INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67, D–80538 München

Ludwig-
Maximilians-
Universität
München

INTERDIP - Ein Interaktiver Constraint-basierter Dienstplaner für Krankenstationen

Slim Abdennadher, Hans Schlenker

http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-1997-5, September 1997

INTERDIP – Ein Interaktiver Constraint-basierter Dienstplaner für Krankenstationen

Slim Abdennadher und Hans Schlenker*
Institut für Informatik, Ludwig-Maximilians-Universität
Oettingenstraße 67, 80538 München
{Slim.Abdennadher,Hans.Schlenker}@informatik.uni-muenchen.de
{http://www.pms.informatik.uni-muenchen.de/software/}

Zusammenfassung

In einem Krankenhaus muss regelmäßig, meist monatlich, ein Dienstplan für das Pflegepersonal jeder Krankenstation erstellt werden. Dafür müssen viele Bedingungen wie Sollbesetzung einer Station, Arbeitszeitregelungen und Wünsche des Personals berücksichtigt werden. Diese Planung ist sehr aufwendig und wird heute noch meist von Hand erledigt. INTERDIP bietet eine teilautomatische Erstellung von Dienstplänen für Pflegepersonal einer Krankenstation über einen beliebigen Zeitraum. Dafür werden Teile der händischen Planung mit Constraint-Technologie nachgebildet. Mit INTERDIP kann ein Dienstplan teilautomatisch und interaktiv innerhalb weniger Minuten statt von Hand einiger Stunden erstellt werden.

1 Einführung

Constraints sind Relationen, die es erlauben, in deklarativer Weise Probleme durch Angabe von Nebenbedingungen zu formulieren, die im Vergleich zu Logikprogrammiersprachen die Suche nach geeigneten Lösungen beschleunigen [Van89, FM94]. Die Constraints werden dann nicht mittels Resolution, sondern durch spezielle Algorithmen gelöst. Die Programme, durch die diese Algorithmen implementiert werden, nennt man Constraintlöser.

Mit der Einbettung von Constraints in Logikprogrammiersprachen wurde es möglich, schnell und elegant komplexe kombinatorische Probleme durch eine Verbindung aus Constraint-Erfüllung und Suche zu lösen. Konkret ergeben sich vor allem folgende Vorteile:

- deklarative Modellierung unter Verwendung problemnaher Constraints
- die Möglichkeit zur Darstellung sowohl unvollständiger und spärlicher als auch vollständiger Information durch Constraints, die es ermöglicht, auch mit ungenauen, unsicheren und unscharfen Daten korrekt zu arbeiten
- automatische Propagierung impliziter Effekte während der Suche nach einer Lösung
- Trennung der Constraintbehandlung vom eigentlichen Suchverfahren

Seit Anfang der neunziger Jahre wird constraintbasierte Programmierung mit großem Erfolg von mehreren Firmen weltweit kommerziell eingesetzt. Anwendungsgebiete sind

*Diese Arbeit wurde im Rahmen einer Diplomarbeit von der Siemens Nixdorf Informationssysteme AG unterstützt.

Produktions- und Ressourcenplanung (insbesondere Zeit- und Kapazitätswirtschaft), Transportoptimierung, Layoutgenerierung, CAD-Systeme und Personalplanung. Dazu gehört die Generierung von Dienstplänen für Krankenstationen, die heute noch Gegenstand der Forschung ist.

Immer noch werden an den meisten Krankenhäusern die Dienstpläne von Hand erstellt. Dabei wird für jedes Mitglied einer Station und für jeden Tag des Planungszeitraums – üblicherweise ein Monat – eine oder keine Schicht zugeteilt, wofür verschiedene harte Bedingungen unbedingt erfüllt werden müssen, und weiche Bedingungen nach Möglichkeit. Eine solche Planungstätigkeit ist, speziell bei größeren Krankenstationen, sehr aufwendig: typischerweise benötigt eine Person einen ganzen Tag zur Planung eines Monats.

Unser System, INTERDIP, bildet Teile der händischen Dienstplanerzeugung nach. Damit kann ein mit der Dienstplanung vertrauter Benutzer, in einer Krankenstation die sogenannte *Stationsschwester*, interaktiv innerhalb weniger Minuten einen Plan erstellen. Wir erwarten, dass dieses System

- leicht und schnell zu einem korrekten Plan führt, da der Benutzer nicht sein komplettes Wissen über die Planung dem System mitteilen muss,
- vom Benutzer eher akzeptiert wird, als ein vollautomatisches System, da er an der unmittelbaren Planung beteiligt bleibt,
- bessere Lösungen als ein vollautomatisches System und oft sogar als die händische Generierung liefert und damit
- akzeptabel ist für das von der Planung betroffene Personal.

Ein Dienstplan wird mit INTERDIP in mehreren Phasen generiert. Ausserdem werden nach vorgegebenen Mustern jeweils mehrere Tage auf einmal belegt. Dadurch erreichen wir im Vergleich mit existierenden Verfahren eine enorme Reduzierung des Suchraums wodurch gute Ergebnisse für realistische Krankenstationen (20 Schwestern) innerhalb weniger Minuten erzeugt werden können.

Implementiert wurde INTERDIP mit IF/Prolog [Sie96b] der Siemens-Nixdorf-Informationssysteme AG, das basierend auf den Forschungsarbeiten von CHIP [DVS+88] ein Constraint-Paket enthält [Sie96a]. Dieses umfasst u.a. Lineare Gleichungen, Constraints über endlichen Wertebereichen (Finite Domänen) und Boolesche Constraints.

Dieses Papier ist wie folgt gegliedert: Abschnitt 2 beschreibt das Problem der Dienstplangenerierung und Abschnitt 3 dessen Modellierung als *Partial Constraint Satisfaction Problem*. In Abschnitt 4 sehen wir, wie Dienstpläne von Hand erzeugt werden und unseren daraus abgeleiteten inkrementellen Ansatz. Abschnitt 5 beschreibt dann die Bedienung von INTERDIP. In Abschnitt 6 vergleichen wir unseren Ansatz mit verwandten Arbeiten, Abschnitt 7 ist Zusammenfassung und Ausblick.

2 Beschreibung des Problems

Dieser Abschnitt beschreibt das Problem der Dienstplangenerierung für Krankenstationen (engl. *Nurse Scheduling Problem*, NSP).

Jeden Monat muss für jede Krankenstation ein neuer Dienstplan erstellt werden. Eine Station eines Krankenhauses ist eine organisatorische Einheit. Eine bestimmte Station hat gewisse spezialisierte Aufgaben zu übernehmen, verfügt über Räume im Krankenhaus und über Personal: die Krankenschwestern. Die Stationen eines Krankenhauses sind praktisch völlig voneinander getrennt: jede hat ihre eigenen Räume und ihr eigenes Personal. Dadurch können die Dienstpläne aller Krankenstationen getrennt voneinander erzeugt werden. Wir betrachten im folgenden *genau eine* Krankenstation.

Ein Dienstplan eines Monats ist eine Zuordnung des Personals der Krankenstation zu den Schichten der Tage dieses Monats. Eine Schicht ist eine Arbeitseinheit: jeder Tag hat die Einheiten *Frühschicht*, *Spätschicht* und *Nachtschicht* und unter Umständen weitere. Jeder Schicht jedes Tages muss Personal zugeordnet werden.

Bei der Erzeugung eines Dienstplans müssen unterschiedliche Constraints berücksichtigt werden. Man unterscheidet verschiedene Arten von Constraints:

- *Gesetze* regeln unter anderem die maximale Arbeitszeit einer Person pro Tag und pro Woche, Freizeitausgleich oder Mutterschaftsurlaub. Zum Beispiel ist die gesetzliche monatliche Regelarbeitszeit für ein Krankenhaus mit 37,5 Stunden-Woche für einen typischen September 161,7 Stunden. Bei einer durchschnittlichen Schichtlänge von 8 Stunden ergibt das im Schnitt 20 Arbeitstage für jede Schwester. Ein anderes Gesetz erzwingt für eine Schwester zwischen zwei Schichten eine Pause von 11 Stunden („11-Stunden-Regel“). Wenn eine Schwester an einem Tag in der Nachtschicht arbeitet, darf sie dadurch am nächsten Tag nicht in der Früh- oder der Spätschicht arbeiten. Genauso darf nach einer Spätschicht keine Frühschicht kommen.

- *Dienstliche Regelungen* sind solche, die für ein spezielles Krankenhaus, ein Teil des Krankenhauses oder auch nur für eine Station gelten. Sie werden von der jeweiligen Leitung festgesetzt und sind meist organisatorischer Art. Das sind vor allem Art der Schichten und – im gesetzlichen Rahmen – Mindestbelegungen einer Station. Wir benutzen im folgenden ein Modell mit drei Schichten: Frühschicht, Spätschicht und Nachtschicht. Mindestens belegt muss die Frühschicht und die Spätschicht mit drei Schwestern sein und die Nachtschicht mit zwei.

- *Personaldaten* geben den individuellen Rahmen für eine Person. Dazu zählen vor allem die vertraglich feststehende monatliche Arbeitszeit, noch ausstehender Urlaub und angefallene Überstunden. Wenn z.B. eine Schwester 16 Überstunden aus dem Vormonat mitbringt sollten ihr diesen Monat möglichst zwei Schichten weniger zugeteilt werden.

- *Wünsche* sind schließlich Anforderungen an den Dienstplan, die das Personal selbst formuliert. Meist sind das Freiwünsche, z.B. für Wochenenden oder Feiertage oder Geburtstage. Dazu gehören auch konkrete Urlaubswünsche.

Bei der Erzeugung eines Dienstplans kann man nicht immer davon ausgehen, dass alle Constraints erfüllt werden. Deswegen unterscheidet man zwei Arten von Constraints. Harte Constraints müssen immer erfüllt werden, weiche Constraints können verletzt werden. Gesetze, dienstliche Regelungen und Personaldaten sind harte Constraints, Wünsche können harte oder weiche Constraints sein. So kann die Urlaubsplanung von der eingentlichen Dienstplanung getrennt über einen längeren Zeitraum passieren. Damit wäre ein Urlaubswunsch ein harter Constraint, da nicht die Dienstplanung über dessen Erfüllung entscheidet, sondern schon vorher darüber entschieden ist. Andere Wünsche sind meist weiche Constraints. Oft aber bekommen die Schwestern die Möglichkeit, ihre Wünsche in verschiedenen Wichtigkeits-Kategorien anzugeben. Nach Möglichkeit werden bei der Planung dann die Wünsche einer dieser Kategorien als harte Constraints betrachtet.

Ein Dienstplan ist dann korrekt, wenn alle harten Constraints erfüllt sind. Die Güte eines Dienstplans ergibt sich aus der Anzahl der erfüllten weichen Constraints und ihrer Präferenzen.

3 Modellierung des Problems

Dieser Abschnitt beschreibt die Formulierung des Dienstplangenerierungsproblems als *Partial Constraint Satisfaction Problem* (PCSP). Allgemein besteht ein PCSP aus einer Menge von Variablen und einer Menge von Relationen zwischen den Variablen (Constraints). Für jede dieser Variablen existiert ein Wertebereich, aus dem ein Wert für die Variable ausgewählt werden muss. Eine Variablenbelegung, die alle *harten* Constraints erfüllt ist eine Lösung des PCSP.

Die Dienstplanung kann folgendermassen (quasi naiv) als PCSP formuliert werden: Für alle Schwestern einer Krankenstation muß für jeden Tag der Planungsperiode eine bestimmte Schicht, Urlaub, Freier Tag, Feiertagsausgleich, Arbeitszeitverkürzungstag, Sonderurlaub, unbezahlter Urlaub, Mutterschutz usw. zugeordnet werden. Entsprechend wird für jede Schwester pro Tag eine Constraint-Variable angelegt. Der Wertebereich dieser Variablen besteht nun gerade aus den möglichen Schichten, Urlaub, Freier Tag, usw.

Die Größe C des Suchraums von Lösungen für das PCSP ist die Anzahl der möglichen Belegungen der Variablen. Sei N die Menge der Schwestern und D die der Tage, dann haben wir $|N| * |D|$ Variablen[1]. Der Wertebereich jeder Variable sei W. Dann ist die Größe des Suchraums:

$$C = |W|^{|N|*|D|}$$

Typischerweise sind $|D| = 30$, $|N| = 20$ und für die drei Schichten und die oben genannten Freischichten ist $|W| = 10$. Damit ist

$$C = 10^{20*30} = 10^{600}$$

In [WH95, HW96] wird, basierend auf der Arbeit [Fre91], nachgewiesen, dass die Werte für die Freischichten, also Urlaub, Freier Tag, Feiertagsausgleich usw., auf einen reduziert werden können, wodurch sich der Wertebereich der Variablen verringert: $|W| = 4$. Damit reduziert sich der Suchraum drastisch:

$$C = 4^{20*30} \approx 10^{361}$$

Wir benutzen dieses Ergebnis und beschränken uns auch auf eine Art von Freischichten.

Abbildung 1 zeigt einen typischen Dienstplan: eine Tabelle mit einer Zeile für jede Schwester der Station und einer Spalte für jeden Tag des Planungszeitraums. Jede Zelle der Tabelle enthält maximal eine Zuordnung zu einer Schicht. Ist am Tag d der Schwester n der Wert $s_{n,d} \in W := \{0, 1, 2, 3\}$ zugeordnet, dann heißt das:

- $s_{n,d} = 0$: Schwester n hat am Tag d keinen Dienst („Freischicht“)
- $s_{n,d} = 1$: Schwester n wird eine Frühschicht am Tag d zugeordnet.
- $s_{n,d} = 2$: Schwester n wird eine Spätschicht am Tag d zugeordnet.
- $s_{n,d} = 3$: Schwester n wird eine Nachtschicht am Tag d zugeordnet.

Wir werden im nächsten Abschnitt sehen, wie unser mehrphasiges Verfahren die Komplexität des Problems weiter verringert, indem die Variablen boolesche Wertebereiche bekommen.

[1] Wenn M eine Menge ist, dann ist $|M|$ die Kardinalität der Menge.

Datei Wünsche Bereiche Muster Optionen Debugger Phasen

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
KarinG	0	0	1	1	1	2	3	0	3	3	0	0	2	2
Hilde	0	1	1	3	3	3	0	0	1	1	1	3	3	0
Gerda	3	3	3	0	0	0	1	1	2	3	3	0	0	2
Gerd	0	1	1	3	3	3	0	0	1	1	1	3	3	0
Hubert	0	1	2	2	2	2	0	0	1	1	1	1	1	0
Anna	0	2	2	2	2	2	0	0	0	2	2	2	2	3
Norbert	1	3	3	0	0	1	2	2	2	2	0	0	2	3
EddaA	2	2	2	0	0	1	1	1	2	2	0	1	0	1
EddaB	2	2	0	2	2	0	2	3	3	0	0	1	1	1
Ina	1	0	0	1	1	1	2	2	0	0	2	2	2	2

Legende: '0' Frei(schicht), '1' Frühschicht, '2' Spätschicht, '3' Nachtschicht

Abbildung 1: Beispiel eines Dienstplanes für 10 Schwestern und 14 Tage

4 Inkrementelle Lösung

Dieser Abschnitt beschreibt zuerst die heute übliche Dienstplanung von Hand und dann unsere darauf aufbauende teilautomatische Dienstplangenerierung: die inkrementelle Erzeugung in mehreren Phasen, die Belegung in Mustern und die Suche nach optimalen Lösungen.

4.1 Dienstplanerzeugung von Hand

Aufgrund des immens großen Suchraums wird ein Dienstplan üblicherweise von Hand in zwei Phasen erzeugt. In der ersten Phase hat man für die Belegung der Zellen noch alle Freiheiten. Deshalb wird da die „komplizierteste“ Belegung (die an die meisten Bedingungen geknüpft ist) erledigt: die Belegung der freien Tage oder der „Freischichten“. Diese sind an viele Constraints gebunden: sie bestimmen, wieviele Tage eine Schwester im Belegungszeitraum zu arbeiten hat, wieviele Schwestern insgesamt der Station jeden Tag zur Verfügung stehen und nicht zuletzt sind dabei die meisten Wünsche der Schwestern an den Dienstplan zu berücksichtigen: die Freiwünsche.

Eng mit den Freischichten verknüpft sind die Nachtschichten. Die 11-Stunden-Regel erzwingt ja für die Belegung der Schichten einer Schwester, dass am Tag nach einer Nachtschicht nur entweder eine Nachtschicht oder aber eine Freischicht kommen darf[2].

Bei der händischen Dienstplanerzeugung werden die Frei- und die Nachtschichten in einer Phase, der ersten, erledigt. In der zweiten Phase werden dann Früh- und Spätschichten auf die noch nicht zugeordneten Zellen des Dienstplans verteilt.

Der offensichtliche Vorteil der Erzeugung in zwei Phasen gegenüber der in einer Phase ist die Reduktion der Komplexität: pro Phase müssen weniger Constraints und vor allem

[2] Die 11-Stunden-Regel gilt zwar auch für die anderen Schichten, doch knüpft sie nur die Nachtschichten (außer an die Nachtschichten selbst) an die Freischichten. Auf alle anderen Schichten dürfen auch Nicht-Freischichten folgen.

weniger mögliche Belegungen berücksichtigt werden. In der ersten Phase haben wir für jede Zelle drei Möglichkeiten: Freischicht, Nachtschicht oder *unbestimmt*. In der zweiten werden die noch unbestimmten Zellen mit entweder Frühschicht oder Spätschicht belegt[3]. Die Größe des Suchraums ist also

$$C = 3^{600} + 2^{600} \approx 3^{600} \approx 10^{286}$$

4.2 Phasenweise Generierung

Schon 1993 wurde an der Universität Leuven für dieses Problem ein automatischer Planer entwickelt [vdB93], der zwei sehr unterschiedliche Phasen verwendet, und damit flexibel gute Dienstpläne erzeugt, leider aber keine Planung der Nachtschichten zulässt. INTERDIP benutzt mehr als zwei Phasen, die vom gleichen Constraintlöser in der gleichen Weise durchgeführt werden. Durch dieses mehrphasige Verfahren reichen boolesche Wertebereiche für die Belegung der Variablen.

Die Idee ist, die Anzahl der Phasen soweit zu erhöhen, dass in jeder Phase für jede Zelle nur die minimale Entscheidung zwischen zwei Möglichkeiten bleibt und trotzdem nach allen Phasen ein vollständiger Dienstplan generiert ist. Dazu müssen wir nur die erste Phase der händischen Dienstplanung in zwei teilen. Wir erhalten drei Phasen:

1. Phase Verteilung der Freischichten auf die Zellen des Dienstplans.

2. Phase Verteilung der Nachtschichten auf die noch nicht belegten Zellen.

3. Phase Verteilung der Früh- und der Spätschichten auf die restlichen Zellen.

In jeder Phase wird jede Variable an einen Wert aus dem booleschen Wertebereich $\{0, 1\}$ gebunden. Ein Dienstplan ergibt sich dann aus allen drei Phasen. Je nach Phase haben die Werte Null und Eins verschiedene Bedeutung. Wenn einer Variable in der ersten Phase der Wert Null zugeordnet wird, tragen wir in die entsprechende Zelle des Dienstplans eine Freischicht ein. Die Zellen, deren Variablen an die Eins gebunden sind, bleiben unentschieden. Die zweite Phase behandelt ausschließlich die unentschiedenen Zellen: erhält eine Variable den Wert Eins, wird in die Zelle eine Nachtschicht eingetragen, der Rest bleibt weiterhin unentschieden. In der dritten Phase werden die noch unentschiedenen Zellen vollständig mit Frühschicht oder Spätschicht belegt, je nachdem, ob die entsprechende Variable an Eins bzw. Null gebunden ist (siehe Abbildung 2).

Dieses Verfahren reduziert den Suchraum auf

$$C = 2^{600} + 2^{600} + 2^{600} \approx 10^{181}$$

4.3 Belegung in Mustern

Wenn die gegebenen Constraints nicht stark genug sind, um eine eindeutige Lösung zu erhalten, ist die Verwendung von Suchverfahren notwendig. Dabei wechseln Constraintbehandlung und Suche einander ab. Zur Steuerung der Suche wendet man ein Enumerationsverfahren (engl. labeling) an, das die Variablen an ihre möglichen Werte bindet.

[3] Die Belegung der ersten Phase wird in der zweiten nicht mehr verändert, es sei denn, es findet sich in der zweiten Phase keine Lösung und eine Veränderung der Nacht- und Freischichten ermöglicht eine Lösung. Diese Veränderungen sind vom Umfang her aber gegenüber dem Umfang des Suchraums in jeder der beiden Phasen zu vernachlässigen: bei den konkret beobachteten händischen Dientplanungen gab es jeweils maximal etwa 10 nachträgliche Änderungen. Im übrigen sollen die Zahlenangaben eher die Dimension der Suchräume andeuten, als genau die Anzahl der Möglichkeiten angeben. Insofern können wir die Phasen als unabhängig voneinander betrachten.

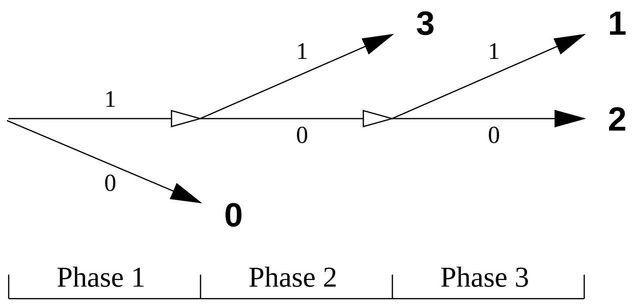

Abbildung 2: Belegung der Zellen in drei Phasen

Das hier eingesetzte Enumerationsverfahren verwendet phasenspezifische *Muster*, die vom Benutzer definiert werden. Wie wir oben gesehen haben, werden die Variablen in jeder Phase mit 0 oder 1 belegt und entsprechend Schichten in den Dienstplan eingetragen. Muster sind dann *sinnvolle* Kombinationen von *Dienstplan*-Einträgen, wobei eine Kombination für aufeinanderfolgende Tage steht. Zum Beispiel halten wir folgende Kombination für sinnvoll: fünf Tage arbeiten, zwei Tage frei. Das entsprechende Muster für die erste Phase, in der die Arbeitstage ermittelt werden, ist dann: (unbestimmt,unbestimmt,unbestimmt,unbestimmt,unbestimmt,0,0). Unter der Annahme, dass es besser ist, an drei aufeinanderfolgenden Tagen in derselben Schicht zu arbeiten, als in unterschiedlichen, formulieren wir für die zweite Phase und damit für die Nachtschichten: (3,3,3). Jede Phase hat einen eigenen Satz von Mustern. Die Muster einer Phase haben eine Reihenfolge, nach der sie ausgewählt werden: zuerst solche, die ein *gutes* Ergebnis unterstützen – hier sollen im obigen Sinne sinnvolle Kombinationen stehen – und am Schluss triviale einelementige, die notwendig sind um Lösungen zu generieren, wenn sie existieren. Die Variablen und damit die Zellen des Dienstplans werden nur nach den vorgegebenen Mustern belegt.

Indem mehrere Variable auf einmal belegt werden, wird der Suchraum weiter eingeschränkt. Werden im Schnitt drei Variable auf einmal belegt, und das war in unseren Tests tatsächlich der Fall, hat der Suchraum in jeder Phase die Größe $2^{600/3}$ und insgesamt:

$$C = 3 * 2^{200} \approx 10^{61}$$

Trotz der genannten Verbesserungen ist der Suchraum immer noch zu groß. Diese theoretische Komplexität wird in der Praxis weiter verringert, da bei konkreten Problemen viele Constraints berücksichtigt werden, die den Suchraum bei Verwendung eines Constraintlösers unmittelbar einschränken. Bei einer typischen Anzahl von Constraints generiert und optimiert INTERDIP eine Lösung im allgemeinen binnen einiger Minuten, was bei den oben genannten größeren Suchräumen seltener möglich ist.

4.4 Optimale Dienstpläne

Was ist ein optimaler Dienstplan? Wie wir oben gesehen haben, beeinflusst die Auswahl der Muster die Güte eines Dienstplanes. Da im allgemeinen nicht alle Constraints erfüllt werden können, hängt die Güte des Dienstplanes auch von der Anzahl der erfüllten weichen Constraints und ihrer Präferenzen ab.

Der Constraintlöser von IF/Prolog kann von Hause aus keine weichen Constraints behandeln. Jede Variablenbelegung erfüllt alle Constraints, die an ihn gestellt werden. Deshalb formulieren wir nur die harten Constraints als IF/Prolog-Constraints und weisen jedem weichen Constraint des Problems fiktive *Kosten* zu, die *entstehen*, wenn das Constraint verletzt wird. Kosten entstehen dabei für jede Schwester getrennt.

In jeder Phase wird zuerst eine Variablenbelegung für alle Zellen generiert. Diese Lösung L wird zu einer Lösung L' optimiert, indem alle Variablenbindungen rückgängig gemacht werden und dieselbe Enumeration, die zu L geführt hat, erneut versucht wird, mit dem Unterschied, dass alle Kosten von L' kleiner sein müssen als das Maximum der Kosten von L (*minimize-maximum-branch-and-bound*). INTERDIP optimiert dann L' genauso weiter zu L'' und so fort. Dieser Prozess und damit diese Phase ist zu Ende, wenn entweder eine vom Benutzer vorgegebene Zeitschranke erreicht ist oder der Benutzer selbst die Berechnung abbricht oder eine optimale Lösung gefunden ist. Eine Lösung ist optimal, wenn sie nicht mehr verbessert werden kann, also kein L' gefunden werden kann, deren Kosten alle unter dem Maximum von L liegen.

Kosten entstehen also getrennt für jede Schwester und das Maximum dieser Kosten wird minimiert. INTERDIP versucht zu erreichen, dass keine Schwester besonders schlecht wegkommt und damit zum Beispiel die unerfüllten Wünsche gleichmässig über alle Schwestern verteilt wird. Das führt zu einem Dienstplan, der allgemein für gerecht gehalten wird.

INTERDIP besitzt damit zwei Techniken, optimale Lösungen zu erzeugen:

- Art und Reihenfolge der Muster für die Enumeration
- Verletzungen weicher Constraints werden minimiert

5 Bedienung des Systems / Interaktivität

Dieser Abschnitt beschreibt die Bedienung des Systems und damit dessen interaktive Möglichkeiten.

Die Bedienung von INTERDIP erfolgt über eine graphische Benutzerschnittstelle. Diese wurde in IF/Prolog und dessen Tcl/Tk-Interface [Ous94] implementiert. Abbildung 1 zeigt das zentrale Fenster der Anwendung mit Menüleiste und berechnetem Dienstplan.

Es lassen sich alle Parameter wie Größe des Dienstplans (Anzahl der Schwestern und der Tage des Planungszeitraums), Mindest- und Höchstbelegung der Schichten, Wünsche oder Muster grafisch oder in Tabellen erfassen.

Abbildung 3 zeigt als Beispiel die Eingabe der Belegungsgrenzen für die Phase zwei: die Nachtschichten. Die Zahlen links in jeder Zeile stehen für die Anzahl der Schwestern, denen in Phase eins für einen bestimmten Tag keine Freischicht zugeordnet wurde, die also an diesem Tag arbeiten. Sind das fünf, so soll in Phase zwei genau eine Schwester an diesem Tag in die Nachtschicht eingeteilt werden, bei sechs und sieben mindestens eine und höchstens zwei und bei acht genau zwei. Diese Grenzen werden übrigens wie harte Constraints behandelt.

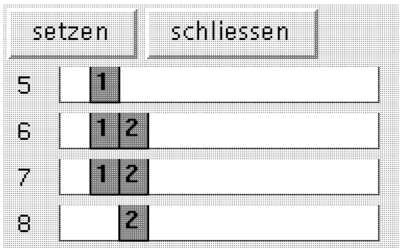

Abbildung 3: Belegungsgrenzen für die zweite Phase (Nachtschichten)

Die Wünsche werden in drei Kategorien eingegeben: *rot* für unbedingt zu erfüllende (z.B. Urlaub), *schwarz* für wichtige und *weiss* für weniger wichtige Wünsche. Die weissen Wünsche sind gewissermassen die *Standardwünsche*, die jeder hat: z.B. am Wochenende nicht zu arbeiten. Rote Wünsche werden später als harte Constraints eingetragen, die anderen wie weiche Constraints behandelt. Ein Wunsch bezieht sich immer auf genau eine Schwester und genau einen Tag.

Es kann leider passieren, dass es keinen Dienstplan gibt, der alle harten Constraints erfüllt. Man nennt das Problem dann auch *over-constrained.* INTERDIP erkennt das zum Teil schon beim Erzeugen der IF-Prolog-Constraints und gibt dann dem Benutzer Hinweise auf die Constraints, die zu der Inkonsistenz geführt haben. Es gibt aber komplexe Widersprüche, die nicht automatisch erkannt werden. INTERDIP besitzt unter anderem dafür einen Debugger (s. unten).

Üblicherweise läuft die Erzeugung eines Dienstplans wie folgt ab. Nachdem der Benutzer die Rahmenbedingungen eingegeben hat, wird er die Phasen anstoßen. Eine Phase beginnt damit, dass INTERDIP die Constraints einträgt. Dann wird nach obigem Verfahren eine optimale Lösung berechnet. Nach Beendigung der Berechnung wird die beste gefundene Lösung gespeichert und alle Variablenbindungen werden rückgängig gemacht. Dadurch muss nicht für jede Phase ein kompletter Satz von Variablen erzeugt werden, was den Speicherbedarf reduziert und unter Umständen damit die Berechnungsgeschwindigkeit erhöht. Ist eine Phase beendet, wird die nächste angestoßen, die sich auf die besten Lösungen der vorhergehenden Phasen stützt. Das ist die automatische Generierung.

Als interaktives Tool ermöglicht INTERDIP dem Benutzer verschiedenartigen Einfluss auf die Dienstplangenerierung. Zum einen sind da natürlich die Rahmenbedingungen, bei deren Formulierung jedes konkrete Planungsproblem üblicherweise einige Freiheiten lässt. Unmittelbar kann der Benutzer zum Beispiel durch Angabe roter Wünsche konkrete Belegungen erzwingen. Aber auch in den Prozess der Berechnung kann er eingreifen: mit dem Debugger kann man die Berechnung manuell anhalten oder Zellen des Dienstplans als Breakpoints definieren, vor deren Belegung dann automatisch angehalten wird. Am Haltepunkt werden alle dort anwendbaren Muster zur Auswahl angeboten. Die Berechnung geht dann mit dem ausgewählten weiter. Im Einzelschrittmodus wird nach jeder Belegung eines Musters angehalten. Außerdem kann der Benutzer bereits erfolgte Belegungen rückgängig machen und zwar in einzelnen Schritten oder gezielt bis zu einer Zelle des Dienstplans.

Damit kann der Benutzer von Hand einzelne Stellen des Dienstplans generieren um einerseits automatisch angebotene Lösungen zu verbessern und andererseits, wenn der Generator keine Lösung für das konkrete Planungsproblem findet, eine solche ermöglichen.

6 Verwandte Arbeiten

Unserer Arbeit verwandt sind vor allem solche, die das Dienstplanproblem für Krankenstationen unter Verwendung der Constraint-Technik behandeln. Alle beschriebenen Artikel legen, bis auf dort genannte Ausnahmen, der Dienstplanung im wesentlichen dieselben Rahmenbedingungen zugrunde, wie wir. Das unterstützt im übrigen die Annahme, dass es sich dabei um einen allgemeingültigen Rahmen handelt.

Das System SEPHOS [LA96] geht von einer anderen Organisation der Krankenstation als in unserem Fall aus. Dort können Schwestern offenbar keine Wünsche zur Dienstplanung äussern. Nur der Urlaub wird individuell berücksichtigt. Das unterscheidet den Rahmen doch wesentlich von unserem und vereinfacht das Problem wohl auch. Ansonsten benutzt SEPHOS die Constraint-Bibliothek ILOG und ist eine Impementieung in C. Anders als unser System kann SEPHOS die Anzahl der erforderlichen Schwestern minimieren.

HOROPLAN [DFM+94] implementiert einen interaktiven Dienstplaner als Teil eines mehrschrittigen Planungsverfahrens. Die Arbeit gibt Vorschläge, wie für die Schwestern angenehme Schichtfolgen erzeugt werden können. Die Modellierung scheint aber spezifischer zu sein, als unsere.

Diese beiden Arbeiten waren für uns wenig hilfreich. Die Beschreibung der dort entwickelten Verfahren ist zum einen zu knapp, um im Einzelnen nachvollzogen werden zu können. Zum anderen stand uns keine Testversion zur Verfügung, mit der wir hätten überprüfen können, wie mit den Verfahren gearbeitet werden kann, was für unseren interaktiven Ansatz ja sehr interessant gewesen wäre, und wie „gut“ die daraus resultierenden Dienstpläne sind, nach unserem Ermessen.

Die Arbeit [HW96] schlägt einen Dienstplaner vor, der einen Plan für typischerweise zwei Wochen generiert. Die Modellierung des Planungsproblems als CSP ist ähnlich der in Abschnitt 3 angedeuteten naiven Modellierung. Die Autoren benutzen (wie bei SEPHOS) den ILOG-Constraint-Solver, allerdings als Lisp-Bibliothek.

Die für die Entstehung von INTERDIP wichtigste, unter anderem weil am besten dokumentierte Arbeit, ist [vdB93]. Der dort vorgeschlagene Dienstplaner generiert Pläne in zwei sehr unterschiedlichen Phasen, was nicht zuletzt uns zu der Formulierung des mehrphasigen Planers angeregt hat. Ein wesentlicher Unterschied des dort verfolgten Ansatzes zu unserer Arbeit ist die Benutzung von *work-patterns*: jeder Schwester ist eins von vier möglichen *work-patterns* zugeordnet. Ein Work-Pattern hat die Länge von ein oder zwei Wochen. Die Belegung des Plans geschieht dann streng anhand dieser Muster, wobei zunächst der komplette Zeitraum mit Mustern aufgefüllt wird und anschliessend diese Muster um ein bis zwei Tage nach vorne oder hinten verschoben werden, um Belegungsgrenzen zu erfüllen. Insofern ist dieser Ansatz weniger flexibel als der von INTERDIP. Im übrigen läßt das Verfahren von [vdB93] keine Planung der Nachtschichten zu, diese müssen vor der Planung feststehen.

Für zukünftige Verbesserungen von INTERDIP interessant ist auch das Projekt CONPLAN [Con]. Die Arbeiten hierzu [aHT95, aH96] beschäftigen sich mit weichen Constraints und deren Repräsentation durch „Preference Orderings“. In diesem Rahmen wird ein eigener Constraintlöser implementiert und damit ein Dienstplaner für Krankenstationen entworfen. Im Rahmen unserer Arbeit konnten diese Ergebnisse nicht berücksichtigt werden, da wir den vorliegenden Constraintlöser nicht verändern wollten.

7 Zusammenfassung und Ausblick

Mit der Constraint-Technologie können Zeitplanungsprobleme effizient gelöst werden. INTERDIP ist ein interaktiver Dienstplaner für Krankenstationen, der mit der deklarativen Constraint-Sprache IF/Prolog implementiert wurde. Mit diesem Prototyp können tatsächlich Dienspläne erzeugt werden. Damit hat der Benutzer ein Werkzeug, mit dem er zum einen unter Anwendung der gegebenen Möglichkeiten fast alle Rahmenbedingungen der Planung berücksichtigen kann und zum anderen durch geschickte Ausnutzung der interaktiven Möglichkeiten gute Pläne bekommt. Insgesamt haben wir damit also einen pragmatischen Ansatz verfolgt, der nicht sämtliche Funktionen, die Dienstplanung betreffen, vollständig automatisch erfüllt.

INTERDIP ahmt die händische Dienstplangenerierung nach und reduziert damit den Suchraum erheblich. Um optimale Dienstpläne zu generieren, benutzt das System die weichen Constraints und die Muster der Enumeration. Die Muster sind ein Knackpunkt des Verfahrens. INTERDIP braucht für jede Phase Muster, die eine gültige Belegung in dieser und allen folgenden Phasen ermöglichen und gleichzeitig zu einer *guten* Gesamtlösung beitragen. Tatsächlich konnten wir der händischen Dienstplanung einige sinnvolle Muster abschauen und haben heute eine Sammlung, mit der fast immer automatisch eine Lösung gefunden wird und wenn eine gefunden wird, diese auch brauchbar ist. Wir haben das für eine tatsächlich existierende Krankenstation (des münchener Krankenhauses *Rechts der Isar*) anhand konkreter Wunschpläne getestet. Für 20 Schwestern und 30 Tage konnte INTERDIP innerhalb eines Zeitlimits von 60 Sekunden pro Phase, insgesamt also nach drei Minuten, eine brauchbare bis gute Lösung generieren. Stellenweise sind solche Lösungen besser als von Hand generierte.

Unsere Erwartungen wurden insofern erfüllt:

- korrekte und gute Dienstpläne können in kurzer Zeit generiert werden
- das System scheint den potentiellen Benutzern, den Stationsschwestern, akzeptabel

Um die Qualität der von INTERDIP erzeugten Dienstpläne weiter zu verbessern, sind folgende Erweiterungen geplant:

- Die Möglichkeit von kombinierten Wünschen wie „Schwester Anna möchte nicht mit Schwester Berta in einer Schicht arbeiten" oder Constraints wie „Jeden Montag muss mindestens eine der Schwestern Carla, Dora und Eva in der Frühschicht arbeiten".
- Verwendung bzw. Implementierung eines Constraintlösers, der weiche Constraints behandeln kann. Zum einen vereinfacht das die Modellierung konkreter weicher Bedingungen. Vor allem können dann aber moderne Techniken der Constraint-Behandlung verwendet werden, wie sie z.B. in [aH96] für das Dienstplanproblem vorgeschlagen wurden.

Danksagung Wir danken Norbert Trapp, Klaus Däßler, Peter Richter, Norbert Eisinger und François Bry und vor allem Karin Grossmann für ihre freundliche Unterstützung.

Literatur

[aH96] Harald Meyer auf'm Hofe. Representation of Requirements Through Preference Orderings of Soft Constraints. Technical report, DFKI Kaiserslautern, Feb. 1996.

[aHT95] Harald Meyer auf'm Hofe und Bidjan Tschaitschian. PCSPs with Hierarchical Constraint Orderings in Real World Scheduling Applications. In *Notes on the CP'95 Workshop on Over-Constrained Systems*, Seiten 69–76, Cassis, France, 1995.

[Con] ConPlan. http://www.dfki.uni-kl.de/conplan/Welcome.html.

[DFM+94] S. J. Darmoni, A. Fajner, N. Mahe, A. Leforestier, M. Vondracek, O. Stelian und M. Baldenweck. HOROPLAN: Computer-Assisted Nurse Scheduling Using Constraint-Based Programming. *Journal of the Society for Health Systems*, 1994.

[DVS+88] Mehmet Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf und F. Berthier. The Constraint Logic Programming Language CHIP. Technical Report TR-LP-37, ECRC, Munich, Germany, Mai 1988.

[FM94] E. C. Freuder und A. K. Mackworth. *Constraint-Based Reasoning*. MIT Press, Cambridge, Massachusetts, 1994.

[Fre91] Eugene C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *AAAI-91 – Proceedings of the 9th national conference on artificial intelligence*, Seiten 227–233, 1991.

[HW96] Kamel Heus und Georges Weil. Constraint Programming A Nurse Scheduling Application. In *PACT 96 – Proceedings of the Second International Conference on the Practical Application of Constraint Technology*, Seiten 115–127. The Practical Application Company, 1996.

[LA96] J. M. Lazaro und P. Aristondo. Job Rostering with Constraints. In *PACT 96 – Proceedings of the Second International Conference on the Practical Application of Constraint Technology*, Seiten 155–168. The Practical Application Company, 1996.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison Wesley, 1994.

[Sie96a] Siemens Nixdorf Informationssysteme AG. *IF/Prolog Constraint Problem Solver*, 1996.

[Sie96b] Siemens Nixdorf Informationssysteme AG. *IF/Prolog Users Guide*, 1996.

[Van89] Pascal Van Hentenryck. *Constraint Satisfaction in Logic Programming*. Logic Programming Series. MIT Press, Cambridge, MA, 1989.

[vdB93] Bart van den Bosch. Implementation of a CLP library and an application in nurse scheduling. Diplomarbeit, Katholieke Universiteit Leuven, Belgium, 1993.

[WH95] Georges Weil und Kamel Heus. Eliminating Interchangeable Values in the Nurse Scheduling Problem Formulated as a Constraint Satisfaction Problem. In *CONSTRAINT'95*, 1995.