Informatik Forsch. Entw. (2004) 19: 65–73
Digital Object Identifier (DOI) 10.1007/s00450-004-0160-x

Informatik
Forschung und Entwicklung

Automatische und interaktive Stundenplanung

Ulrich Geske, Hans-Joachim Goltz

Fraunhofer FIRST, Kekuléstr. 7, 12489 Berlin (e-mail: {geske,goltz}@first.fraunhofer.de)

Eingegangen am 15. März 2003 / Angenommen am 9. März 2004
Online publiziert: 1. Juli 2004 –

Zusammenfassung. Das System für die interaktive, automatische Stundenplanung ist im Rahmen der Forschungsarbeiten des Bereichs Planungstechnik und Deklarative Programmierung in Fraunhofer FIRST zur Erweiterung der *Constraint-basierten Programmierung* entwickelt worden. Mit dem System wird die Stundenplanung der Medizinischen Fakultät Charité seit dem Sommersemester 1998 vorgenommen. Seitdem wurde das System kontinuierlich weiterentwickelt. Der erfolgreiche Einsatz des Systems zeigte, dass die gewählten Methoden und Verfahren sehr geeignet für die Behandlung derartiger Probleme sind. Die Vorteile einer kombinierten interaktiven und automatischen Stundenplanerzeugung konnten eindeutig nachgewiesen werden.

Schlüsselwörter: automatische Stundenplanung, constraint-basierte Programmierung, Constraints, Scheduling

Abstract. The interactive, automatic timetabling system was developed in Fraunhofer FIRST's department for Planning Technology and and Declarative Programming in the frame of research on the constraint-based programming paradigm. Since 1998, the system has been used to generate timetables for the Charité Medical Faculty at Berlin's Humboldt University and has undergone continuous extensions. Successful deployment of the system shows that the chosen methods and techniques are well suited for dealing with problems of this sort. The quality of the plans and their permanent use demonstrate the advantages of combined interactive and automatic generation of timetables.

Keywords: Automated timetabling, Constraint-based Programming, Constraints, Scheduling

CR Subject Classification: I.2.8, I.2.3, J.1, K.3.2, D.3.3, D.1.6

1 Einleitung

Ein Stundenplan ist eine Zuordnung von Ereignissen zu Zeitpunkten bzw. Zeitintervallen, so dass die geforderten Bedingungen erfüllt sind. Bei dieser Zuordnung sind problemspezifische Bedingungen zu beachten. Neben den Bedingungen, die immer erfüllt sein müssen, existieren auch Bedingungen, die möglichst erfüllt sein sollen und auch als weiche Bedingungen bezeichnet werden können.

Stundenplanungsprobleme sind im allgemeinen sehr komplexe Probleme. Seit langem ist bekannt, dass diese Probleme zu der Klasse der NP-vollständigen Problemen gehören (siehe z.B. [6]). Existierende Softwareprodukte für die Stundenplanung schränken häufig die möglichen erzeugbaren Pläne zu stark ein. Außerdem sind sie oft nicht flexibel genug, um abweichende Anforderungen integrieren zu können und Besonderheiten zu berücksichtigen, die es bei fast jedem Anwender gibt.

Die automatische Stundenplanung ist ein aktuelles Forschungsgebiet. Verbesserungen bei Lokalen Suchverfahren, neuere Paradigmen sowie Genetische Algorithmen oder Constraint-basierte Verfahren berechtigen zu der Hoffnung, dass eine Behandlung des Problems mit Rechnerunterstützung unter bestimmten Voraussetzungen möglich ist.

Die constraintlogische Programmierung konnte in vielen Anwendungen zur Lösung komplexer diskreter Probleme sehr erfolgreich eingesetzt werde. Verschiedene Ansätze zur Anwendung dieser Programmierung zur Lösung von Stundenplanungsproblemen werden beispielsweise in [3,12,13,15,2] diskutiert. Unser Ansatz ist, das Computersystem als Assistenten anzusehen und eine kombinierte automatische und interaktive Arbeitsweise anzustreben.

Auf Grund der bisherigen Erfahrungen ([9]) mit Constraint-basierter Programmierung bei der Behandlung kombinatorischer Probleme, wie Werkstatt- oder Dienstplanung, wurde dieses Paradigma auch als Grundlage für die Stundenplanerzeugung verwendet. Ein Ziel unserer Forschung ist die Entwicklung von Methoden, Verfahren und Konzepten für eine interaktive, automatische Stundenplanung, die in Universitäten und Schulen angewendet werden kann. Die automatische Lösungssuche soll dabei so realisiert werden, dass möglichst eine Lösung in relativ kurzer Zeit gefunden wird, falls eine existiert. Für die Realisierung einer effizienten automatischen Lösungssuche ist die Problemmodellierung von entscheidender Bedeutung. Wichtige Forschungsschwerpunkte sind für uns deshalb die Entwicklung von Konzepten und Methoden der Modellierung von Problemen der Stundenplanung, sowie Untersuchungen zum Einfluss verschiedener Problemmodellierungen auf die Lösungssuche.

Die Systeme der Stundenplanung sollen so flexibel sein, dass Besonderheiten der Anwender berücksichtigt und Bedin-

gungen leicht geändert werden können, falls keine grundsätzliche konzeptionelle Änderung der Stundenplanung erforderlich ist. Die Constraint-basierte Programmierung bildet die Grundlage zur Verwirklichung unserer Forschungsziele.

2 Problembeschreibung

Stundenpläne zu erstellen, kann allgemein beschrieben werden, durch die Aufgabe Zeiten, Fächer, Lehrer/Dozenten, Klassen/Semester und Räume den Lehrveranstaltungen so zuzuweisen, dass keine Überschneidungen auftreten. Allerdings haben offenbar unterschiedliche Einrichtungen verschiedene und weitere spezielle Bedingungen, die zusätzlich einzuhalten sind. Wir verwenden hier als Grundlage der Darstellung die Bedingungen der Medizinischen Fakultät Charité der Humboldt-Universität Berlin. Schon hier zeigt sich, dass es unterschiedliche Ausprägungen der Stundenplanung gibt, da für Vorlesungspläne, Seminarpläne und Praktikumspläne unterschiedliche Regelungen zu beachten sind. An der Charité sind beispielsweise Seminare von den Vorlesungen zeitlich getrennt und durch ein Seminargruppensystem geregelt. Die Breite der behandelten Phänomene erleichtert eine Übertragung des Planungsansatzes auf andere Einrichtungen.

In der medizinischen Ausbildung müssen die Studenten relativ viele Pflichtveranstaltungen absolvieren. Neben den Vorlesungen eines Semesters finden Seminare, Untersuchungskurse und Praktika statt, die in Gruppen von bis zu 20 Studenten durchgeführt werden. Dazu werden die Studenten eines Semesters in Seminargruppen unterteilt. Für jede Seminargruppe wird ein konfliktfreier Stundenplan mit den Pflichtveranstaltungen erzeugt. Die Studenten schreiben sich dann für eine Seminargruppe ein und können so alle Pflichtveranstaltungen des Semesters absolvieren. Zusätzlich hat jeder Student die Möglichkeit, sich für weitere Veranstaltungen einzuschreiben.

In bestimmten Studienabschnitten müssen die Studenten Praktika verschiedener Fächer absolvieren, die in der Regel jeweils an aufeinanderfolgenden Werktagen stattfinden, wobei die Anzahl der Tage bzw. Wochen vorgegeben ist. Diese Praktika werden als Blockpraktika bezeichnet.

Der gesamte Stundenplan besteht aus verschiedenen Komponenten (Vorlesungen, Seminare/Kurse, Praktika). Für die Planung kann jede Komponente relativ unabhängig betrachtet werden, wobei aber eine bestimmte Reihenfolge einzuhalten ist. Zuerst sind die Vorlesungen für alle Semester zu planen. Anschließend können dann für jedes Semester die Lehrveranstaltungen geplant werden, die in Seminargruppen durchgeführt werden, wobei die relevanten Vorlesungszeiten als "Sperrzeiten" zu berücksichtigen sind. Die Planung der Blockpraktika kann völlig unabhängig erfolgen, da diese tageweise in vorgegebenen vorlesungsfreien Wochen stattfinden.

Im folgenden werden wichtige Bedingungen dieses Problems der Stundenplanung kurz beschrieben.

2.1 Vorlesungen, Seminare und Kurse

Für Vorlesungen, Seminare und Kurse ist ein Wochenplan zu erzeugen, der im allgemeinen für alle Wochen eines Semesters gilt. Einige Lehrveranstaltungen finden dabei nur in bestimmten Wochen statt. Die Lehrveranstaltungen haben unterschiedliche vorgegebene Längen. Für jede Lehrveranstaltung existieren zeitliche Einschränkungen bezüglich ihrer Durchführung. Vorlesungen eines Semesters bzw. Lehrveranstaltungen einer Seminargruppe dürfen sich zeitlich natürlich nicht überlappen, damit jeder Student sie auch besuchen kann.

Die Fakultät hat mehrere Standorte, die sich räumlich in unterschiedlicher Entfernung befinden. Für jede Seminargruppe sind folglich die Wege zwischen den möglicherweise verschiedenen Veranstaltungsorten der Lehrveranstaltungen zu berücksichtigen.

Als Ressourcen sind den Vorlesungen Dozenten und Räume konfliktfrei zuzuordnen. Die Raumplanung für die Lehrveranstaltungen kann auch von der Zeitplanung getrennt erfolgen, was insbesondere angebracht ist, wenn die Räume in den verschiedenen Standorten der Fakultäten lokal verwaltet werden. Als sogenannte *weiche* Bedingungen werden außerdem Wunschzeiten und Raumwünsche für Lehrveranstaltungen mit Priorität, soweit möglich, berücksichtigt.

Für jedes Fach ist die Anzahl der Kurse und Seminare, die gleichzeitig durchgeführt werden können, beschränkt.

Weitere Bedingungen der Stundenplanung, die berücksichtigt werden, sind:

1. Die Lehrveranstaltungen können zu jeder Viertelstunde beginnen.
2. Für jede Lehrveranstaltung existieren zeitliche Einschränkungen bezüglich ihrer Durchführung.
3. Zwei Vorlesungen des gleichen Faches sind an verschiedenen Wochentagen einzuplanen.
4. Einige Lehrveranstaltungen finden nur in bestimmten Wochen statt (u.a. Unterteilung in A- und B-Wochen).
5. Eine Lehrveranstaltung, die eine Seminargruppe zu absolvieren hat, kann zur fast gleichen Zeit auch an verschiedenen Orten angeboten werden. Dies kann auch bei Vorlesungen der Fall sein. Der Student kann entscheiden, welche er besucht.
6. Vorlesungen eines Semesters bzw. Lehrveranstaltungen einer Seminargruppe dürfen sich zeitlich nicht überlappen.
7. Den Vorlesungen können Räume entweder unmittelbar zugeordnet werden oder es wird zunächst nur ihre zeitliche Einordnung bestimmt. Viele Vorlesungen können dabei nur in bestimmten Räumen stattfinden.
8. Die Anzahl der Seminare, Untersuchungskurse und Praktika, die eine Klinik gleichzeitig durchführen kann, ist beschränkt, wobei die Schranke auch zeitabhängig sein kann.
9. Die Wunschzeiten und Raumwünsche für Vorlesungen sind möglichst zu berücksichtigen.

2.2 Blockpraktika

Blockpraktika finden in einem vorgegebenen Zeitraum statt und werden in Gruppen von bis zu 20 Studenten durchgeführt. Die Dauer der Praktika ist jeweils vorgegeben. Bei der Planung von Blockprakika sind nur die Tage zu bestimmen, an denen sie stattfinden sollen. Die zugeordnete Tageszeit ist für die Planung nicht von Bedeutung. Ein gegebenes Praktikum ist von allen Gruppen oder einer angegebenen Menge von Gruppen zu absolvieren. Als *Praktikumseinheit* oder *Einheit eines Praktikums* bezeichnen wir im folgenden das Praktikum, das für eine bestimmte Gruppe stattfindet. Folglich kann jedem Praktikum eine Menge von *Praktikumseinheiten* zugeordnet werden.

An der Charité werden zwei Arten von Blockpraktika unterschieden: *Wochen-Praktika* und *Tages-Praktika*. Den Wochen-Praktika sind eine bestimmte Anzahl von Wochen zuzuordnen. Wenn mehr als eine Woche zuzuordnen ist, müssen diese Wochen aufeinanderfolgen. Diese Praktika beginnen immer am ersten Werktag der Woche. Bei der Planung dieser Praktikumsart werden Feiertage ignoriert. Folglich kann die Anzahl der Tage, an denen diese Praktika tatsächlich stattfinden, unterschiedlich sein. Dagegen ist bei den Tages-Praktika diese Anzahl fest. Den Tages-Praktika ist eine bestimmte Anzahl von aufeinanderfolgenden Werktagen zuzuordnen, wobei freie Tage (Wochenende, Feiertage) nicht mitgezählt werden. Praktika dieser Art können an jedem Werktag beginnen. Die wichtigsten Bedingungen für Blockpraktika sind außerdem:

1. Die Praktikumseinheiten, die einer Gruppe zugeordnet sind, dürfen sich nicht überlappen.
2. Die Anzahl der Einheiten eines Praktikums, die zur gleichen Zeit stattfinden dürfen, ist beschränkt. Diese Schranke muss nicht für die gesamte Zeit gleich sein.
3. Einige Praktika dürfen an bestimmten Tagen nicht beginnen.
4. Für jeweils zwei Einheiten eines Praktikums kann die folgende Bedingung gefordert werden: Entweder sie beginnen am gleichen Tag oder sie dürfen sich nicht überlappen.
5. Die Anzahl der Tage, an denen die Praktikumseinheiten eines Praktikums beginnen dürfen, kann als minimal oder maximal vorgegeben sein, wobei auch eine genaue Anzahl dieser Tage vorgegeben werden kann.

3 Constraint-Systeme

3.1 Constraint-basierte Programmierung

Das Paradigma der Constraint-basierten Programmierung unterscheidet sich wesentlich von dem der klassischen Programmierung. Durch den deklarativen Charakter der Constraints ist eine effiziente und flexible Softwareentwicklung möglich und die Entwicklung korrekter Software wird unterstützt. Problemmodifikationen sind relativ einfach zu realisieren. Die Art und Weise der Verarbeitung bzw. der Berücksichtigung der Constraints führt zu großen Suchraumeinschränkungen und ermöglicht die schnelle Erzeugung von Lösungen, die nahe dem Optimum liegen. Die Constraint-basierte Programmierung kann insbesondere zum Lösen komplexer Suchprobleme, die mittels Constraints (Bedingungen) formulierbar sind, erfolgreich eingesetzt werden. Anwendungsgebiete sind beispielsweise, außer der Stundenplanung, Probleme aus dem Bereich Produktionsplanung, wie Ablaufplanung, Kapazitätsplanung, Transportplanung und Personalplanung, und aus dem Bereich Konfiguration, wie Schaltkreisentwurf, Konfiguration von Produkten und Konfiguration technischer Anlagen.

Probleme werden auch in der Constraint-Programmierung durch eine Menge von Variablen sowie durch Funktionen und Relationen – insbesondere arithmetischen – zwischen den Variablen repräsentiert. Derartige Relationen beschreiben im allgemeinen Bedingungen (auch Restriktionen oder Constraints genannt), die an eine zulässige Variablenbelegung gestellt werden. In klassischen Systemen können diese Bedingungen, die dort zuweilen auch bereits als Constraints bezeichnet werden, erst nach dem Belegen der Variablen berechnet werden (Trial-and-Error-Verfahren). In der Constraint-Programmierung dagegen wird das Behandeln der Constraints (wie z.B.: `BeginnVorlesung1` $<$ `BeginnVorlesung2`) vor die Generierung von Variablenbelegungen gezogen. Die spezifizierten Constraints mit zum Teil gemeinsamen Variablen bilden ein Constraint-Netz mit den Variablen als Knoten und den relationalen Beziehungen als Kanten. Wenn die Domäne D_v einer Variablen v zu D_v^1 eingeschränkt wird, werden durch den als Dämon arbeitenden ConstraintLöser sofort die Konsequenzen für die anderen Variablen, die mit dieser Variablen über Constraints verbunden sind, berechnet. Bei diesem Vorgang, der Constraint-Propagation, werden aus den Domänen der Variablen die Werte gelöscht, die das Constraint nicht erfüllen können. Wird dabei die Domäne einer Variablen leer - kann die Variable also keinen Wert mehr annehmen – liegt eine Sackgasse vor, die durch Backtracking zu einer alternativen Einschränkung D_v^2 umgangen werden kann. Der entscheidende Vorteil liegt in der Erkennung dieser Situation bevor allen Variablen Werte zugewiesen wurden. Daraus resultiert eine sehr effiziente Abarbeitung, insbesondere bei einer großen Anzahl zu berücksichtigender Constraints.

Durch Constraint-Propagation wird der Suchraum verkleinert, ohne dass potentielle Lösungen abgeschnitten werden. Eine eindeutige Lösung liegt vor, wenn die Domänen aller Variablen genau einen Wert enthalten. Im Allgemeinen werden eindeutige Lösungen erreicht, indem im verbleibenden Lösungsraum, der durch die reduzierten Domänen der Variablen definiert ist, durch Auswahl je eines Wertes für die Variablen (mit Hilfe von Heuristiken) nach einer Lösung gesucht wird (Labeling).

Constraints wie das oben erwähnte "$<$"-Constraint oder auch Gleichheits- und Ungleichheitsconstraints ("$=$", "$\backslash=$") sind Basis-Constraints. Daneben bieten Constraintsysteme häufig auch zusätzliche Constraints, die komplexere Teilprobleme behandeln. Derartige Konstrukte werden als *globale Constraints* bezeichnet. Durch den Aufruf *eines* globalen Constraints kann beispielsweise die folgende Bedingung für jede Seminargruppe formuliert werden: *Die Lehrveranstaltungen, die der Seminargruppe zugeordnet sind, dürfen sich zeitlich nicht überlappen.* Im Folgenden geben wir einige Beispiele globaler Constraints an.

Das globale Constraint

```
alldifferent(Variablenliste)
```

spezifiziert, dass alle im Argument aufgeführten Variablen paarweise unterschiedliche Werte annehmen müssen. Natürlich könnte stattdessen auch die logisch äquivalente Konjunktion von paarweisen Ungleichungen verwendet werden. Der Nachteil liegt neben dem größeren Schreibaufwand in der schlechteren Constraint-Propagation, d.h. die verbleibenden Suchräume können noch erheblich größer sein.

Mit Hilfe des Element-Constraints

```
element(N, Liste, Wert)
```

lässt sich ausdrücken, dass das n-te Element einer nichtleeren Liste natürlicher Zahlen den Wert `Wert` haben muss.

Das Cumulative-Constraint

$\texttt{cumulative}([S_1,\ldots,S_n],[D_1,\ldots,D_n],[R_1,\ldots,R_n],L)$

beschreibt das folgende Ressourcen-Problem: Wenn S_i die Startzeit eines Ereignisses, D_i die Dauer und R_i die benötigte Ressourcenanzahl für dieses Ereignis ausdrücken, dann sichert dieses Constraint, dass zu keinem Zeitpunkt mehr als `L` Ressourcen gleichzeitig benötigt werden.

Die mathematische Formulierung dieser Bedingungen ist:

Für jedes $k \in [min\{S_i\}, max\{S_i + D_i\} - 1]$ gilt:

$\Sigma H_j \leq L]$ für alle j mit $S_j \leq k \leq S_j + D_j - 1$

Durch das globale Constraint

$$\texttt{diffn}([\ldots,[X_i,\ldots,L_i,\ldots],\ldots])$$

können abstrakte Bedingungen der folgenden Art direkt modelliert werden:

> *Eine Liste n-dimensionaler Rechtecke ist überlappungsfrei in einem gegebenen n-dimensionalen Rechteck zu platzieren.*

Die X_i beschreiben den Koordinatenursprung der n-dimensionalen Rechtecke und die L_i die entsprechenden Seitenlängen. Dieses Constraint sichert, dass sich die gegebenen n-dimensionalen Rechtecke nicht überlappen. Die Abbildung 1 zeigt ein Beispiel für drei zweidimensionale Rechtecke, die sich nicht überlappen. Das größte Rechteck ist bestimmt durch die zweite Liste `[1,2,3,2]`: der Ursprung ist der Punkt `(1,2)` und die Längen in den beiden Dimensionen sind `3` und `2`.

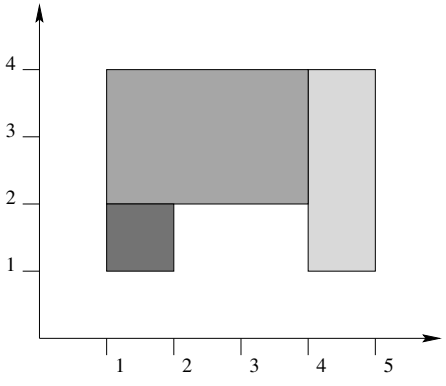


Abb. 1. `diffn([[1,1,1,1],[1,2,3,2],[4,1,1,3]])`

Für das `diffn`-Constraint existieren weitere Varianten mit weiteren Argumenten, mit deren Hilfe noch zusätzliche Bedingungen ausgedrückt werden können.

Neben problemnaher Spezifikation von Problemen, Effizienz in der Abarbeitung und Flexibilität in der Anpassung an geänderte Bedingungen sind die inkrementelle Behandlung der Constraints und dadurch eine gute Verbindung von automatischer und interaktiver Problembearbeitung weitere Vorteile der Constraint-Programmierung.

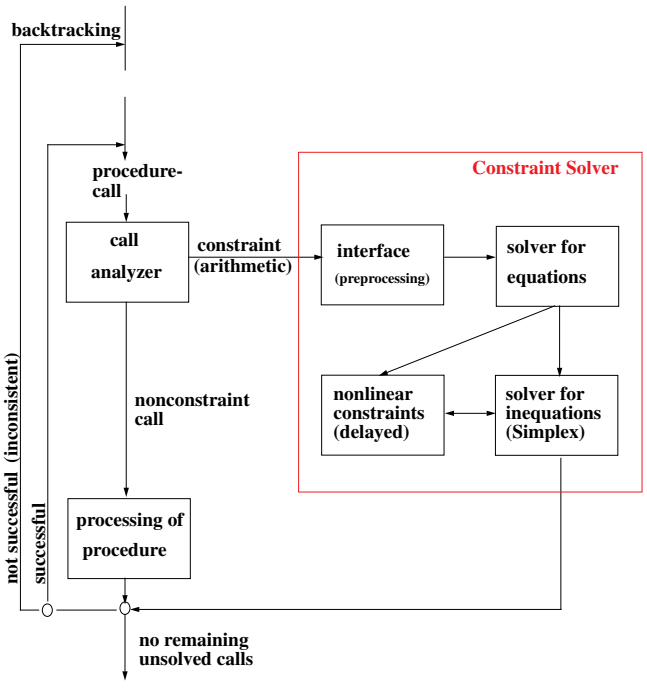


Abb. 2. System mit Constraintlöser

3.2 Constraint-Verarbeitung

Constraints dienen zur Repräsentation von relationalen Beziehungen zwischen Variablen. Durch Constraints wird ausgedrückt, welche Bedingungen zulässige Belegungen der Variablen erfüllen müssen. Eine Lösung einer Menge von Constraints ist folglich eine Belegung der in dieser Menge vorkommenden Variablen, so dass alle diese Constraints erfüllt sind.

Die Verarbeitung von Constraints erfolgt durch einen *Constraint-Löser* (s. a. Abb. 2). Seine Arbeitsweise besteht im Vereinfachen von Constraintmengen, in Konsistenzprüfungen und im Erzeugen von Konsequenzen und Lösungen. Ein Constraint-Löser grenzt dabei im allgemeinen die Menge der potentiell möglichen Lösungen ein oder er erkennt Widersprüche. Sind die Lösungen variablenfrei, handelt es sich um konkrete Lösungen. Die Lösungen können aber auch noch Variablen enthalten. Es handelt sich dann um allgemeine Lösungen. Um konkrete Lösungen zu finden, sind in diesem Fall in einem zusätzlichen Schritt noch weitere Constraints hinzuzufügen oder einem Teil der Variablen schrittweise Werte zuzuordnen.

Constraint-Löser unterscheiden sich bezüglich der Wertebereiche (Domänen) für die Variablen und bezüglich Syntax und Semantik der Ausdrücke, mit denen Constraints formuliert werden können. Für die Behandlung des Stundenplanungsproblems werden Constraint-Löser für Variablen mit endlichem Wertebereich verwendet.

Die Arbeitsweise eines Constraint-Lösers kann folgendermaßen grob skizziert werden:

- Zuerst werden den Variablen endliche Wertebereiche zugeordnet.
- Dann werden die Bedingungen (Constraints) definiert, die diese Variablen erfüllen müssen.

- Der Constraint-Löser streicht aus allen Variablendomänen Werte heraus, die nicht Bestandteil einer Lösung sein können (Propagation). Dadurch ist eine drastische Suchraumeinschränkung möglich.
- Anschließend werden nach einer gewählten Methode/Heuristik schrittweise den Variablen Werte zugeordnet (Suche). Jeder Suchschritt zieht Propagation nach sich.

Der Constraint-Löser wacht darüber, dass die Lösungen die Constraints erfüllen. Ein wichtiger Bestandteil der Arbeitsweise ist der sogenannte "*Delay-Mechanismus*", durch den die verzögerte Constraintauswertung und damit eine reihenfolgeunabhängige (deklarative) Constraintverarbeitung ermöglicht wird.

Einige Vorteile der Constraint-basierten Programmierung sind unter anderem:

- *Die Zuordnung eines zulässigen Wertes zu einer Constraintvariablen führt unmittelbar zu einer Suchraumeinschränkung. Eine unzulässige Wertzuordnung wird unmittelbar erkannt und zurückgewiesen.*
- *Durch Constraints wird festgelegt,* was *zu erfüllen ist; es wird nicht festgelegt,* wie *es zu erfüllen ist. Folglich sind Constraints deklarativ und die Modellierung von Problemen ist weitgehend unabhängig von den Lösungsmethoden, wodurch eine größere Flexibilität bei Änderungen der Problemstellung gewährleistet ist.*

4 Modellierung des Stundenplanungsproblems

Die in der Problembeschreibung aufgeführten Bedingungen werden als Constraints formuliert, d.h. durch die Angabe von relationalen Beziehungen, die die gesuchten Variablenbelegungen zu erfüllen haben. Alle notwendigen Bedingungen konnten mittels der vordefinierter Constraints modelliert werden. Um die Effizienz der Lösungssuche (hier der Propagation von Variablenbelegungen) zu verbessern, können relationale Beziehungen, die aus den notwendigen Bedingungen logisch folgen, zusätzlich angegeben werden. Die automatische Ableitung derartiger redundanter Constraints und die Bestimmung der Bedingungen unter denen diese Ableitungen zur Effizienzsteigerungen ausgeführt werden müssen, sind Bestandteil weiterer Forschungsarbeiten.

Für die unterschiedlichen Komponenten eines Stundenplanes (Pläne für Vorlesungen, Pläne für Seminare und Kurse, Pläne für Blockpraktika) sind unterschiedliche Techniken der Problemmodelierung erforderlich, da sich die Bedingungen entsprechend unterscheiden. Im folgenden werden die Ansätze zur Modellierungen einiger Bedingungen angegeben.

4.1 Seminare/Kurse

Es handelt sich um ein vorrangig kombinatorisches Problem. Für jedes Fach sind Zeiten angegeben, an denen das Seminar bzw. der Kurs durchgeführt werden kann, wobei auch eine bestimmte Anzahl parallel durchgeführt werden kann. Jeder Seminargruppe können für alle zu absolvierender Seminare bzw. Kurse Zeiten aus den gegebenen Zeiten so zugeordnet werden, dass sich diese zeitlich nicht überlappen und die gegebene Schranke für die gleichzeitige Durchführung von Lehrveranstaltungen eines Faches nicht überschritten wird. Falls die gegebenen Wunschzeiten nicht bereits einfach überprüfbare Kriterien der erforderlichen Kapazitäten verletzen, war es mit unserem System bisher immer möglich für unsere Anwendungsprobleme eine Lösung zu finden, da potenziell sehr viele Varianten der Belegung in Betracht gezogen werden.

Die Stundenplan-Bedingung "*Die Anzahl der Seminare, Untersuchungskurse und Praktika, die eine Klinik gleichzeitig durchführen kann, ist beschränkt*" kann beispielsweise mit dem Cumulative-Constraint leicht modelliert werden. Seien `S1,S2,...,Sn` die Domänenvariablen für die Startzeiten der Lehrveranstaltungen, die eine Klinik durchführen muss, und seien `D1,D2,...,Dn` die jeweilige Dauer dieser Lehrveranstaltungen, `Max` die maximale Anzahl von Lehrveranstaltungen, die diese Klinik gleichzeitig durchführen kann. Dann kann diese Bedingung durch das folgende Constraint ausgedrückt werden:

```
cumulative([S1,S2,...,Sn],
           [D1,D2,...,Dn],
           [1,1,...,1],Max)
```

4.2 Vorlesungen

Vorlesungen, die Studenten eines Semesters hören müssen, dürfen sich zeitlich nicht überlappen. Zur Modellierung der Information, dass Unterrichtsstunden bezüglich der zwei Dimensionen *Zeit, Raum* überlappungsfrei sein müssen, kann das `diffn`-Constraint verwendet werden.

Seien `S1,S2,...,Sn` die Domänenvariablen für die Startzeit, `D1,D2,...,Dn` natürliche Zahlen für die Dauer und `R1,R2,...,Rn` die Domänenvariablen für die möglichen Räume dieser Lehrveranstaltungen. Dabei werden den Domänenvariablen nur Werte zugeordnet, die auch potenziell möglich sind. So sollten beispielsweise den Raumvariablen nur die Räume zugeordnet sein, die jeweils für die entsprechende Lehrveranstaltung geeignet sind. Die konfliktfreie Zuordnung wird durch

```
diffn([[S1,D1,R1,1],
       [S2,D2,R2,1],...,
       [Sn,Dn,Rn,1]])
```

gesichert. Es werden also zweidimensionale Rechtecke in den Dimensionen *" Zeit"* und *"Raum"* betrachtet. Um eine Lösung zu finden, ist i.Allg. eine Suche im verbleibenden Lösungsraum erforderlich, wobei der Constraint-Löser nur Lösungen zulässt, die dieses `diffn`-Constraint erfüllen. Falls Lehrveranstaltungen nicht in allen Wochen eines Semesters stattfinden, sind entsprechend dreidimensionale Rechtecke mit der weiteren Dimension *"Woche"* zu betrachten.

Wenn die Wunschzeiten zweier Vorlesungen zusammenfallen, kann nur für eine davon die Wunschzeit erfüllt werden. Bei der Planung ist es vorrangig das Ziel, einen Plan zu finden, in dem möglichst viele Zeit- und Raumwünsche erfüllt werden. Es wird deshalb zunächst versucht, die Wertebereiche der Variablen nicht zu stark einzuschränken, sondern die Wunschwerte mit einer gewissen akzeptablen Abweichung (z.B. +/- 30 Minuten) zuzulassen.

4.3 Blockpraktika

Für jede Gruppe, die ein gegebenes Blockpraktikum absolvieren muss, wird eine Praktikumseinheit definiert. Allen Praktikumseinheiten sind Zeiten so zuzuordnen, dass alle Bedingungen erfüllt sind. Da die Dauer eines Praktikums vorgegeben ist, genügt es, jeweils die *Startzeiten* der Praktikumseinheiten zu bestimmen. Die Startzeit wird als Domänenvariable betrachtet. Die möglichen Ausgangswerte einer Domäne ergeben sich aus der fortlaufenden Numerierierung aller Tage, an denen Praktika stattfinden können. Die Domäne der Startzeitvariable eines Wochen-Praktikas besteht anfangs nur aus den natürlichen Zahlen, die der Menge der ersten Werktage der gegebenen Wochen entsprechen. Wenn ein Praktikum an bestimmten Tagen nicht beginnen darf, kann diese Bedingung durch Streichen der entsprechenden Elemente aus der Domäne der Startzeitvariablen gesichert werden.

Weil die Dauer der Wochen-Praktika in Abhängigkeit von der zugeordneten Startzeit durch mögliche Feiertage variieren kann, wird die Dauer von Praktikumseinheiten solcher Praktika als Domänenvariable betrachtet. Für die Modellierung der Beziehung zwischen Dauer und Startzeitvariable kann das vordefinierte Constraint `element/3` verwendet werden. Sei zum Beispiel für eine Praktikumseinheit `S` die Startzeitvariable, `D` die Domänenvariable für die Dauer und `[D1,D2,...]` die geeignet sortierte Liste (natürliche Zahlen) der verschiedenen Dauern dieser Einheiten in Abhängigkeit vom Startzeitpunkt. Dann kann durch

```
element(S,[D1,D2,...],D)
```

die Beziehung zwischen diesen beiden Domänenvariablen `S` und `D` modelliert werden.

Für die Formulierung der Bedingungen 1 und 2 für Blockpraktika kann jeweils das globale Constraint `cumulative` direkt verwendet werden. Dieses Constraint sichert, dass von einer Ressource zu jedem Zeitpunkt nicht mehr als die angegebene Anzahl benötigt wird. Falls diese Schranke für den gesamten Zeitraum nicht gleichmäßig ist, können "Dummy"-Einheiten definiert werden, die zu den entsprechenden Zeiten Kapazitäten blockieren.

5 Lösungssuche

In der Lösungssuche sind zwei Arten von Nichtdeterminismus enthalten: Auswahl einer Variablen und Auswahl der Wertebereicheinschränkung (bzw. Wertzuordnung als Spezialfall) für die ausgewählte Variable. Natürlich ist es im Allgemeinen nicht möglich, alle Auswahlmöglichkeiten der Suche zu probieren. Für die Lösungssuche sind folglich Heuristiken erforderlich, die die jeweilige Auswahl unterstützen.

In der von uns gewählten Heuristik für die Variablenauswahl ist die Priorität des zugehörigen Faches entscheidend. Die Ausgangspriorität kann in der Problemdefinition festgelegt werden. Wenn es erforderlich ist, kann diese Priorität während der Lösungssuche auch verändert werden, wobei dabei auch ein Zufallsgenerator eingesetzt wird. In den Heuristiken für die Wertebereicheinschränkung ist die Berücksichtigung der Wünsche integriert. Folglich wird im ersten Schritt versucht, einen Wertebereich so einzuschränken, dass die Wünsche enthalten sind.

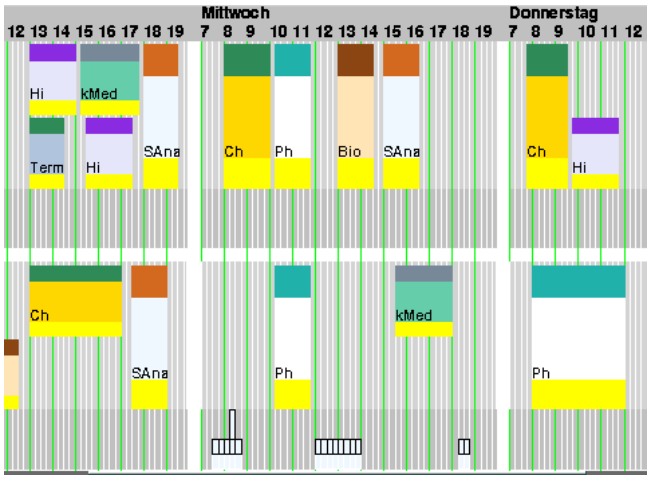


Abb. 3. Interaktive Vorlesungsplanung

In vielen Fällen unserer Tests konnte eine Lösung entweder in wenigen Suchschritten gefunden werden oder es wurde eine außerordentlich große Anzahl von Suchschritten benötigt, falls das Problem überhaupt lösbar war. Deswegen wählten wir die folgende grundsätzliche Vorgehensweise bei der Lösungssuche, um trotz einer eventuell ungünstig gewählten Heuristik in akzeptabler Zeit eine Lösung zu finden: Die Anzahl der erlaubten Suchschritte wird stark eingeschränkt und es werden verschiedene Heuristiken der Variablenauswahl bzw. der Wertebereicheinschränkung verwendet. Somit wird ein Backtracking über verschiedene Heuristiken durchgeführt.

Die Lösungssuche kann über eine grafische Repräsentation interaktiv beeinflusst werden (s.a Abb 3). Folgende Möglichkeiten der interaktiven Planerzeugung, die beliebig kombinierbar sind, wurden realisiert:

- Lehrveranstaltungen einzeln einplanen (nur widerspruchsfreie Zeiten können zugeordnet werden),
- markierte Lehrveranstaltungen automatisch einplanen,
- markierte geplante Lehrveranstaltungen ausplanen,
- alle noch nicht geplanten Lehrveranstaltungen automatisch einplanen.

Bei der interaktiven Beeinflussung der Lösungssuche werden Entscheidungen des menschlichen Planers beim Wechseln in den automatischen Verarbeitungsmodus nicht zurückgenommen. Dadurch ist es möglich, dass der Anwender bei der interaktiven Planung keinen widerspruchsfreien Plan findet, obwohl die automatische Planung eine Lösung erzeugen kann. Entscheidungen des menschlichen Planers können aber individuell zurückgenommen werden. Außerdem besteht die Möglichkeit, jeden Planungszustand abzuspeichern. Ausgehend von einer Teillösung können folglich auch leicht verschiedene Varianten der Fortsetzung der Lösungssuche probiert werden.

Bei der interaktiven Einzelplanung und bei der automatischen Planung einer Menge von markierten Veranstaltungen werden neben den einzuplanenden Einheiten sowohl die bereits geplanten als auch die noch zu planenden Lehrveranstaltungen berücksichtigt, insbondere werden auch die erforderlichen Kapazitäten der noch zu planenden Einheiten in Betracht gezogen (es erfolgt eine Vorausschau auf noch zu planende Ereignisse).

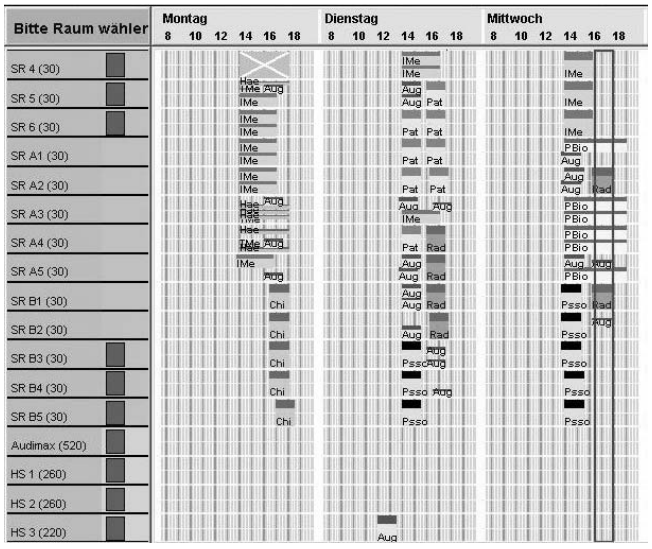


Abb. 4. Interaktive Raumplanung

Im interaktiven Modus werden beim Einplanen einer Veranstaltung grafisch die Zeiten angezeigt, die in diesem Schritt konfliktfrei ausgewählt werden können (in der Abb. 3 sind das die in der untersten Zeile aneinandergefügten Rechtecke, von denen jedes einen Viertelstundentakt kennzeichnet, an dem die Veranstaltung beginnen könnte). Das Erkennen von möglichen Konflikten wird durch diese grafische Darstellung unterstützt.

Wenn während der zeitlichen Planung keine gleichzeitige Raumzuordnung gewünscht wurde, können den Lehrveranstaltungen im Raumplanungssystem geeignete Räume zugeordnet werden. Neben der automatischen Raumplanung aller Lehrveranstaltungen bzw. einer Menge ausgewählter Lehrveranstaltungen, können auch in diesem System den Lehrveranstaltungen interaktiv Räume zugeweisen werden. Natürlich kann die Raumzuweisung nur so erfolgen, dass keine Widersprüche entstehen. Gegenüber der automatischen Raumplanung kann im interaktiven Modus aber die Bedingung über die notwendige Raumkapazität verletzt werden, was natürlich grafisch angezeigt wird.

Die Abbildung 4 zeigt einen Ausschnitt der grafischen Oberfläche während der interaktiven Raumplanung. Im rechten Teil dieses Ausschnitts ist ein langes Rechteck erkennbar, das durch die gegebenen Zeiten (Beginn und Ende) der ausgewählten Lehrveranstaltung bestimmt wird (in diesem Beispiel beginnt diese Lehrveranstaltung etwa um 16 Uhr am Mittwoch). Durch Rechtecke sind im linken Teil der Grafik die auswählbaren Räume markiert.

Im Raumplanungssystem wird i. Allg. den Lehrveranstaltungen für alle Wochen des Semesters ein einheitlicher Raum zugeordnet. Es besteht aber auch die Möglichkeit für jede Woche einen anderen Raum zu wählen. Insbesondere können dadurch natürlich leicht Abweichungen von einer einheitlichen Raumzugeordnung für bestimmte Wochen erzeugt werden. Außerdem können im Raumplanungssystem auch Lehrveranstaltungen und andere Veranstaltungen hinzugefügt werden, die bei den zeitlichen Einplanungen nicht betrachtet werden, die aber einen Raum benötigen (wie z.B. Versammlungen, Räume für Klausuren, Sonderveranstaltungen).

6 Systembeschreibung

Zum Stundenplanungssystem gehören folgende Bestandteile:

Grafischer Editor : Spezifikation eines Stundenplanungssystems,
Transformation: Transformation einer externen Problembeschreibung in eine interne Repräsentation und umgekehrt,
Grafisches Interface: Grafische Darstellung von Stundenplänen und Interaktionen des Benutzers mit dem System,
Suche: Aufbau und Verwaltung des Constraint-Netzes und heuristisch gesteuerte Lösungssuche,
HTML-Code: Erzeugung der HTML-Darstellung der Lösung.

Für die Darstellung der externen Beschreibung des Stundenplanungsproblems wurde eine Problembeschreibungssprache entwickelt. Alle Komponenten (Vorlesungen, Kurse/Seminare, Praktika) eines Planungsproblems können unter Verwendung dieser Sprache spezifiziert werden. Die externe Darstellung wird vom System bei der Eingabe der Daten eines Stundenplanungsproblems in den grafischen Editor erzeugt. Die externe Darstellung wird vom System außerdem verwendet, wenn ein Systemzustand während der Abarbeitung (zur späteren Wiederaufnahme) gesichert werden soll. In der externen Darstellung müssen für alle Definitionskomponenten nur für die Eigenschaften Werte angegeben werden, die vom Standardwert abweichen. Beispiele für einige Komponenten in externer Darstellung sind:

```
def_plan
   wochen 14,
   semester_start  12/4/2004,
   zeiten_von 8:00,
   zeiten_bis 20:00.

def_raum
   id hs1,
   max 200.

def_raum
   id hs2,
   max 175,
   nicht mo(13:00,15:30),
   nicht do(10:30,12:00).

def_dozent
   id 'Meier',
   nicht mi(8:00,12:30).

def_vorlesg
   id anatomie1,
   semester 2,
   fach 'Anatomie' ,
   studenten 110,
   wunsch di(10:15),
   raum_wunsch hs2,
   dozent 'Reiche'.
```

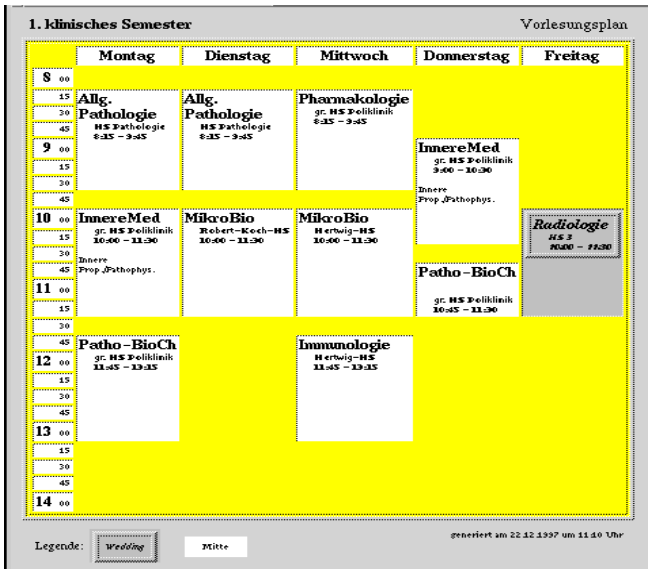


Abb. 5. Semester-Vorlesungsplan im Internet

```
def_vorlesg
   id biochemie,
   semester 1',
   fach 'Biochemie',
   dauer 150,
   wochen a,
   studenten 135,
   raum_wunsch hs1,
   dozent 'Meier'.
```

Aus der externen Darstellung wird durch die Transformationskomponente eine objektorientierte interne Darstellung erzeugt. Auf der Grundlage der internen Darstellung erfolgt die Lösungssuche.

Die Ausgabe der erzeugten Stundenpläne erfolgt als HTML-Dateien (siehe Abb. 5). Dadurch sind die Ergebnisse der Planung universell weiter verwendbar.

Alle Komponenten des Systems, einschließlich der grafischen Bestandteile, wurden in der Constraint-basierten Sprache CHIP ([7]) implementiert. Das System besteht aus ungefähr 1.6 MMytes Programmcode. Der Abarbeitungskern, die Suche, besteht aus nur 190 KBytes Programmcode.

7 Erfahrungen

Die erste Testphase konnte erfolgreich abgeschlossen werden. Die Pläne konnten schnell erzeugt werden, falls nicht Widersprüche (Verletzung harter Constraints) auftraten. Diese Widersprüche konnten aber mit Hilfe der interaktiven Planungsmöglichkeiten relativ schnell lokalisiert werden. Die Konfliktbeseitigung erfolgte dann in enger Zusammenarbeit mit der für den Stundenplan verantwortlichen Mitarbeiterin der Fakultät. Dabei zeigte sich auch, dass Hintergrundwissen sehr wichtig ist. Die erzeugten Pläne wurden als HTML-Datei ausgegeben und sind im Internet unter

```
http://www.charite.de/fakultaet/lehre/
stundenplan/stuplan.htm
```

zu finden.

Seitens der Fakultät wurde die automatisierte Stundenplanung sehr positiv beurteilt. Die Vorteile einer interaktiven, automatischen Stundenplanerzeugung konnten eindeutig nachgewiesen werden – ausgedehnte Curriculumskonferenzen und Beschwerden (der Studenten) gehören der Vergangenheit an. Gegenüber der Situation in den Jahren vor der Verwendung des Stundenplanungssystems sind die Pläne wesentlich eher verfügbar.

Die Zuordnung der Studenten zu Seminargruppen, bei der die Wünsche der Studenten möglichst berücksichtigt werden, kann nun noch zum Ende des vorhergehenden Semesters erfolgen.

Testplanungen für andere Fakultäten haben den folgenden Fragespiegel ergeben, der gleichzeitig die zur Zeit behandelbaren Phänomene beschreibt:

- Veranstaltungsbezeichnungen
- Dauer der einzelnen Veranstaltungen
- eventuell: Liste der für jede Veranstaltung potenziell passenden Räume
- für jede Veranstaltung: Name des Dozenten (oder: NN)
- für jede Veranstaltung: Wunschzeiten (Menge der Wünsche mit hoher Priorität, Menge der Wünsche mit geringerer Priorität)
- für jede Veranstaltung: Wunschräume (Menge der Wünsche)
- für jede Veranstaltung: Priorität bezüglich Raum oder Zeit bei der Planung
- für jede Veranstaltung: Teilnehmeranzahl
- Menge der verfügbaren Räume
- für jeden Raum: Platzanzahl
- eventuell für jeden Raum: Ausstattungsbesonderheiten
- eventuell: müssen Wegezeiten zu einzelnen Räumen, zu Raumgruppen oder zu Standorten berücksichtigt werden, die nicht mit der normalen Pausenwechselzeit zu schaffen sind?
- Täglicher frühester Beginn der Veranstaltungen
- Tägliches spätestes Ende der Veranstaltungen
- Bei welchen Veranstaltungen gibt es Reihenfolgenabhängigkeiten?
- Welche Veranstaltungen werden für mehrere Studiengänge gleichzeitig (im gleichen Raum) angeboten?
- Gibt es Veranstaltungen für einen Studiengang, die parallel laufen können?
- Sollen mehrere Vorlesungen des gleichen Fachs für einen Studiengang an verschiedenen Tagen stattfinden?
- Soll ein Seminar/eine Übung zu einer Vorlesung unmittelbar folgend stattfinden?

8 Ausblick

Die seit 1998 erfolgreiche Verwendung des Systems zeigt, dass die eingesetzte Methodik geeignet ist. Durch diesen Testeinsatz gewannen wir auch wertvolle Informationen für unsere weitere Arbeit. So zeigte sich beispielsweise, welche Bedeutung die interaktive Einflussnahme auf die Suche hat und welche Anforderungen an sie gestellt werden. Die Akzeptanz ließe sich weiter durch den Einsatz inzwischen von uns untersuchter und ausgebauter Verfahren der Behandlung von Constrainthierarchien mit unterschiedlich priorisierten Constraints und unterschiedlichen Fehlerfunktionen verbessern. Eine weitere Verbesserung (der Effizienz) ist durch dynamische Constraintverarbeitung möglich, bei der im Fall eines als wider-

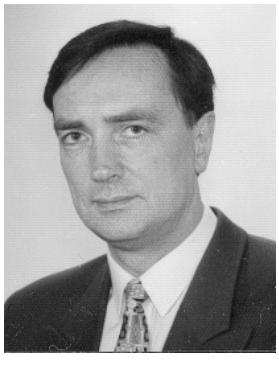
sprüchlich erkannten Constraints nicht ein Neustart der Verarbeitung mit einer relaxierten Spezifikation, sondern eine inkrementelle Relaxation erfolgt. Schließlich ist der Übergang zu einer dezentralen Verarbeitung wünschenswert. Dadurch können mehrere Fakultäten gemeinsam Ressourcen nutzen, jede ihre aber vorrangig. Voraussetzung ist die Möglichkeit des verteilten Constraintlösens, um die Konsistenz des Gesamtplans zu wahren.

Auf der Basis unseres Stundenplanungssytems lassen sich schließlich weitere Anwendungsgebiete erschließen. Naheliegend ist es, ein Planungssystem für Studenten zu entwickeln, die sich – unter Berücksichtigung ihrer persönlichen Randbedingungen – eine geeignete Seminargruppe ermitteln wollen (z.B. weil sie wegen des Nachholens von Prüfungen mehr als die im Semester erforderlichen Veranstaltungen besuchen müssen). Andere Anwendungsgebiete sind Planung für Schulen oder die Kursplanung von Weiterbildungsveranstaltungen.

Literatur

1. Abdennadher S, Marte M (1998) University timetabling using constraint handling rules. In: Ridoux O (ed) Proc JFPLC'98, Journées Francophones de Programmation Logique et Programmation par Constraintes, pp 39–49, Paris, Hermes.
2. Abdennadher S, Saft M, Will S (2000) Classroom assignment using constraint logic programming. In: Gervat C (ed) Proceedings PACLP 2000, pp 179–191. The Practical Application Company Ltd
3. Boizumault P, Delon Y, Peridy L (1996) Constraint logic programming for examination timetabling. J Logic Programming, 26(2):217–233
4. Burke E, Carter M (eds) (1998) Practice and Theory of Automated Timetabling II, volume 1408 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg
5. Burke E, Ross P (eds) (1996) Practice and Theory of Automated Timetabling, volume 1153 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg
6. Cooper TB, Kingston JH (1996) The complexity of timetable construction problems. In: [5], pp 183–295
7. Dincbas M, van Hentenryck P, Simonis H, Aggoun A, Graf T, Berthier F (1988) The constraint logic programming language CHIP. In: Int Conf Fifth Generation Computer Systems (FGCS'88), pp 693–702, Tokyo
8. Frühwirth T, Abdennadher S (1997) Constraint-Programmierung. Springer-Verlag, Berlin, Heidelberg, New York
9. Geske U, Goltz H-J, John U, Matzke D, Wolf A (1998) Constraint-basierte Planung und Simulation von Multiressourcen-Problemen. GMD Report 28, Sankt Augustin
10. Goltz H-J (2000) Combined automatic and interactive timetabling using constraint logic programming. In: Burke E, Erben W (eds) PATAT 2000, Proceedings Int Conf on the Practice and Theory of Automated Timetabling, pp 78–95
11. Goltz H-J, Matzke D (1999) University timetabling using constraint logic programming. In: Gupta G (ed) Practical Aspects of Declarative Languages, volume 1551 of Lecture Notes in Computer Science, pp 320–334, Berlin, Heidelberg, New York, Springer-Verlag
12. Henz M, Würtz J (1996) Using Oz for college timetabling. In: [5], pp 162–177
13. Lajos G (1996) Complete university modular timetabling using constraint logic programming. In: [5], pp 146–161
14. Marriott K, Stucky PJ (1998) Programming with Constraints: An Introduction. The MIT Press, Cambridge (MA), London
15. White GM, Zhang J (1998) Generating complete university timetables by combining tabu search with constraint logic. In: [4], pp 187–198

Ulrich Geske leitet den Bereich Planungstechnik und Deklarative Programmierung im Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik in Berlin und lehrt am Institut für Informatik der Universität Potsdam. Sei Arbeitsgebiet sind Logische und Constraint-Logische Programmierung sowie KI-Techniken und deren Anwendung auf kombinatorische Probleme.



Hans-Joachim Goltz studierte an der Humboldt-Universität zu Berlin Mathematik und promovierte 1982 an dieser Universxität im Fachbereich *Mathematische Logik*. Zunächst arbeitete er am Fachbereich *Mathematische Logik und Grundlagen der Mathematik* der Humboldt-Universität und ist jetzt am Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik (FIRST) im Bereich Planungstechnik und Deklarative Programmierung tätig. Seine Forschungsschwerpunkte sind Logische und Constraint-Logische Programmierung und deren Anwendung für die Modellierung und Lösung kombinatorischer Probleme.