme

Wie in Abschnitt 4.3.2 dargestellt, kann das „Timetabling“-Problem auf Multicommodity-Flüsse reduziert werden, für die ein randomisierter Approximationsalgorithmus mit begrenztem Erwartungswert existiert [Hoc97]. Voraussetzung für dessen Anwendung ist aber, daß die Kantenkapazitäten logarithmisch in der Zahl der Kanten wachsen, was bei der Modellierung der automatischen Stundenplanerstellung leider nicht der Fall ist.

4.10.6 Approximationsschemata für dichte Graphen

Ist die Kantenzahl im Konfliktgraphen bzw. die Zahl der Bedingungen in der entsprechenden logischen (KNF) Formel sehr groß, und gilt es, die Zahl der Konflikte zu minimieren, wie dies z.B. beim Kurswahlproblem der Fall ist, so kann das polynomielle Approximationsschema nach Arora, Karger und Karpinski [AKK98] angewandt werden: Davon ausgehend, daß sich die optimale Zahl erfüllbarer Nebenbedingungen wie δn^k verhält, erreicht dieses Verfahren dadurch, daß es zu jedem ϵ eine konstante Abweichung von ϵn^k garantieren kann, eine multiplikative Abweichung von $\frac{OPT - \epsilon n^k}{\delta n^k} \approx (1 - \frac{\epsilon}{\delta})$, wobei δ meist kleiner gleich 1 ist.
Das Verfahren, dessen genaue Darstellung auch wegen seiner Vielschrittigkeit den Rahmen dieser Arbeit sprengen würde, basiert darauf, ganzzahlige polynomielle Programme (IPP) - in Anlehnung an ILPs - mit einer additiven Abweichung zu lösen, die exponentiell im Grad k des Polynoms ist.

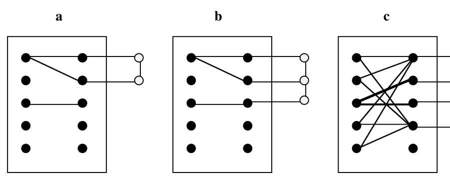
Hierzu wird zunächst eine „Probe“ des Problems logarithmischer Größe genommen, so daß diese mittels eines „Backtrackingverfahrens“ in exponentieller Zeit ihrer Größe, also in polynomieller Zeit der Gesamtgröße des Problems gelöst werden kann. Mit Hilfe der Werte dieser Probe kann ein lineares Programm bestimmt werden, dessen (nichtganzzahlige) Lösung der des Ausgangsproblems mit hoher Wahrscheinlichkeit sehr nahe kommt. Nimmt man nun immer einen Teil der Parameter des Polynoms als konstant an, so daß der verbleibende Ausdruck linear in den übrigen Parametern ist, so können diese gemäß den Rundungsschemata für lineare Programme gerundet werden [Hoc97]. Dabei ist die mögliche Abweichung bei jedem Schritt und mit hoher Wahrscheinlichkeit linear in der Größe des Problems, was dazu führt, daß sie sich jedesmal potenziert und somit am Ende eine Abweichung der Größenordnung von höchstens n^k zu erwarten ist.
Man kann dieses Verfahren allerdings nicht unmittelbar auf das Problem der Kurswahl anwenden, da jeweils eine Methode gefunden werden muß, um eine Lösung zu finden,

die die zunächst nur approximativ erfüllten Nebenbedingungen exakt erfüllt ohne Einbußen bei der Qualität hinnehmen zu müssen. Das am ehesten anwendbare Problem wäre also hier dichtes „Max2SAT", wobei sich zusätzlich noch das Problem ergäbe, daß eine Lösung ohne Stunden natürlich ebenfalls alle Nebenbedingungen erfüllen würde. Zudem bleibt in der Praxis auch beim Kurswahlproblem der Maximalgrad durch eine Konstante beschränkt, da andernfalls die Kurse jeweils aufgeteilt würden. Ein Festhalten von ϵ bei konstant gehaltenem Maximalgrad würde aber die maximale Größe des Graphen beschränken, was aus denselben Gründen abzulehnen wäre wie ein vollständiges Durchsuchen, das mit Unterstützung durch „ILP"-Verfahren auf den nächsten Computergenerationen vermutlich möglich wäre. Allerdings stellt, selbst wenn der Grad hoch genug wäre, die Laufzeit ein Problem dar, da es sich nicht um ein voll polynomielles Approximationsschema handelt und die Laufzeit mit n^{1/ϵ^2} wächst, was bereits für relativ große ϵ nicht mehr praktikabel ist.
Eine Möglichkeit, diese Algorithmen dennoch anzuwenden, bestünde darin, zuerst solange nach Heuristiken Blockungen zu bilden, bis die benötigte Dichte erreicht ist; hier könnte aber nur experimentelle Laufzeitanalyse endgültigen Aufschluß bieten.

Die ebenfalls in [AKK98] erwähnten Algorithmen für „Min-k-CUT" ließen sich zwar unmittelbar auf das UTP anwenden, da ein minimaler k-Schnitt auf dem komplementären Graphen einer Färbung entspräche, bei der möglichst wenige Kanten monochromatisch sind, aber die Voraussetzungen sind noch höher, nämlich linear wachender Grad jedes Knotens und $o(n)$ Farben, so daß ein Einsatz im „Timetabling"-Bereich nicht sinnvoll erscheint.

4.10.7 Sonstiges

Nichtbipartites Matching:
Durch den Übergang zu nichtbipartitem Matching können paarweise Gleichzeitigkeitsbedingungen modelliert werden, indem man zwei Knoten durch einen Pfad ungerader Länge verbindet, und, nach Einfügen einer ausreichenden Zahl von Hilfsknoten, jeweils versucht, perfekte Matchings zu bestimmen (a). Hierbei ergäben die „inneren" Kanten (Kasten) dann die Stunden, während ein Objekt, daß über eine „äußere" Kante überdeckt wäre, zum jeweiligen Zeitpunkt „frei hätte".
Dieses Verfahren läßt sich aber nicht auf größere Mengen von Knoten ausdehnen, da die einzelnen Konstrukte einander beeinflussen (b). Des weiteren kann nicht mehr garantiert werden, daß alle Knoten maximalen Grades überdeckt werden, da man zwei dieser Knoten mit zwei weiteren „synchronisieren" könnte, die nicht gleichzeitig gesetzt werden können.



Nichtbipartite Kantenfärbungen:
Gemäß dem Satz von Vizing(1964) kann jeder einfache Graph G mit $\Delta(G)+1$ Farben kantengefärbt werden [Die96]. Um dies effektiv im „Timetabling" einsetzen zu können, müßte dies aber auf Multigraphen übertragbar sein. Wie man aber an einem K_3 mit je n Kanten zwischen den Knoten sieht, bei dem $3n$ Farben benötigt werden, aber

nur ein Maximalgrad von $2n$ gegeben ist, kann hier die Differenz zwischen χ' und Δ beliebig groß werden. Das Edge-Coloring-Problem ist als $\mathcal{NP}$-vollständig bewiesen worden, dennoch wird im allgemeinen angenommen, daß es einen Algorithmus gibt, der nur $\chi'+1$ Farben benötigt [CR98]; dieser wurde bisher jedoch noch nicht gefunden. Analog zum gerade beschriebenen Matchingansatz könnten dann die Termine zweier Objekte x, y mit höchstens einer Ausname synchronisiert werden, indem man sie mit $\chi' - \max(\delta(x), \delta(y))$ zusätzlichen Kanten verbindet, was im bipartiten Fall nur für Objekte verschiedenen Typs möglich ist.

Master-Slave Matchings:
Wie von DeWerra et al. [dWE96] gezeigt wurde, führt die Nebenbedingung, daß einige (wenige!) Stunden parallel liegen müssen dazu, daß das School-Timetabling-Problem $\mathcal{NP}$-hart wird. Ein schwächere Forderung wäre es zu verlangen, daß bestimmte Lehrer oder Klassen parallel tätig sein sollen. Untersuchungen dazu wurden von [Hef97] vorgenommen, es konnten aber nur Fälle als „vielleicht in $\mathcal{P}$" identifiziert werden, bei denen der Graph, der Implikationen der Form „wenn A Unterricht hat, dann auch B" modelliert, aus isolierten Kanten besteht.

Matchings mit Partitionsbeschränkungen:
Eine Möglichkeit, zumindest die Raumbedingungen in das STP zu integrieren, wäre es, bei der jeweiligen Bildung von Paarungen vorschreiben zu können, daß von bestimmten Mengen von Kanten nur jeweils eine bestimmte Anzahl verwendet werden darf. Dieses Problem ist im allgemeinen ebenfalls $\mathcal{NP}$-vollständig und nicht konstant approximierbar, wohl aber für eine bestimmte Klasse von Graphen, die sogenannten Pfaffschen Graphen in $\mathcal{P}$. Es wurde hier nicht eingehend untersucht, ob die in der automatischen Stundenplanerstellung auftretenden Graphen tatsächlich nicht pfaffsch sind, aber sämtliche der in [Lec87] angegebenen einfach nachprüfbaren, aber nur hinreichenden Kriterien treffen nicht zu. Selbst wenn die Graphen pfaffsch sein sollten, so hängen beim „Timetabling" Zahl und Größe der Partitionen zumeist linear von der Gesamtproblemgröße ab, was bei den Verfahren in [Lec87], die eine konstante Zahl von Partitionen verlangen, ebenfalls exponentielle Laufzeit hervorruft.
Lediglich bei Beschränkung auf ein STP mit wenigen Raumtypen könnte dieses Verfahren anwendbar sein.

Gradientenverfahren:
Der Übergang zu differenzierbaren Funktionen z.B. auf dem Einheitswürfel dürfte kaum Vorteile bringen, da bei sinnvoller Fortsetzung der auf den Ecken gegebenen Zielfunktion, die hohe Symmetrie des Problems die Gradienten in der Nähe des Mittelpunktes nahezu Null werden ließe. Es verblieben dort also nur die Auswirkungen der schwachen Nebenbedingungen, die ihr Minimum normalerweise nicht in der Nähe eines Minimums der harten Bedingungen haben.

4.11 Zusammenfassung

Während die älteren Verfahren noch sehr spezifisch für die automatische Stundenplanerstellung waren, setzen sich heute mehr und mehr allgemeine Optimierungsverfahren durch, wobei nur noch die Wahl einiger Funktionen direkt vom Sachkontext beeinflußt wird. Es ist bisher nicht gelungen, eine Eigenschaft des „Timetabling" zu finden, die es für einen bestimmten Algorithmus prädestinieren würde, und die einzige Regelmäßigkeit scheint zu sein, daß in Abständen von vielleicht fünf Jahren weite-

re Problemklassen mit Ganzzahliger Linearer Programmierung gelöst werden können bzw. dies versucht wird.
Eine Leistungsbewertung der Heuristiken ist kaum möglich, und Wren und Kwan [WK99] führen z.B. lediglich an, daß das von ihnen verwandte Verfahren zm Erstellen von Busplänen in vielen tausend Versuchen nie versagt habe. Versuche, durch Einschränkungen der Problemstellung zu neuen Lösungen zu gelangen bzw. Versuche, bestehende Verfahren zu erweitern, scheitern, wie an einigen Beispielen demonstriert wird, zumeist an Komplexitätsgesichtspunkten, woran deutlich wird, wie genau die bestehenden Methoden die Grenzen des Möglichen erreichen.
Übersichten zu verschiedenen Lösungsverfahren finden sich in fast allen Beiträgen zur automatischen Stundenplanerstellung und natürlich wieder in [Sch95].

Kapitel 5

Grundlegende Architektur, Modell und Algorithmen

Um möglichst effektiv, aber auch realitätsnah verschiedene Lösungsverfahren testen zu können, wurde zunächst ein Programm entwickelt, das die Charakteristika besserer Satzprogramme aufweist und dieses so konzipiert, daß problemlos eine beliebige Zahl von Verfahren integriert werden kann. Somit werden diese alle auf das gleiche Modell angewandt und später nach denselben Kriterien bewertet. Dieses Kapitel beschreibt diejenigen Teile des Rahmenprogramms, die für die Besprechung der Verfahren notwendig sind. In Kapitel 8 wird auf die Implementierungsdetails, aber auch auf besondere Eigenschaften der Satzhilfen eingegangen.

5.1 Problemstellung

In dieser Arbeit wurde als Problem das „University Timetabling" mit festen Zeitstunden gewählt, da dieses genau den angestrebten Einsatzgebieten (vgl. Kapitel 1) entspricht.
Um einige Nebenbedingungen einfacher darstellen zu können, wurde allerdings die Sichtweise etwas geändert: Zunächst wird das Ganze nicht mehr aus Sicht der Curricula, sondern aus Sicht der Kurse betrachtet. Ein Kurs, im folgenden Gruppe genannt, „besteht" aus den „Curricula", an denen er beteiligt ist, d.h. eine Gruppe enthält die Schüler, Lehrer und Räume, die für die entsprechende Schulstunde relevant sind. Dieses Problem ist entspricht somit in der Grundform einem mehrfachen „Set-Packing" bzw. einem mehrdimensionalen „Bin-Packing" Problem.

Da aber nicht alle Ressourcen „statisch" sind, wird das Ganze so weiter relaxiert, daß jede Gruppe aus Ressourcenanforderungen der Form „3 Biologieräume" besteht. Die „statischen" Ressourcen gehen darin auf, indem hier stets die volle Zahl der zur Verfügung stehenden Objekte angefordert wird („1 Herr Meier"). Hierbei wird explizit vorgesehen, daß sich diese Anforderungen in beliebiger Weise überschneiden können. D.h. es kann durchaus auch vorkommen, daß ein Raum sowohl Biologie- als auch Chemieraum sein kann. Die Aufgabe besteht also in der Auswahl der flexiblen Ressourcen und dem kollisionsfreien Zuordnen zu den Zeitstunden. Es ist hier nicht sinnvoll, dies in mehrere Probleme aufzuteilen, da stets Beziehungen zwischen den flexiblen Objekten und den Zeitstunden gegeben sind. Als Nebenbedingungen werden

- Abwesenheitszeiten,
- bevorzugte Zeiten,
- bevorzugte Ressourcen,
- und Springstundenminimierung

angenommen. Kompaktheit und gleichmäßige Verteilung wurden, wenn möglich, fest integriert, können also nicht deaktiviert werden. Die Modellierung ausgefallener Anforderungen, wie geographischer Bedingungen wurde nicht vorgesehen, siehe hierzu auch Kapitel 10. Wie auch bei Cooper und Kingston [CK95b] könnte das System auch zur Lehrerverteilung verwendet werden, dies wird aber von den Lösungsverfahren nicht gesondert unterstützt.
Im folgenden wird das hier beschriebene Problem als „RSTP" („real school timetabling problem") bezeichnet.

5.2 Modell

5.2.1 Das betrachtete Problem als ILP

Als Eingabe seien gegeben: Eine Menge von Objekten $O = \{o_1 \dots o_n\}$, eine Menge von Mengen $\mathbb{P}(M) \supset G = \{g_1, \dots, g_m\}$, T als Menge von Timeslots und eine Menge von Kursen $K = \{(k_1, v_1), \dots (k_r, v_r)\} \subset \mathbb{N}^m \times \mathbb{N}$, wobei jeder Koeffizient von k angibt, wieviele Objekte der entsprechenden Gruppe benötigt werden, und v_j die Vielfachheit des jeweiligen Kurses angibt. Der Einfachheit halber seien die Gruppenbeziehungen durch eine Matrix $B \in \{0,1\}^{mn}$ kodiert.

Dann stellt sich das Suchproblem in der folgenden Form dar[1]: Da es der Verständlichkeit dient, werden die Variablen mit den entsprechenden Objekten anstatt ihrer Indices in den jeweiligen Mengen indiziert: Finde

$$\mathcal{X} = (x_{k,s,t}) \in \{0,1\}^{|K|(\max_j v_j)|T|}$$

und

$$\mathcal{Y} = y_{k,s,n,t,f} \in \{0,1\}^{|K|(\max_j v_j)|O||T||G|}$$

mit

$$\begin{aligned}
\forall (k,v) \in K, \forall s \leq v : \sum_{t \in T} x_{k,s,t} &= 1 && (5.1)\\
\forall (k,v) \in K, \forall s \leq v, \forall t \in T : \sum_{o \in O} y_{k,s,o,t,*} &\geq x_{k,s,t} k && (5.2)\\
\forall (k,v) \in K, \forall o \in O, \forall t \in T : y_{k,s,o,t,*} &\leq B^t e_o && (5.3)\\
\forall o \in O, \forall t \in T : \sum_{(k,v) \in K} \sum_{s<v} \sum_{g \in G} y_{k,s,o,t,g} &\leq 1 && (5.4)
\end{aligned}$$

$\mathcal{X}$ gibt an, zu welchen Zeiten eine Stunde eines Kurses liegt. $\mathcal{Y}$ beschreibt, welche Objekte zu welchen Zeitpunkten welchem Kurs in welcher Funktion zugeordnet werden.

[1] Sind bei Indizierungen weniger Indizes angegeben als in der Definition (*), ist der Vektor gemeint, der entsteht, wenn man alle Werte des entprechenden Parameters einsetzt. Auf Vektoren bezogen sind die Relationszeichen stets komponentenweise zu verstehen.

- Bedingung 5.1 gibt an, daß jede Stunde eines Kurses genau einmal stattfindet.
- 5.2 sorgt dafür, daß zu dieser Stunde dem Kurs ausreichend Ressourcen zugeordnet werden.
- Bedingung 5.3 sorgt dafür, daß die Ressourcen nur zu Aufgaben verwandt werden, für die sie geeignet sind. Hier sei e_o der charakteristische Vektor für o aus $\{0,1\}^{|O|}$.
- 5.4 legt fest, daß jedes Objekt pro Zeiteinheit höchstens einmal verwendet werden darf.

Das oben vorgestellte Modell erlaubt es noch, mehrere Stunden eines Kurses, der ausschließlich aus flexiblen Ressourcen besteht, derselben Zeitstunde zuzuordnen. Dies kann durch

$$\forall (k,v) \in K \forall t \in T : \sum_{s \leq v} x_{k,s,t} \leq 1$$

unterbunden werden. Dies wird hier nicht explizit gefordert; einige der Lösungsverfahren produzieren aber nur Lösungen, die dies beachten. Da es sehr unwahrscheinlich ist, daß dieser Fall vorkommt, wird darauf aber nicht weiter eingegangen.
Daß diese Konstruktion so relativ kompliziert ist, liegt an der Anforderung, die flexiblen Ressourcen nur zu bestimmten Zeiten auszuwählen. Ließe man hier quadratische Constraints zu, so könnte man bei y die Zeitdimension sparen. Bei einer durchschnittlichen Schule von 30 Klassen, 40 Lehrern, 40 Räumen, 35 Zeitstunden, im Schnitt 3 Zeitstunden pro Kurs, 10 Kursen pro Klasse und 100 Gruppen ergibt sich eine Zahl von 300 Millionen Variablen. Entfernte man das Problem flexibler Ressourcen vollständig, wären es nur ca. 31500, d.h. ca. um einen Faktor $|G||O| \approx 10000$ weniger. Eine ebenfalls mögliche Art der Darstellung ist die, jeweils statt $y_{k,s,n,t,g}$ nur $y_{n,t}$ zu verwenden und als weitere - nur mit ähnlichem Aufwand wie oben darstellbare - Nebenbedingung zu verlangen, daß es eine Abbildung von $N \times T$ nach $K \times k_v \times G$ gibt, so daß die obigen Bedingungen erfüllt sind. Im Zusammenhang mit Abschnitt 8.4.3 erkennt man, daß bei Festhalten von $\mathcal{X}$ das Bestimmen von $\mathcal{Y}$ durch ein Matchingverfahren geschehen kann. Daher wird zumeist darauf verzichtet, $\mathcal{Y}$ mit zu verwalten, so daß auch hier die Variablenzahl - allerdings auf Kosten der Komplexität - sehr stark verringert werden kann.

5.2.2 Zielfunktion

Allen nachfolgend aufgeführten Verfahren wurde eine einheitliche Zielfunktion zu Grunde gelegt. Aus der Menge der Nebenbedingungen konnten hier natürlich nur die einbezogen werden, für die ein Modellierungsschema vorgesehen ist. Insbesondere können also Anforderungen, welche Stunden nicht aufeinanderfolgen dürfen o.ä., nicht kodiert werden (siehe hierzu Kapitel 10). Die Zielfunktion wurde in mehreren Schichten organisiert, so daß sie am besten als lexikographische Ordnung auf einem Vektor von Zielwerten angesehen werden kann. Dies läßt sich natürlich auch realisieren, indem jeder Koeffizient des Vektors mit dem Maximalwert des nächsten Koeffizienten gewichtet wird (vgl. Kapitel 6.6.5.1), dies wurde hier aber der Übersichtlichkeit halber nur in der Kurzanzeige während des laufenden Optimerungsprozesses getan. Die oberste Schicht repräsentiert nochmals die Kollisionsfreiheit, da diese bei einem der Verfahren nicht als Nebenbedingung auftritt.
Im schulischen Umfeld läßt sich das Konzept „Fach“ nicht umgehen. Daher sei ψ :

$K \to F$ eine Abbildung, die jedem Kurs das zugehörige Fach zuordnet. Des weiteren sei die Menge der Zeiteinheiten in Tage zerlegt: $T = D_1 \dot\cup \ldots \dot\cup D_r$ und dem Zeitverlauf entsprechend angeordnet. Dann baut sich das Ganze wie folgt auf:

Kollisionen	$\lvert\{(o,t) \in O \times T \lvert \sum_{(k,v)\in K, s<v, g\in G} y_{k,s,o,t,g} > 1\}\rvert$
Maximale Stundenzahl einer Fach-Klassen-Paarung	$\lvert\{(o,f,d) \lvert \sum_{t\in D_d,(k,v)\in K,\psi((k,v))=f,s<v,g\in G} y_{t,s,k,o,g} \geq 3$ $\wedge\, o$ ist Klasse$\}\rvert$
Fach-Klassen-Springstunden	$\lvert\{(o,f,t) \lvert \exists t' < t < t'' \sum_{(k,v)\in K,\psi((k,v))=f,s<v,g\in G} y_{k,s,t',o}, g \geq$ 1 $\wedge \sum_{(k,v)\in K,\psi((k,v))=f,s<v,g\in G} y_{k,s,(t),o,g} = 0$ $\wedge \sum_{(k,v)\in K,\psi((k,v))=f,s<v,g\in G} y_{k,s,(t''),o,} \geq 1 \wedge o$ ist Klasse$\}\rvert$
Klassenspringstunden	$\lvert\{(o,t) \lvert \exists t' < t < t'' \sum_{(k,v)\in K,s<v,g\in G} y_{k,s,t',o,g} \geq 1$ $\wedge \sum_{(k,v)\in K,s<v,g\in G} y_{k,s,(t),o,g} = 0$ $\wedge \sum_{(k,v)\in K,s<v,g\in G} y_{k,s,(t''),o,g} \geq 1 \wedge o$ ist Klasse$\}\rvert$
Bevorzugte Zeitpunkte	$\sum_{(k,v)\in K,s<v,t\in T} \frac{\sum_{o\in O,g\in G} w(o,t) y_{k,s,o,t,g}}{\lvert\{(o',g') \lvert y_{k,s,o',t,g'}=1 \wedge w(o,t)\neq 0\}\rvert}$
Bevorzugte flexible Objekte	$\sum (k,v) \in K, s < v, t \in T$ $\frac{\sum_{o1,o2\in O,g_1,g_2\in G} y_{k,s,o_1,t,g_1} y_{k,s,o_2,t,g_2} w(o_1,o_2)}{\lvert\{(o',g',o'',g'') \lvert w(o',o'')\neq 0 \wedge y_{k,s,o',t,g'} y_{k,s,o'',t,g''}=1\}\rvert}$
Lehrerspringstunden	$\lvert\{(o,t) \lvert \exists t' < t < t'' \sum_{(k,v)\in K,s<v,g\in G} y_{k,s,t',o,g} \geq 1$ $\wedge \sum_{(k,v)\in K,s<v,g\in G} y_{k,s,(t),o,g} = 0$ $\wedge \sum_{(k,v)\in K,s<v,g\in G} y_{k,s,(t''),o,g} \geq 1 \wedge o$ ist Lehrer$\}\rvert$

Wobei w sowohl das Verhältnis zwischen Objekten, als auch das von Objekten zu Zeiteinheiten angibt. Diese Bewertung kann neutral sein. In diesem Falle wird ein Wert aus den Relationen der betreffenden Übermengen ermittelt (s. Abschnitt 8.4). Der Übersichtlichkeit halber wurde in der Tabelle weggelassen, daß t' und t'' zum selben Tag gehören müssen, wie t.

Der Unterschied zwischen Klassen- und Lehrerspringstunden muß vom Benutzer angefordert werde (s. Paragraph 8.2.3), da zunächst kein Unterschied zwischen den Objekttypen vorgesehen ist. Die Normalisierung der Gruppen zueinander erfolgt unter Ausschluß der Paarungen mit Wert 0, da deren große Zahl Veränderungen in diesem Bereich fast völlig nivellieren würde (vgl. Abschnitt 8.4.2).
Die letzte Subfunktion wird in der Implementierung allerdings nicht exakt so ausgewertet wie hier beschrieben, sondern es fließen nur diejenigen Paare ein, bei denen das eine ein „statisches“ und das andere ein „dynamisches“ Objekt ist, Die Beziehung zwischen den „festen“ Objekten ergäbe ohnehin nur eine Konstante, die für alle Lösungen gleich ist; die Hinzunahme der Beziehungen zwischen „flexiblen“ Objekten ließe selbst die Bestimmung von $\mathcal{Y}$ bei feststehendem $\mathcal{X}$ $\mathcal{NP}$-hart werden (s. Unterkapitel 3.6.1). An einigen Stellen, an denen die Geschwindigkeit der Auswertung von besonderer Bedeutung ist, wird hier nur festgestellt, wieviele dieser Ressourcen verfügbar sind.

5.3 Gesamtaufbau

5.3.1 Stammdatenverwaltung und Mengensystem

Wegen der Verschiedenheit der möglichen Einsatzgebiete wurde, was Bezeichnungen und Objekttypen angeht, dem Benutzer maximale Freiheit belassen: Es gibt keine Festlegung auf Lehrer und Klassen etc., sondern es kann eine beliebige Zahl von Objekttypen erzeugt werden, die mit (bis zu sechs) verschiedenartigen Attribut-Typen ausgestattet werden können. Jedes dieser Merkmale kann entweder aus einem Text, einer Zahl oder einem Boolschen Wert bestehen. Die Gesamtheit der Objekte und Attribute wird im folgenden mit „Stammdaten" bezeichnet. Zeiteinheiten werden auch als Objekte behandelt, wobei allerdings in der momentanen Implementierung kein Unterschied zwischen verschiedenen Typen von Zeiteinheiten vorgesehen ist (vgl. Kapitel 10).

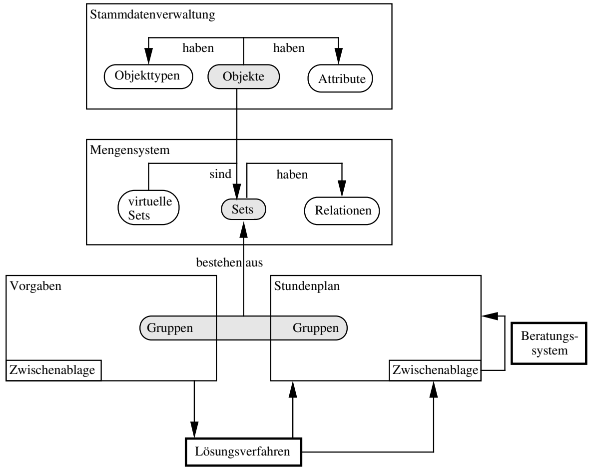
Anhand dieser Attribute und durch Auswahl von Hand werden die Stammdaten dann zu Mengen zusammengefaßt. Damit folgt dieser Teil des Aufbaues im wesentlichen dem AUGUST-System von Burke et al. [BJKW96]. Um das Prinzip eines „Schulfaches" mit entsprechenden Anforderungen, was Räume, Zeiten etc. angeht, zu modellieren, wurden leere Mengen, d.h. Mengen, die kein Objekt beinhalten, zugelassen. Diese Mengen stehen somit in unbegrenzter Zahl zur Verfügung, fließen aber in die Parameterbestimmung ebenso ein wie andere Mengen. Die Mengen wiederum können zu Übermengen vereinigt oder auch im Sinne einer Mengenbildung zusammengefaßt werden, wobei sich dieser Unterschied allerdings nur auf die spätere Art der Darstellung auswirkt. Von der ursprünglichen Idee, Anforderungen wie „1 Tag" spezifizieren zu können, mußte Abstand genommen werden, da dies wiederum einige weitere Teilprobleme hätte komplexer werden lassen. (s. Abschnitt 3.6).
Das Mengensystem stellt Funktionen zur Verfügung, die es erlauben, den Inhalt einer Menge ohne Mehrfachnennungen auszulesen, so daß es kein Problem darstellt, mehrere konkurrierende Semantiken zu erzeugen. So wäre z.B sowohl die Amtshierarchie als auch zusätzlich eine soziale Ordnung auf den Objekten möglich.

5.3.2 Gruppen

Um die Unterrichtsverteilung bzw. den fertigen Stundenplan zu beschreiben, können dann aus den Mengen Gruppen gebildet werden, wobei jede Gruppe mit einer Zahl, die angibt, wie oft diese zu wiederholen ist versehen werden kann. Jeder Menge wird eine Zahl zugeordnet, die angibt, wie viele Objekte der jeweiligen Menge an der Gruppe beteiligt sein sollen. Zusätzlich kann angegeben werden, daß bei „dynamischen" Ressourcen bei jeder Wiederholung der Gruppe das selbe Objekt verwandt werden soll. Dieser Teil des Systems ist somit nahezu identisch mit dem von Cooper und Kingston [CK93], wurde jedoch, da das Paper erst nach Fertigstellung der entsprechenden Strukturen vorlag, selbständig entwickelt. Hier kann dann die Fachzuordnung mittels Hinzufügen der leeren Mengen geschehen.

Alternativ wäre es denkbar gewesen, bei der Konstruktion des Mengensystems Vielfachheiten zuzulassen, was es z.B. erlaubt hätte, fertige „Beispiel-Gruppen" zu erstellen und diese dann in anderen Gruppen wie „Makros" anwenden zu können. Es ist aber auch bei diesem System, das Vielfachheiten nur in der obersten Ebene der Hierarchie bereitstellt, schwierig, als Benutzer den Überblick zu behalten.

Unterrichtsverteilung und Stundenplan bestehen je aus einer Liste von Gruppen, wobei beiden eine jeweils eigene Zwischenablage[2], vgl. Abschnitt 8.2.6, angegliedert ist. Der Zwischenspeicher des Stundenplans dient hierbei zusätzlich zur Aufnahme von Stunden, die der entsprechende Lösungsalgorithmus nicht verplanen konnte und als Basis für die interaktive Unterstützung. Somit präsentiert sich die Architektur des Systems in der folgenden Form:



Die Gruppen im fertigen Plan besitzen keine Vielfachheiten mehr und enthalten statt der Mengen jeweils die ausgewählten Objekte. Die Struktur ist an zwei Stellen redundant ausgelegt: Zunächst ist es so, daß mit Ausnahme der Unterscheidung zwischen Zeitobjekten und konkreten Objekten sämtliche Verfahren „oberhalb des Mengensystems" nur noch mit Mengen operieren und nicht mit konkreten Objekten, die nun als Menge der Größe 1 angesehen werden. Zweitens könnte man die Mengen ganz weglassen und die Objekte direkt in den Gruppen angeben.
Diese beiden zusätzlichen Schichten erlauben es, das System den jeweiligen Bedürfnissen anzupassen, so daß sich das Ganze nach deren einmaliger Einstellung, an der Oberfläche als schulisches Stundenplanprogramm, als FH-Planungsprogramm oder auch als Dienstplanprogramm präsentiert, das der Benutzer mit Hilfe der Dialoge für „Objektbearbeitung" und „Bewertungsverwaltung" benutzen kann.

5.4 Wahl der Algorithmen

Bei den im vorangegangenen Kapitel dargestellten Algorithmen neueren Datums wurde überwiegend nur das Suchproblem, nicht aber das Optimierungsproblem betrachtet. Von einer optimalen Lösung des Suchproblems aus könnte zwar versucht werden, durch den Wechsel zu anderen optimalen Lösungen die schwachen Nebenbedingungen zu optimieren, aber dies scheitert zumeist daran, daß diese Algorithmen nicht gezielt zu einer anderen Lösung geführt werden können bzw. das beständige Reevaluieren zu aufwendig wäre. Zudem entspricht bereits die einfachste Form derartiger Veränderungen, das Vertauschen von Zeitstunden, einer beweisbar nichtapproximierbaren Variante des

[2] „Copy-Puffer"

Problems des Handlungsreisenden.

Schon nach Implementierung der ersten Versuche stellte sich heraus, daß die Betrachtung eines einzelnen Verfahrens nicht ausreichend ist, da kein zuverlässiges Maß für die Schwierigkeit der Probleme gegeben ist, und dessen Funktionsfähigkeit daher nicht unabhängig bewertet werden kann. Das in Kapitel 7 beschriebene Verfahren lieferte, nachdem es auf einem kleinen Testfall annehmbar funktioniert hatte, erstmals auf ein reales Problem angewendet überwiegend schlechtere Ergebnisse, was im scheinbaren Widerspruch zu den Beobachtungen des vorangegangenen Praktikums stand, bei dem für größere, realistisch scheinende Testfälle stets funktionierende Pläne erzielt werden konnten.
Um diesen Effekt näher zu untersuchen, wurde das in Kapitel 6 beschriebene Verfahren implementiert, das, wenn man die Iterationszahl auf 1 beschränkt und das Verfahren auf ein reines STP anwendet, das gleiche Verhalten zeigen würde wie der im Praktikum betrachtete Algorithmus, durch zahlreiche Modifikationen dennoch aber auch in der Lage ist, die komplizierteren Problemstellungen des RSTP zu verarbeiten.
Da dieses zunächst das gleiche Verhalten zeigte, war hier ein prinzipielles Umdenken notwendig, da somit nicht mehr wie beim STP die Optimierung im Vordergrund stehen konnte, sondern zunächst nach einer einfachen Lösung gesucht werden mußte.

Der erste hier genauer untersuchte Algorithmus verbindet daher - wie zur Zeit im „Timetabling“-bereich häufiger versucht - zwei ältere Methoden miteinander, da diese in der Lage sind, relativ komplexe Nebenbedingungen zu berücksichtigen. Als grundlegendes Verfahren wird die von Glover 1989 eingeführte Tabusuche [Sch96] verwendet, da sie unter allen lokalen Suchverfahren am ehesten für das Problem der automatischen Stundenplanerstellung geeignet scheint. Um einen geschickteren Wechsel zwischen lokalen Optima zu erlauben, wird die Tabusuche dann mit einer gewichteten Variante des Algorithmus nach König hybridisiert, wobei diese von einem Reparaturschema unterstützt wird.

Der zweite Algorithmus stellt eine völlige Neuentwicklung dar, die die Vorteile von Metaheuristiken mit denen genetischer Algorithmen zu verbinden sucht. Ausgehend von der Beobachtung, daß bei einer Kodierung des UTP als GA zur Beschreibung einer Species bei einer mittleren Schule mindestens ca. 600 Kilobyte benötigt werden, erscheint es unrealistisch, mehrere hundert dieser „Specimen“ effektiv verwalten und insbesondere auch gezielt mutieren zu wollen, auch wenn ein Speicheraufwand von 1 Mbyte heute natürlich keine Grenze mehr darstellt. [3]

Daher wird hier jeweils nur mit einem Stundenplanmodell gearbeitet, dessen Kodierung aber auf viele Einzelentitäten verteilt. Jedes dieser Objekte - im weiteren als Agenten bezeichnet - versucht, das ihm zugeteilte Problem optimal zu lösen. Da sich die jeweils benötigten Ressourcen aber überschneiden, kommt es zu Konflikten zwischen den Optimierungsbestrebungen der einzelnen Agenten. Diese sollen durch ein Handels - und Auktionssystem bereinigt werden. Dabei wird insbesondere einen frei handelbaren Wert (Geld) mit einbezogen, was es nach [SS66] ermöglicht, gewisse lokale Minima zu verlassen. Neben dem Suchen brauchbarer Voraussetzungen für eine solidere theoretische Grundlage als sie Bachem [BHM96] bietet, wurde eine erste Implementierung vorgenommen und deren Performanz im Vergleich mit dem obigen Verfahren bewertet.

[3] In [Laj95] wurde mit Speicheranforderungen im Gigabyte - Bereich gearbeitet, um ein größeres Problem der automatischen Stundenplanerstellung zu behandeln.

Kapitel 6

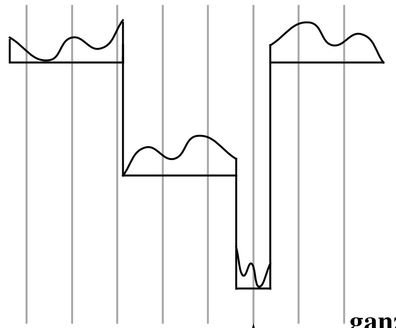
Verfahren 1: Tabusuche

6.1 Allgemeine Betrachtungen zum Lösungsraum

Im weitaus größten Teil der Literatur wird das Timetabling Problem analog zu seiner Darstellung als ILP kodiert. In den jeweiligen Implementierungen werden auch häufig Vektoren ganzer Zahlen verwandt, da somit die Dimension erheblich reduziert werden kann. Die so kodierten Probleme sind damit aber LP-Methoden vollkommen unzugänglich, da Ungleichheits-Nebenbedingungen auftreten.
Läßt man beim ILP zunächst alle Nebenbedingungen weg, so erhält man einen Hyperwürfel der Dimension $|\text{Gruppen}| \times |\text{Zeitstunden}|$. Bereits die Anforderung, daß genau die gewünschte Zahl von Stunden zu setzen ist, reduziert die Dimension des Lösungsraums derart um 1, daß ein einer Lösung auf dem Hyperwürfel benachbarter Punkt unmöglich gültig sein kann. Dementsprechend ist bei den meisten Verfahren nicht das Umschalten eines einzelnen Bits, sondern das gleichzeitige Ändern eines Bits von Null zu Eins und eines anderen von Eins zu Null die Basisoperation. Da zusätzlich die Stunden pro Gruppe konstant bleiben müssen, reduziert sich dies sogar auf das „Verschieben" einer Gruppe auf einen anderen Zeitpunkt.
Da die diskrete Zielfunktion nur auf den ganzzahligen und somit am Rande liegenden Punkten definiert ist, stellen abgesehen von der Ganzzahligkeitsforderung Konvexitätsüberlegungen kein Problem dar, da die konvexe Hülle einer beliebigen interpolierten Zielfunktion stets an diesen Eckpunkten mit der Funktion selbst übereinstimmt.

Faltet man den Hyperwürfel auseinander, wobei hier wie bei einem Gray-Code stets nur zwischen benachbarten Punkten gewechselt werden soll, so stellt sich die „Oberfläche" der Zielfunktion in etwa wie folgt dar: Auf die von einer „Bestrafungs-" Funktion zur Erzwingung der harten Bedingungen gebildeten Stufen ist schwächer die Zielfunktion der weichen Nebenbedingungen aufmoduliert. Da sich harte und weiche Anforderungen zumeist widersprechen, ist eher damit zu rechnen, daß deren Gradient von den „Löchern" der Bestrafungsfunktion wegführt, als zu diesen hin.
Wie schon in Unterabschnitt 4.6 erwähnt, tritt bei insgesamt k Zeitstunden jede Lösung der harten Bedingungen $k!$ mal auf, was aber die Menge der Lösungen im Hyperwürfel nicht dicht[1] werden läßt, da jede nicht gültige Belegung ebenfalls $k!$ mal auftritt.

[1] In Anlehnung daran, aber nicht im eigentlichen mathematischen Sinne.



6.2 Wahl der Heuristik: Die Tabuliste als notwendiges Element

Wie obige Überlegungen und auch die in Abschnitt 4.6 zeigen, stellt die hohe Symmetrie der harten Nebenbedingung, die aber wegen der weichen Nebenbedingungen nicht durch Wahl einer anderen Kodierung beseitigt werden darf, ein wesentliches Problem dar. Die einzige gängige und auch im „Timetabling"-Bereich eingesetzte Metaheuristik, die in der Lage ist, dieses Problem zu bewältigen, ist die Tabusuche, da die Tabuliste so gestaltet werden kann, daß sie permutierte Wiederholungen genauso ausschließt wie vollständige. STÜBER[Stü97] bezeichnet Tabusuche ebenfalls als das einzig geeignete Verfahren für größere Probleme, zieht aber direkte Heuristiken noch vor.
Die meisten Implementierungen der Tabusuche (vgl. [Dow97]) ignorieren diesen Symmetriezusammenhang dennoch und verwenden überwiegend direkte Kriterien, z.B. daß eine bestimmte Schulstunde zu einem bestimmten Zeitpunkt stattfindet.

In dieser Arbeit wurde bei der Konstruktion der Tabuliste dementsprechend ein anderes System verwandt: Sobald eine Schulstunde aus einer konkreten Zeitstunde entfernt wird, wird ein Paar, bestehend aus einem Verweis auf sie und eine zufällig gewählte andere Schulstunde, die in derselben Zeitstunde liegt, in die Tabuliste eingetragen. Die Tabubedingung ist nun, daß keine Stunden der beiden zugehörigen Kurse mehr zusammengelegt werden dürfen. Diese Orientierung nach Kursen und nicht nach Stunden fördert ein gleichartiges Verhalten zu einem Kurs gehöriger Stunden, was prinzipiell Blockungen („time-coherence" [CK93]), d.h. das Parallellegen mehrerer Stunden zweier Kurse fördert, ohne es allerdings hart zu erzwingen.

Des weiteren wird in [KMSV94] gezeigt, daß mittels direkter Abstiegsverfahren für k-Sat Probleme eine Approximationsgüte von 2^{-k} erreicht werden kann. Da bei geeigneter Wahl der „Aspirationsfunktion" (s. unten) die Tabusuche zu Beginn mit dieser einfachen Form der lokalen Suche identisch ist, könnte diese Schranke somit erzwungen werden, wenn man die Zielfunktion analog zu [KMSV94] modifiziert. Wenn man aber annimmt, daß die Differenzierungsgröße in $\mathcal{O}(\log n)$ wächst, erreicht man damit also bereits kein besseres Ergebnis als der triviale Algorithmus.

6.3 Tabusuche als allgemeines Optimierungsverfahren

Auch wenn im vorangegangenen Kapitel und in Abschnitt 4.7 bereits einige Aspekte vorweggenommen wurden, soll in diesem Kapitel ein kurzer Überblick über die Prinzipien der Tabusuche gegeben werden.
Im Gegensatz zu anderen lokalen Suchverfahren untersucht die Tabusuche nicht einen zufällig gewählten Punkt der Nachbarschaft, sondern die gesamte bzw. eine ausgewählte Teilmenge der Nachbarschaft und geht dann zu deren günstigsten Punkt über. Auf diesem Wege werden sehr schnell lokale Minima gefunden, aber das Verlassen eines solchen wird nahezu unmöglich. Um dieses „Kreisen“ zu verhindern, wird eine sog. Tabuliste geführt, in der „verbotene Operationen“ vermerkt werden. Im allgemeinen werden dort Charakteristika der letzten Operationen gespeichert und ein Vorgang wird genau dann als nicht zulässig gewertet, wenn sein Inverses in der Tabuliste auftritt. Sind diese Charakteristika allgemeiner gefaßt, was bei größeren Suchräumen sinnvoll ist, kann es vorkommen, daß ein „Nachbar“ als verboten gewertet wird, obwohl er noch nicht besucht wurde. Damit er dennoch betrachtet wird, falls er eine deutliche Verbesserung erlaubt, wird eine sogenannte „Aspiration“-Funktion hinzugefügt, die bei deutlichen Verbesserungen einem Wechsel zustimmt. Insgesamt läßt sich das Übergangsverhalten ϕ der Tabusuche für ein Minimierungsproblem mit Zielfunktion f wie folgt charakterisieren:

$$\phi(X) = \arg\min_{f(Y)} \{Y | Y \in N(X) \wedge (\neq \text{Tabu(Y)} \vee \text{Asp(Y)})\} \tag{6.1}$$

Beliebte Modifikationen der Tabusuche sind ([Dow97]):

- Verwendung wechselnder Zielfunktionen
- Variation der Tabulistenlänge
- Die Auswahl von Teilmengen der Nachbarschaft
- Ketten: Die Nachbarschaft entsteht nicht als Bild einfacher Operationen, sondern als Ergebnis iterierter Anwendung dieser Übergänge

Typische Implementierungen der Tabusuche im Bereich der automatischen Stundenplanerstellung verwenden als Übergangsoperationen einfache Tausch- oder Verschiebungsoperationen ([Sch96]). Die meisten Formen von Tabulisten umfassen einen oder mehrere der folgenden Ereignistypen ([Dow97],[Cos94]):

- Veränderung einer Stunde
- Veränderung einer Zeitstunde
- Verschieben einer bestimmten Stunde aus oder in eine Zeitstunde

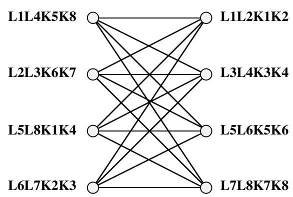
Die Anwendung anspruchsvollerer Methoden der Tabusuche scheint mehr durch empirische Ergebnisse und individuelle Entscheidungen geleitet zu werden als durch tiefere Kenntnis der Problemstruktur.
Ein effektiver Vergleich der Verfahren ist kaum möglich, es sei denn man entschiede sich, die Zielfunktion eines anderen Autors vollständig zu übernehmen, da als Bewertung eines Verfahrens zumeist auf den Vergleich des erreichten Zielfunktionswertes mit dem nachberechneten Funktionswert eines handerstellten Stundenplans zurückgegriffen wird [Sch96].

6.4 Kodierung

Als Kodierung wird die Zuordnung von Schulstunden zu Zeitstunden gewählt, wobei diese als $0, 1$ Matrix dargestellt wird. Eine Kodierung via Listen wäre weniger speicheraufwendig, hätte aber den Nachteil, daß entweder der Zugriff auf die Tabuliste oder der Zugriff auf kollidierende Stunden verlangsamt würde. Die Ressourcenzuteilung wird - wie in Abschnitt 5.2.1 angedeutet - jeweils neu berechnet. Im Gegensatz zu bitstringorientierten Verfahren wie genetischen Algorithmen ist hier die konkrete Kodierung nicht relevant für die ohnehin kaum quantifizierbare Konvergenzgeschwindigkeit, da sämtliche Operationen auf einer abstrakteren Ebene angesiedelt sind, d.h. in Notationen von Stunden und Klassen, und die Implementierung diese lediglich umsetzt. Analog werden in der Tabuliste auch nicht einzelne Bitpositionen vermerkt, sondern Paare von Schulstunden.
Etwas formaler gesprochen: Seien zwei Kodierungen C, C' gegeben, die durch eine Funktion f ineinander überführt werden können, und eine Übergangsfunktion g, die mit C operiert, so ergäbe sich die entsprechende Übergangsfunktion für C' einfach als $f^{-1}gf$. Insofern ist die Kodierung nur für die Laufzeit relevant, wobei zu berücksichtigen ist, daß das Verfahren natürlich eine hohe Iterationszahl verlangt.
Eine Umkodierung des gesamten Problems in eine Form, die es erlaubt hätte, die Tabuliste in Form des „Festhaltens einzelner Bits“ zu kodieren, hätte zwar die Möglichkeit eröffnet, die zurückgelegte „Strecke“ im Form der Hamming-Distanz zu bestimmen und z.B. zur Steuerung der Tabulistenlänge zu verwenden [Bat96], dies scheint aber den damit verbundenen Aufwand nicht zu rechtfertigen, da dies zudem verlangen würde, wiederkehrende Punkte zu erkennen.

6.5 Wahl der Nachbarschaft

Im allgemeinen ([Cos94],[Sch96],[Her92]) werden zwei Stundenpläne als „benachbart“ angenommen, wenn sie durch eine einfache Operation, wie etwa das Verschieben einer Stunde, auseinander hervorgehen. Im folgenden sollen die Auswirkungen einer solchen Verschiebung kurz untersucht werden:
Sei angenommen, daß eine Stunde S aus n Objekten besteht. Es ist natürlich nicht sinnvoll, diese Stunde in eine Zeitstunde zu verschieben, die von einer zur selben Gruppe gehörenden Stunde belegt wird. Ansonsten wird normalerweise jedes Objekt in der neuen Zeitstunde bereits in einer anderen Stunde verwendet. D.h. um diese eine Verschiebung gültig werden zu lassen, müssen bis zu n weitere Gruppen bewegt werden. Im Idealfall sieht das so aus, daß $n-1$ weitere Gruppen aus der alten Zeitstunde in die neue bewegt werden, so daß in der alten Zeitstunde Platz für die n aus dem neuen Timeslot zu verschiebenden Gruppen gewonnen wird (Kanten deuten Kollisionen an):



Jede andere Variante würde offenbar noch mehr Stunden involvieren. Für $n = 2$(STP) müssen also insgesamt 4 Stunden bewegt werden. In [Sch96], wo lediglich das STP betrachtet wird, wird dies realisiert, indem als „elementare“ Operation ein Austausch vorgesehen wird, und stets dann, wenn Widersprüche auftreten, ein Korrekturversuch durch eine weitere Austauschoperation unternommen wird, was zum Erfolg führen kann, da somit die benötigte Zahl von Mengenverschiebungen erzielt wird. Für größere n wird dieses Vorgehen aber sinnlos, da der Zeitaufwand für die entstehende „$2n$-OPT“ Optimierungsaufgabe zum Lösen der Widersprüche zu aufwendig wird. Gibt man aber diesen „harten“ Bedingungen die größtmögliche Priorität, wird es somit offenbar nahezu unmöglich, ein lokales Minimum der harten Nebenbedingungen zu verlassen. Da somit das Hintereinanderausführen mehrerer Schritte auch kein Verlassen des entsprechenden Gebietes garantieren kann, wird in dieser Arbeit der einfachere Nachbarschafts-Ansatz nach Costa [Cos94] gewählt, der nur elementare Verschiebeoperationen vorsieht und die Behandlung dieses Problems in den Bereich der Ziel- bzw. Aspiration-Funktion bzw. der Hybridisierung verlagert.

Es sei noch angemerkt, daß diese gerade beschriebene Eigenschaft des Lösungsraums unter den gängigen Metaheuristiken nur mit Tabusuche und Genetischen Algorithmen effektiv angegangen werden kann, da die entsprechenden großen Sprünge der Zielfunktion die Wahrscheinlichkeit, daß z.B. „Simulated Annealing“ ein solches „Loch“ verlassen kann, aufgrund des exponentiellen Charakters der Akzeptanzfunktion exponentiell gering werden läßt. Elmohamed et al. [ECF97] versuchen daher z.B. die Wahl des neuen Punktes mittels einer Heuristik zu verbessern, was andererseits eine Analyse des erwarteten Verhaltens unmöglich werden läßt.

Aufgrund des obigen Beispiels wurde die Tabuliste asymmetrisch gewählt; d.h. wenn eine Stunde A von einer Stunde B weggelegt wurde, so kann B A folgen, A aber nicht zurückverlegt werden. Ansonsten wäre ein Austausch, wie er oben angedeutet wird, unmöglich. Der relativ große Aufwand zur Berechnung der Zielfunktion läßt es nicht zu, stets die gesamte Nachbarschaft zu betrachten. Es muß daher eine Teilmenge, in diesem Fall eine konkrete Schulstunde ausgewählt werden, deren Verschiebbarkeit getestet werden soll. Dies ließe sich natürlich auch durch eine komplexere Konstruktion der Nachbarschaft erreichen, aber es scheint üblich zu sein, hier diese Terminologie zu verwenden ([Cos94]).

Dieses Vorgehen birgt allerdings ein gewisses Risiko: Als eine der Voraussetzungen für die Funktionsfähigkeit von Tabusuche wird stets angenommen, daß die reflexiv-transitive Hülle der Nachbarschaftsrelation alle Punkte des Lösungsraums paarweise verbindet. Dies ist bei den relativ einfachen Übergangsoperationen stets gesichert. Durch dieses Vorgehen, nun von einer Teilmenge der Nachbarschaft zu sprechen, bleibt der Anschein gewahrt, daß dies nach wie vor gelte. Bei [Cos94] werden hier z.B. nur Stunden betrachtet, die Kollisionen aufweisen. Dies heißt z.B., daß unabhängig von der Wahl der Zielfunktion ein Verlassen einer Konstellation, die die „harten“ Nebenbedingungen erfüllt, nicht möglich ist.
Dieses Vorgehen stellte sich - in ersten Experimenten - als nicht ausreichend heraus, da, wenn die vollständige Beseitigung dieser Probleme nicht möglich ist, nach Ablauf des Programms simple Optimierungschritte möglich bleiben, die aber wegen der zu eingeschränkten Suchweise nicht beachtet wurden. Daher wurde eine gemischte Bewertung nach der Zahl der Widersprüche und der Wartezeit eingeführt. Die Priorität einer Zeitstunde ergibt sich dabei einfach als Summe der Zahl der Ressourcenüberschreitungen und der Wartezeit seit dem Zeitpunkt der letzten Betrachtung dieser Stunde. Da die Zahl der Überschreitungen durch die der Zahl Objekte O beschränkt

ist (sei Z die Zahl der Zeitstunden), genügt es also, mindestens $O+Z$ Iterationen durchzuführen, um sicherzustellen, daß jede Zeitstunde mindestens einmal überprüft wurde. Spätestens sobald eine Stunde Wartezeit O erreicht, nimmt die Zahl der Stunden mit höherer Priorität mit jedem Schritt um 1 ab, da jede dieser auf einen Wert kleinergleich O zurückgestellt wird. Damit muß die betrachtete Stunde nach spätestens Z weiteren Schritten die höchste Priorität aufweisen.[2] Alle in dieser Zeitstunde gelegenen Schulstunden werden dann der Reihe nach auf Verschiebbarkeit geprüft. Dies stellt zumindest prinzipiell den Zusammenhang des Lösungsraums unter der „Teilmenge der Nachbarschaft“-Relation sicher, wobei zugleich die verstärkte Untersuchung kollidierender Stunden gesichert wird.

Zusätzlich wird hier jeder Stunde, die eine Kollision aufweist, das Verbleiben in der Zeitstunde verweigert, so daß diese nach Abschluß ihrer Untersuchung sicher kollisionsfrei ist. Dies erscheint zunächst übertrieben, stellt sich aber wiederum im Zusammenhang mit der Zahl der für eine Verbesserung benötigten Operationen als sinnvoll heraus, da somit allein die Möglichkeit geschaffen werden kann, hier mehrere andere Stunden unterzubringen.
Sehr interessant ist hier auch der Ansatz in [Cat98], wo - wenn auch im Kontext des „Simulated Annealing“ - die Möglichkeit vorgesehen wird, Stunden ganz zu entfernen. Hierdurch können die starken Sprünge an den Rändern eines Optimums abgeschwächt werden. Im schulischen Kontext ist die Zahl der möglichen, aber abzulehnenden (Nachmittags-) Stunden ausreichend hoch, um diesen Effekt über eine Verletzung schwächerer Nebenbedingungen zu erzielen. Abramson [AD93] führt dies ebenfalls als möglichen Ausweg an.

6.5.1 Bewertungs und „Aspiration“ - Funktion

Die Bewertungsfunktion ist identisch mit der eingangs beschriebenen. Es werden aber zunächst zwei getrennte Zielfunktionen bestimmt: Sei x eine Konfiguration so ist $h(x)$ die Zahl der Kollisionen und $s(x)$ repräsentiert die verbleibenden Summanden. Anstelle einer Diversifizierung wird nach einer bestimmten Anzahl von Schritten ohne Verbesserung von h dessen Anteil an der Zielfunktion reduziert, was, von den schwachen Nebenbedingungen getrieben, zu einer verstärkten Kompaktifizierung des Planes führt. In dieser Situation wird die Zielfunktion zusätzlich um eine schwache Randomisierung ergänzt, da bei reiner Berücksichtigung der schwachen Nebenbedingungen häufig mehrere Stunden gleich bewertet werden, was zur einer nur von der Stundenreihenfolge geprägten und daher immer identischen Auswahl führen und damit die Zahl der Kollisionen unnötig in die Höhe treiben würde. Da die Größe der Zufallswerte unterhalb der „Auflösung“ der Zielfunktion liegt, könnte die vorliegende Methode im Rahmen der Klassifikation von Batitti [Bat96] als „fixed TS with stochastic tie breaking“ eingeordnet werden.
Dem Charakter des Lösungsraums entsprechend wurde die „Aspiration“-Funktion so gewählt, daß sie sowohl auf Verbesserungen der „harten“, als auch auf eine deutliche Verbesserung im Bereich der „weichen“ Nebenbedingungen anspricht. Konkret bedeutet dies, daß die „Aspiration“-Funktion dann einem Wechsel zustimmt, wenn die Zahl der Kollisionen um mehr als die halbe Gruppengröße reduziert werden kann, was bei kleineren Gruppen kaum möglich ist, oder wenn ein Ergebnis erzielt werden kann, das besser als das bisher erreichte ist.

[2] In Betriebssystem-Terminologie: Es tritt keine „Starvation“ auf.

Es besteht natürlich nicht die Notwendigkeit, hier die Funktionen einzusetzen, die den tatsächlichen Zielfunktionen des Problems entsprechen; so z.B. erzielen KHANNA et al.[KMSV94] mit abgewandelten Funktionen im Sinne der Ausgangsfunktion bessere Ergebnisse. Ein möglicher Ansatz hier wäre es, die erzielte Dichte in Schulstunden pro Zeitstunde mal Kollisionszahl einfließen zu lassen, da eine kollisionsarme Kompaktheit in einer Stunde das Erreichen der echten Zielfunktion in anderen Stunden begünstigen würde. Der Versuch, die schwachen Constraints stärker zu berücksichtigen, indem zu Beginn die harten Bedingungen völlig ignoriert wurden, führte in jeder Hinsicht zu schlechteren Ergebnissen, was sich mit der Beobachtung von STÜBER [Stü97] deckt, daß die Tabusuche nur erfolgreich ist, wenn die Startposition nicht mehr allzuviele Kollisionen aufweist.

6.6 Hybridisierung mit einer Variante des Algorithmus nach KÖNIG

6.6.1 Anwendungsvoraussetzungen

Da dieses Verfahren dem Algorithmus von KÖNIG folgt, verlangt es die Vorgabe eines bipartiten Graphen. Dieser ist im RSTP im allgemeinen nicht gegeben, die Abweichungen sind aber insbesondere im Bereich der Sekundarstufe I häufig auf wenige Differenzierungsstunden beschränkt. Voraussetzung ist somit in erster Linie die folgende Bedingung:

Bemerkung 6.1 *Der im folgenden beschriebene Algorithmus setzt voraus, daß sich aus der Menge der Objekte (Lehrer, Klassen, Räume) zwei Teilmengen auswählen lassen, so daß in jeder Gruppe mindestens ein Element aus jeder dieser Teilmengen auftritt.*

Bei der konkreten Umwandlung des UTP wird aus jeder Gruppe je ein Element aus jeder Teilmenge ausgewählt. Eine Lösung nach dem Satz von KÖNIG garantiert also somit, daß bezüglich dieser Elemente keine Widersprüche auftreten können.
Im folgenden sei n die größere der beiden Knotenzahlen in dem so gewonnenen bipartiten Graphen und k die Zahl der Farben. Des weiteren sei basierend auf obiger Bemerkung vereinbart:

Definition 6.1 *Eine Stunde ist genau dann eine Differenzierungsstunde, wenn sie aus einer der beiden oben definierten Mengen mehr als ein Element enthält.*

Definition 6.2 *Ein Stundenplan wird als „KL-geeignet" bezeichnet, wenn er gemäß dem STP, das sich durch Reduktion aller Gruppen auf die oben ausgewählten Objekte ergibt, geeignet ist.*

Die hiermit im Zusammenhang stehenden Überlegungen entspringen allerdings vollständig dem schulischen Kontext, da hier angenommen werden kann, daß Kollisionen in der im allgemeinen dritten „Ressource Raum" seltener sind und im Bereich von Schülern und Lehrern selten die Wahl zwischen verschiedenen Objekten möglich ist. Dies muß bei einer Universität oder Fachhochschule nicht mehr der Fall sein.

6.6.2 Gewichtetes Matching

Beim Algorithmus nach KÖNIG (Satz 4.3) spielen Matchings, die alle Knoten maximalen Grades überdecken, eine entscheidende Rolle (vgl. dort). Wie die Matchings abgesehen von dieser Eigenschaft aussehen, ist für das Funktionieren des Algorithmus nicht von Bedeutung.
Es ist daher möglich, andere Kanten nach Belieben hinzuzunehmen. Ersetzt man nun das im Algorithmus nach KÖNIG auftretende einfache Matching durch gewichtetes Matching, so muß man lediglich dafür sorgen, daß stets alle Knoten maximalen Grades im zugehörigen Multigraphen von den jeweiligen Matchings überdeckt werden. Sei hierzu im Graphen $G = (V, E)$ von einer Gewichtung $w(e)$ $e \in E$ des Graphen ausgegangen, die in irgendeiner Weise gewünschte Eigenschaften des Matchings kodiert. Sei A die Menge der Knoten mit maximalem Grad und $c := \sum_{e \in E} w(e) + 1$. Dann definiere $w'(e) := w(e) + c * |e \cap A|$.

Lemma 6.1 *Ein gemäß w' maximales Matching überdeckt (1) alle Knoten maximalen Grades und ist (2) unter den Matchings mit dieser Eigenschaft gemäß w maximal.*

Beweis:
1. Satz 4.2 zeigt, daß es ein Matching gibt, das alle Knoten maximalen Grades überdeckt. Es genügt also zu zeigen, daß jedes Matching mit dieser Eigenschaft gemäß w' schwerer ist als jedes andere, da damit das schwerste Matching ingesamt eines von diesen sein muß. Ein Matching mit $k < |A|$ überdeckten Knoten maximalen Grades hat aber offenbar ein Gewicht von höchstens $\sum_{e \in E} w(e) + k * c = (k+1) * c - 1$, während ein Matching mit $|A|$ überdeckten Kanten mindestens das Gewicht $|A| * c \geq (k+1) * c$ hat.
2. Da alle Matchings, die alle Knoten maximalen Grades überdecken, gleichviele dieser Knoten überdecken, gilt für ein derartiges Matching stets $w'(M) = w(M) + c * |A|$ und somit ist für diese $w(M) \geq w(M') \Leftrightarrow W'(M) \geq W'(M')$.

Eine ähnliche Idee findet sich in [CdW89], wobei diese Autoren die Gewichtungsmöglichkeit ausschließlich zur Erzwingung der gewünschten Überdeckung von Knoten maximalen Grades verwenden[3], aber nicht explizit auf die Garantien eingehen. Des weiteren finden sich ähnliche Überlegungen in [dW71]. Eine Frage, die sich an dieser Stelle ebenfalls stellt, ist, ob alle Kanten um einen solchen, die übrige Bewertung dominierenden Betrag vergrößert werden sollten, um stets maximale Matchings maximalen Gewichts zu erhalten. Auch wenn dies im Sinne der harten Nebenbedingungen sinnvoll wäre, wurde hier davon abgesehen, da hiervon insbesondere die Nichtverfügbarkeiten betroffen würden, die zumeist ähnlich wichtig sind.
Zur Gewichtung können nun mehrere Kriterien herangezogen werden:

- Die Tatsache, ob für ein Objekt die Zahl der noch zu setzenden Stunden gleich der Zahl der noch verbleibenden Stunden ist. Dieses Kriterium ist nicht hinreichend für eine Lösbarkeit des RSTP, da nicht berücksichtigt wird, ob sich diese Stunden auch tatsächlich plazieren lassen.
- Analog bei flexiblen Ressourcen, ob die Zahl der noch zu benutzenden Objekte gleich der Objektzahl multipliziert mit der Zahl der verbleibenden Zeitstunden ist.

[3] Und das Ganze damit als Flußproblem auffassen.

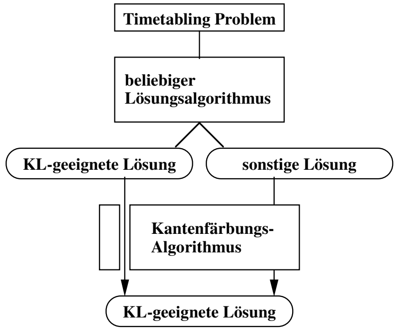
Die folgenden Varianten könnten sinnvolle Erweiterungen darstellen:

- Die Tatsache, daß die entsprechende Stunde „tight" nach den Kriterien von GOTTLIEB ist, was eine Verbesserung im obigen Sinne darstellt, aber auch nicht hinreichend ist.
- Werte, die repräsentieren, wie weit die Stunde noch von den gerade beschriebenen Tatsachen entfernt ist, das heißt zum Beispiel, um wieviel die Zahl der verbleibenden Stunden größer ist, als die der benötigten. Dies führt zu einer Art „Lookahead".
- Die zeitbezogenen Anforderungen bzw. Wünsche der Objekte.
- Relative Nebenbedingungen, sofern bereits andere Stunden gesetzt wurden.

In der bisherigen Implementierung wurden die Gewichtungen der übrigen Parameter noch unmittelbar aus den von Benutzer gewählten Werten bezogen. Hierbei ist aber zu beachten, daß diese zumeist relativ sind. Gibt der Anwender z.B. an, daß zwei Stunden gleichermaßen am Montag liegen können, die eine aber besonders für Dienstage geeignet ist, während die andere möglichst nicht an diesem Tag liegen sollte, so können diese beiden bei der Betrachtung des Montags nicht mehr die gleiche Bewertung erhalten, da die Auswahl somit zufällig wäre. Daher wäre es vermutlich sinnvoll, die absoluten Parameter durch ihren relativen Anteil an der Summe der Parameter zu ersetzen.

6.6.3 Für die Hybridisierung günstige Eigenschaften

Auch mit einigen Modifikationen läßt sich dieses Verfahren, das lange Zeit das einzige polynomielle des gesamten Gebietes war, nicht dazu verwenden, kompliziertere Zielfunktionen zu verfolgen bzw. Differenzierungen und Raumzuteilungen zufriedenstellend zu lösen. Es hat aber, und aus diesem Grunde wurde es dem Verfahren nach GALVIN vorgezogen, die Eigenschaft, eine in der Bewertungsmatrix kodierte gültige Färbung zu erhalten: Die Lehrer-Klassen-Kanten (oder eine Auswahl derselben beim RSTP) müssen bei einer gültigen Lösung zu jeder Zeitstunde ein Matching bilden, das alle Knoten kritischen Grades überdeckt. Versieht man also in einem entsprechenden gewichteten Graphen diese Kanten mit dem Gewicht 1 und wertet die andern mit 0, so wird der obige Algorithmus ein Matching finden, das diese Kanten verwendet. Faßt man also den Algorithmus als eine Funktion ϕ auf, die aus dem $\mathbb{R}^{L\times K\times T} \times \{G|\ G \text{bipartit}\}$ in die Menge S der KL-geeigneten Stundenpläne über G führt, so ist diese surjektiv, da man jeden Zielpunkt erreichen kann, indem man ihn in die Eingabe $0,1$-kodiert. Sei $\phi_G(x) := \phi(x, G)$. Damit könnte der Algorithmus also als „Postoptimizer" für beliebige andere Verfahren verwandt werden.



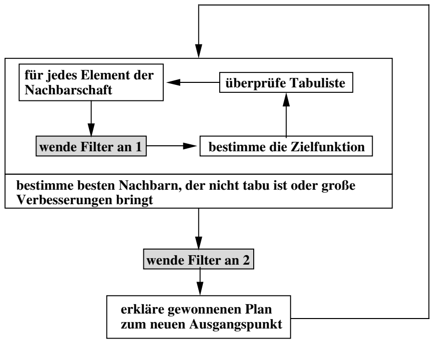
Andererseits erlaubt es dies aber auch, diesen Algorithmus als Bestandteil einer komplexeren Zielfunktion zu sehen. Mit einer gegebenen Bewertung $\psi : S \to \mathbb{R}$ erhält man durch $\zeta_G := \psi \circ \phi_G$ eine Abbildung von $\mathbb{R}^{L\times K\times T}$ nach $\mathbb{R}$. Nimmt man G als fest an, so erhält man also eine Optimierungsaufgabe der Form:
Finde $\mathcal{X} \in \mathbb{R}^{L\times K\times T}$, so daß $\zeta_G(\mathcal{X})$ maximal wird. Im folgenden sollen die Auswirkungen dieser Veränderung diskutiert werden.

6.6.4 Auswirkungen der Hybridisierung

Häufig wird in der Literatur zum Timetabling angegeben [Dow97], daß die „Oberfläche" der Zielfunktionen durch steile, schmale Täler bzw. Löcher geprägt sein kann. Zudem verrät die Struktur der umliegenden Funktionswerte, die von Springstundenminimierung o.ä. geprägt wird, selten die günstigste Richtung zu einem solchen Tal, da es z.B. ungültige Lösungen mit wenigen Springstunden geben kann.
Durch das oben beschriebene Zwischenschalten des Kantenfärbungsalgorithmus werden diese Täler künstlich verbreitert, da nun die benachbarten Lösungen zu KL-geeigneten Lösungen umgeformt werden.
Es bleiben insbesondere beim RSTP natürlich auch Täler erhalten, die durch die nichtbeachteten Gruppenmitglieder (s. Abschnitt 6.6.1) erzeugt werden; diese sind aber nicht so „tief". Dies wirkt sich allerdings andererseits auch so aus, daß nun große Gebiete mit gleichem Funktionswert entstehen, da für viele Punkte bereits die Eingabe der Zielfunktion identisch ist. Gerade auf der Traversierung solcher Flächen beruhten aber die Fortschritte moderner Metaheuristiken gegenüber klassischen Minimierungsverfahren.
Zudem muß man sich - im Gegensatz zum Problem der Täler - hiermit nicht abfinden, da man diesen Flächen eine zusätzliche Struktur aufprägen kann:

- Die weichen Bedingungen können z.T. über der Eingabe des Graphenalgorithmus ausgewertet werden.
- Der Kontrast zwischen einzelnen Werten kann reduziert werden, so daß ein schnellerer Wechsel bei der Ausgabe des Graphenalgorithmus erfolgt.
- Der Abstand zu der Lösung eines relaxierten Problems, z.B. eines LPs, kann hinzugenommen werden, so daß vornehmlich in dessen Nähe gesucht wird.

Innerhalb der Tabusuche gibt es zwei Stellen, die es erlauben, diesen Filterprozess zu integrieren, abgesehen von einer Verwendung als Pre- oder Postprocessor:



Den oben geäußerten Ideen würde es am besten gerecht, wenn das Filterverfahren in der inneren Schleife (1) integriert würde, also zu jeder Berechnung der Zielfunktion herangezogen würde. Dies würde aber, um noch eine ausreichende Geschwindigkeit gewährleisten zu können, verlangen, die Nachbarschaft ausreichend klein zu wählen, so daß die Zahl der Iterationen entsprechend gedrückt würde.
Eine Möglichkeit dies zu realisieren, wäre eine Form von „Metropolis-Verfahren mit Tabuliste“, da hier aus der Nachbarschaft jeweils nur ein Element ausgewählt würde. Hier träte allerdings das Problem auf, daß dieses Verfahren insgesamt mehr Einzelschritte vollziehen muß als die Tabusuche.
In dieser Arbeit wurde der andere Weg beschritten und das Suchverfahren als nachträglicher Filterschritt vorgesehen, was wesentlich weniger laufzeitaufwendig ist, aber eher einem memetischen Algorithmus entspricht (vgl. [BNW95]). Dies wiederum kann dazu führen, daß diese Nachbearbeitung die Veränderungen, die durch das Suchverfahren vorgenommen werden, revidiert.[4] Um diesem entgegenzuwirken, werden die Zeitstunden vor Anwendung des Filterschritts zufällig permutiert. Die Wahrscheinlichkeit, ob dies funktioniert, ist allerdings vom jeweiligen Problem abhängig, da es durchaus vorkommen kann, daß eine bestimmte Schulstunde nur in einer bestimmten Zeitstunde liegen darf.
Geht man allerdings davon aus, daß keine Zeitwünsche gegeben sind, und daß die entsprechende Stunde nicht mehr von der Wahl der übrigen Stunden beeinflußt wird, so sinkt sie unter $\frac{1}{|\text{Zeitstunden}|-|\text{Gruppengröße}|}$, da der Filter sie unabhängig von der gewählten Permutation stets im gleichen Bearbeitungsschritt setzt. Die Wahrscheinlichkeit, daß eine Stellung vollständig rekonstruiert wird, ist entsprechend geringer, aber nicht das Produkt dieser Wahrscheinlichkeiten, da die Ereignisse nicht unabhängig sind.

6.6.5 Nachbearbeitung

Offenbar hat das obige Verfahren den Nachteil, daß bezüglich der nichtbeachteten Gruppenobjekte Widersprüche auftreten können. Ließe man nun einfach die entsprechenden Kanten weg, so würde dies nur dazu führen, daß diese entweder nicht gesetzt werden oder irgendwann zu „unbedingt notwendigen“ Kanten würden, so daß diese dann trotz ihrer Kollisionen eingesetzt werden müßten.

[4] Diese Möglichkeit scheint von den Befürwortern dieser Methode vollkommen ignoriert zu werden.

6.6.5.1 Gestaffelte Ressourcenoptimierung

Wurde nach obigem Verfahren eine Menge von Stunden ermittelt, die in die momentan betrachtete Zeitstunde gelegt werden sollen, so ist noch zu überprüfen, daß diese nicht wegen nicht im Graphen repräsentierter Gruppenmitglieder kollidieren. Im Gegensatz zur Vorgehensweise in Abschnitt 8.4.3 reicht es hier allerdings nicht, nicht setzbare Stunden nur zu markieren, sondern es sollen möglichst viele Stunden gesetzt werden. Dies entspricht aber dem „Set-Packing“ Problem, welches selbst wieder $\mathcal{NP}$-schwer ist. Standardansatz wäre hier eine LP-Relaxation mit anschließendem Runden (vgl. [Hoc97]), aber der zeitliche Aufwand wäre für eine Subroutine einer Metaheuristik zu hoch. Glücklicherweise liegt aber ein zusätzliches Kriterium vor, nämlich die Wichtigkeit der Stunden. Sei also angenommen, es existiere eine vollständige Ordnung O auf den m Stunden, die ausgewählt wurden, und seien diese so angeordnet, daß $i > j \Leftrightarrow O(i) > O(j)$ $i, j \in [1, \ldots, n]$. Dann kann im Abschnitt 8.4.3 analog zu und mit demselben Beweis wie in Abschnitt 6.6.2 eine Bewertung der Kanten vorgenommen werden, bei der alle Kanten, die in bezug zur Gruppe S_i stehen, den Wert $(m+1)^{i-1}$ erhalten. Bestimmt man nun ein maximal gewichtetes Matching in diesem Graphen, so werden zunächst die Gruppen mit hohem Index befriedigt, bevor eine mit niedrigem Index überhaupt eine einzige Kante erhält. Dieses Verfahren ist natürlich nicht optimal - es wäre es nur, wenn die Gruppen mit den größten Indices ein maximales Packing bilden würden, aber es ist dennoch dem Greedy-Verfahren, das Gruppen wählt solange sie passen, überlegen, da dieses auch dann, wenn eines existiert, kein Matching der bevorzugten Komponenten garantieren kann.
Auch wenn nur $(m-1)\log_2(m+1)$, also polynomiell viele Bits zur Kodierung dieser Gewichtungen benötigt würden, scheint es im Rahmen dieser konkreten Anwendung sinnvoller, sich mit den zur Verfügung stehenden Zahldarstellungen zu beschränken, und stattdessen die Gruppen in mehrere Mengen aufzuteilen.

Tritt eine Ressource stets im Zusammenhang mit den gleichen dynamischen Ressourcen auf, so könnte wenn man das Matching- durch das analoge Flußproblem ersetzt, diese Ressource den dynamischen „vorgeschaltet“ werden, was die Kollisionszahl vermindern würde, da diese Ressourcen nur noch gleichzeitig oder gar nicht benutzt werden können, wobei die selbe Konstruktion entstehen würde, wie in Abbildung b) in Kapitel 3.6.1. Verzichtet man darauf, die optimale Zuordnung zu wählen, so kann die Existenz auch mit Hilfe des nichtgewichteten Algorithmus geprüft werden (s. Abschnitt 6.7).

6.6.5.2 Alternierende Kreise

Wird festgestellt, daß eine der gewählten Stunden die vorhandenen Ressourcen überschreitet, so wird versucht, diese zu entfernen. Das hier eigentlich notwendige Konzept wäre das von Matchings mit Ressourcenbeschränkungen, die aber, wie schon in Abschnitt 4.10.7 erwähnt für den gegebenen Fall nicht effizient bestimmt werden können. Wie in Abschnitt 4.3.3.1 dargestellt, ist es bei der Untersuchung verschiedener Matchings in Graphen interessant, deren symmetrische Differenz zu betrachten. Vergleicht man nun zwei maximale Matchings miteinander, so besteht ihre symmetrische Differenz aus Pfaden gerader Länge und Kreisen, da es keine augmentierenden Wege mehr geben darf und somit ein Pfad ungerader Länge immer fortgesetzt werden kann. Wird das Matching über einen alternierenden Kreis modifiziert, bleibt, im Gegensatz zu der

Verwendung von Pfaden, die Menge der vom Matching überdeckten Knoten gleich. Insbesondere erhält eine Veränderung nach diesem Schema also die Eigenschaft eines Matchings, Knoten kritischen Grades zu überdecken. Somit bietet sich das Suchen solcher Kreise an, um Matchings nachträglich zu verbessern.
Hierbei muß dafür gesorgt werden, daß die neu hinzugekommenen Kanten nicht andere Ressourcenüberschreitungen verursachen. Daher werden hierfür nur Kanten zugelassen, die nicht mit einer Differenzierungsstunde korrespondieren. Dieses Vorgehen folgt der Überlegung, daß im Bereich des RSTP damit zu rechnen ist, daß sich räumliche Probleme relativ leicht beheben lassen, da es häufig möglich ist, Unterricht auch in Nicht-Fachräumen durchzuführen. Diese Annahme muß bei anderen Planungsaufgaben natürlich nicht gegeben sein.
Betrachtet man den Gesamtwert eines solchen Matchings, so verringert sich dieser bei Alternieren über einen solchen Kreis um das Gewicht der beteiligten Matchingkanten und es vergrößert sich um das der Nicht-Matching-Kanten. Da es $\mathcal{NP}$-vollständig ist, in einem Graphen den längstmöglichen Weg zu suchen, kürzeste Wege auf Netzwerken mit positiven Kantengewichten schneller bestimmt werden können als auf allgemeinen und es insgesamt wünschenswert ist, den Graphen so wenig wie möglich zu verändern, werden die Kantengewichte wie folgt modifiziert und als Entfernungen interpretiert, damit der Algorithmus von DIJKSTRA [Jun90] anwendbar wird: Sei $W := \max_{e \in G} w(e)$. Dann setze

$$w(e) := \begin{cases} W + 1 - w(e) & \text{für } e \notin M \\ w(e) & \text{für } e \in M \end{cases}$$

Danach wird solange noch Widersprüche auftreten, aber nur $n$mal (es muß nicht notwendigerweise möglich sein, alle Kollisionen zu beseitigen) die folgende Routine durchgeführt, wobei die Partitionen des Graphen im folgenden mit „links“ und „rechts“ bezeichnet seien.

- Bestimme eine Kante $e = (l, r)$, die maximal viele Kollisionen auslöst.
- Bestimme den alternierenden Kürzeste-Wege-Baum [Blu98] von r zu allen anderen Knoten auf der rechten Seite (s.u.)
- Bestimme denjenigen Knoten $p \neq r$ auf der rechten Seite, bei dem die Summe von „Entfernung“ im Baum und Abstand zu l minimal sind.
- Vertausche entlang des durch (l, p), den kürzesten Weg von p zu r und (l, r) gegebenen Kreises Matching - und Nicht-Matching - Kanten.

Einer genaueren Erläuterung bedarf noch die Konstruktion des alternierenden Baumes. Die ausgewählte Kante sei wie oben (l, r). Hierzu betrachte man das gerichtete Netzwerk $H = (R, E, w)$, wobei R die Knoten der rechten Seite im bipartiten Graphen sein sollen (analog L) und

$$E := \{(x, y) | \exists z \in L : z \neq l \wedge (x, z) \notin M \wedge (z, y) \in M\}$$

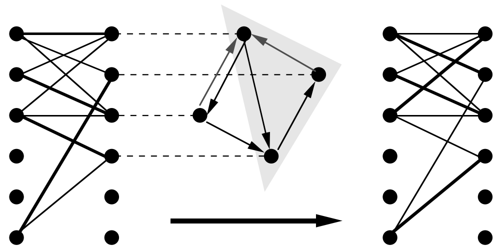
Da dieses z eindeutig bestimmt ist, sei $\phi((x, y)) := z$.

$$w((x, y)) := w_G((x, \phi((x, y)))) + w_G(\phi((x, y)), y)$$

Lemma 6.2 *Ein Kürzester-Wege-Baum mit Wurzel r auf H entspricht einem alternierenden Kürzeste-Wege-Baum auf R in $G' := G \backslash (l, r)$.*

Beweis
Hierzu genügt es, zu zeigen, daß es zu jedem einfachen alternierenden Pfad von R nach R in G einen Pfad gleichen Gewichtes in H gibt und umgekehrt.
„$\Rightarrow$“: Jeder von r ausgehende alternierende Pfad muß mit einer Nicht-Matching-Kante beginnen, da $(l, r) \in M$ entfernt wurde. Damit erfolgt der Übergang zu einem anderen $y \in R$ stets über eine Abfolge von Nicht-Matching-Kante und Matching-Kante und somit muß man, um einen Weg in H zu erhalten, einfach jeden zweiten Knoten des Weges in G' entfernen. Das Gewicht des Pfades ist nach Definition die selbe.

„$\Leftarrow$“: Sei ein einfacher Pfad in H $P := p_1, p_2, p_3, \ldots, p_o$ gegeben. Dann wähle den Pfad $P' := p_1, \phi((p_1, p_2)), p_2, \phi(p_2, p_3)), p_3, \ldots, p_o$ in G. Dieser hat offenbar die selbe Länge wie P und ist alternierend. Es bleibt nur zu zeigen, daß er einfach ist. Keines der p_i kann doppelt auftreten, da dann P nicht einfach wäre. Sei also angenommen, daß ein $\phi((p_i, p_{i+1}))$ mehrfach auftritt z.B. als $\phi((p_j, p_{j+1}))$. Da aber die Kante $(\phi((p_i, p_{i+1})), p_{i+1}) \in M$ sein muß und $\forall z \in L : |\{y | \exists x : \phi((x, y)) = z\}| = 1$ gilt, da jede Matching-Kante nur zu je einem Knoten in L und R inzident ist, folgt daraus unmittelbar $p_{j+1} = p_{i+1}$, was wiederum implizieren würde, daß P nicht einfach war.
Das Entfernen von l bzw. (l, r) ist notwendig, da diese benötigt werden, um den Weg zu einem Kreis zu schließen.



Die Abbildung zeigt einen Alternierungsvorgang, wobei als Kreis in H die grau unterlegten Kanten gewählt wurden.

Gruppen, die nun immer noch nicht mit Ressourcen versehen wurden, werden entfernt bzw. sollte die zugehörige Kante im Graphen kritisch sein, als Kollision in Kauf genommen. Eventuell könnte hier noch versucht werden, weitere Gruppen „greedy“ hinzuzufügen.

6.7 Implementierung dieses Verfahrens

6.7.1 Die Tabuliste

Die Tabuliste wurde als Feld der Größe $|Gruppen|^2$ implementiert. Jeder Eintrag gibt an, bei der wievielten Iteration das entsprechende Element in die Tabuliste eingetragen wurde. Dies ermöglicht es insbesondere, die Länge der Tabuliste - auch zwischenzeitlich - beliebig zu wählen bzw. zu verändern. Die Tabulistenlänge wurde in diesem ersten Versuch wegen der Größe des Lösungsraums so lang gewählt, daß gerade noch gesichert bleibt, daß stets ein weiterer Schritt möglich ist. Hierzu sei s die Stundenzahl, G die Gruppenzahl und t die Zahl der Zeitstunden. $z(x)$ sei die Stundenzahl der Gruppe x.

Dann ist die Zahl der möglichen Bewegungen gegeben durch:

$$B := \sum_{x \in G} (t - z(x)) = gt - s = g(t - \bar{z}), \tag{6.2}$$

wobei $\bar{z}$ die durchschnittliche Stundenzahl einer Gruppe ist.
Mit jeder Eintragung in die Tabuliste wird die Anzahl der Möglichkeiten im Schnitt um $\bar{z}$ reduziert, wobei außer acht gelassen wurde, daß damit eine bereits gesperrte Zeitstunde nochmals gesperrt werden kann. Somit sollte die Tabuliste kürzer als $\frac{B}{\bar{z}} = \frac{tg^2}{s} - 1$ sein.

6.7.2 Der Filterschritt

Der Filterschritt besteht im wesentlichen darin, den Graphenalgorithmus dazu zu verwenden, die von der Tabusuche gewählte Konfiguration in eine solche umzuwandeln, die bezüglich der je zwei aus jeder Gruppe ausgewählten Elemente keine Kollisionen aufweist. Hierzu wird eine beliebige Reihenfolge der Zeitstunden gewählt und dann der Algorithmus angewandt. Hierbei werden im Gegensatz zum einfachen gewichteten Matching drei Klassen von Kanten betrachtet. Die oberste enthält die Kanten mit anliegenden Knoten kritischen Grades wie oben beschrieben. Die zweite Klasse enthält diejenigen Kanten, die von der Tabusuche für das betrachtete Zeitslot vorgesehen wurden. Die dritte beinhaltet all diejenigen Kanten, die für eine der bereits gesetzten Zeitstunden vorgeschlagen worden waren, aber nicht gesetzt werden konnten. Gibt es fest gesetzte Stunden, wird für diese eine Klasse 0 vorgesehen. Bei der Bestimmung werden die Kanten so gewichtet, daß jede Kante einer höheren Klasse Vorrang vor der Summe aller Kanten einer niedrigeren Klasse hat. Verzichtet man auf eine zusätzliche Gewichtung, so kann dieser Effekt dadurch erreicht werden, daß man zunächst ein Matching für Klasse 1 bestimmt, dieses dann zu einem von Klasse 2 erweitert, usw. Die Tatsache, daß bereits überdeckte Knoten überdeckt bleiben, sichert die Gültigkeit dieses Vorgehens. Dies wurde in der Form integriert, daß nur gegen Ende der Laufzeit eine Optimierung verlangt wird.

6.7.3 Startpunkt

Das Verfahren startet jeweils vom letzten gegebenen Stundenplan aus, wobei fehlende Stunden zufallsgesteuert hinzugefügt werden. Dies ermöglicht es einerseits, verschiedene andere Verfahren als Startpunktgeneratoren zu verwenden, andererseits können durch erneutes Starten der Tabusuche die Auswirkungen eines Löschens der Tabuliste untersucht werden.

6.7.4 Laufzeitanalyse

Mehr noch als bei genetischen Algorithmen ist eine Abschätzung der Konvergenzgeschwindigkeit der Tabusuche nahezu unmöglich. Die einzige Möglichkeit, die es hier gibt, ist die geschickte Wahl der „Aspiration-Funktion“, die zumindest garantiert, daß das Verfahren nicht langsamer ist als eine herkömmliche Abstiegsmethode.
Der Filterschritt hat wegen der Anwendung des „Ungarischen“ Algorithmus zur Prüfung der Ressourcenzuteilung eine Komplexität von $\mathcal{O}(|\text{Objekte}|^3||\text{Zeitstunden}|)$, wobei die

Nachbearbeitung hier vollkommen dominiert. Verwendet man dort ausschließlich das schnellere Matchingverfahren, so dominiert mit $\mathcal{O}(|\text{Lehrer}|^3|\text{Zeitstunden}|)$ Operationen das erste Matching.

6.8 Variationen des Graphenalgorithmus ohne Tabusuche

Wechsel der Reihenfolge mit Erinnerung
Der oben vorgeschlagene Algorithmus läßt Freiheiten bezüglich der Reihenfolge, in der die einzelnen Farben vergeben werden. Hier besteht also die Möglichkeit, die Reihenfolge zu wählen. Da der Graphenalgorithmus kollidierende Stunden stets nur dann setzt, wenn es sich nicht vermeiden läßt, treten Kollisionen immer erst später, d.h. bei höheren Farbwerten, auf. Die Idee dieses Verfahrens besteht darin, den Algorithmus mehrmals laufen zu lassen und dabei die Ergebnisse des letzten Durchlaufs, absteigend nach der Zahl der Kollisionen sortiert, als Eingabe für die Parameterwahl zu verwenden, so daß die Kollisionen nun andere Zeiten betreffen. In Analogie zu dem Nivellieren eines Sandhaufens durch Hin- und Herschieben wird dieses Verfahren im folgenden als „Sandkasten-Heuristik“ beueichnet.

Rückschritte
Durch Bewertung normaler Kanten mit 1 und kritischer Kanten mit n^2 bzw. $2n^2$ erhält man mit der Bewertung n eine Möglichkeit, Kanten allen anderen vorzuziehen, ohne dabei die Korrektheit des Algorithmus zu gefährden (vgl. Abschnitt 6.6.5). Es sei nun jeder Kante e des Multigraphen ein Wert $\zeta(e) \in \mathbb{N}_{\leq k}$ zugewiesen. Dieser Wert gibt an, zu welchem Zeitpunkt der Berechnung das Gewicht einer Kante von 1 auf n wechseln soll. Der Wechsel auf n^2 ist weiterhin nur abhängig vom Knotengrad. Der Algorithmus basiert im wesentlichen auf der Beobachtung, daß beim UTP Kanten trotz ausreichend kleinem Knotengrad nicht gesetzt werden können, da Kollisionen in den nicht repräsentierten Objekten auftreten können. Deren Auftreten kann also hier nur nachträglich beobachtet werden. Der Algorithmus läuft wie folgt:

Algorithmus 6.1

1. Initialisiere alle $\zeta(e)$ mit k

2. $i := 0$

3. Berechne den i-ten Matchingschritt (s.o.)

4. Hat eine Kante e des Matchings Wert n oder n^2 und kann wegen einer Kollision nicht gesetzt werden, und ist $i > 0 \wedge \zeta(e) > 1$
Setze $\zeta(e) := \zeta(e) - 1$ dann $i := \zeta(e)$ und gehe zu 3

5. Falls $i < k$: $i = i + 1$ gehe zu 3

Da bei jedem Rückschritt der ζ-Wert einer Kante reduziert werden muß, terminiert der Algorithmus offenbar nach spätestens $k * m$ Schritten.

6.9 Zusammenfassung

Durch die Kombination der beiden Verfahren konnten die Vorteile des Graphenalgorithmus für das STP auf die Tabusuche übertragen werden. Diese wiederum ist recht leistungsfähig, was das Reduzieren von Kollisionen angeht, da die andere Verfahren ablenkenden Symmetrien kontrolliert werden können. In der Kombination scheint allerdings die Filterwirkung des Graphenalgorithmus die Wirkung der Tabusuche etwas zu nivellieren (vgl. Kapitel 9). Hier stellt sich allerdings die Frage, inwieweit weitere Kriterien zur Kantenbewertung (s. Abschnitt 6.6.2) einfließen sollten. Diese vermögen das Verhalten des Filterverfahrens in bezug auf Kollisionen zumeist noch zu verbessern. Hierbei müsste natürlich darauf geachtet werden, daß die Surjektivität erhalten bleibt. Dies könnte z.B. geschehen, indem die Gewichtungen nachträglich mit den aus $\{0, 1\}$ gewählten Parametern multipliziert werden.

Kapitel 7

Verfahren 2: Handels - Heuristik

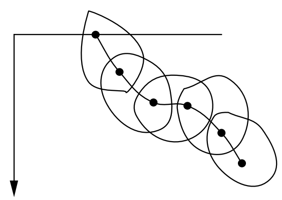
Der im folgenden beschriebene Ansatz stellt einen völlig neuartigen Versuch dar, das „Timetabling"-Problem zu bearbeiten. Inspiriert wurde dieser Ansatz dadurch, daß ein häufig bei automatisch erstellten Stundenplänen auftretendes Problem ist, daß zwar die Gesamtzahl der Kollisionen, Springstunden etc. minimiert wird, dabei aber in Kauf genommen wird, daß einzelne Objekte überproportional stark belastet werden. Während dies in betriebswirtschaftlichen Kontexten oder im „Shop-Scheduling" häufig akzeptabel ist, können derartige Ungerechtigkeiten in einer Schule, in der die meisten „Objekte" Personen sind, nicht akzeptiert werden.
Aus dieser Beobachtung entstand die Idee, diese Objekte untereinander simuliert „handeln" zu lassen. In Organisationen, in denen nach der Erstellung der ersten Pläne Wunschänderungen seitens der Betroffenen möglich sind [Rom82], wird dies sogar teilweise durch eine reale Auktion gehandhabt. Ebenso läßt es sich beobachten, daß die beteiligten Personen nach Erstellung eines fertigen Stundenplans versuchen, untereinander Termine zu tauschen. Insofern empfiehlt sich dieses Verfahren zumindest als Nachbearbeitungsmethode, da die Fähigkeiten eines Stundenplankonstrukteurs - bzw. eines Programms - häufig danach beurteilt werden, wieviele Verbesserungen auf diese Weise noch möglich sind.

7.1 Grundlagen

Der hier versuchte und bis zu einer ersten Testimplementierung gediehene Ansatz verwendet eine Kombination von „Simulated Trading" [BHM96], „Negotiation" [RW97] und „Predictor-Korrektor" Verfahren [AG82]. Der Kombination der letzten beiden entspricht in gewisser Weise das sogenannte „Equilibrium Programming" [ZG81]. Um das Zusammenwirken dieser Prinzipien zu verdeutlichen, soll hier eine kurze Beschreibung dieser Verfahren gegeben werden:

Predictor-Korrektor Verfahren gehören zum Methodenrepertoire der nichtlinearen Optimierung. Die Grundidee besteht darin, daß man eine Homotopie von einer (stetigen) Funktion mit leicht bestimmbarem Minimum $g : \mathbb{R}^n \to \mathbb{R}$ zur gewünschten (stetigen) Zielfunktion $f : \mathbb{R}^n \to \mathbb{R}$ betrachtet, das heißt eine stetige Abbildung $F : \mathbb{R}^n \times [0,1] \to \mathbb{R}$ mit $F(x,0) = g(x)$ und $F(x,1) = f(x)$.
Nun spekuliert man darauf bzw. kann in bestimmten Fällen beweisen, daß die Menge aller Punkte an denen F den Wert 0 annimmt, einen stetigen Pfad in

dem so entstandenen um eins höherdimensionalen Raum bildet [AG82].
Will man eine gegebene Optimierungsaufgabe mit Zielfunktion $g = f'$ lösen, sucht man also jeweils einen optimalen Punkt, d.h. einen Nullpunkt der Ableitung. Dann erhöht man den Homotopiekoeffizienten um einen diskreten Wert. Differenzierbarkeit annehmend prognostiziert man, daß sich das Minimum entlang der letzten Tangente an den Pfad ein Stück weiterbewegt hat und sagt somit die ungefähre Position des neuen Minimums voraus („Predictor"-Schritt). Danach wendet man lokale Suchverfahren, z.B, das Newtonverfahren, an, um sich dem tatsächlichen Minimalpunkt wieder zu nähern. Das Verhalten des Verfahrens kann über die Parameter „Schrittweite des Predictor-Schritts" und „Zahl der nachfolgenden „Korrektor"-Schritte" kontrolliert werden. Auf dem Predictor-Korrektor Verfahren basierten die ersten effizienten Algorithmen zum Lösen semidefiniter Programme ([VB96]). Wie Puzicha, Hofmann und Buhmann [PHB99] zeigen, kann das „Deterministische Annealing" zu diesen Verfahren gezählt werden.



Negotiation: Die „Negotiation" entstammt zu großen Teilen der künstlichen Intelligenz. Verschiedene Agenten sollen gemeinsam eine Aufgabe lösen, indem sie darüber eine Verhandlung (Negotiation) führen. Dies wird häufig bei verteilten Agentensystemen genutzt, um z.B. jedem Agenten eine Aufgabe zuzuordnen. Eine Variante dieser Methode wurde bereits von Ram und Warren[RW97] im „Timetabling"-Bereich eingesetzt (vgl. Abschnitt 4.8).

Simulated Trading ist eine lokale Simulation eines solchen Handelsvorgangs. Die „Spieler" teilen ihre Strategie einem „Schiedsrichter" mit, der unmittelbar das Ergebnis ermittelt, das nach k Handelsvorgängen, bei denen mehrere Spieler gleichzeitig handeln können, eine im Schnitt optimale Verbesserung darstellt. In der Terminologie von [BHM96]: Ein optimales Trading-Matching wird ermittelt (s.u.).

Equilibrium Programming verbindet „Negotiation" und Predictor-Korrektor Verfahren. Zangwill und Garcia [ZG81] zeigen, daß sich unter moderaten Voraussetzungen an Zielfunktion und Nebenbedingungen Probleme des wirtschaftlichen Gleichgewichts (Equilibriums) durch Einführung eines zusätzlichen Parameters so modifizieren lassen, daß die Punkte, die den jeweiligen Gleichgewichtszuständen entsprechen, eine Kurve im Parameterraum bilden. Diese läßt sich, analog zu den Prediktor-Korrektor-Verfahren, in der Form verfolgen, daß man die Schrittweite der Parameterveränderung ausreichend klein wählt und dann jeweils die Marktkräfte walten läßt, um das neue Equilibrium zu erreichen.

Im Prinzip beruht z.B. „Simulated Annealing“ auf der gleichen Idee: Bei einer bestimmten Temperatur erreicht das System ein „thermales“ Equilibrium. Dann wird die Temperatur geringfügig gesenkt und es besteht die Hoffnung, daß sich die alte und die neue stabile Verteilung soweit überlappen, daß nur wenige Schritte notwendig sind, um zwischen den beiden zu wechseln.

Das hier vorgestellte Verfahren wird im folgenden als „Simulated Negotiation“ bezeichnet: „Simulated“, da ebenso wie bei Bachem, nicht der Prozeß als solcher, sondern allein das Ergebnis interessiert. Die Agenten dienen also nur der besseren Veranschaulichung und könnten durch ein anderes System, das auf anderem Wege das spätere Verhandlungsergebnis bestimmt, ersetzt werden. „Negotiation“ wird hier statt „trading“ verwandt, da jeder Handelsprozeß sofort durchgeführt wird, also gewissermaßen nur das erste „Level“ des „Trading“-Graphen betrachtet wird.

7.2 Das Prinzip des „Simulated Trading“

In diesem Kapitel soll zunächst der Rahmen der Problemstellung besprochen werden. Als Ausgangspunkt bietet sich hier das Verfahren von [BHM96] an, da diese Arbeit und die auf ihr aufbauenden Arbeiten [FMP96], die sich allerdings mehr mit dem dynamischen Fall beschäftigen, die bisher einzigen nachweisbaren Untersuchungen zu genau diesem Thema darstellen. Beide Arbeiten beschäftigen sich mit dem sog. „Vehicle Routing“-Problem, einer Erweiterung des Problems des Handlungsreisenden, in dem es darum geht, mehrere Punkte mit einer fest vorgegebenen Zahl von Routen zu überdecken, wobei je nachdem sowohl die benötigte Zeit als auch die gefahrene Gesamtstrecke zu berücksichtigen sind. Das klassische Problem ist die Versorgung mehrerer Kunden von einem zentralen Depot aus durch eine gegebene Zahl von Fahrzeugen.
Basierend auf der Vorstellung, daß sich die Fahrer untereinander absprechen, schlägt [BHM96] folgende Heuristik vor: Jeder Fahrer bewertet seine Kunden nach dem zusätzlichen Aufwand, den sie für seine Tour bedeuten würden. Mit den entstandenen Verhältnissen als Basis für eine Wahrscheinlichkeitsverteilung wählt er dann zufällig einen der Kunden aus. Danach betrachten alle Fahrer die zum Verkauf stehenden Objekte und wählen mit einer Wahrscheinlichkeit, die durch die Differenz zwischen dem eingesparten Aufwand des Kollegen und dem Eigenen zusätzlichen Aufwand gegeben wird, ein Objekt zum Kauf aus.
Nun wird angenommen, daß alle diese Geschäfte tatsächlich möglich sind und auf Basis des potentiellen Ergebnisses wird der ganze Entscheidungsprozeß mehrere Schritte oft wiederholt. Jeder hierbei für möglich gehaltene Tauschvorgang wird dabei durch eine Kante in einem Graphen repräsentiert: Dieser besteht aus sovielen „Levels“, wie es Wiederholungen gegeben hat. Auf jedem „Level“ repräsentieren die Knoten die Angebote bzw. Anfragen und die Kanten mögliche Handelsvorgänge.
Danach wird durch vollständige Enumerierung der Teilmengen der Kantenmenge des so erstandenen Graphen festgestellt, ob sogenannte „Trading Matchings“ existieren und wenn ja, das mit maximalem Gewinn ausgewählt und umgesetzt. Hierbei unterscheidet sich dieses „Trading Matching“ von einem herkömmlichen Matching insofern, als verlangt wird,

- daß jedes Objekt, das ein Händler verkauft, auch vorher von diesem besessen wurde,

- daß alle Handelsvorgänge, die der Händler auf einem niedrigeren Entscheidungslevel beantragt hat, ebenfalls stattgefunden haben, da sein Vorsatz, den Handel auf dem höheren „Level“ durchzuführen, auf dieser Annahme beruht,
- daß einige problemspezifische Nebenbedingungen ebenfalls erfüllt sind,

was die vollständige Enumerierung notwendig macht. Danach beginnt der gesamte Prozess von neuem.

Das Bestimmen eines optimalen „Trading-Matchings“ ist, wie in [BHM96] gezeigt wird, $\mathcal{NP}$-hart, es wird aber ausgesagt, daß die beteiligten Graphen so wenige Kanten aufweisen, daß dies kein Problem darstellt. Im weiteren werden noch einige Vorschläge zur dynamischen Parametervariation bzw. zum Einbetten in eine Tabusuche vorgebracht.

Läßt man die Anschauung weg, handelt es sich bei diesem Verfahren allerdings lediglich um eine heuristisch eingekleidete Variante von randomisiertem k-Opt[1], da eine bestimmte Anzahl von Objekten zufällig herausgegriffen und dann lokal optimal wiedereingesetzt wird. Es ist fraglich, ob herkömmliches k-Opt, bei dem das Enumerieren vielleicht etwas länger dauern kann, das dafür aber alle Tauschvarianten prüft, nicht effektiver wäre. So z.B. wird im Programm „GPUntis“ [GPU98] deterministisches 3-Opt verwandt.

7.3 Erweiterung auf eine „individuell ausgeglichene“ Methode

Auch die obige Vorgehensweise zeichnet sich dadurch aus, daß sie den Bedürfnissen des Einzelnen nicht gerecht wird. Übernimmt Fahrer A von Fahrer B einen zusätzlichen Auftrag, so erhöht dies primär seine persönliche Belastung, auch wenn es die Gesamtlast senkt. In der Realität - auch in der der Schule - würde dies, sofern es keine Vorschrift für ein derartiges Vorgehen gäbe, so gehandhabt, daß A bei B etwas „gut hat“ und die entsprechende Formulierung wäre, daß B an A einen (moralischen) monetären Ausgleich leisten würde. Es ist auch sofort klar, welcher Spielraum für diese Zahlung sinnvoll wäre: B muß A mindestens seine Unannehmlichkeit ausgleichen und wird höchstens soviel bezahlen, wie er durch Abgabe des Auftrags gewinnt. D.h. wenn der entsprechende Auftrag A p_A Einheiten kostet und B p_B Einheiten gekostet hätte, ergibt sich als Verhandlungsspielraum, in dem das Geschäft zustande kommen kann, das Intervall $[p_A, p_B]$.

7.4 Spieltheorie

7.4.1 Terminologie

Bei dem gerade beschriebenen Prozeß handelte es sich um eine Situation bei der mehrere Objekte (Kunden) auf mehrere Händler (Fahrer) verteilt werden sollen. Um dieses Problem zu lösen, kommt es zu Kooperationen zwischen verschiedenen Gruppen von Fahrern (Knotenmenge eines „Trading Matchings“), bei der durch Zusammenarbeit ein in der Summe günstigeres Ergebnis entsteht. Dies führt sofort zu einigen zentralen

[1] k-Opt ist ein beim TSP häufig angewendetes Optimierungsverfahren. [Jun90]

Begriffen der Spieltheorie [Cur96], wobei an dieser Stelle eine Unterscheidung zwischen kooperativer und nichtkooperativer Theorie vorläufig nicht vorgenommen werden soll:

Kooperatives Spiel Ein Kooperatives Spiel ist ein Paar $< N, v >$ mit $N = \{n_1, \dots, n_n\}$ und $v : \mathbb{P}(N) \to \mathbb{R}$, wobei N die Menge der Spieler ist und v eine Funktion, die jeder Teilmenge der Spieler den Gewinn zuordnet, den sie macht, wenn sich ihre Mitglieder verbünden.

Auszahlungsvektor $x \in \mathbb{R}^n$ gibt an, wieviel von dem Gewinn einem einzelnen Spieler zugeordnet wird. Es sei für $S \subset N$ $x(S) = \sum_{y \in S} x(y)$.

Individuell rational verhalten sich die Spieler, wenn sie ein x mit $x(y) \geq v(y)$ $\forall y \in N$ wählen.

Kern: Der Kern eines Spiels ist $\{x | \forall S \subset N v(S) \leq x(S) \wedge x(N) = v(N)\}$, d.h. die Menge der Auszahlungsvektoren, der gegenüber keine Koalition eine Verbesserung erreichen kann.

Ökonomie mit unteilbaren Elementen Eine „economy with indivisibilities" ergänzt die Menge der Spieler um eine endliche Menge von Objekten P, eine Startkonfiguration, die jedem Spieler einige der Objekte, sowie eine gewisse Menge Geld zuordnet, sowie Relationen $\psi_i \subset (\mathbb{P}(P), \mathbb{R}) \times (\mathbb{P}(P), \mathbb{R})$ $i \in N$, die zu jedem Spieler angeben, ob ihm eine bestimmte Objekt-Geld-Kombination relativ zu einer anderen mehr wert ist. Bei vielen gleichartigen Objekten werden diese häufig zu Vektoren zusammengefaßt. Die obigen Definitionen übertragen sich entsprechend auf diese spezielle Form eines Spiels.

„Price Equilibrium" Das „Price Equilibrium" beruht auf der Vorstellung, daß die Preise sämtlich exogen sind, d.h. daß die Handlungen eines einzelnen Spielers irrelevant für die Preisbildung sind. Unter dieser Voraussetzung ist ein „Price-Equilibrium" ein Preisvektor p, der jedem Objekttyp einen Preis zuordnet und eine Verteilung der Waren, so daß der Markt „leer" (cleared) ist, d.h. alle Ansprüche gesättigt sind. Für eine formale Definition siehe [Cur96].

7.4.2 Resultate

Ausgehend von den obigen Definitionen seien die für diese Anwendung zunächst wichtigsten Resultate genannt.

Satz 7.1 *([Cur96] [HK76]) Ein Equilibrium liegt immer im Kern.*

Satz 7.2 *([Cur96]) In einer „economy with indivisibilities" mit mehr als einem Objekttyp kann der Kern leer sein.*

Bemerkung 7.1 *([HK76]) Die Existenz eines Equilibriums setzt entweder konvexe Präferenzen der Agenten oder aber eine unbegrenzt große Zahl von Agenten voraus.*

Bemerkung 7.2 *([Cur96]) Die Verhandlungsmenge ist nie leer.*

Bemerkung 7.3 *([Gal84]) Will jeder Spieler nur ein Objekt erwerben, so gibt es unter sehr allgemeinen Voraussetzungen ein Equilibrium.*

[HK76] führt zudem noch aus, daß selbst bei Existenz eines Equilibriums nicht gesichert ist, daß die „Kräfte des Marktes" das System gegen ein solches konvergieren lassen. Dieses Verhalten tritt nur in den z.B. in [Bin92] besprochen Fällen auf, in denen nur zwei Typen von Spielern vorkommen. Wie [SS66] zeigen, stellt sich mit wachsender Zahl von Händlern eine approximative Verbesserung der Situation in dem Sinne ein, daß die Gewinne durch Koalitionswechsel gegen Null konvergieren.

7.4.3 Reinterpretation im Rahmen der kombinatorischen Optimierung

Für den unmittelbaren Einsatz in der kombinatorischen Optimierung bringen diese allgemeinen Theorien von Spielen kaum Vorteile, da bereits das Bestimmen von $v(N)$ äquivalent zum Lösen des Problems ist, was sich auch darin äußert, daß einige andere spieltheoretische Begriffe als Mittel über die Summe aller Permutationen der Spieler gebildet werden. Die von [Cur88] und [Cur96] gelösten Spielprobleme mit bezug zur kombinatorischen Optimierung erweisen sich demzufolge auch allesamt als Repräsentationen von polynomiell lösbaren Aufgaben, wie Matchings oder Spannbaumproblemen. Die Nichtexistenz des Kerns bzw. eines kompetetiven Equilibriums oder eines konvergierenden Verfahrens ist dabei nicht so hinderlich, wie es zunächst erscheint. Diese ist die Folge der Tatsache, daß sämtliche Teilnehmer bzw. der hier immer präsente Beobachter, bereits vor Beginn des Spiels über all ihre Möglichkeiten informiert sind. Da die Menge der Objekte, die des Geldes und die der Spieler begrenzt sind, müßte, da somit nur endlich viele Zustände gegeben sind, im Rahmen tatsächlicher Handelsvorgänge ein Kreisen auftreten. Da aber nicht alle auf einem solchen Kreis beständig Gewinn machen würden, würde die Verweigerung eines der Verlierer ausreichen, um das Kreisen zu stoppen. Dies steht nicht im Widerspruch zu den obigen Sätzen, da der blockierende Spieler bei der reinen Vorüberlegung natürlich kein derartiges Vetorecht hätte, da er die entsprechenden Objekte/Gelder noch nicht besäße. Es steht vielmehr im Einklang mit [HK76], die ausführen, daß das jeweilige Equilibrium nur im Sinne einer vorher gegebenen Verteilung der Werte gegeben sein kann.

Beeindruckend ist die Allgemeinheit des Ergebnisses von GALE[2], allerdings können die angegebenen Bedingungen nur als heuristische Hinweise dienen, da es in dem dort gegebenen Modell dennoch nicht möglich ist, mehrere Spieler so aufeinander abzustimmen, daß sie in ihrer Gesamtheit so handeln wie ein einzelner.

7.4.4 Fazit

Da sich somit ergibt, daß für den im „Timetabling" betrachteten Fall mit einer großen, aber endlichen Zahl von Spielern, einer endlichen Menge von Objekten und strikt nichtkonvexen Präferenzen weder die Existenz eines Equilibriums sicher ist, geschweige denn die Konvergenz eines ökonomischen Prozesses gegen ein solches gezeigt werden kann, können die Ergebnisse dieses Bereichs der Spieltheorie hier nur insoweit umgesetzt werden, als die Hoffnung bestehen kann, daß die Wahrscheinlichkeit einen stabilen Zustand zu erreichen, steigt, wenn man diejenigen notwendigen Kriterien berücksichtigt, die bei einer konvexen, unendlichen oder speziellen Situation die Existenz von Equilibria sichert.

[2] Wobei mir nicht bekannt ist, ob es eine Anwendbarkeit in der Matchingtheorie gibt.

Ansonsten verbleibt nur das externe Steuern des Marktes. Hier gibt es z.B. die Möglichkeit von Steuern, Abgaben oder aber auch die des direkten Eingriffs in die Zielfunktionen der Händler. Eine Variante hierzu findet sich insbesondere beim Equilibrium Programming, bei dem das in [HK76] geschilderte Verfahren durch eine externe Manipulation der Preise realisiert wird, wobei allerdings streng konvexe Vorlieben der Spieler gefordert werden.
Eine andere Möglichkeit besteht darin, die Händler in ihren Möglichkeiten, und vor allem auch in ihrem Wissen einzugrenzen. Binmoore [Bin92] bezeichnet solche Situationen mit unwissenden Spielern als Fälle von „Bounded Rationality“ und zeigt auf, wie ein Spiel in solchen Situationen konvergieren kann.
Der Rest dieses Abschnitts beschäftigt sich daher mit Experimenten zu eingeschränkten Handelssituationen. Hierbei wird dennoch versucht eine Verbindung zur Spieltheorie aufrecht zu erhalten, da, wenn man auf die Ausnutzung dieser Analogie verzichtet, die einzige Möglichkeit das Verfahren zu bewerten in einer rein numerischen Überprüfung bestünde, wie sie auch bei Bachem vorgenommen wird.

7.5 Ablauf der Auktion und Strategien

7.5.1 Auktionsschemata

Eine erste Einschränkung der Möglichkeiten der Händler kann dadurch erzielt werden, daß ihnen ein bestimmtes Handelsschema auferlegt wird, so daß beliebige Tauschoperationen nicht mehr möglich sind.
Im Verlauf der Zeit wurden zahllose Methoden entwickelt, Auktionen abzuhalten, um Ressourcen jedweder Art mit maximalem Profit oder nach irgendwelchen Gerechtigkeitskriterien zu verteilen. Am günstigsten wäre es natürlich, im Rahmen einer einzigen Auktion sofort für allumfassende Zufriedenheit zu sorgen. Dies ist aber im gegebenen Fall nicht möglich, da sich die Wünsche der Teilnehmer mit Erwerb von Objekten ändern bzw. bei vorausschauender Berücksichtigung dieser Änderungen das Auktionsschema eine Lösung des Gesamtproblems darstellen würde.
In der ökonomischen Literatur liegt ein wesentlicher Schwerpunkt beim Design von Auktionen auf Überlegungen, die die Manipulierbarkeit des Marktes durch Händler betreffen, die besser über die Wünsche ihrer Konkurrenten informiert sind, als diese über die ihren. Dies tritt natürlich im Rahmen einer Simulation nicht auf[3], so daß es kein Problem darstellt. Dennoch werden im folgenden Auktionsschemata bevorzugt, bei denen diese Gefahr relativ gering wäre, da damit auch der probabilistische Anteil bei der Strategiebestimmung umgangen werden kann. In der Sprache der Spieltheorie, die bezüglich Auktionen wesentlich brauchbarere Ergebnisse liefert, heißt dies, daß die „ehrliche Strategie“ jeweils dominant sein soll, d.h. bei gleichem Verhalten der Gegner stets den anderen Strategien vorzuziehen ist.

- Das einfachste Beispiel für eine solche Auktion ist „Vickrey's Second Price Auction“. Bei dieser Form gibt jeder Spieler sein Angebot geheim ab und der mit dem höchsten Angebot erhält das Objekt zum Preis des zweithöchsten Angebots. In [Bin92] wird gezeigt, daß das erwartete Ergebnis dieser Auktionsform mit dem der „herkömmlichen“ Auktion übereinstimmt.

[3] Bzw. es tritt nur dann auf, wenn man es explizit vorsieht.

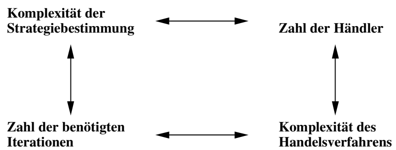
- Laut Miyake [Miy98] eine approximative Verallgemeinerung des obigen Vorgehens ist die „Englische Auktion“. Hier wird ein gemeinsames Preislevel für alle Beteiligten nach und nach angehoben, wobei die Teilnehmer jeweils angeben, ob sie noch weiter mitbieten. Ein Objekt zählt dann als zum momentanen Preis verkauft, wenn nur noch ein Käufer an ihm interessiert ist.
- Keine Auktion im eigentlichen Sinne, sondern mehr ein Verteilungsverfahren, das in einer gewissen Weise dem Heiratsalgorithmus [Sed92] ähnelt, ist das Verfahren von Crawford und Knoer [CK81]. Ausgehend von einem Minimalpreis werden auch hier die Preise immer weiter angehoben. Ein Verkäufer kann sich temporär für einen Käufer entscheiden; dann bleibt dieser Preis stehen. Entscheidet er sich später um, so wächst dieser Preis von dem angegebenen Niveau aus weiter.

7.5.2 Aufgabenbereich der Händler und Strategiebestimmung

7.5.2.1 Durch die Komplexität gesetzte Grenzen

Da in die Überlegungen der Spieltheoretiker die Zeit, die zur Bestimmung der Strategien und zur Ermittlung eines Handelsprozesses notwendig ist, meist nur insofern einfließt, als überlegt wird, daß die entsprechenden Funktionen überhaupt (in endlicher Zeit) berechenbar sind, muß die Wahl einiger entscheidender Grundeigenschaften unter weitgehender Vernachlässigung dieses Kontextes getroffen werden.

Ist es, um die Existenz eines Equilibriums zu fördern, auch sinnvoll, die Zahl der Spieler sehr groß zu wählen, so setzt die Umsetzbarkeit der entsprechenden Handelsmechanismen hier harte Grenzen. Insgesamt sieht man sich mit der Situation konfrontiert, daß, bezeichnet man die Zahl der Spieler mit s, den Aufwand für die Strategiebestimmung mit a, die Zahl der benötigten Iterationen mit i, und die Komplexität des Handelsverfahrens mit h, sich der benötigte Gesamtaufwand ungefähr gemäß $i(h(s)+as)$ verhält, während diese Werte untereinander in einem reziproken Wachstumsverhältnis stehen.



Hierbei muß allerdings beachtet werden, daß der Aufwand für das Handelsverfahren ($h(s)$) mit wachsender Händlerzahl ansteigt, auch wenn das Verfahren als solches einfacher wird.

Als erstes Problem bei einer Modellierung des „Timetabling“-Problems im Rahmen des oben dargestellten Kontextes stellt sich also die Frage, was die Objekte sein sollen und wie sie auf die Händler zu verteilen sind. Will man hier nicht bestimmte Objekte den anderen vorziehen, so bietet es sich an, jeder Gruppe einen Händler zu assoziieren, was der intuitiven Vorstellung „Englisch in der 5 soll so gut behandelt werden, wie Mathematik in der 7“ wohl auch am nächsten kommt. Die ersten in dieser Arbeit vorgenommenen Testimplementierungen folgten auch dieser Überlegung.
Es ist aber so, daß diese Methode insbesondere was die betrachtete Realschule anbelangt, bei der viele Kurse nur aus einer einzelnen Stunde bestehen, zu einer sehr großen Zahl von Händlern führt (ca. 300) . Z.B. das Verfahren von Bachem, das zur Bestimmung des „Trading Matchings“ Backtracking verwendet, kann mit einer so

großen Zahl von Händlern nicht effizient implementiert werden.[4] Einfachere Auktionsschemata hingegen können dies bewältigen.
Das nächste Problem liegt in der Häufigkeit der Strategiebestimmung. Bei der „Englischen Auktion“ und dem Verfahren von [CK81] muß nach jeder Erhöhung des Preisniveaus jeder Händler seine Strategie von neuem überdenken. Dies ist wiederum zu aufwendig, da auch bei diesem Verfahren nur bei hohen Iterationszahlen gute Ergebnisse zu erwarten sind.

7.5.2.2 Mögliche Modelle

Wie bereits angedeutet besteht eine Möglichkeit darin, für jeden zu verteilenden Kurs einen Händler einzurichten, was insbesondere den Vorteil hat, daß er nur einen Typ von Gruppe erwerben muß und daher die Entscheidung entfällt, welche der erworbenen Ressourcen er welcher Gruppe zuordnen soll. Der damit sehr großen Zahl von Händlern muß entsprechend dadurch Rechnung getragen werden, daß z.B. ein einfaches „Second-Price“ Auktionsmodell gewählt wird. Da dieses im Gegensatz zu den beiden anderen keine unmittelbare relative Preisentwicklung bietet, muß hier als „Preis“ das jeweils beste Angebot der letzten Runde gewählt werden.

Im rein schulischen Kontext hingegen könnte eine Bevorzugung der Klassen sinnvoll sein, da der pädagogische Auftrag hier Vorrang vor den anderen Überlegungen hat. Zudem ist es so, daß die Klassen nahezu soviele Stunden Unterricht haben, wie für sie zur Verfügung stehen, so daß trotz des mit dem Zusammenfassen von Klassen verbundenen Sinkens der Händlerzahl die Komplexität der Strategiebestimmung nicht steigt, da nun nur die wenigen Stunden bestimmt werden müssen, an denen die Klasse frei hat. Ein Vorgang der in etwa der Tatsache entspricht, daß das Finden von sehr großen Cliquen in Graphen ähnlich einfacher ist wie das von solchen geringer Größe.

7.5.3 Strategiebestimmung

Die einfachste Form der Strategiebestimmung bestünde darin, keine Strategie zu verwenden, sondern die Händler ihr Guthaben je nach Wert auf alle Objekte verteilen zu lassen. Dies hätte aber den Nachteil, daß bei diesem Vorgehen der größte Teil des Geldes unberührt bliebe, da stets nur ein geringer Teil der möglichen Objekte erworben wird.

Zunächst scheint es sinnvoll zu sein, daß jeder Händler die beste Konfiguration erwerben will, die er für das ihm zur Verfügung stehende Geld bekommen kann. Wie sich aber herausstellt, ist die Bestimmung dieser Menge bereits wieder $\mathcal{NP}$-hart. Würde man zunächst noch vermuten, daß das genaue Wissen um die benötigte Zahl der Objekte hier eine Hilfe wäre, so erkennt man doch, daß sich das allgemeine „Bin-Packing“-Problem darauf reduzieren läßt, sofern man die gewünschte Anzahl frei, d.h. insbesondere größer als die Zahl der Objekte in einem optimalen „packing“ wählen kann:
Dazu genügt es, bei diesem zu jeder Paketgröße eine feste (große) Konstante zu addieren, sowie einige „Hilfspakete“ hinzuzufügen, deren Größen genau dieser Konstante entsprechen. Setzt man dann die neue „Bin“-Größe auf ein Vielfaches dieser Konstante zuzüglich der alten „Bin“-Größe, so werden offenbar stets entsprechend viele

[4] Die Zahl der Händler bei [BHM96] liegt in der Größenordnung von 30.

Pakete verwandt, und diejenigen unter ihnen, die keine Hilfspakete sind, stellen eine Lösung des Ausgangsproblemes dar. Dabei ist allerdings anzumerken, daß es für dieses Problem sowohl einen pseudopolynomiellen Algorithmus, der auf dynamischer Programmierung beruht [Sed92], als auch polynomielle Approximationsschemata [Hoc97] gibt. Obwohl hier im Schnitt nur drei oder vierelementige Mengen durch vollständiges Durchsuchen geprüft werden müssen, was im Prinzip in $\mathcal{P}$ liegt, ist dies zu langwierig um entsprechend häufig durchgeführt zu werden.

Eine erste Möglichkeit besteht darin, das verbleibende Budget gleichmäßig aufzuteilen, und die jeweils besten Objekte zu ermitteln, die einzeln diese Grenzen nicht überschreiten. Dies führt aber natürlich zu einer hohen Wechselrate bei den ausgewählten Gruppen.

Daher könnte nur eine Veränderung von je einem Zeitslot zugelassen werden. D.h jeder Händler kann pro Handelsrunde sein Interesse an einer Zeitstunde aufgeben und stattdessen für Objekte aus einer neuen Zeitstunde bieten. Diese einfache Form des Übergang erlaubt es zudem, bei der Wahl einer neuen Zeitstunde zu berücksichtigen, wieviel beim Verkauf einer passenden alten Stunde gewonnen werden kann. Im folgenden sei dieser Vorgang so er angewandt wird - damit bezeichnet, daß ein Händler die eine Zeitstunde zugunsten der anderen aufgibt. Hierbei muß allerdings darauf geachtet werden, daß der Händler auch die nichtveränderten Zeitstunden immer wieder prüft, da sich hier die Preise der dynamischen Objekte seiner Gruppen ändern können.

Zusätzlich tritt hier aber noch das Problem der flexiblen Ressourcen auf. Wie auch bei den Zeitstunden ist es $\mathcal{NP}$-hart, aus einem gegebenen Angebot die besten zu einem vorgegebenen Maximalpreis auszuwählen. Hier kann entweder „greedy“ nach dem besten Preis/Leistungsverhältnis vorgegangen werden oder aber wiederum eine feste Obergrenze für jedes einzelne Objekt verwandt werden.

7.5.4 Erzwingen des Anhaltens

Auch wenn die Existenz eines Equilibriums nicht gesichert werden kann, so sollte es, in Anlehnung an die obigen Überlegungen zu Homotopien, einen Punkt geben, an dem das Ganze anhält, so daß die Wahrscheinlichkeit vergrößert wird, daß bei Übergang von einer Konfiguration zur nächsten nur wenige Schritte notwendig sind, um wieder in einen solchen Stundenplan zu gelangen.
Hierzu werden in der ersten Testimplementierung zwei Verfahren eingesetzt: Das erste Verfahren ist,vom Handelsverhalten her noch sinnvoll, daß ein Händler ein Objekt, an dem er selbst noch interessiert ist, auf jeden Fall zu einem höheren Preis als seinem Wert für ihn verkaufen muß. Würden die Agenten also nie ihre Wünsche ändern, würden die Preise monoton steigen und die begrenzte „Geldmenge“ würde irgendwann ein Anhalten erzwingen. Es muß also noch dafür gesorgt werden, daß es nicht zu einem Kreisen in der Menge der von den Händlern angeforderten Zeitstunden kommt.
Hier hilft die folgende Beobachtung: Im Falle eines Kreisens müßte es stets vorkommen, daß ein Händler eine Zeitstunde zugunsten einer anderen Zeitstunde aufgibt, zugunsten der er erstere bereits schon einmal aufgegeben hat.
Es erscheint im Sachkontext nicht vollkommen abwegig anzunehmen, daß er gelernt hat, daß ihm dies nichts bringt und dies somit zu unterbinden. Da es nur endlich viele Zeitstunden gibt, die Menge aller geordneten Paare also ebenfalls beschränkt ist, heißt dies, daß alle Händler nach einer endlichen Zahl von Schritten ihre Wunschzeitstunden

nicht mehr ändern könnten, woraufhin die obige Preispolitik zu einem Erliegen des Handels führen würde. Bei den anderen Methoden der Strategiebestimmung läßt sich dieses Verfahren allerdings nicht anwenden. Hier könnte man z.B. verlangen, daß jedes Objekt nur über seinem Einkaufspreis verkauft werden darf. Dies würde zwar ein Anhalten garantieren, erscheint aber im Sachkontext nicht sinnvoll.

7.5.5 Kooperation und Wahl des „Homotopie“ - Parameters

Zunächst würde es kanonisch erscheinen, den Geldwert in Abhängigkeit von der Zeit sinken zu lassen. Da dieser aber für alle Spieler gleichmäßig sinken würde, würde der Markt dies durch eine gleichmäßige Erhöhung aller Preise ausgleichen, so daß der Effekt nur der wäre, daß Geschäfte ab einen bestimmten Umsatz aus Geldmangel nicht mehr möglich sind. Daher sollte der „Zeitparameter“ derart in die individuellen Zielfunktionen der Spieler integriert werden, daß deren Wunsch nach vollständigen Gruppen immer größer wird. Somit könnte man mit konvexen Zielfunktionen beginnen, die die Existenz eines ersten Equilibriums sichern. Dies würde aber das System bereits ab einem sehr frühen Zeitpunkt erstarren lassen, da beim Kauf eines Objektes sozusagen alle Objekte bezahlt werden müßten, die der Verkäufer in dieser Zeitstunde besitzt. Daher könnte so vorgegangen werden, daß der Käufer einen festgelegten Anteil an den übrigen Besitztümern des Verkäufers erwirbt und anteilhaft ausgezahlt wird, wenn diese verkauft werden. Im Verlauf des Handels wird dieser Anteil immer weiter erhöht, bis am Ende stets der ganze Set bezahlt werden muß.
Diese Art der Verzerrung liefert aber noch kein einfaches Minimum für den Anfang. Um dies zu erreichen, werden die Objekte zu Beginn von außen, d.h. von einem reinen Verkäufer und mit stetig sinkenden Preisen in den Markt eingeführt. Damit erhält man als triviales Ausgangsequilibrium den Zustand, in dem keiner der Händler einen Gegenstand besitzt. Voraussetzung hierfür ist aber wiederum, daß Geld als solches für die Händler einen Wert besitzt, da er ansonsten sein gesamtes Geld für den Ersterwerb ausgibt und dieses somit für die Handelssituation verloren wäre.
Dies deckt sich insbesondere mit den Bedingungen von Gale [Gal84], der verlangt, daß kein Spieler für irgendein Objekt sein gesamtes Geld ausgeben darf, und daß jeder Spieler eine Grenze hat, jenseits der er nichts investiert.

7.6 Erste Versuche einer Implementierung

Allgemeine Anmerkung zu Simulationen
Eine Simulation nach der obigen Beschreibung weist keine weiteren Besonderheiten auf. Da sie keinen Zeitvorgaben unterworfen ist, treten die bei Simulation anderer reeller Vorgänge zu beachtenden Synchronisationsprobleme nicht auf. Des weiteren kann momentan kein Vorteil darin erkannt werden, die Handelsvorgänge in einer besonderen Reihenfolge ablaufen zu lassen, so daß hier jeweils eine beliebige gewählt wurde, um Symmetrien vorzubeugen. Wie auch bei der Tabusuche war es notwendig, auf die Zielfunktionen ein leichtes Rauschen aufzumodellieren, so daß sich nicht alle Händler, denen alle Zeitstunden recht wären, unnötig lange um die gleiche Stunde bemühen. Das Ganze wurde so implementiert, daß die Handelsstrategien unabhängig von der Auktion modifiziert werden können.

Der erste Versuch
Zunächst wurde ein Verfahren implementiert, bei dem jeder Händler in jeder Handelsrunde seine Wunschobjekte vollständig neu nach dem besten Preis-Leistungs-Verhältnis bestimmen kann und sein Geld einfach jeweils zu gleichen Teilen verteilt. Die bestehenden Besitzstände wurden nur insofern berücksichtigt, als der jeweilige Händler wußte, daß er diese kostenlos erhalten könnte. Auf einem kleinen Testfall gestartet, zeigte dieses Modell ein konvergentes Verhalten, bei dem das Handeln häufig schon nach einigen hundert Iterationen zum Stillstand kam. Die Qualität der gefundenen Lösungen zeigte sich aber unabhängig davon hochgradig sensitiv gegenüber kleinsten Parameterveränderungen.

Der zweite Versuch
Als nächstes wurde versucht, die Bestimmung der neuen Zeitstunden durch vollständiges Durchsuchen vorzunehmen, da es erträglich schien, alle drei oder vierelementigen Teilmengen der ca. 30 Zeitstunden zu betrachten. Da aber wegen der dynamischen Ressourcen zu jeder Überprüfung ein Neubestimmen der Ressourcenmenge notwendig ist, erwies sich dies doch als wesentlich zu langwierig, und das Verfahren konnte in einigen Testläufen nie in annehmbarer Zeit die erste Iteration abschließen.

Eine fortgeschrittenere Implementierung
In einem dritten Versuch wurde das fluktuationsbeschränkte Modell der Strategiebestimmung verwandt und auch die zusätzliche „Tabuliste“ implementiert. Dem Geld wurde ein Wert zugewiesen, der in Abhängigkeit davon, ob ein Händler seine Bedürfnisse befriedigen konnte, gehoben oder gesenkt wird: Besitzt ein Händler zu viele Objekte, wird seine Wertschätzung des Geldes solange angehoben, bis er den Überschuß verkauft hat. Bei zu wenigen Objekten sinkt sein Geldwert, allerdings nur bis zu einer bestimmten unteren Grenze, damit die Bedingungen von Gale eingehalten werden.
Da das Ganze trotz dieser Vereinfachungen immer noch zu langsam wäre, wurde jeder „Händler“ zusätzlich noch in ein „Frontend“ und ein „Backend“ aufgespalten. Das „Backend“ übernimmt dabei die eigentliche Strategiebestimmung und wird nur jeweils nach einer vollständigen Handelsrunde, die aus je einer „Second-Price-Auktion“ für jedes Objekt besteht, aufgerufen. Das „Frontend“ wird nach jedem Handelsvorgang, an dem der betreffende Händler beteiligt ist, aktiviert und paßt bei konstant gehaltenen Objektwünschen, die einzelnen Angebote der veränderten Geldsituation durch einfaches Skalieren an. Da hierbei jedesmal mehrere Objekte gehandelt werden, an denen der Händler interessiert ist, kann dabei natürlich die langfristige Strategie verfehlt werden, so daß die entsprechenden Objekte in der nächsten Runde möglicherweise mit Verlust wieder abgestoßen werden müssen. Dieses Verfahren setzte immerhin schon 50% der Schulstunden,wobei die Probleme in erster Linie darin bestanden, daß die Händler am Ende manchmal nicht zusammenpassende Objekte besaßen, und teilweise der Geldwert noch zu langsam sank, so daß keine Objekte erworben wurden.

Beide Implementierungen haben damit noch nicht den Stand erreicht, der es erlauben würde, auswertbare Ergebnisse zu erzeugen. Beim ersten Versuch war dies von vorneherein nicht beabsichtigt; er diente lediglich dazu, einen Teil der Schwierigkeiten, die normalerweise nur im Rahmen einer praktischen Umsetzung erkannt werden können, sichtbar werden zu lassen. Hier fiel besonders das Problem der fehlenden Preisbildung auf.

Das zweite Verfahren stellt vermutlich bereits einen zu großen Sprung gegenüber dem ersten dar. Obwohl die erwähnte Möglichkeit, Anteile zu erwerben, noch nicht integriert wurde, erwies sich die Abstimmung der Parameter insbesondere auch auf wechselnde Problemgrößen als zu aufwendig, um in der diesen Versuchen von mir zugebilligten Zeit erfolgreich durchgeführt werden zu können. Andererseits ist aber auch fraglich, ob ein Test, der nicht dem vollen Modell entspricht, erfolgreich verlaufen kann, da doch eine sehr hohe Interaktion der einzelnen Bestandteile notwendig ist. Das in [Bin92] beschriebene Prinzip der „Bounded Rationality", d.h. der Tatsache, daß relativ einfach konstruierte Spieler in einer Simulation zum Equilibrium gelangen können, das ein Vorgehen nach dem „Prototyping"-System nahelegte, scheint hier doch nicht so weit zu reichen.

7.7 Zusammenfassung

Idee des Verfahrens war es, die in unserer Gesellschaft übliche Form der Ressourcenzuteilung durch die Simulation von Handelsvorgängen auf das Problem der automatischen Stundenplanerstellung zu übertragen. Angesichts der Tatsache, daß hier Ergebnisse aus einem Gebiet außerhalb der Informatik herangezogen werden müssen, wurde zunächst eine theoretische Fundierung versucht, wobei aber lediglich falsifizierende Ergebnisse der Form erzielt wurden, daß nur für nicht anwendbare Spezialfälle ein gesichertes gutartiges Verhalten der Methoden zu erwarten ist.

Erste Versuche mit einfachen Spielern zeigten sich zwar einer randomisierten Verteilung überlegen, aber die Abstimmung aller Parameter in einer so komplexen Situation kann nur im Rahmen eines größeren Projektes effektiv durchgeführt werden. Versuche diese dynamisch während der Laufzeit festlegen zu lassen, wurden unternommen, führten aber nur zu geringfügigen Verbesserungen.
Dennoch erscheint insbesondere im Rahmen einer von persönlichen Wünschen geprägten Planungsaufgabe dieses Vorgehen als sinnvoll und weitere Anstrengungen sollten vielleicht zunächst im Rahmen wesentlich einfacher strukturierter Probleme - wie z.B. des Vehicle Routing (s. Abschnitt 7.2) - unternommen werden. Die realistische automatische Stundenplanerstellung erwies sich jedenfalls für die Ersteinführung einer neuen Methode als zu vielschichtig.

Kapitel 8

Implementierung des Rahmenprogrammes

Dieses Kapitel beschreibt die Implementierung des Bearbeitungs- und Rahmenprogrammes. Die Implementierung der Lösungsverfahren wird in den jeweiligen Kapiteln beschrieben.

8.1 Datenstrukturen

- Wesentlichstes Element ist ein assoziatives Array, das als universelles und dynamisches Hash [Blu98] gestaltet wurde[1]. Hierbei wurden zur Kollisionsbehandlung nicht verkettete Listen, sondern Binärbäume verwandt, um eine weitere Beschleunigung zu erhalten. Von der ursprünglichen Absicht, diese zusätzlich zu balancieren, um so eine Datenstruktur mit Zugriffszeiten von $\mathcal{O}(1)$ im „Average-Case“ und $\mathcal{O}(\log n)$ im „Worst-Case“ zu erhalten, wurde abgesehen, da erste Versuche zeigten, daß selten mehr als 5 Elemente pro Baum auftraten. Mit dem Umkopieren bei Vergrößerung des Hashs wird jeweils gewartet, bis der Belegungsfaktor den Wert 2 erreicht hat.
- Auch wenn keine Balancierung integriert wurde, finden in obiger Datenstruktur immer noch relativ viele Kopieroperationen statt. Daher wurde diese Struktur in erster Linie dazu verwendet, Zeiger auf die Elemente einer normalen verketteten Liste zu verwalten. Der Zugriff verlängert sich hierdurch nur um einen weiteren Dereferenzierungsschritt. Zudem tritt häufig der Fall auf, daß die Objekte unabhängig von den Schlüsseln durchmustert werden müssen. Beides wird durch diese Kombination gewährleistet.
- Die ansonsten komplizierteste Struktur stellt das „Mengensystem“ dar. Jede Menge wird mit je zwei Listen versehen, die Zeiger auf die jeweiligen Ober- und Untermengen beinhalten. Bei der Speicherung werden diese durch die Indizes der entsprechenden Mengen im „Mengensystem“ ersetzt; beim Laden werden die Zeiger erneut bestimmt. Da die Größe jeder Menge somit nur rekursiv bestimmt werden kann, wird sie stets in einer Variablen gehalten, so daß sie bei

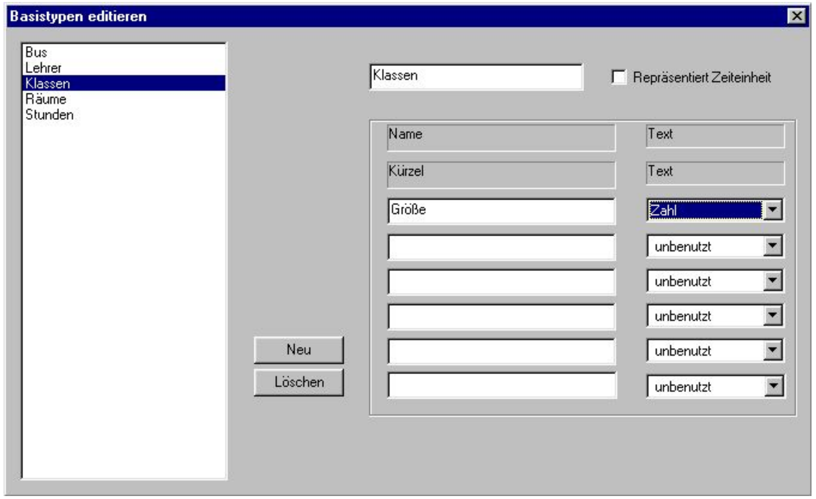
[1] Visual C++ bietet ein ähnliches Konstrukt an, warnt aber, daß dieses sehr viele Kopieroperationen benötige.

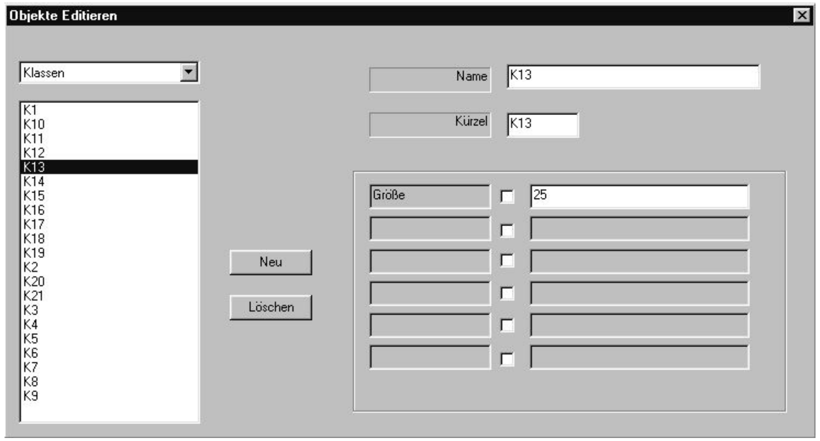
Vereinigungs- oder Löschvorgängen unmittelbar aus den ebenfalls gespeicherten Größen der Teilmengen bestimmt werden kann.

- Während der Bearbeitung des Stundenplans ändern sich die Daten vom Mengeneditor abwärts nur selten. Daher werden die wichtigsten Ergebnisse, wie z.B. die Struktur des Mengensystems, zusätzlich in Matrixform zwischengespeichert, die einen schnelleren Zugriff erlaubt. Da diese Matrizen nach größeren Veränderungen jeweils anhand der eigentlichen Strukturen (vgl. Abschnitt 8.4.1) neu erstellt werden, kann hier die Indizierung wesentlich effektiver erfolgen.

8.2 Funktionalität und Implementierung des Editors

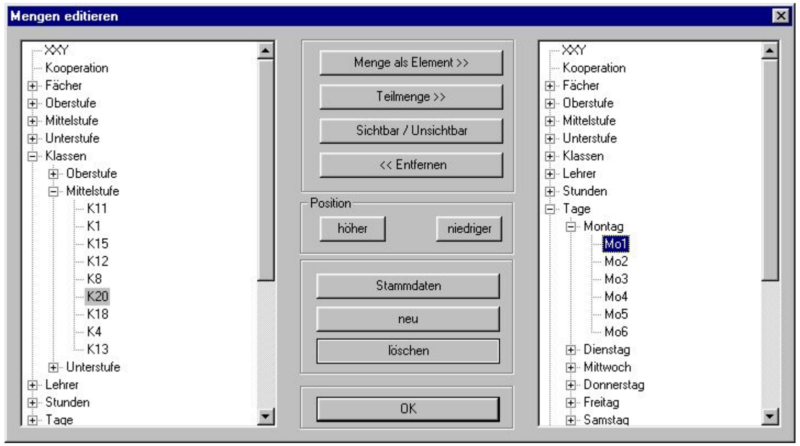
8.2.1 Objekttypen- und Objekteditor





Die beiden Abbildungen zeigen den Objekttypen- und den Objekteditor. Der Typeneditor dient in erster Linie dazu, den Objekteditor zu konfigurieren, indem eingegeben werden kann, welche Bezeichnungen im Objekteditor auftreten sollen bzw. wie die entsprechenden Felder zu interpretieren sind. Als einzige „wirkliche“ Information wird hier festgelegt, welche Objekttypen als Zeiteinheiten zu verstehen sind.
Der Objekteditor erlaubt das Erzeugen neuer Objekte und das Zuordnen der jeweiligen Attribute, wobei diese im Mengensystem dazu verwandt werden, automatisch Mengen zu generieren (s. Abschnitt 8.2.2). Der Übersichtlichkeit halber werden die Objekte nach Typen sortiert angezeigt.

8.2.2 Mengeneditor

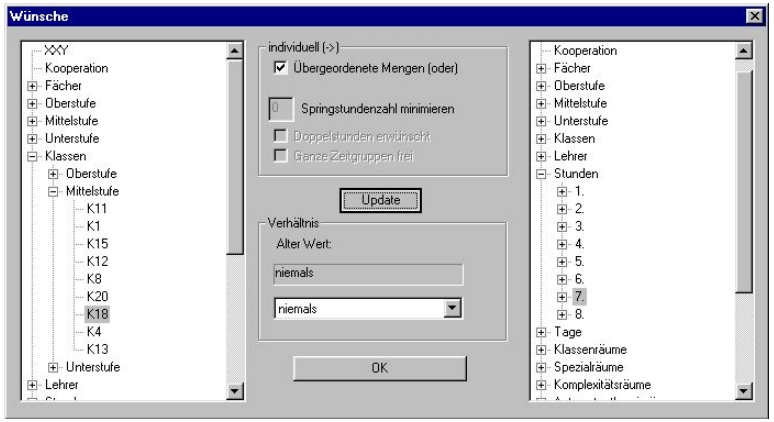


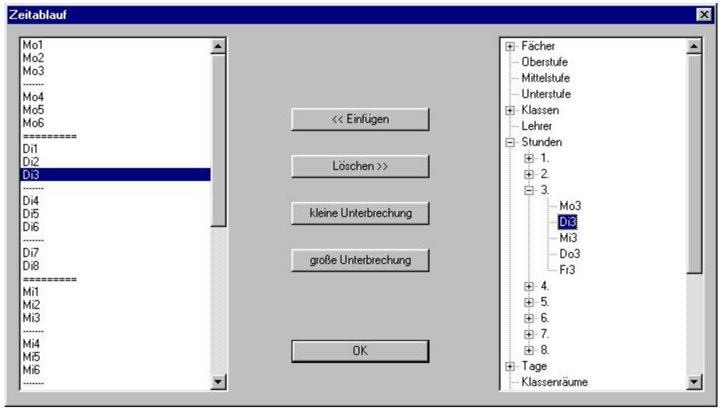
Der Mengeneditor ermöglicht es, Mengen zu erstellen, zu löschen und Teil bzw. Übermengenbeziehungen herzustellen. Das Mengensystem, das azyklisch, aber nicht im ungerichteten Sinne kreisfrei (i.e. kein Wald) ist, wird durch die „CTree“ Struktur von Visual C++ dargestellt, indem jede Menge mit der Zahl ihrer Übermengen vielen Repräsentanten assoziiert wird. Entsprechend kann jede Menge sooft dargestellt werden, wie es Pfade von der (virtuellen) Wurzel des Mengensystems zu der jeweiligen Menge gibt. In der praktischen Anwendung ist aber nicht damit zu rechnen, daß ein Pfad eine Länge größer als 4 annimmt. Die Unterscheidung zwischen Teilmengen und Elementmengen wirkt sich nur auf die Ansichten aus. Einzelne Objekte können als „unsichtbar“ markiert werden. Dies führt dazu, daß sie auf der obersten Ebene der Hierarchie des Mengeneditors nicht mehr angezeigt werden und als Ausgangsmenge für Ansichten (s. Unterkapitel 8.2.4) nicht mehr zur Verfügung stehen. Hierbei wird darauf geachtet, daß dennoch für jede Menge eine Repräsentation im Mengeneditor existiert, was wegen der Möglichkeit, daß die Übermenge über die dieser Zugriff erfolgt, gelöscht werden kann, relativ kompliziert ist. Nach Abschluß der Editierarbeiten findet eine Überprüfung dahingehend statt, daß die gewählten Relationen azyklisch sind.
Der Mengeneditor war von der Implementierung her besonders aufwendig, vor allem, was die unsichtbaren Mengen angeht. Fügt der Benutzer z.B. einen Zyklus ein (vor dem er erst bei Beenden des Editors gewarnt wird) und macht dann alle Mengen, die

zu diesem Zyklus gehören unsichtbar, so besitzt jeder der Knoten einen Vorgänger, so daß dies zunächst erlaubt ist. Auf diesem Wege können aber alle Mengen des Zyklus unsichtbar werden. Dieses Problem wurde auf dem Wege behoben, daß bei Finden eines Zykluses automatisch alle beteiligten Mengen wieder sichtbar werden.

8.2.3 Bewertungs- und Zeiteditor

Der Bewertungseditor dient zur Eingabe absoluter und relativer Parameter der Zielfunktion. Im oberen Kästchen in der Mitte stehen die absoluten Parameter. Einstellungen hier wirken sich auf die im rechten Listenfeld selektierten Objekte aus. Im unteren Kästchen werden der berechnete und der eingestellte Wert der Relation zwischen dem im linken Listenfeld und dem im rechten Listenfeld selektierten Objekt angezeigt. Hierbei dominiert stets der eingestellte Wert den berechneten Wert vollständig, es sei denn, es wurde „egal“ ausgewählt bzw. bei den absoluten Parametern das entsprechende Kästchen aktiviert. Die berechneten Werte der absoluten Parameter ergeben sich als Mittelwert der entsprechenden Einstellungen der Obermengen, wobei durch Auswertung in topologischer Reihenfolge, was aufgrund der Azyklität des Mengensystems möglich ist, die Obermengen stets vor der Bestimmung dieses Parameters ausgewertet werden können. Zur Propagierung der relativen Parameter siehe Abschnitt 8.4.2.

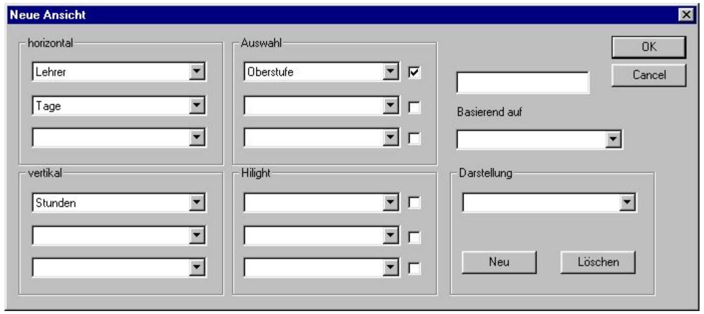


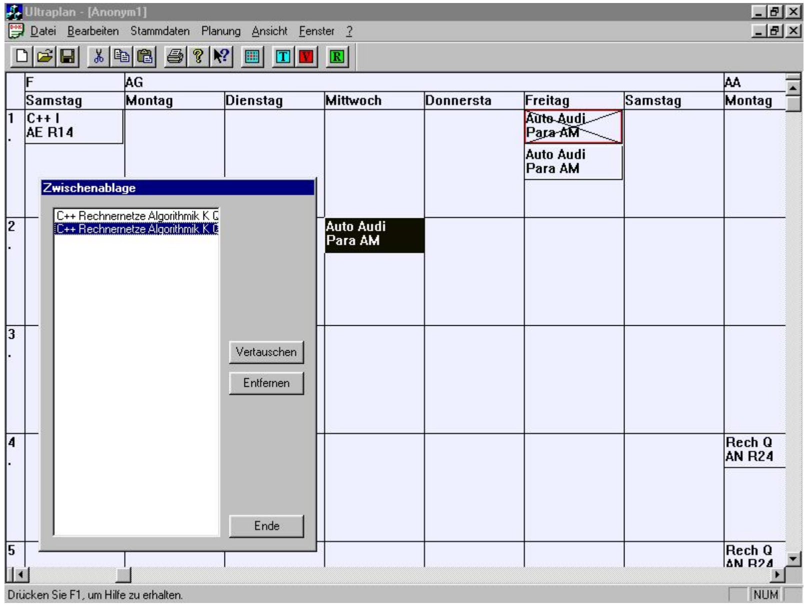


Der Zeiteditor erlaubt es, die relative Anordnung der Zeiteinheiten zueinander anzugeben und festzulegen, welche Stunden am selben Tag bzw. zwischen zwei Pausen stattfinden. Diese Anordnung wird für die relativen Nebenbedingungen, insbesondere für die Berechnung der Springstundenzahl, benötigt. Alternativ wäre es möglich, die im Bewertungseditor vorgebbaren, aber in der momentanen Implementierung ignorierten Relationen zwischen zwei Zeitobjekten auszuwerten, was zwar eine größere Flexibilität, aber mit |Zeitstundenzahl| einzugebenden Relationen eine große Unübersichtlichkeit mit sich bringen würde. [CK95b] führen ein dem Zeiteditor äquivalentes Konstrukt in ihrer Eingabesprache auf, was daran liegt, daß sie ebenfalls Zeitstunden als Objekte betrachten.

8.2.4 Ansichten

Die völlige Abwendung von festliegenden Entitäten wie „Schülern“ und „Lehrern“ erfordert eine flexiblere Art der Darstellung der Datensätze. Hierzu wurde eine Tabelle mit einstellbaren Kategorien geschaffen, die es erlaubt, nahezu jede beliebige in „Timetabling“-Programmen übliche Ausgabeform zu wählen. Insbesondere sind - eine geeignete Wahl im Mengeneditor vorausgesetzt - auch die „Outline“ Darstellungen nach [Hil81] möglich. Die vollständige Übertragung dieses Schemas auf die Druckausgabe erlaubt zugleich, nun wieder im Schulkontext interpretiert, das automatische Erstellen von Klassen-, Raum-, Lehrer- und sämtlichen Übersichtsplänen.
Um eine erträgliche Geschwindigkeit des Bildaufbaus bei Verschiebungen zu gewährleisten, wird diese Tabelle stets vollständig im Speicher gehalten, wobei allerdings zu aufwendige Tabellen - der Benutzer kann de facto eine Tabelle mit |Anzahl der Objekte|^6 Feldern anfordern - nicht angefertigt werden. In die Tabelle werden diejenigen Felder der jeweiligen Gruppe bzw. Stunde bevorzugt eingetragen, die sich nicht aus dem Tabellenkontext ergeben. Die anderen, deren Namen zumeist ohnehin in den Tabellenüberschriften stehen, werden nur angegeben, sofern noch Platz zur Verfügung steht; diese werden bei Ausschneide-Operationen entfernt (s. dort). Die übernächste Darstellung zeigt einen Ausschnitt aus einer solchen Tabelle.



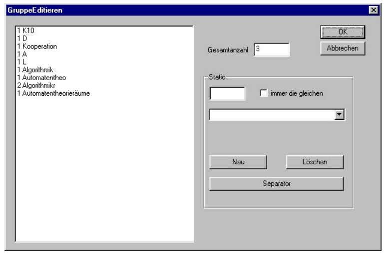


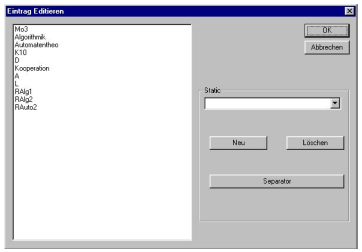
Die erste Abbildung zeigt den Dialog, der die Auswahl der Tabellenformen ermöglicht. Sowohl horizontal, als auch vertikal ist ein bis zu dreistufiger Hierarchieaufbau möglich. Im Feld Auswahl kann die Anzeige auf bestimmte Datensätze beschränkt werden, wobei die Auswahlbox angibt, daß die Teilmengen der angegebenen Menge nach und nach als Grundlage verwandt werden sollen. Bei der Bildschirmdarstellung kann mittels „+“ und „−“ zwischen verschiedenen Teilmengenkonfigurationen gewählt werden; sämtliche möglichen Kombinationen aus den drei „Auswahl“-Feldern werden durchlaufen. Beim Druck werden automatisch alle Konfigurationen ausgegeben, wobei zwischen zweien ein Seitenvorschub erfolgt und bei übergroßen Tabellen diese korrekt paginiert auf mehrere Seiten verteilt werden.

Es gibt zwei getrennte, farblich voneinander abgehobene Editoren für Vorgaben und Ergebnisse (vgl. Abschnitt 5.3.2). Beim Wechsel vom einen zum anderen wird die jeweils selektierte Position in der Tabelle beibehalten, so daß ein leichter Vergleich von Vorgaben und Zustand möglich ist. Um Tabellen dieser Größe verwalten zu können, mußte eine eigene Bildlauf-Funktion[2] entwickelt werden, da der von Windows für ein Fenster zur Verfügung gestellte Speicher nicht ausreichte. Diese eigene Funktion erlaubt es zudem vorzugeben, daß mit jedem Schritt um ein ganzes Feld weitergesprungen wird bzw. bei einem Klick auf die Bildlaufleiste um eine ganzzahlige Anzahl von Feldern. Die Implementierung erwies sich als recht aufwendig, da die Feldbegrenzungen von der jeweiligen Hierarchieebene abhängen. In Anhang A findet sich ein Beispiel für die Druckausgabe.

Selektionen werden durch Anklicken eines Tabellenfeldes vorgenommen, betreffen aber stets die gewählte Gruppe bzw. Stunde, so daß alle Repräsentationen dieser Gruppe (Stunde) in der Tabelle markiert werden. Klickt man unterhalb der Gruppen in ein Tabellenkästchen, so wird dieses ganz markiert. Dies dient z.B. der Vorbereitung einer Einfügeoperation. Beim Anzeigen von Stunden wird zudem auf auftretende Widersprüche hingewiesen. Kollidierende Stunden werden mit einem roten Rahmen versehen. Zudem wird auf Basis der den Stunden zugrundeliegenden Gruppen versucht, eine „Ressourcen-Optimierung“ (s. Abschnitt 8.4.3) durchzuführen. Stunden, bei denen dies nichts hilft, werden im Plan durchgestrichen. Bei nicht durchgestrichenen Mengen kann durch Selektion und anschließendes Aktivieren des Befehls „Zeitstunde Optimieren“ zu dem bestimmten Matching übergegangen werden. Des weiteren können Stunden an einer Position festgehalten werden, was es erlaubt, einen gegebenen Plan zu ergänzen bzw. Abwesenheitszeiten unter Umgehung des Bewertungssystems direkt einzugeben.
An dieser Stelle tritt ein semantisches Problem auf. Im Rahmen von „Zeitslot Optimieren“ kann jede Stunde mit anderen Ressourcen versehen werden, je nachdem wie es die zugehörige Gruppe vorgibt. Hierbei werden sämtliche Änderungen, die der Benutzer von Hand an dieser Gruppe vorgenommen hat, gelöscht. Dem Anwender steht es aber durchaus frei, die Stunde soweit zu verändern, daß sie zu einer anderen Gruppe zu gehören scheint, so daß die Gefahr besteht, daß die Optimierung der Zeitstunde Verwirrung stiftet, da scheinbar Stunden die Gruppen, auf denen sie basieren, wechseln. Es tritt also die Frage auf, welche Eigenschaften charakteristisch für die Gruppenzugehörigkeit einer Stunde sind.

8.2.5 Gruppen- und Stundeneditor





Die beiden dargestellten Dialoge ermöglichen das Editieren von Gruppen bzw. Stunden. Es besteht die Möglichkeit, diese nochmals in Untermengen aufzugliedern, da z.B. bei einer Differenzierungsstunde, die Mathematik und Schwimmen beinhaltet, die Bevorzugung von Schwimmhallen zum Schwimmen keine Auswirkungen auf die

[2] „Scrolling“

Raumwahl für die Mathematikstunden haben sollte. Diese Trennung wird zur Zeit noch nicht unterstützt; derselbe Effekt kann aber erzielt werden, indem man die „speziellen“ Räume unter einer Menge „Fachräume“ zusammenfaßt und generell die Verwendung von Fachräumen empfiehlt. Der Zwang, zum Schwimmen die entsprechende Sportstätte zu verwenden, kann dann über die Konstruktion des Mengensystems umgesetzt werden.

8.2.6 Löschen, Kopierfunktionen und Zurücknahmefunktion

Das Programm erlaubt es, Einträge von einer Position einer Tabelle in eine andere zu kopieren, wobei die jeweiligen Gruppen „kontextsensitiv“ angepaßt werden, indem die entsprechenden Mengen, die zum Einfügen der Gruppe an der Anfangsposition geführt haben, entfernt werden und die für das Erscheinen an der neuen Position erforderlichen Mengen hinzugefügt werden. Hierbei werden die Mengen durch etwaige gemeinsame Teilmengen ersetzt, so daß z.B. statt „Dienstag“ und „4. Stunde“ „Dienstag 4.Stunde“ eingesetzt wird. Die Bestimmung der minimalen Menge von Mengen, die zur Einfügung an dieser Stelle der Tabelle notwendig ist, entspricht dem Set-Cover Problem und ist daher $\mathcal{NP}$-vollständig. Auch bei kleinen Objektzahlen ließe sich der Aufwand hierfür an dieser Stelle nicht rechtfertigen, da noch nicht einmal gesichert ist, daß der Benutzer die Auswahl verstehen bzw. gutheißen würde. Daher wurde so vorgegangen, daß jede Menge der Reihe nach mit allen nachfolgenden Mengen verglichen wird; wird eine gemeinsame Teilmenge gefunden, so wird die erste durch diese ersetzt und die zweite gelöscht.

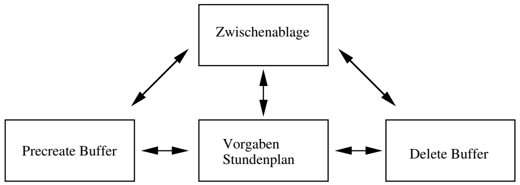
Bemerkung 8.1 *Dieses Verfahren führt zu einer Menge von Mengen, die unter obiger Operation nicht mehr weiter verkleinert werden kann.*

Angenommen, es gäbe am Ende noch zwei Mengen, die eine gemeinsame Teilmenge haben, dann betrachte den Zeitpunkt, zu dem die erste Menge mit den folgenden verglichen wurde. Die zweite Menge ist als gemeinsames Element im Schnitt einer oder mehrerer anderer Mengen entstanden. Da auf dieser Ebene nur Teilmengen betrachtet werden[3], muß das Element, das später die Vereinigung der beiden Mengen erlaubt also bereits Element jeder dieser Ausgangsmengen sein. Damit wären diese aber alle bereits zum betrachteten Zeitpunkt entfernt worden.
Der Benutzer sollte allerdings während Kopiervorgängen die Tabellenform nicht wechseln, da somit natürlich die hier eingeführte Logik durchbrochen würde. Es besteht die Möglichkeit, die erzeugten Stunden „von Hand“ nachzubessern.

Die Kopierfunktion kann durch Hinzunahme der „Zwischenablage“ in ihrem Funktionsumfang erweitert werden.
Ist die „Zwischenablage“ ausgeblendet, so wird beim Kopieren oder Ausschneiden der jeweils oberste Eintrag mit den neuen Daten überschrieben, so daß sich das Ganze wie eine herkömmliche Kopierfunktion verhält. Ist sie aber eingeblendet, so wird das Kopierte (Ausgeschnittene) der Zwischenablage hinzugefügt und kann vom Benutzer in beliebiger Reihenfolge entnommen werden,[4] so daß dem Benutzer eine nur durch den Speicherplatz begrenzte Zahl verschiedener „Zwischenspeicher“ zur Verfügung steht. Das Einfügen kann als das Einsetzen einer leeren Gruppe interpretiert werden.

[3] Und keine Mengen als Elemente anderer Mengen.
[4] Durch Klick mit der rechten Maustaste wird automatisch der oberste Ablageneintrag eingefügt.

Die Operationen Einfügen, Löschen, Ausschneiden, Kopieren und Einsetzen können bis zu 30 Schritte rückgängig gemacht und wiederhergestellt werden, wobei dies ein willkürlich gewählter Wert ist.[5] Hierbei können alle rückgängig gemachten Schritte so lange wiederhergestellt werden, bis der Benutzer eine neue Operation vornimmt. Dem Ganzen wurde das folgende System zugrundegelegt:



Sämtliche Aktionen werden als das Bewegen von Objekten entlang der dargestellten Pfeile interpretiert, wobei die Trennung zwischen `Precreate Buffer` und `Delete Buffer` nicht unbedingt notwendig ist, da die Objekte durch einen eindeutigen Bezeichner unterschieden werden können. „Ausschneiden“ und ähnliche zerlegbare Operationen werden z.B. als „Kopieren“ und „Löschen“ interpretiert, so daß der Benutzer ggf. einmal häufiger die „Wiederherstellen“-Funktion aufrufen muß, um das gewünschte Ergebnis zu erzielen. Liegt die letzte Operation, bei der eine Stunde oder Gruppe benötigt wurde, mehr als 30 Schritte zurück, wird das entsprechende Objekt gelöscht.[6] Wird eine Stunde aus den Voraussetzungen gelöscht, so werden automatisch alle Stunden im Stundenplan gelöscht, die mit dieser im Zusammenhang stehen. Ein Rückgängigmachen dieser Auswirkung ist z.Zt. nicht möglich, da dies für Stunden und Gruppen getrennt ginge und sich die entsprechenden Stunden somit auf nichtexistierende Anforderungen beziehen würden.

8.3 Die interaktive Unterstützung

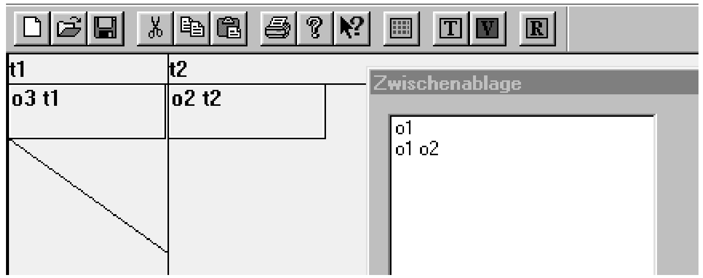
Die Unterstützung beruht auf tiefenbeschränktem Backtracking. Es wird jeweils versucht, die nächste Stunde aus dem Zwischenspeicher in eines der aktuellen Tabellenfelder einzufügen und dann eine bestimmte Anzahl weiterer Stunden aus dem Zwischenspeicher zu verteilen. Hierbei werden Gruppen, die bereits Zeitstunden enthalten diesen direkt zugeteilt.[7] Für die übrigen werden alle möglichen Zeitstunden durchprobiert. Wurde dies für alle Felder der momentanen Tabelle versucht, so wird das Level um 1 erhöht und der Vorgang erneut begonnen, wobei nun aber nur Felder betrachtet werden, die bei den niedrigeren Ebenen nicht aussortiert wurden.
Da sich die Qualität der Ratschläge mit der Dauer der Wartezeit verbessern soll, wird die interaktive Unterstützung immer dann aufgerufen, wenn der Benutzer gerade nichts tut. Das Visual C++-Framework bietet hierzu eine spezielle Funktion, die immer dann aufgerufen wird, wenn die Ereignisliste des Frameworks leer ist. Bei jedem Aufruf der Funktion wird ein weiteres Blatt des aktuellen Suchbaums überprüft, was mit sich bringt, daß dessen Position bis zum nächsten Aufruf gespeichert bleiben muß. Eine

[5] MIMOSA [Mim98] bietet 1000 Schritte an, aber sowohl dies als auch eine dynamische Implementierung, die beliebig viele Schritte unterstützen könnte, erschienen übertrieben.

[6] Das Funktionieren dieses Features konnte nur im Debugger beobachtet werden. Es gibt keinen für den Benutzer transparenten Test, der dies später verifizieren könnte.

[7] Es wird kein Versuch unternommen, etwaige fehlende Objekte zu ergänzen, da durch deren Hinzufügen zumeist der Bezug zu einer Voraussetzungs-Gruppe verloren würde.

derartig feine Unterbrechung ist hier aber unumgänglich, damit die Reaktionszeit auf parallel ablaufende Benutzeraktivitäten nicht zu groß wird. Dies machte es ebenfalls notwendig, die Initialisierungsphase des Hilfssystems in zwei Schritte zu zerlegen.
Da die Zahl der möglichen Verteilungen weniger Schulstunden meist sehr groß ist, mußte darauf verzichtet werden, die gültigen Lösungen des jeweils vorangegangenen Levels zwischenzuspeichern, so daß die Suche jedesmal von neuem beginnen muß. Genauer gesagt bedeutet dies, daß beim Untersuchen eines Blattes der Tiefe k zwar bekannt ist, daß der zugehörige Knoten auf Ebene $k-1$ erreicht werden konnte, aber nicht auf welche Weise. Das folgende (kleine) Beispiel zeigt, wie die Funktion vor dem Einfügen der nächsten Gruppe in die „linke“ Zeitstunde warnt, wobei dieses zwar keinen unmittelbaren Widerspruch auslöst, aber das Einfügen der folgenden Stunde unmöglich macht.



8.4 Besonderheiten

8.4.1 Integrität und kaskadierende Löschvorgänge

Die starke Abhängigkeit der Datenstrukturen voneinander und das Bestreben, daß durch das Löschen eines Objektes nicht die übrigen in Mitleidenschaft gezogen werden, führt zu dem Auftreten kaskadierender Löschvorgänge, wie sie aus dem Bereich relationaler Datenbanken bekannt sind. Zwei Mechanismen wurden vorgesehen, um dieses Problem zu behandeln:

- Eindeutige Schlüssel für alle auftretenden Datenobjekte (also auch Mengen und Gruppen): Dies ermöglicht es, einzelne Objekte zu löschen, ohne daß durch Indexverschiebungen Relationen verloren gehen. Eine „normale“ Implementierung mit Zeigern ist hier ohnehin nur schwer möglich, da das „Serialisierungskonzept“ von Visual C++ verlangt hätte, daß diese nach Speichern und Neuladen noch gültig sein müßten. Über die verwandten „Hashs“ (vgl. Abschnitt 8.1) wird es zudem möglich, die Tatsache, daß ein Objekt gelöscht wurde, nachträglich festzustellen.
- Zeitmarken: Da das verwendete Visual C++ Framework aus diversen Aufrufen heraus überladene „Update“-Funktionen aufruft, war durch eine herkömmliche Kette von Funktionsaufrufen das Problem, alle Datensätze konsistent zu halten, nicht zu lösen, da entweder auf bereits gelöschte Objekte zurückgegriffen werden sollte oder aber unendliche Rekursionen auftraten. Daher beinhaltet jedes größere Konstrukt (d.h. das Mengensystem, Gruppensystem usw.) jeweils eine Zeitmarke, die die letzte Veränderung bezeichnet, und eine Marke, die angibt,

wann zum letzten Mal ein Abgleich mit dem zugrundeliegenden System stattgefunden hat. Lediglich bei den Puffern kann eine „Dirty“-Marke verwendet werden. Vor jeder Operation kann so überprüft werden, ob sich an der eigenen Basis etwas geändert hat. „Zeitmarke“ beschreibt hier nur einen Zähler, da die „Zeit“ mit jeder Veränderung und nicht mit jeder Sekunde o.ä. inkrementiert wird. Die Verwendung entspricht aber der von Zeitmarken.

8.4.2 Objektbeziehungen

Wie oben bereits ausgeführt, umfassen die Objektbeziehungen auch die Beziehungen von Objekten zu Zeitstunden. Beziehungen zwischen Zeitstunden könnten auch eingegeben werden und die Angabe des Zeitablaufs ersetzen; sie werden aber nicht berücksichtigt, da dies den Benutzer wohl zu sehr verwirren würde.

Da nicht für jedes Objektpaar vom Benutzer ein eigener Wert angegeben werden kann, wurde ein System vorgesehen, nach dem die Bewertungen von einer Menge auf ihre jeweiligen Untermengen übertragen wird. Dies ermöglicht z.B. Vorgaben wie die, daß Rollstuhlfahrer am besten in Räumen mit Rampen unterrichtet werden (vgl. [BJKW96]). Als Werte wurden $\{-3, \ldots, 0, \ldots, +3\}$ vorgesehen, wobei 0 andeuten soll, daß keine konkreten Vorlieben bestehen, sondern der von den Obermengen her berechnete Wert eingesetzt werden soll. Wesentliche Anforderungen an die so zu bestimmende Relation war, daß sie

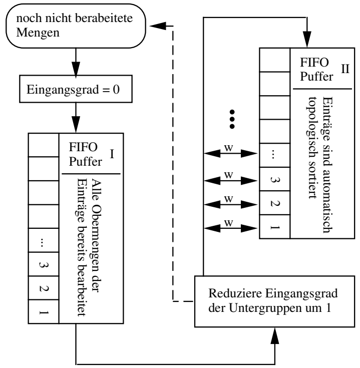
- symmetrisch und
- von der Reihenfolge der Berechnung unabhängig ist.

Daher wurde die folgende Formel zugrundegelegt:

$$w(x,y) := \frac{\left(\sum_{z \supset y} w(z,x)\right) + \left(\sum_{z \supset x} w(z,y)\right)}{\left(\sum_{z \supset y \wedge w(z,x) \neq 0} 1\right) + \left(\sum_{z \supset x \wedge w(z,y) \neq 0} 1\right)} \tag{8.1}$$

Hierbei bezeichnet $\supset$ direkte Übermengen ohne Berücksichtigung der Mengen, die diese wiederum umfassen. Damit ist es möglich, Vorgaben dieser indirekten Obermengen vollständig zu überstimmen. Bei der Bewertung bleiben Obermengenbeziehungen mit dem Wert „0“, d.h. „unbestimmt“ unberücksichtigt, da die große Zahl der Beziehungen dieses Typs ansonsten über Rundungsprozesse die vom Benutzer gewählten Eigenschaften völlig nivellieren würde. Bei der Berechnung der einzelnen Werte wurde vorausgesetzt, daß der Graph der Obermengen- bzw. Untermengenrelation azyklisch ist, was wie oben beschrieben sichergestellt wird.

Zur Berechnung werden die einzelnen Mengen dem folgenden System unterworfen:

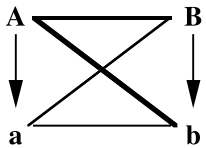


Die oben geforderten Eigenschaften ergeben sich unmittelbar aus der Formel. Diese setzt aber voraus, daß die entsprechenden Werte für die Übermengen bekannt sind: Es sei also angenommen, daß x neu in `FIFO II` aufgenommen werdon und seine Beziehung w zu einer Menge y aus `FIFO II` zu bestimmen ist.

- Die Werte $w(y,z)$ mit $z \supset x$ sind bekannt, da `FIFO I` topologisch sortiert ist und somit die Mengen z bereits bearbeitet worden sein müssen, bevor x betrachtet wird.
- Die Werte $w(z,x)$ mit $z \supset y$ sind bekannt, da `FIFO II` die topologische Sortierung von `FIFO I` übernommen hat und somit die Beziehung der Mengen z zu x geprüft wurde, bevor y betrachtet wird.

Da die Zahl der Mengen im voraus bekannt ist und die beiden Pufferinhalte stets disjunkt sind, konnte das Ganze in einer Schlange (als Array implementiert) mit zwei Pointern zu den jeweiligen „Schlangenenden“ realisiert werden.

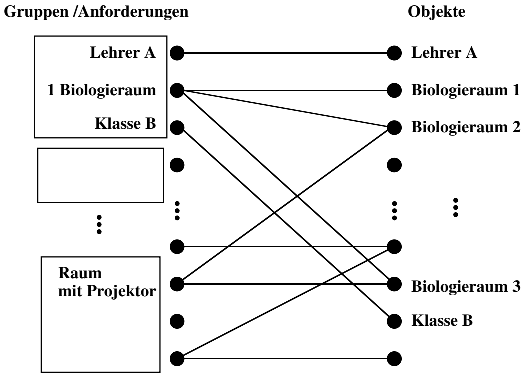
In einigen Versuchen zeigte sich, daß das Verhalten des Systems so am ehesten dem vom Benutzer erwarteten entspricht; dennoch verhält es sich in wenigen Fällen noch nicht der Intuition entsprechend, wie das folgende Beispiel illustriert: Seien a, b Teilmengen von A bzw. B. Die durch die dickeren Linien gekennzeichneten Relationen seien vom Benutzer vorgegeben.



Zunächst wird die gegebene Relation (A, B) auf (a, B) übertragen. An der Bestimmung des Wertes von (a, b) sind nun (A, b) und (a, B) gleichberechtigt beteiligt, was also - vorausgesetzt alle Werte sind ungleich Null - dazu führt, daß diese Relation als Wert den Mittelwert der beiden anderen erhält. Intuitiv würde man aber annehmen, sie müßte (A, b) erhalten, da dies (A, B), von dem (a, B) ja abstammt, überstimmt hat. Ein einfaches Bevorzugen der Benutzervorgaben an dieser Stelle erscheint aber ebenfalls nicht sinnvoll, da eine etwaige Beziehung (a, B') nicht neutralisiert werden dürfte.

8.4.3 Prüfung der Ressourcenzuteilung

Sind die Gruppen einmal Zeitpunkten zugewiesen, so kann mittels eines Matchings eine optimale Ressourcenzuteilung vorgenommen werden.[8] Hierzu wird ein bipartiter Graph gebildet, dessen eine „Seite“ aus sämtlichen Objekten und dessen andere Seite aus allen von den Gruppen gestellten Anforderungen besteht, wobei für mehrfache Anforderungen entsprechend mehrere Knoten eingefügt werden:



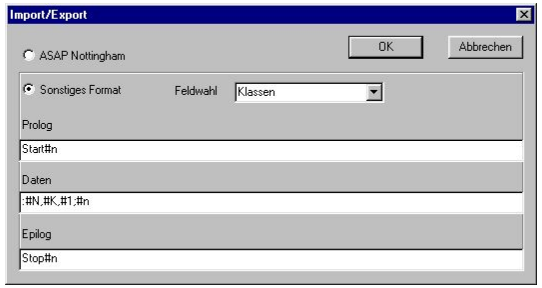
Der hierzu verwandte „Ungarische“ Algorithmus ist im allgemeinen mit einer Komplexität von $\mathcal{O}(n^3)$ zu langsam, um interaktiv eingesetzt zu werden. Die Möglichkeit einer solchen Zuordnung hingegen kann durch ungewichtetes Matching überprüft werden. Dies hat bereits eine prinzipiell niedrigere Komplexität ($\mathcal{O}(n^2)$), man kann den Vorgang aber noch weiter beschleunigen, da der Algorithmus darauf beruht, bereits bestehende Matchings zu vergrößern. Die Tatsache, daß die Mengen im Schulbereich mit wenigen Ausnahmen disjunkt sind, führt hier dazu, daß man mit einfachen Greedy-Verfahren Matchings erhalten kann, die nur wenige Kanten weniger haben als ein maximales Matching. Bei im Test verwendeten ca. 40 Timeslots, also ebensovielen zu berechnenden Matchings, fällt der Aufwand zu deren Berechnung gegenüber dem zur Erstellung der Ansicht nicht merklich ins Gewicht. Wurden die entsprechenden Stunden automatisch erstellt, so kann ihr Bezug zu einer Stunde in den Voraussetzungen leicht festgestellt werden. In diesem Fall wird die Voraussetzung und nicht die konkrete Menge für das Ressourcen-Matching verwandt, so daß Vorschläge für die Verbesserung gemacht werden können. In der Implementierung sieht dies so aus, daß Kollisionen, die auf diesem Wege aufgelöst werden können, anders angezeigt werden als solche, bei denen dies nicht der Fall ist.

8.4.4 Import und Export

Da zahlreiche Schulen ihre Stundenplanvorgaben in der EDV erfaßt haben, schien die Implementierung einer flexiblen Import/Export - Funktion sinnvoll. Hierzu mußte dem Benutzer die Möglichkeit gegeben werden, ein eigenes Eingabeformat zu definieren. Da nicht anzunehmen ist, daß die auszutauschenden Datensätze eine allzu komplizierte Struktur aufweisen, wurde auf die Integration eines Parsergenerators verzichtet, und

[8] Siehe auch [CK95b].

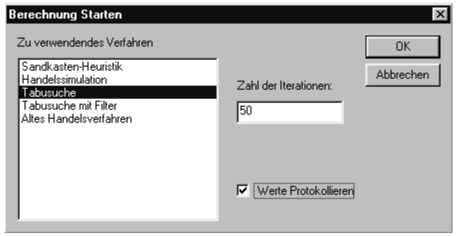
stattdessen eine einfache „Wildcard“ - Substitution implementiert, die lediglich Ausdrücke der Form $ab*c$ für Datensätze und $a(bc*d)*e$ für Ressourcen-Gruppen zuläßt, was sogar eine Teilmenge der regulären Sprachen bleibt. Die Buchstaben stehen hier für Zeichenketten bestehend aus Feldbezeichnern (z.B. `#N` für „Name“ etc.), einigen Makros (Dateiname, Datum, etc.) und beliebigen Zeichen. Gemäß dieser Schemata wird dann die Eingabedatei zerlegt bzw. die Ausgabe generiert. Eine Importmöglichkeit nach den Vorschlägen von Burke [BKP97] wurde vorgesehen, aber nicht implementiert.



8.4.5 Das Einbinden von Lösungsverfahren

Um ein neues Lösungsverfahren einzubinden, muß dies lediglich als eine neue C++-Klasse erzeugt werden, die von „SolutionMethod“ abgeleitet ist. Im Konstruktor muß einmal die Funktion „RegisterSolution“ mit dem im Auswahlmenü anzuzeigenden Text als Argument aufgerufen werden. Und es muß mindestens ein Objekt dieser Klasse erzeugt werden. Letzteres hat den Nachteil, daß jedes Lösungsverfahren den vom ihm benötigten Speicher erst bei Aufruf und nicht schon im Konstruktor allokieren darf. Dies war aber dem expliziten Einbinden von Zeigern in die Klasse „SolutionMethod“ vorzuziehen. Die Erzeugung dieses Objektes muß in einer bestimmten Datei geschehen, da die Lösungsverfahren in einer statischen Liste gespeichert werden, die von Visual C++ nicht zeitgleich mit ihrer Klasse, sondern erst bei ihrer Definition initialisiert wird. Die einzige Möglichkeit in dieser Situation sicherzustellen, daß dies vor der Initialisierung des ersten Verfahrens geschieht, ist die, diese Schritte in einer Quellcodedatei zusammenzufassen.

Ein neues Lösungsverfahren sollte jeweils vier Funktionen überladen: `Initialize`, `Step`, `ReadoutSolution` und `MakeClean`, wobei die Namen selbsterklärend sind.



Die Klasse „SolutionMethod“ stellt zusätzlich eine Funktion zur Verfügung, die es erlaubt, beliebige Ausdrücke in eine vorher vom Benutzer angegebene Datei zu schreiben. Dies, sowie die Möglichkeit zwischen den Verfahren zu wählen, dienen der Unterstützung einer einfacheren Evaluation verschiedener Verfahren.

8.5 Sonstiges

8.5.1 Darstellung und Symbolik

Das direkte „Aussehen“ der Oberfläche wird von den von Visual C++ angebotenen Dialogelementen geprägt. An den wenigen Stellen, an denen eigene Darstellungen notwendig waren, wurden diese zumeist auf farbig unterlegte Buchstaben begrenzt. Die Möglichkeit, die Form der Einzeleinträge der Tabelle vorzugeben, wurde vorgesehen. Diese - im Rahmen einer akademischen Arbeit kaum zu umgehende - Schlichtheit stellte sich im Nachhinein im Vergleich mit der Vielfarbigkeit und der Symbolmenge von anderen Programmen zur automatischen Stundenplanerstellung[9] aber als „Beschränkung zum Besseren“ heraus.

8.5.2 Codeoptimierung

Der vorliegende Programmcode kann an mehreren Stellen beschleunigt werden; insbesondere die Tatsache, daß die meisten Objekte nur an wenigen Gruppen beteiligt sind, kann durch eine umfassendere Verwendung von Listen anstelle von Matrizen noch wesentlich stärker ausgenutzt werden. Zu diesem Zweck könnte ebenfalls sinnvoll sein, die Trennung zwischen Gruppen und Mengen aufzugeben, was beim ersten Entwurf als für den Anwender zu verwirrend eingeschätzt wurde.
Da die Entscheidung, das Mengensystem azyklisch zu gestalten, erst relativ spät getroffen wurde, verwenden einige Routinen noch Algorithmen, die auch für den nicht azyklischen Fall geeignet wären. So verwendet die Bestimmung der Übermengenrelation CORMAN et al. [CLR96] folgend noch den Algorithmus von FLOYD, während für die azyklische Version deutlich schnellere Verfahren existieren.

8.5.3 Ergonomie

Da die Schulen die Daten in Papierform übergaben, und die Möglichkeit einer Erfassung z.B. per Scanner wegen der wechselnden Formate nicht sinnvoll schien, entstand die Notwendigkeit, diese vollständig manuell zu erfassen; hierbei wurden mehrere Abfolgen des ersten Programmentwurfs als unpraktisch empfunden, so daß nachträglich zahlreiche - auch zeitaufwendige - Verbesserungen eingebracht wurden, die sich aber durch die resultierende höhere Eingabegeschwindigkeit amortisiert haben dürften. So wurde z.B. der Wechsel der Eingabefelder im Objekteditor nicht mehr dem Framework überlassen, sondern direkt gesteuert, so daß eine ähnliche Geschwindigkeit erreicht werden konnte wie bei einer rein zeilengesteuerten Eingabe, da es - eine genaue Kenntnis der Daten vorausgesetzt - wesentlich schneller möglich ist, ein Kürzel zu tippen, als die entsprechende Menge aus einer ungeordneten Liste mit mehreren hundert Elementen herauszusuchen. In diesem Sinne wäre es des weiteren vorstellbar, Gruppen- und Stundenansichten mit effizienten Möglichkeiten zur Mehrfachselektion auszustatten.

[9] z.B. Mimosa [Mim98]

8.6 Qualitätsprüfung und -management

8.6.1 DIN ISO 9001-3, DIN ISO/IEC 12119

Die DIN ISO 9001-3 beschreibt generelle Vorgehensweisen zur Qualitätssicherung bei Produktionprozessen; eine Anwendung auf die Softwareerstellung findet sich in SCHMAUCH [Sch94]. Es zeigt sich, daß die vorgeschlagenen Verfahren überwiegend für die Abstimmung größerer Teams gedacht sind und der wesentliche Schwerpunkt auf der Erstellung vorgeschriebener Dokumentation beruht. Im Rahmen dieser Diplomarbeit war eine Kooperation nicht gegeben und die Dokumentation erfolgte zeitgleich mit der Erstellung des Programms in der vorliegenden Arbeit. Daher beschränkte sich die Anwendung der entsprechenden Norm auf die Erstellung einer „Qualitätspolitik“ und dem Festlegen einiger Verfahrensweisen („guter Vorsätze“), die im wesentlichen eingehalten werden konnten.
Die DIN ISO/IEC 12119 beschreibt die Verfahren, die zur Produktprüfung verwandt werden sollen. Auch hier liegt wiederum der Schwerpunkt auf festgelegten Formularen (s. [LS95]), die eine Klassifikation der verschiedenen Teile und die jeweilige Prüfungstiefe festlegen. Da in diesem Projekt nur ein einziger Entwickler - der Diplomand - und zwei Tester - die betreuenden Assistenten - gegeben waren, reduzierte sich die Dokumentation hier auf ein Notizbuch mit „Testideen“ für den „White-Box“ Test - entsprechend dem analogen Formular aus [LS95] - und elektronischer Post für die „Black-Box“ Tests. Eine weitere Formalisierung erschien nicht angemessen.

8.6.2 Sperrvariablen

Im Programm kommt es an einigen Stellen vor, daß Variablen einer Klasse von verschiedenen Routinen verwandt werden und eine Implementierung mit mehreren Variablen zu speicheraufwendig wäre. Hier wurde in der Test-Version[10] ein umfangreiches Schema von Sperr-Methoden eingerichtet, das eine Fehlermeldung auslöst, sobald eine Routine auf eine Variable zugreifen will, die gerade von einer anderen benutzt wird. Da hier keine echte Nebenläufigkeit auftritt, konnte dies jeweils mit einer einzigen Boolschen Variablen realisiert werden. In der Endversion des Programms werden diese Konstrukte vom Preprozessor entfernt. Als Nebeneffekt ermöglichte dieses Vorgehen eine schnelle Aufspürung und Diagnostizierung ungewollter Rekursionen, die entstanden, da die Aufrufketten innerhalb des Frameworks nicht bekannt sind.

8.6.3 Datumsabhängigkeiten

Das Programm verwendet das aktuelle Datum als Zusatzinformation beim Ausdruck und bei der Erzeugung eindeutiger Schlüssel für Objekte. Beim Druck kann es nicht zu Fehlern kommen, ein Schlüssel hingegen kann sich nach ca. 40 Jahren wiederholen. Es genügt in diesem Fall, das doppelte Objekt zu entfernen. Datensätze die mit verschiedenen Kopien des Programms zur gleichen Zeit - oder bei verstelltem Zeitgeber - erstellt wurden, dürfen nur mit der Import-Funktion, aber keinesfalls von Hand gemischt werden, da es ansonsten ebenfalls zu doppelt auftretenden „eindeutigen“ Schlüsseln kommen kann. Momentan könnte dieses Problem umgangen werden, indem

[10] Visual C++ bietet hier eine Trennung an.

jedes Dokument mit einem eigenen Zähler versehen wird, aber eine Möglichkeit, Datensätze zwischen Dokumenten zu kopieren, würde hier neue Probleme schaffen. Der Austausch zwischen verschiedenen Installationen sollte grundsätzlich nur über die Im- und Exportfunktionen vorgenommen werden.

8.6.4 Unterstützung durch die Entwicklungsumgebung

Als Entwicklungsumgebung wurde „Visual C++ 5.0“[11] verwandt. Die Fehlermeldungen der lexikalischen Analyse bewegen sich im Rahmen des Üblichen; der Parser ist relativ anfällig für ein fehlendes Semikolon am Ende einer Klassendefinition. Einige erwartete (Parser-) Warnungen werden nicht oder nur sporadisch angezeigt. Insbesondere fehlt eine Warnung bei unerreichbarem Code.

Die Sprache selbst bietet mit der Möglichkeit, eigene Klassen von „CObject“ abzuleiten, Features zur genauen Überprüfung der Speicherallokation und der Integrität von Objekten, die bei der Fehlersuche ebenfalls sehr hilfreich sind.
Als zusätzliche Hürde machte sich die Verwendung des Multi-Document-Environment bemerkbar, da es kaum mehr möglich war, globale Variablen zu benutzen, weil mehrere Dokumente konkurrierend darauf zugreifen könnten. Dies führte zu einem erhöhten Dereferenzierungsaufwand und war auch während der gesamten Programmentwicklung eher störend. Da keine direkte Möglichkeit der Datenübertragung zwischen verschiedenen Dokumenten vorgesehen wurde, bringt es dem Benutzer zudem kaum Vorteile.

8.7 Zusammenfassung

Das Programm wurde in Visual C++ implementiert und umfaßt ca. 8500 Zeilen Programmcode, wobei die von Visual C++ automatisch erzeugten nicht berücksichtigt wurden. Es bietet zahlreiche Funktionen an, von denen einige in der automatischen Stundenplanerstellung in dieser Form bisher nicht auftreten. Insbesondere die hohe Flexibilität gegenüber verschiedenen Einsatzsituationen und die davon unabhängige Funktionsfähigkeit der Verfahren zeichne diesen Teil der Arbeit gegenüber „herkömmlichen“ Programmen zur Verwaltung von Stundenplänen aus. Es wurde ein hoher Aufwand betrieben, um das Programm trotz vieler untereinander abhängiger Einstellungsmöglichkeiten für den Benutzer robust gegenüber dem Löschen von Objekten zu gestalten.

[11] ©Microsoft

Kapitel 9

Evaluation

9.1 Testdaten

Zur Evaluation wurden ein selbst erstellter Datensatz (3), sowie zwei reelle Stundenplanprobleme (1),(2) verwandt. (1) entspricht einem aktuellen Stundenplanproblem eines Gymnasiums. Es treten 230 verschiedene Gruppen auf, was zu insgesamt 590 zu setzenden Stunden führt. Die Blockung der Oberstufe wurde vollkommen übernommen. Da die Möglichkeit einzelne Stunden parallel zu legen, aufgrund der fehlenden Information über die Verteilung der Schüler nicht zur Verfügung stand, konnte nur so gesichert werden, daß der spätere Plan auch gültig ist. Da diese beiden Teile des gymnasialen Stundenplans häufig getrennt erstellt werden, entspricht dies sogar eher den in der Praxis auftretenen Fällen. (2) Stammt von einer Realschule. Es treten 202 Gruppen und 491 Stunden auf, die auf 30 Zeitstunden verteilt werden müssen. Es handelt sich hierbei aber nur um den STP-Anteil dieses Problems. Die Differenzierungsstunden könnten aus den gegebenen Daten nur mit hohem Aufwand zurückgewonnen werden, da das an dieser Schule verwandte Programm nur das STP unterstützt, und daher mehrere virtuelle Klassen eingerichtet wurden, um dies noch darstellen zu können. Es läßt sich aber kaum rekonstruieren, welche Schüler an diesen Kursen teilnehmen. (3) Besteht aus 62 Gruppen und 190 zu setzenden Stunden. Es stehen 31 Zeitstunden zur Verfügung.

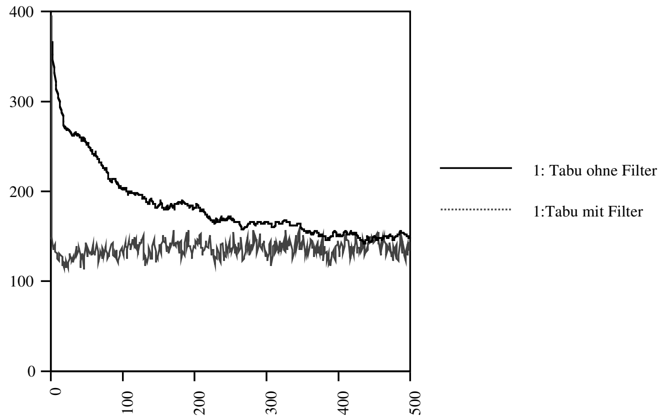
9.2 Ergebnisse

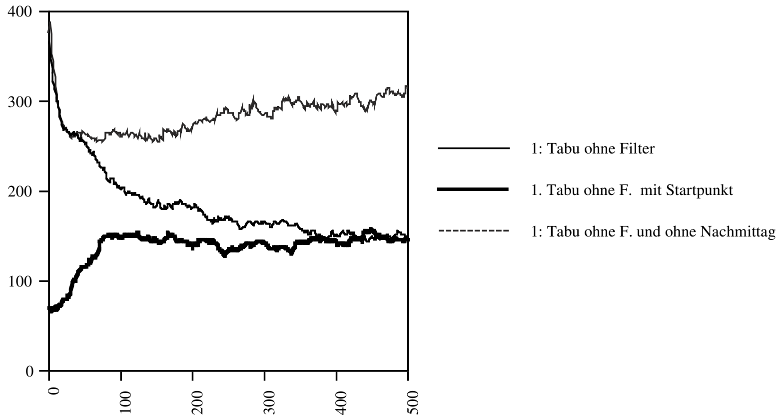
9.2.1 Parameterwahl

Bei allen Verfahren wurde zunächst mit derselben Zielfunktion gearbeitet. Im Laufe der Zeit stellte sich aber heraus, daß die fertigen Pläne eine zu geringe Kompaktheit aufwiesen, da mit der Kollisionsvermeidung als oberstem Ziel stets auch die Nachmittagsstunden voll ausgenutzt wurden. Daher wurde der Parameter, der über die Bewertung „niemals“ in die Zielfunktion einfließt extrem erhöht. Die Ergebnisse fielen bezüglich der Kollisionszahl entsprechend schlechter aus.

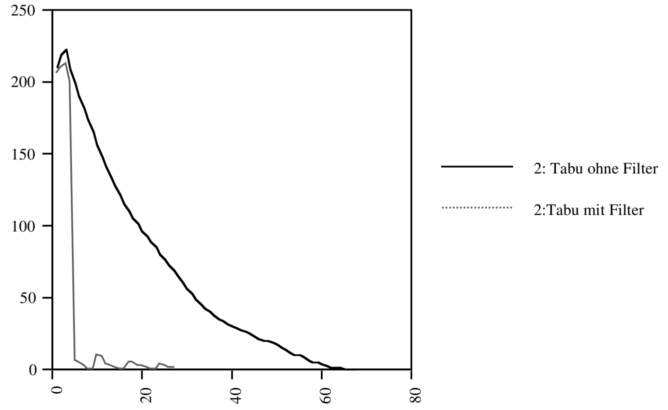
9.2.2 Diagramme

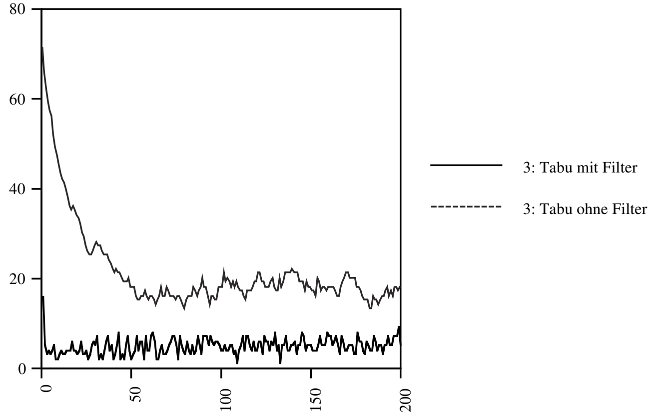
Zunächst wurde Fall 1 ohne Einschränkung der Nachmittagsstunden behandelt. Die Abbildung zeigt das Verhalten der Tabusuche mit und ohne zusätzlichem Filter. Das Filterverfahren erreicht natürlich wesentlich schneller gute Kollisionszahlen, kann dann aber keinen Vorteil mehr erzielen, wobei es aber stets besser ist als das Verfahren ohne Filter.



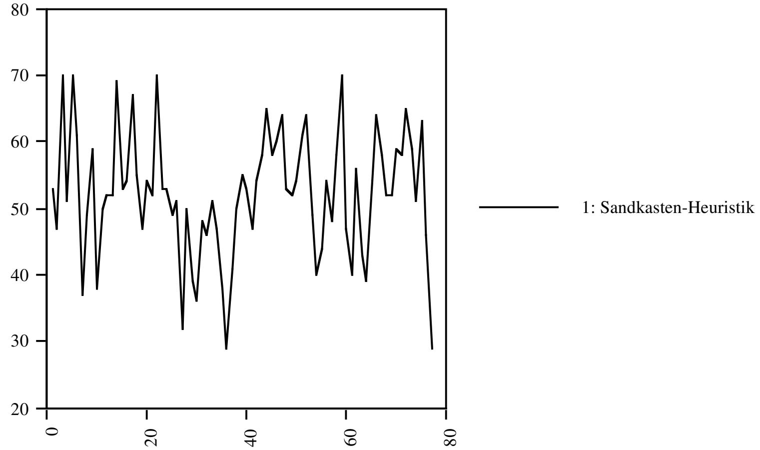


Dieses Diagramm zeigt das gleiche Problem, wobei eine Kurve aus der oberen Darstellung übernommen wurde. Die erste zusätzliche Kurve zeigt das Verhalten, wenn Mittel- und Unterstufe nachmittags keinen Unterricht haben dürfen. Hier konnte die Zahl der Kollisionen entsprechend weniger gesenkt werden und ca. 40% der Stunden hätten von Hand neu verteilt werden müssen. Die andere Kurve zeigt das Verhalten der Tabusuche, wenn sie mit einem Startpunkt, der mit der „Sandkasten-Heuristik“ erzeugt wurde, gestartet wird. Diese Heuristik nimmt keinerlei Rücksicht auf Springstunden, so daß die Tabusuche hier zunächst Kollisionen hinzufügt, um deren Zahl zu reduzieren. Die beiden folgenden Diagramme zeigen das Verhalten für die Fälle 2 und 3. Insbesondere, da bei der Realschule die Differenzierungsstunden nicht erfaßt wurden, ist in beiden Fällen die Zahl der Stunden relativ zu Objekten und Zeitstunden gering. Mit der „Sandkasten-Heuristik“ gelang in beiden Fällen in einem Schritt ein kollisionsfreier Plan. Wie das obige Beispiel zeigt, sind diese mit den realen Problemstellungen nicht vergleichbar.





In Bezug auf die reine Kollisionsvermeidung erzielte die „Sandkasten-Heuristik“ die besten Ergebnisse. 30 Kollisionen entsprechen maximal 6% neu zu setzenden Stunden. Dies zeigt, daß das Problem prinzipiell lösbar ist, wobei hier allerdings wiederum der Nachmittag für alle Klassen zugelassen wurde.



9.3 Weitere Anmerkungen

- Im Rahmen dieser Diplomarbeit wurde eine Befragung für die Stundenplanerstellung verantwortlicher Lehrer durchgeführt und ausgewertet. Hierzu wurde ein zweiseitiger Fragebogen an ca 30 Lehrer verteilt, die an ihren jeweiligen Schulen mit der Erstellung der Stundenpläne betraut sind. Zwei der Fragen beschäftigten sich mit der Leistungsfähigkeit der im Einsatz befindlichen Programme: Zum einen, andererseits welcher Anteil der Stunden automatisch gesetzt wird und welcher Anteil davon nachträglich korrigiert werden mußte. Läßt man die Antworten mit 0% weg, da hier vermutlich reine Satzprogramme verwendet wurden, so erhält man einen durchschnittlichen Anteil von 50% von denen im Schnitt 30% nachträglich korrigiert werden mußten. Dies entspricht in etwa den Ergebnissen, die mittels Tabusuche erzielt werden konnten, wobei allerdings anzumerken ist, daß hier die Zahl der Fach-Klassen-Springstunden noch nicht berücksichtigt wurde.

- Durch das Hinzufügen der Nachbearbeitung mit alternierenden Kreisen konnte die Zahl der Kollisionen, die von der hier verwendeten Variante des Algorithmus nach König verursacht wurden von 76 auf 62 gesenkt werden.

Kapitel 10

Ausblick

Mit der hier vorgelegten Arbeit wurden zwei Ziele verfolgt:

Eine Aufgabe der Arbeit war es, einen umfassenden und bewertenden Überblick über das Gebiet der automatischen Stundenplanerstellung zu geben.
Zweitens sollten im Rahmen der Erstellung eines Programms Erfahrungen sowohl mit neuen Lösungsverfahren, als auch mit den Anforderungen einer möglichen praktischen Einsetzbarkeit gewonnen werden.

Bereits bei den ersten Nachforschungen bezüglich realistischer Problemstellungen zeigte sich, daß mit einer einfachen Erweiterbarkeit der im Praktikum[1] entwickelten Verfahren nicht zu rechnen war, da selbst bei Realschulen inzwischen eine große Zahl von Differenzierungsstunden auftritt. Deren Auftreten verschiebt die Problemstellungen nicht nur aus $\mathcal{P}$ heraus, sondern bis hinein in den Bereich der Nichtapproximierbarkeit[2].
Um in diesem Bereich gute Ergebnisse zu erzielen, wäre entweder ein nicht zu erwartender theoretischer Durchbruch oder aber jahrelange Erfahrung im Umgang mit derartigen Problemstellungen und den zugehörigen Parameterwahlen notwendig. Aus diesen Gründen wurde daher der Schwerpunkt des Vorgehens zugunsten interaktiver Möglichkeiten verschoben, da sich die Probleme in diesem Bereich zwar auch als komplex, zumindest aber als handhabbar erwiesen. Dennoch wurde - unter einem zunächst eher experimentellen Aspekt - die Untersuchung möglicher Heuristiken parallel weitergeführt.

Diese Verschiebung kam der Erstellung des Satzprogrammes zugute, die sich ohnehin als wesentlich aufwendiger erwies, als dies ursprünglich erwartet worden war. Zeigte der erste Prototyp noch das Verhalten, daß das Löschen eines einzelnen Datensatzes die Neueingabe sämtlicher Parameter verlangte, so zog die Verhinderung dieses Problems eine große Zahl von Konsistenzproblemen nach sich, deren Behebung wiederum sehr aufwendig war, auch da sich das verwendete „Framework“ hier nicht sonderlich „kooperativ“ zeigte. Zudem führte die Allgemeinheit, in der das Problem angegangen wurde, dazu, daß viele Operationen jeweils auf Mengen von Objekten und Relationen zwischen diesen ausgeführt wurden, was den Aufwand hier noch weiter steigerte. Bei anderen Implementierungen, die ein ähnlich komplexes System anbieten [CK95b], war noch auf eine Kodierung über eine Eingabesprache und „Batch-Läufe“ zurückgegriffen

[1] Cremers,Karpinski: Effiziente Algorithmen für kombinatorische Optimierungsprobleme; SS1997
[2] Vorausgesetzt $\mathcal{P} \neq \mathcal{NP}$.

worden.

Dennoch konnte dieses Programm mit einigen Funktionen versehen werden, die bisher im Bereich der automatischen Stundenplanerstellung noch nicht angeboten wurden, insbesondere, was die schnellere Vorgabe komplexer Parametermengen und die Unterstützung während des Editierens angeht. Auch die Möglichkeit, beliebige Tabellenformen verwenden und ausdruckenzu können, wird von anderen Produkten nicht so umfassend geboten. Das Programm hat somit in einigen Bereichen den Status eines Prototyps weit hinter sich gelassen, und da durch eine einheitliche Schnittstelle die Integration neuer Algorithmen sehr einfach gestaltet wurde, bietet es sich somit als Basis für deren Weiterentwicklung an.

Im Bereich der Lösungsverfahren konnte eine auf Tabusuche beruhende Methode entwickelt werden. Obwohl das gewählte Verfahren auch in der Lage ist, Problemstellungen mit differenziertem Unterricht zu behandeln, schließt es durch Hybridisierung mit dem Algorithmus nach KÖNIG ebenso die Vorteile der für Unterricht ohne Differenzierungsstunden vorhandenen Verfahren mit ein, wobei hier sogar immer ein gültiger Stundenplan garantiert werden kann.

Des weiteren wurden erste Anfänge einer „Simulated Negotiation“ - Heuristik entwickelt, wobei die Tatsache, daß die vorgenommenen Implementierungen noch kein vollständig zufriedenstellendes Verhalten zeigen, nicht über das große Entwicklungspotential dieses Verfahrens hinwegtäuschen sollte. Dies ist auch insofern nicht sinnvoll, da sich durch die geeignete Wahl einzelner Routinen im Rahmen dieses Modells herkömmliche Verfahren wie „Simulated Annealing“ emulieren ließen, deren Funktionsfähigkeit inzwischen allgemein anerkannt ist.

Beide Verfahren sind, wie erwartet, vorläufig nur in Ansätzen für den praktischen Einsatz geeignet. Die Einschätzung der Parameter der verschiedenen Teile der Zielfunktion erweist sich hierbei unabhängig vom gewählten Verfahren als sehr schwierig, insbesondere wenn man intuitiv Formulierungen verwenden würde wie „aber nur wenn es nicht anders geht“. Dies führt dazu, daß bei Zwischenlösungen, bei denen tatsächlich keine andere Möglichkeit gegeben sein kann, von diesem Angebot rege Gebrauch gemacht wird, was es wiederum erlaubt, bei anderen Kriterien einen besseren Wert zu erzielen, als er in der besten Endlösung erwartet werden kann. Hier mit wechselnden Zielfunktionen zu arbeiten erwies sich als wenig ergiebig, da die ersten zu Beginn noch relativ frei setzbaren Stunden später über die relativen Nebenbedingungen den gesamten Plan determinieren. Dies läßt die Wirkung der geänderten Zielfunktion teilweise „verpuffen“.

Da diese Probleme bei rein theoretischen Betrachtungen nie sichtbar werden bzw. beim Stand der momentanen formalen Durchdringung auch noch nicht sichtbar werden können, bleibt hier somit nach wie vor das Verfolgen ganzheitlicher - d.h. Theorie und Praxis umfassender - Ansätze die einzige Möglichkeit, den Eigenheiten der automatischen Stundenplanerstellung gerecht zu werden.

10.1 Bemerkungen zur betrachteten Problemstellung

Da das Programm daraufhin konzipiert war, in möglichst verschiedenen Umgebungen eingesetzt zu werden, mußte bzw. konnte auf zahlreiche problemspezifische Eigenschaften verzichtet werden.

Das Weglassen statischer Typen wie „Lehrer“, „Klassen“ etc. führte dazu, daß die Ausgabe im Allgemeinen unstrukturierter wurde, da die entsprechenden Objekte nicht mehr unterschieden werden und daher (in den einzelnen Feldern) in beliebiger Reihenfolge auftreten können. Des weiteren machte dies eine typengebundene Unterscheidung zwischen „statischen“ und „dynamischen“ Ressourcen zunichte, was z.B. bei der Anpassung des Algorithmus dazu führte, daß zunächst Typen mit einem hohen Anteil an statischen Ressourcen identifiziert werden mußten, und Gruppen, die diese nicht aufwiesen, nicht verplant werden konnten.
Die umfassende Berücksichtigung dynamischer Ressourcen führte zu einem unerwartet hohen Anstieg der Komplexität, was mit sich brachte, daß z.B. die innere Schleife der Tabusuche eine Laufzeit von $\mathcal{O}(n^3)$ aufweist, was das Durchführen der hier normalerweise verwendeten mehreren zehntausend Iterationen unmöglich macht.
Vor diesem Hintergrund muß festgestellt werden, daß es nahezu unabdingbar ist, sich bei der Erstellung eines solchen Problems auf eine konkrete Institutionsform zu beschränken. Die gleichzeitige Zielvorgabe von Realschule und Fachhochschule ließ das Ausnutzen probleminhärenter Möglichkeiten nicht zu. Üblicherweise gibt es an der Realschule zumeist genügend Klassenräume, während die dort kritischen Klassenspringstunden an einer typischerweise unter Raumnot leidenden Fachhochschule keinerlei Problem darstellen.
Es ist aber sogar fraglich, ob selbst die Beschränkung auf eine Schulform ausreichend ist. In der Befragung (s. Abschnitt 9.3) ergab sich, daß sowohl Raumprobleme, als auch Personalprobleme in allen Schulformen gleichermaßen als schwierigster Teil genannt wurden. Einer der befragten Lehrer äußerte, daß das Problem in der „Constraint interaction“ begründet läge, wie sie z.B. in [Cor97] erwähnt wird.

10.2 Erweiterungsmöglichkeiten

10.2.1 Datenbankfunktionalität

Im Verlaufe der Anwendung zeigte sich, daß trotz der komplexen Tabellenformen[3] wesentlich mehr Funktionen benötigt werden, um die mehreren hundert Stunden zu verwalten. Hier wäre es sinnvoll, die Funktionalität einer externen Datenbank zur Verfügung zu stellen, insbesondere, da sich hier Makros zur Stundenplanfortschreibung etc. vom Anwender implementieren ließen, die in das Programm aufzunehmen unsinnig wäre, da sie sich von Institution zu Institution zu sehr voneinander unterscheiden. Auch die teilweise sehr aufwendigen aber heterogenen Verfahren zur Sicherstellung von Konsistenz, die in dieser Implementierung verwendet wurden, könnten auf diesem Wege durch eine einheitliche Methode ersetzt werden, wobei natürlich noch zu prüfen bliebe, ob sich dies mit der „Update“-Politik des „Visual C++ Frameworks“ vereinbaren ließe.

10.2.2 Berichtsgenerator

Der weitaus überwiegende Teil der befragten (s. Abschnitt 9.3) Anwender gab an, daß eine genaue Beschreibung eventuell auftretender Probleme für sie von großer Bedeutung sei. Bei der momentanen Implementierung werden diese zumeist farblich angedeutet. Eine genauere statistische Erfassung und Auswertung, im Falle von „personenbezogenen Härten“ könnten auch Histogramme hilfreich sein, wäre also mit Sicherheit sinnvoll. Allerdings müßte man, um dies einigermaßen effektiv zu gestalten, zu einer statischeren Betrachtung der Zeiteinheiten zurückkehren. Das momentan verwandte hochflexible Modell verlangt zumeist ein vollständiges Durchsuchen; eine Indizierung würde hier nur eine Beschleunigung in der Größenordnung der Zeitstundenzahl erreichen, da diese sehr häufig vorkommen.

[3] Die es in den meisten Datenbankprogrammen nicht gibt.

10.2.3 Weitere relative Nebenbedingungen

Stundenübergreifende relative Nebenbedingungen lassen sich leicht hinzunehmen, da für sie das selbe Bewertungsschema verwandt werden kann, das bereits für „dynamische“ Ressourcen eingesetzt wird, da diese zumeist nicht an den Kursen, sondern an ihren Teilnehmern ansetzen. Hier ließen sich nahezu beliebige Semantiken auf Paaren von Stunden einführen, wie etwa, daß diese am selben Tag liegen o.ä. Das dargestellte Propagierungsverfahren akzeptiert hierbei auch asymmetrische Relationen. Das Einführen von Relationen höherer Ordnung hingegen scheint nicht angebracht. Es gibt zwar mit Sicherheit Anforderungen seitens der Benutzer, die sich nur mit solchen Relationen formulieren ließen, aber es ist davon auszugehen, daß die korrekte Eingabe den Anwendern doch nahezu unmöglich sein dürfte. Da die Zahl möglicher Parameter mit der Relationsordnung exponentiell wächst, müßten noch komplexere Schemata als das hier verwandte Mengensystem eingesetzt werden, um eine effektive Eingabe der Werte zu ermöglichen.

10.2.4 Lehrerverteilung

Wie schon von Cooper und Kingston [CK95b] erwähnt, ermöglicht es das obige Gruppenschema im Prinzip, die Lehrerverteilung gleichzeitig zu erstellen. Auch wenn dies in der Realität selten zum Einsatz kommt, wäre es interessant festzustellen, wie stark sich die Komplexität des Problemes unter Hinzunahme derartiger Anforderungen vergrößert. Hierbei stellt sich sofort die Frage stellt, ob sie sich auch in der Praxis tatsächlich vergrößern muß, oder ob hier nicht die hinzugewonnenen Freiheitsgrade überwiegen, da es hin und wieder vorkommt, daß seitens der Stundenplanersteller eine Änderung des Dienstplanes verlangt wird, um leichter zu einem gültigen Stundenplan gelangen zu können. Im Rahmen dieser Arbeit erschien eine Hinzunahme nicht sinnvoll, da der Abstand von den ersten Vorarbeiten zu diesem Modell bereits ausreichend groß war.

10.3 Perspektiven

10.3.1 Problemstellungen

Es ist auffällig, daß bei bei den meisten Untersuchungen zum Timetabling die betrachteten Testfälle bestenfalls oberflächlich und statistisch untersucht werden, was zu Charakterisierungen wie „dicht“ oder „dünn“ führt. An keiner Stelle wird auf die dem jeweiligen Problem spezifischen Vorteile eingegangen. Auch wenn natürlich nicht für jede Institution ein neues Modell geschaffen werden kann, so scheint es doch sinnvoll, derartige Eigenschaften bei der Erfassung und Darstellung der Probleme mit zu berücksichtigen, da ohne eine Klassifizierung dieser Möglichkeiten deren Ausnutzung unmöglich bleibt. Diese Spezifika können zumeist nur im direkten Dialog mit den jeweiligen Planern gefunden werden, wobei hier aber zumeist nachgefragt wird, womit diese besondere Probleme haben und nicht, was noch nie schwierig gewesen sei.

10.3.2 Lernfähige Systeme

Zahlreiche „Timetabling"-Programme verlangen die Eingabe konkreter Parameter für den Lösungsalgorithmus. Die entsprechenden Stellen sind meist mit Hinweisen versehen, die den unerfahrenen Benutzer davor warnen, die „magic numbers" zu modifizieren. Da zumeist dieselbe Person mehrere Jahrzehnte mit der Erstellung von Stundenplänen beauftragt ist und zumeist mehrere Stundenpläne pro Jahr erstellt werden müssen, könnte es sich lohnen, ausgehend von einer komfortablen Nachbearbeitungsmöglichkeit festzustellen, welche Parameter zu einem dem nachträglich verbesserten Endplan ähnlichen Ergebnis geführt hätten. Dies würde wohl am ehesten der Tatsache Rechnung tragen, daß jeder Anwender eine andere Zielfunktion bevorzugt, ohne diesen mit der konkreten Parameterbestimmung zu belasten.

10.3.3 Lösungsalgorithmen

Dieses Projekt hat deutlich werden lassen, daß zwischen Theorie und Praxis in diesem Bereich eine große Lücke klafft. Nicht nur, daß es noch keine hinreichenden Modelle gibt, sondern auch weil nahezu alle Verfahren auf kleinen Testproblemen zufriedenstellend arbeiten, auf reellen Problemfällen aber nicht mehr die erwartete Leistung erbringen. All dies zeigt, das seitens der Informatik noch große Anstrengungen unternommen werden müssen, Methoden für die Probleme der „wirklichen" Welt zu entwickeln.

Nach dem momentanen Stand der Dinge wird die automatische Stundenplanerstellung wie in den letzten Jahren darauf angewiesen bleiben, Impulse von außen zu erhalten. Neue „Timetabling" -spezifische Ergebnisse kann es erst dann wieder geben, wenn eine bessere Abgrenzung gegenüber den allgemeinen kombinatorischen Optimierungsproblemen gefunden werden kann.

Literaturverzeichnis

[Aar89] E. H. L. Aarts. *Simulated Annealing and Boltzmann machines.* John Wiley & Sohns Ltd., 1989.

[AD93] D. Abramson and H. Dang. School timetables: A case study using simulated annealing. In *Applied Simulated Annealing, Lecture Notes in Economics and Mathematical Systems*, chapter 5, pages 104–124. Springer-Verlag, 1993.

[AG82] E.L. Allgover and K. Georg. Predictor-corrector and simplicial methods for approximating fixed points and zero points of nonlinear mappings. In A. Bachem, M. Grötschel, and B.Korte, editors, *Mathematical Programming*, The State of the Art, pages 15–57, Bonn, 1982.

[AKK98] Sanjeev Arora, David Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *JcSS*, 1998. noch nicht erschienen.

[Ali95] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.

[Bat96] Roberto Battiti. Reactive search: Toward self-tuning heuristics. In V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith, editors, *Modern Heuristic Search Methods*, chapter 4, pages 61–83. John Wiley and Sons Ltd., 1996.

[Bau97] N. Baumont. Scheduling staff using mixed integer programming. *Europ. J. Operat. Res.*, 98, 1997.

[BF96] Piotr Berman and Toshiro Fujito. On approximation properties of the independent set problem for degree 3 graphs. In *37th IEEE FOCS*, 1996.

[BGH97] R.N. Bailey, K.M. Garner, and M.F. Hobbs. Using simulated annealing and genetic algorithms to solve staff scheduling problems. *Asia-Pacific Journal of Operational Research*, 14:27–43, 1997.

[BHM96] A. Bachem, W. Hochstättler, and M. Malich. The simulated trading heuristic for solving vehicle routing problems. *Discrete Appl. Math.*, 65, 1996.

[Bin92] Ken Binmore. *Fun and games.* D.C. Heath and Company, Lexington, 1992.

[BJKW96] Edmund Burke, Kirk Jackson, Jeff Kingston, and Rupert Weare. Automated timetabling: The state of the art. http://www.asap.cs.nott.ac.uk, 1996.

[BK98] Piotr Berman and Marek Karpinski. On some tighter inapproximability results, further improvements. Technical Report TR98-065, Electronic Colloquium on Computational Complexity (ECCC), 1998. http://www.eccc.uni-trier.de/eccc.

[BKP97] Edmund K. Burke, J.H. Kingston, and P.A. Pepper. A standart data format for timetabling instances. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 98–105, 1997.

[Blu98] Norbert Blum. *Theoretische Informatik.* Oldenbourg Verlag, München,Wien, 1998.

[BNW95] E. K. Burke, J. P. Newall, and R. F. Weare. A memetic algorithm for university exam timetabling. In *1st International Conference on the Practice and Theory of Automated Timetabling (ICPTAT'95, Napier University, Edinburgh, UK, 30th Aug - 1st Sept 1995)*, pages 496–503, 1995.

[Bro95] A. E. Brouwer. Block designs. In M. Graham, L. Grötschel, and L. Lovasz, editors, *Handbook of Combinatorics*, volume 1, chapter 14, pages 693–746. Elsevier, 1995.

[Car] M. Carter. Timetabling literature. ftp://ftp.cs.nott.ac.uk/ttp.

[Cat98] Oliver Catoni. Solving scheduling problems by simulated annealing. *SIAM J. Control Optim.*, 36(5):1539–1575, 1998.

[CCKV96] Michele Conforti, Gerard Cornuejols, Ajai Kapoor, and Krisina Vusovic. Perfect matchings in balanced hypergraphs. *Combinatorica*, 16(3):325–329, 1996.

[CdW89] N. Chahal and D. de Werra. An interactive system for constructing timetables on a PC. *European Journal of Operational Research*, 40:32–37, 1989.

[CG97] William H. Cunningham and James F. Geelen. The optimal path-matching problem. *Combinatorica*, 17(3):315–337, 1997.

[CK81] Vincent P. Crawford and Elsie Marie Knoer. Job mathcing with heterogeneous firms and workers. *Econometrica*, 49.2, March 1981.

[CK93] T. B. Cooper and J. H. Kingston. The solution of real instances of the timetabling problem. *The Computer Journal*, 36:645–653, 1993.

[CK95a] Tim B. Cooper and Jeffrey H. Kingston. The complexity of timetable construction problems. In *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95)*, pages 511–522, 1995.

[CK95b] Tim B. Cooper and Jeffrey H. Kingston. A program for constructing high school timetables. In *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95)*, pages 132–143, 1995.

[CL92] J. Csima and L.Lovasz. A matching algorithms for regular bipartite graphs. *Discrete Applied Mathematics*, 35:197–203, 1992.

[CLR96] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT electrical engeneering and computer science series. MIT Press, 1996.

[CO97] David Corne and John Ogden. Evolutionary optimisation of methodist preaching timetables. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 321–323, 1997.

[Cor97] David Corne. Tradeoffs and phase transitions in the interactions between exam splitting, clash, near clash, and room capacity constraints. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 318–320, 1997.

[Cos94] D. Costa. A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research*, 76:98–110, 1994.

[CR97] J.P. Caldeira and A.C. Rosa. School timetabling using genetic search. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 115–122, 1997.

[CR98] Alberto Caprara and Romeo Rizzi. Improving a family of approximation algorithms to edge color multigraphs. http://www.deis.unibo.it, 1998.

[CT92] M. W. Carter and C. A. Tovey. When is the classroom assignment problem hard? *Operations Research*, 40:28–39, 1992. Supp. No. 1.

[Cur88] Inmaculata Jacinta Curiel. *Cooperative Games*. PhD thesis, Katholieke Universiteit van Nijmegen, 1988.

[Cur96] Inmaculata Jacinta Curiel. Cooperative game theory and applications. Curacao, 1996.

[Die96] Reinhardt Diestel. *Graphentheorie*. Springer, Berlin,Heidelberg,New York et al., 1996.

[Dou95] P. Douchet. Hypergraphs. In M. Graham, L. Grötschel, and L. Lovasz, editors, *Handbook of Combinatorics*, volume 1, chapter 7, pages 381–433. Elsevier, 1995.

[Dow97] Kathryn A. Dowsland. Off-the-peg or made-to-measure? timetabling and scheduling with SA and TS. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 7–26, 1997.

[DS90] Gunter Dueck and Tobias Scheuer. Threshold accepting: A general purpose optimisating algorithm appearing superior to simulated annealing. *J. comp. physics*, 90:161–175, 1990.

[dW71] D. de Werra. Construction of school timetables by flow methods. *INFOR*, 9:12–22, 1971.

[dW85] D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19:151–162, 1985.

[dW95] D. de Werra. Some combinatorial models for course scheduling. In *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95)*, pages 1–20, 1995.

[dW96] D. de Werra. Extensions of coloring modells for scheduling purposes. *Europ. J. Operat. Res.*, 1996.

[dW97] D. de Werra. Restricted coloring models for timetabling. *Discrete Math.*, 165/166:161–170, 1997.

[dWE96] D. de Werra and J. Eschler. Open shop sheduling with some additional constraints. *Graphs and Combinatorics*, 1996.

[dWM95] D. de Werra and N. V. R. Mahadev. Preassignment requirements in chromatic scheduling. ORWP 95/02, EPFL, 1995. Submitted for publication.

[dWMP93] D. de Werra, N. V. R. Mahadev, and U. N. Peled. Edge chromatic scheduling with simultaneity constraints. *SIAM Journal on Discrete Mathematics*, 6:631–641, 1993.

[ECF97] Saleh Elmohamed, Paul Coddington, and Geoffrey Fox. A comparision of annealing techniques for academic course scheduling. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 146–166, 1997.

[EdHH98] A.E. Eiben, J.K. Van der Hauw, and J.I. Van Hemert. Graph coloring with adaptive evolutionary algorithms. *J. Heuristics*, 4:25–46, 1998.

[EIS76] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5:691–703, 1976.

[FMP96] Klaus Fischer, Jörg P. Müller, and Makus Pischel. Cooperative transportation scheduling: an application domain for DAI. Technical report, DFKI GmbH Saarbrücken, 1996.

[Gal84] David Gale. Equilibrium in discrete exchange economy with money. *Int. J. Game Theory*, 13(1):61–64, 1984.

[GLS88] M. Grötschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, chapter 9. Springer, Berlin, 1988.

[Got63] C. C. Gotlieb. The construction of class-teacher time-tables. In *Proc. IFIP Congress Munchen 1962*, pages 73–77, 1963.

[GPS92] Lars Gislen, Carsten Peterson, and Bo Söderbarg. Complex scheduling with potts neural networks. *Neural Computation*, 4:805–831, 1992.

[GPU98] GPUntis98, 1998. Messevorführung auf der C'Bit 98 Hannover.

[Hal93] Magnús M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Information processing Letters*, 45:19–23, 1993.

[Has96] Johan Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *37th IEEE FOCS*, 1996.

[Hax95] P.E. Haxel. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 1995.

[Hef97] A. Hefner. A min-max theorem for a constraint matching problem. *Siam. J. Discrete Math.*, 10, 1997.

[Her92] A. Hertz. Finding a feasible course schedule using tabu search. *Discrete Applied Mathematics*, 35:225–270, 1992.

[HH93] R. Häggkvist and T. Hellgren. Extensions of edge-colourings in hypergraphs I. In *Combinatorics, Paul Erdös is Eighty*, volume 1 of *Mathematical Studies*, pages 215–238. Bolai Society, Keszthely (Hungary), 1993.

[HHW96] Tad Hogg, Bernardo A. Huberman, and Colin P. Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81:1–15, 1996.

[Hil81] A. J. W. Hilton. School timetables. *Annals of Discrete Mathematics*, 11:177–188, 1981.

[HK76] W. Hildenbrand and A.P. Kirman. *Introduction to Equilibrium Analysis*. Elsevier, Amsterdam, 1976.

[Hoc97] Dorit Hochbaum, editor. *Approximation algorithms for NP-hard problems*. PWS Publishing Company, Boston, 1997.

[Huc] A. Huck. Mathematische Methoden des Operations-Research III. Vorlesung, gehalten am Institut für Diskrete Mathematik WS 96/97.

[Jan97] Klaus Jansen. The optimum cost chromatic partition problem. In *LNCS 1203*. Springer, 1997.

[JT95] Tommy R. Jensen and Bjarne Toft. *Graph Coloring Problems*. John Wiley & Sons, New York, 1995.

[Jun86] W. Junginger. Timetabling in germany - A survey. *Interfaces*, 16:66–74, 1986.

[Jun90] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI-Wissenschaftsverlag, Mannheim, Wien, Zürich, 2 edition, 1990.

[KMSV94] Sanjeev Khanna, Rajeev Motwani, Madhu Sudan, and Umesh Vazirani. On syntactic versus computational views of approximability. In *Proccedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 819–830, 1994. http://www.stanford.edu/people/motwani.

[Knu94] Donald E. Knuth. The sandwich theorem. *The Electronic journal of Combinatorics*, 1:#A1, 1994.

[KS96] Sanjeev Khanna and Madhu Sudan. The optimisation complexity of constraint satisfaction problems. Technical Report TR96-028, Electronic Colloquium on Computational Complexity (ECCC), 1996. http://www.eccc.uni-trier.de/eccc.

[Laj95] Gyuri Lajos. Complete university modular timetabling using constraint logic programming. In *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95)*, pages 364–375, 1995.

[Lec87] M. Leclerc. *Algorithmen für kombinatorische Optimierungsprobleme mit Partitionsbeschränkungen*. PhD thesis, Universität Köln, 1987.

[Lew61] C. F. Lewis. *The School Time Table*. Cambridge University Press, 1961.

[Lio66] J. Lions. Matrix reduction using the hungarian method for the generation of school timetables. *Communications of the ACM*, 9:349–354, 1966.

[LS95] R. Lindermeier and F. Siebert. *Softwareprüfung und Qualitätssicherung*. Oldenbourg Verlag, 1995.

[LY94] Carsten Lund and Mihalis Yannakakis. On the hardness of approximation minimization problems. *J. ACM*, 41(5):960–981, 1994.

[Mar98] Michael Marte. Constraint-based grammar school timetabling: A case study, 1998. Diplomarbeit Institut für Informatik TU München.

[Mic92] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Artificial Intelligence. Springer, 1992.

[Mim98] 1998. http://www.solnet.net/ mimosa.

[Miy98] Mitsunobu Miyake. On the incentive properties of multi-item auctions. *Int. J. Game Theory*, 27:1–19, 1998.

[MK98] H.H. Millar and M. Kiragu. Cyclic and non-cyclic scheduling of 12h shift nurses by network programming. *European Journal of operational research*, 104(1998), 1998.

[MM96] Helmut E. Mausser and Michael J. Magazine. Comparision of neural and heuristic methods for a timetabling problem. *Europ. J. Operat. Res.*, 93:271–287, 1996.

[MR97] Nuno Mamede and Tiago Rente. Repairing university timetables using genetic algorithms and simulated annealing. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 187–204, 1997.

[MR98] Andrew J. Mason and David M. Ryan. Integrated simulation, heuristic and optimistation approaches to staff scheduling. *Op. Res.*, 46.2:161–175, 1998.

[PHB99] Jan Puzicha, Thomas Hofmann, and Joachim M. Buhmann. Deterministic annealing: Fast physical heuristics for real-time optimisation of large systems. In *Proceedings of the 15th IMACS World Conference on Scientific Computation Modelling and Applied Mathematics*, 1999? noch nicht erschienen.

[PS89] Nicholas Pippenger and Joel Spencer. Asymptotic behaviour of the chromatic index for hypergraphs. *J. Combinatorial Theory*, A 51:24–42, 1989.

[Pul95] W.R. Pulleyblank. Matchings and extensions. In M. Graham, L. Grötschel, and L. Lovasz, editors, *Handbook of Combinatorics*, volume 1, chapter 3, pages 179–233. Elsevier, 1995.

[Rei90] Karl Rüdiger Reischuk. *Einführung in die Komplexitätstheorie*. Teubner, Stuttgart, 1990.

[RF88] D.M. Ryan and J.C. Falkner. On the integer properties of scheduling set partitioning models. *Europ. J. Operat. Res.*, 35:442–456, 1988.

[RHC97] Peter Ross, Emma Hart, and Dave Corne. Some observations about GA-based exam timetabling. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 55–71, 1997.

[Rom82] Bernardo Prida Romero. Examination scheduling in a large egineering school: a computer assisted participative procedure. *Interfaces*, 12(2):12–24, 1982.

[RW97] Vevek Ram and Peter Warren. Automated timetable generation through distributed negotiation. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 345–346, 1997.

[SAW98] K. Steinhöfel, A. Abrecht, and C.K. Wong. On various cooling schedules for simulated annealing applied to the job shop problem. In Michael Luby, Jose Rolim, and Maria Serna, editors, *Randomization and Approximation techniques in Computer Science*, page 260ff. Springer, 1998. LNCS 1518.

[Sch94] Charles H. Schmauch. *ISO 9000 for Software Developers*. ASQC Quality Press, 1994.

[Sch95] Andrea Schaerf. A survey of automated timetabling. Technical Report CS-R9569, Centrum vor Wiskunde en Informatica, 1995.

[Sch96] A. Schaerf. Tabu search techniques for large high-school timetabling problems. CWI-Report CS-R9611, Centrum voor Wiskunde en Informatica, Amsterdam, 1996.

[Sed92] Robert Sedgewick. *Algorithmen*. Addison-Wesley, Bonn,Reading, 1992.

[Sli96] Tomaz Slivink. Short proof of galvins theorem on the list-chromatic index of a bipartite multigraph. *Combinatorics, Probability and Computing*, 5:91–94, 1996.

[SM97] Pedro Suares and Nuno Mamede. Micro-opportunistic timetabling. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, 1997.

[SS66] L.S. Shapley and M. Shubik. Quasi-cores in a monetary economy with non-convex preferences. *Econometrica*, 34(4):805ff, 1966.

[SS78] L. A. Steen and J.A. Seebach. *Counterexamples in Topology*. Springer, New York, Heidelberg, 1978.

[SS79] G. Schmidt and T. Strohlein. Timetable construction - an annotated bibliography. *The Computer Journal*, 23:307–316, 1979.

[Stü97] Matthias Stüber. Real world timetabling:a pragmatic view. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 258–267, 1997.

[Tuz97] Zsolt Tuza. Graph colorings with local constraints a survey. *Discussiones Mathematicae Graph Theory*, 17(2):161–228, 1997.

[VB96] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.

[Vyg] Jens Vygen. Approximationsalgorithmen. Vorlesung, gehalten am Institut für Diskrete Mathematik SS98. Skript; Heausgabe im Rahmen eines Buches angekündigt.

[WK99] A. Wren and R. S. K. Kwan. Installing an urban transport scheduling system. *J. of Scheduling*, 2(1):3–19, 1999.

[Wre95] Anthony Wren. Scheduling, timetabling and rostering - a special relationship? In *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95)*, pages 474–495, 1995.

[WW98] J. Wood and D. Whitaker. Student centered school timetabling. *Journal of the Operational Research Society*, 49:1146–1152, 1998.

[WZ97] George M. White and Junhan Zhang. Generating complete university timetables by combining tabu search with constraint logic. In *Proceedings of the second conference on practice and theory of automated timetabling (PATAT)*, pages 268–277, 1997.

[ZG81] W.I. Zangwill and C.B. Garcia. Equilibrium programming: The path following approach and dynamics. *Mathematical Programming*, 21:262–289, 1981.

Anhang A

Beispielausdruck

Der folgende Ausdruck zeigt die jeweils ersten Stunden aller Tage für einen Plan, der aus anonymisierten Daten des ersten Testfalls (Kapitel 9) erzeugt wurde. Die Felder sind nicht vollständig gefüllt, da eine andere Zeitstunde eine größere Anzahl von Schulstunden enthält und die Tabellenfelder immer gleich groß gewählt werden.

Anonym1 Sun, 31. January;01:52

	Montag	Dienstag	Mittwoch	Donnersta	Freitag	Samstag
1 .	Para K10 L AH	Audi Comp K10 AA	Algo Geoi Y RGeo	Info K3 AF G	Para Date Algo K3	Algo Auto K10 D
	Rech Q AN RK6	Bioi Audi C++ K3	Algo KI K3 K	KI Date Algo K16	C++ Comp Rech K16	Para Audi Rech K16
	Algo Comp Rech K16	Soft Algo Info K16	KI Info K16 R	Bioi RK6 K11 AH	Comp RCom K1 J	C++ K11 AE RK18
	KI RK8 K1 R	Bild Date Comp K1	Date Bild K8 I	Info K1 AP RK4	Rech RK21 K15 AG	Geoi RGeo K12 H
	Algo REle K15 AJ	Info RK15 K12 G	Komp K20 L RKom	Comp RCom K15 J	Algo RAlg K12 D	Comp K8 Z RCom
	Rech RK12 K12 AO	Audi RAud K8 T	KI RK20 K18 O	Algo RAlg K13 D	Geoi RGeo K8 H	Info K13 G RK20
	Para RK15 K8 AH	Soft K13 AN RK1	Bioi K4 B RK13	Soft Date K18 X	C++ RK9 K13 E	Algo RAlg K20 AB
	KI K20 A RK1	KI K20 A RK11	Info K6 AI RK8	Auto RAut K6 AM	Info RK14 K20 AP	Info RK13 K18 AP
	Rech K9 AL RK11	Algo REel K18 AJ	Info RK12 K14 V	Komp RKom K14 K	C++ K18 F RK6	Komp RKom K4 K
		Algo K17 AK REle	Rech K9 AL RK15	KI RK8 K9 Y	C++ K4 AF RK4	Audi K6 AQ RAud
			Algo K2 P RAlg	KI K21 A RK12	Komp RKom K6 K	KI RK8 K14 Y
			C++ RK1 K19 AH	Komp K2 H RKom	Rech K14 AA RK18	Comp K2 J RCom
			Komp K7 L RKom	Comp K19 O RCom	Para K9 L RK20	Para RK12 K19 AG
			KI K17 AK RK11	Rech RK15 K7 AO	KI K21 A RK13	C++ K7 E RK15
			Audi K5 S RAud	Rech RK1 K17 AL	C++ K2 A RK8	Info RK1 K17 AF
				KI K5 Z RK11	Info K19 V RK12	C++ K5 F RK11
					Info RK15 K7 N	
					C++ K17 I RK1	
					KI K5 Z RK11	