UNIVERSITÄT-GESAMTHOCHSCHULE PADERBORN

PuK '98
12. Workshop „Planen und Konfigurieren"
20./21. April 1998

Sauer, J. / Stein, B. (Hrsg.)

tr-ri-98-193

FACHBEREICH MATHEMATIK-INFORMATIK

Inhaltsverzeichnis

Kooperative Anlagenplanung durch integrierte Modellbeschreibung*

Olaf Wolter, Ulf Böger
Institut für Förder- und Baumaschinentechnik, Stahlbau, Logistik (IFSL)
Otto-von-Guericke-Universität Magdeburg
PSF 4120 - D-39016 Magdeburg
Email: {olaf.wolter|ulf.boeger}@mb.uni-magdeburg.de

Kurzfassung

Die hohen Erwartungen bzgl. einer rechnerunterstützten Gestaltung durchgängig kooperativer Modellentwicklungen materialflußtechnischer Anlagen konnten bisher nur z. T. eingelöst werden. Integrationsfähige Lösungen erfordern eine realitätsnahe, ganzheitliche Sicht auf den Planungsgegenstand. Integrierte Beschreibungsformen können hier eine effiziente und konsistente Modellabbildung anwendungsspezifischer Planungsdaten ermöglichen.
Im Rahmen des Beitrages wird anhand der Domäne „Planung von Materialflußanlagen“ vorgestellt, wie werkzeugabhängige Planungsdaten für z. B. CAD, Prozeßsimulation und -animation zur kooperativen Modellierung durch eine integrierte Modellbeschreibung verknüpft werden. Eine derartige gemeinsame Modellbeschreibung erleichtert eine Daten- und Ablaufintegration über den Konfigurierungsprozeß der Materialflußplanung.

1 Einführung

Die Planung von Materialflußanlagen ist eine Problemstellung der industriellen Praxis, in der kooperative Modellierungsprozesse den Anlagenentwurf vorantreiben. Der Planungsprozeß umschließt u. a. Aufgaben der Anlagenprojektierung, Prozeß- und Bewegungssimulation, Prozeßanimation sowie des Steuerungsentwurfs zur Entwicklung des Materialflußsystems und des Materialflußprozesses. Das Planungsproblem besteht darin, eine Materialflußanlage zu spezifizieren, die eine gestellte materialflußtechnische Aufgabenstellung erfüllt (z. B. eine Stückgutsortierung von Paketen mit bestimmtem Paketdurchsatz und Sortierkriterien – vgl. Abbildung 1). Die Vorgaben der Planung sind dabei neben der technischen Spezifikation des Materialflußgutes, der Systemlast und Problemstellung auch Zielvorgaben (z. B. gewünschte Ausbringung), die erfüllt werden müssen, und sonstige Randbedingungen (insbesondere Restriktionen – z. B. Gebäudevorgaben), die nicht verletzt werden dürfen.

Der Problemlösungsprozeß bei der Planung von Materialflußanlagen wird als evolutionärer, mehrdeutiger, sehr komplexer und iterativer Prozeß mit Stufen- und Schleifen-, Varianten- und Versionenentwicklung und als Fortschreiten vom Ganzen zum Detail charakterisiert. Im Mittelpunkt des Prozesses steht die Modellierung verschiedener Modellwelten (Funktionsmodell, Funktionsträgermodell und Systemgestaltmodell). Das Materialflußsystem (Abbildung 1) und der Materialflußprozeß werden zeitparallel geplant, da ein wechselseitiger Einfluß besteht. Dabei müssen Aufgaben des technischen Systems und Prozesses erfüllt werden und die Modelle bzw. Teilmodelle überführbar und kombinierbar sein. Bei der Entwicklung von Lösungen wechseln sich analysierende Arbeitsschritte mit schöpferischen, synthetisierenden Arbeitsschritten ab, die eine geeignete Kooperation erfordern.

* Teile der in diesem Beitrag dargestellten Arbeiten werden vom Land Sachsen-Anhalt im Rahmen der Vorhaben 1969A/025 und 1957A/025 gefördert.

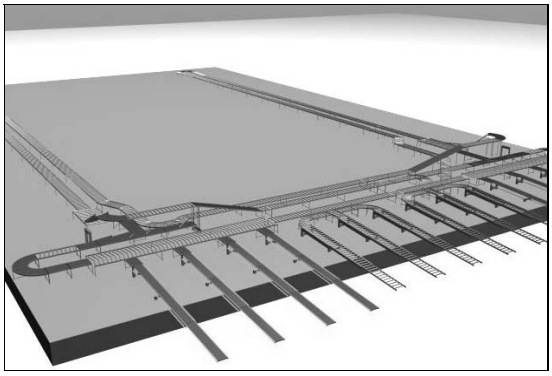


Abbildung 1: Übersichtsmodell einer Stückgutsortieranlage für einen Paketdienst

Die Stufen des Planungsprozesses bilden sich in aufeinander aufbauende Modelle von unterschiedlicher Granularität ab, die eine Assoziation mit der Aufbaustruktur des Systems (Stückliste) aufweisen. Das Modell (die Materialflußanlage) wird im vorliegenden Kontext als System fördertechnischer Ausrüstungen (Modellelemente) beschrieben. Untergeordnete Baugruppen und Einzelteile werden über Merkmalsbeschreibungen dem Modellelement zugeordnet.

2 Problembereich

Gegenwärtig werden in der Anlagenplanung unterschiedliche Werkzeuge eingesetzt, die die Layoutentwicklung, Dokumentationserstellung, CAD-Konstruktion und CAD-Projektierung sowie Prozeßsimulation und -animation unterstützen. Trotz teilweise integrativer Techniken fehlt die Möglichkeit der integrierten und kooperativen Nutzung wesentlicher Werkzeuge im Planungsprozeß. Aktuelle Forschungsarbeiten beziehen sich schwerpunktmäßig auf integrierte, proprietäre Konstruktionssysteme (vgl. z. B. [Fel89, Wei91, Rud91, Sch96]), die nur den konstruktiven Teil der Planung oder besser den Entwurf von Baugruppen und Einzelteilen unterstützen. Vielversprechendere Integrationsansätze finden sich im Umfeld der Entwicklung offener Systeme (integrierte Ingenieursysteme, Frameworks) (vgl. z. B. [CD91, RS92, Göe93, Abe95, GHS96, JKQS96, PSK96, ADJP97, BBK97, ERW97, RM97]) und anwendungsübergreifender Produktmodelle (vgl. z. B. [Pät91, GAP93, KCR+94, MS94, Gen95, Rad95]). Diese Ansätze beziehen sich jedoch ebenfalls vorrangig auf das Anwendungsfeld Konstruktion und lassen somit Aspekte des technischen Prozesses und speziell des Zusammenhangs technisches System und Prozeß unberücksichtigt, wie sie in der Anlagenplanung zu finden sind. Darüber hinaus basieren die meisten Integrationsansätze auf dem a-priori-Ansatz, bei dem neue Umgebungen aus speziell für die Integration entwickelten Systemen zusammengefügt werden. Gegenwärtig existiert in den Unternehmen eine andere Werkzeugbasis, die sich bereits mehr oder minder bewährt hat, so daß geeignete a-posteriori-Ansätze gesucht sind. Weitere Arbeiten beschäftigen sich mit der Integration unterschiedlicher Aspekte der Planungsproblematik und –methoden betreffs der Auslegung und Optimierung von Produktionssystemen mit Abgrenzung auf Fertigungs- und Montageanlagenplanung sowie Kombination von CAD (Projektierung auch in einfachen Funktionsmodellen – Layoutplanung), Simulation und Aktionsplanung (Herstellungsprozeß) (vgl. z. B. [HH90, Lan93, Had95, GKNO96, KVK96]).
Schwachstellen der obigen Ansätze lassen sich, wie auch in der Charakteristik von Bullinger [BMG93] zur Fertigungskonstruktion ausgeführt, vor allem in Bezug auf eine fehlende planungsphasenbezogene Differenzierung und Hierarchisierung der Informationen, einer unzureichenden Versionen-, Varianten- und Historienverwaltung und einer fehlenden erkenntniszielorientierten Vorgehensweise finden.
Ein entscheidendes Anwenderkriterium für zukünftige Integrationsätze von Ingenieursystemen bildet die Berücksichtigung, daß der Planungsablauf nur selten im voraus in Form einer festen Kopplung von Planungsmethoden und Modellwelten definiert werden kann. Die Wahl der Methoden hängt vielmehr von der erreichten Planungsphase, dem Planungsstand, den verfügbaren Daten, dem Informationsumfang, der -tiefe und des -inhalts [BMG93] sowie vom beabsichtigten nächsten Problemlösungsschritt und dem damit verbundenen Erkenntnisziel ab. Dies bedeutet auch, daß die Modellnutzung in Abhängigkeit vom notwendigen Kontext des Werkzeuges erfolgt. Hierzu müssen die notwendigen Modellaspekte spezifiziert und zur Verfügung gestellt werden. Zum Ausdruck kommt dies darin, daß Planungswerkzeuge unterschiedliche Schwer-

punkte auf die Modelle setzen und dabei ausgewählte Partialmodelldaten nutzen.

3 Kooperative Modellentwicklung

3.1 Einführung

Konventionelle Vorgehensweisen, wie das methodische Entwickeln von Lösungsprinzipien (vgl. [VDI96]), müssen mit den zu nutzenden informationstechnisch basierten Planungswerkzeugen verbunden und aufeinander abgestimmt werden, um das Planungsgeschehen logisch konsistent zu halten. Das betrifft sowohl die Funktionalität der einzelnen Planungswerkzeuge als auch den Umfang diskreter, lokal durchzuführender Planungsschritte. Sinnvoll erscheint es, für bestimmte Planungsaufgaben anwendungsspezifische Modellsichten zu definieren, die effiziente Kontrollroutinen zur Konsistenzsicherung und Fortschrittskontrolle zulassen. Partialmodelle wie das Aufbau-, das Ablauf- oder das Kinematikmodell, mit denen die prinzipielle Lösung der Planungsaufgabe beschrieben wird, eignen sich hierfür besonders gut [RW97].

3.2 Planungsszenarium

Als typisches Planungsszenarium wird die schrittweise Entwicklung funktionsfähiger, virtueller Prototypen einer Materialflußanlage (Planungsvarianten), z. B. für Sortieranlagen, mit ausgewählten Werkzeugen zur Modellierung und Analyse mechanischer Baugruppen, Komponenten und Ausrüstungen (Gestalt, Kinematik) und der darauf ablaufenden logistischen Prozesse (Auftragseinlastung, Steuerung, Material- und Informationsflüsse) zugrunde gelegt (Abbildung 2). Der entstehende virtuelle Prototyp der materialflußtechnischen Anlage soll das Materialflußsystem (Gestaltmodell und Kinematikmodell) sowie den Materialflußprozeß (Steuerungsmodell und Flußmodell) repräsentieren und die Aufbaustruktur des Systems (Stückliste) liefern.

Hierzu werden heterogene Planungsumgebungen aus dem Bereich CAD, Simulation und Animation (z. B. VRML-Prototypen-Bibliothek) rechentechnisch für ein kooperatives Modellieren durch eine integrierte Modellbeschreibung gekoppelt.

Mit Hilfe des CAD-Werkzeuges wird die geometrische Gestalt des technischen Systems in statischer Sicht entwickelt und in Form von geometrischen Beschreibungen der Bausteine (Formen, Konturen, Koppelpunkte, Bewegungsbahnen, ...) abgelegt [ZRW96]. Die im Anfangsstadium der Layoutplanung zu definierenden Systemlinien für die Auswahl, Dimensionierung und Positionierung von Fördermitteln sind planungsmethodisch kinematischen Ketten zur Beschreibung der Gutbewegung gleichzusetzen. Der weitere Aufbau des Gestaltmodells orientiert sich an diesen Bewegungslinien. Für die Steuerungsspezifikation übernimmt der Layoutentwurf die Positionierung von steuerungstechnischen Elementen auf den Bewegungslinien. Die Positionierung von Relativlagen des Gutes (oder des Fördermittels) auf der Förderstrecke erlaubt eine Visualisierung von einzuhaltenden Freiräumen im CAD-Layout. Die technischen Parameter des Kinematikmodells dienen direkt der Auswahl von technischen Baugruppen und Ausrüstungen.

Simulations- und Animationssysteme helfen, funktionale und dynamische Sichten auf die Struktur und Parameter technischer Systeme zu definieren und zu beurteilen (Abbildung 2).

Mit der geometrisch definierten kinematischen Kette werden nach Festlegung der bewegungscharakteristischen Kennwerte mit Kinematikwerkzeugen Bewegungssimulationen durchgeführt, die Geschwindigeits- und Beschleunigungsanalysen, aber auch Untersuchungen zur Geometrie (Kollision) und Belastung (Verformung) einzelner Körper (Glieder) der kinematischen Kette ermöglichen. Die Ergebnispräsentation konzentriert sich in Kinematikmodulen auf die Visualisierung von Bewegungsparametern in der fördertechnischen Baugruppe bzw. auf das Zusammenwirken von Fördermitteln im materialflußtechnischen Prozeß. Die Verwendung materialflußkinematisch sinnvoller Gelenkdefinitionen erlaubt derartige Analysen auf beliebigen Abstraktionsebenen. Die im Layoutentwurf positionierten Steuerungselemente werden zusammen mit den Steuerungsstrategien u. a. zur Bewegungssimulation zeitlich versetzter Bewegungsabläufe von Gütern genutzt. Dazu notwendige

prozeßcharakteristische Kennwerte werden im Dialog eingegeben, aber auch der Steuerungslogik oder aus Ereignislisten entnommen.

Animationssysteme, die auf eine möglichst naturgetreue Präsentation von Prozeßabläufen orientiert waren, integrieren zunehmend physikalisch-technische Gesetzmäßigkeiten in ihre Modellbausteine zur realistischen Darstellung bausteinbezogener, lokaler Bewegungsabläufe.

Die Visualisierung von kinematisch realistischen Prozeßabläufen auf einem 3D-Gestaltmodell wird sich zum Normalfall in der Präsentation des technischen Angebotes entwickeln (siehe hierzu [ZLR97, WRH97]).

Für die Bestimmung der Verweilzeiten von Gütern auf oder in Fördermitteln werden kinematische Ersatzmodelle verwendet, die auf Geometriedaten und technische Parameter der jeweiligen Fördermittelklasse zugreifen. Aus kinematischer Sicht bestimmt die Prozeßablaufsteuerung die notwendigen Zwangsbedingungen für die Erzeugung einer definierten Gutbewegung auf einem bestimmten Abstraktionsniveau. Die Materialflußsimulation erzeugt mit der Abbildung dynamischer materialflußtechnischer Prozesse zeitabhängig stellenbezogene Relativlagen materialflußtechnischer Objekte, die dem Animationssystem als Ereignisliste zur Visualisierung des Prozesses zur Verfügung gestellt werden.

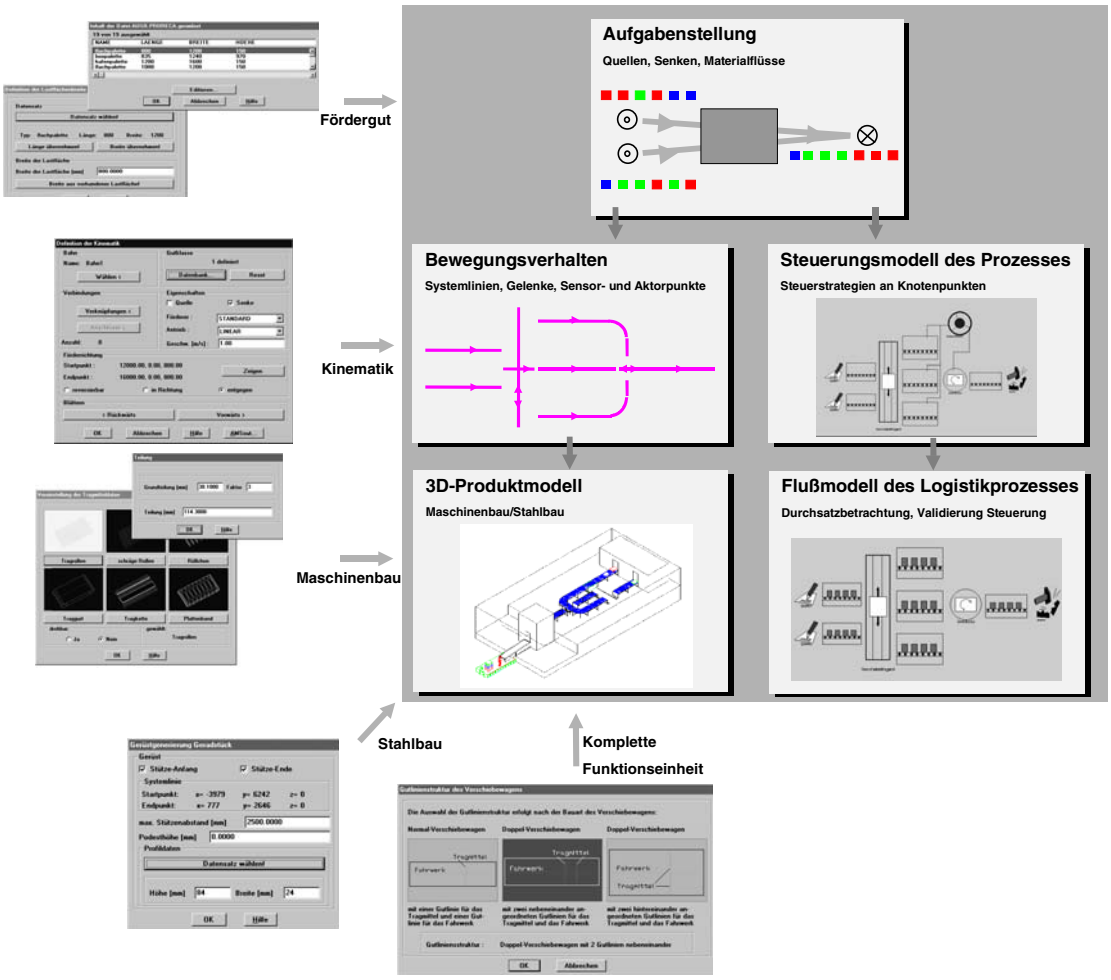


Abbildung 2: Planungsphasen von Materialflußanlagen mit Focus auf die CAD-Projektierung

3.3 Kommunikation der Werkzeuge

Die Kommunikation der Werkzeuge erfolgt durch eine gemeinsame Nutzung von Datenbereichen einer Datenbank und den Austausch von Daten über diesen Datenbereich. Im gemeinsamen Datenbereich werden werkzeugübergreifende Daten (globa-

le Daten) abgelegt und mittels werkzeugorientierter Sichten recherchiert. Durch Mechanismen des Checkin und Checkout werden die Modelle in die Datenbank übertragen bzw. aus der Datenbank gelesen. Zur strukturierten Beschreibung der globalen Daten wurde ein anwendungsorientiertes Datenmodell entwicklet, das durch eine **A**bstrakte **M**odell**S**truktur (AMS) charakterisiert ist und die Schnittmenge gemeinsam genutzter Daten der Werkzeuge definiert (Abbildung 3).

AMS fokussiert auf die effiziente Verknüpfung von steuerungstechnischen, kinematischen und Gestaltinformationen zum effizienten Austausch von Modellen zwischen heterogenen Werkzeugen. Vergleiche mit STEP-10303/105 haben die Konformität in der Beschreibung der Datenstruktur zum Kinematikmodell bestätigt.

Hierdurch wird ein integratives Wechselspiel von Anforderungsbearbeitung, Produktstrukturierung sowie Funktions- und Lösungsfindung gewährleistet, das für einen effizienten und angepaßten Planungsprozeß ein unerläßliches Hilfsmittel darstellt.

Die spezifischen Vorteile der Integration über ein anwendungsorientiertes Datenmodell sind aus informationstechnischer Sicht

- Verbesserung des Zusammenspiels von Teillösungen unterschiedlicher Art (Material, Prozeß, Information, Organisation u. a.),
- Verbesserung der Datenkonsistenz und Reduzierung der Datenredundanz,
- höhere Transparenz der Modelle durch realitätsnahe, einheitliche Weltbeschreibungen (Meta-Modell) und Nutzung von Partialmodellen,
- Sicherung änderbarer und anwendungsgerechter Modelle nach einheitlichem Konzept,
- wahlfreie Zwecksichten auf Modelle und Integration der Planungsdatenbestände,
- Modifizierungen an Modellbeschreibungen im dafür geeigneten Planungswerkzeug und
- werkzeugübergreifende Visualisierung der Modellbeziehungen und Verarbeitungsresultate.

Die Planungswerkzeuge bearbeiten die im Laufe des Planungsprozesses entwickelten Modelle, indem sie auf die globalen Daten und auf die im anwendungsspezifischen Bereich lokal abgelegten Daten zugreifen. In den Werkzeugen existieren die im kooperativen Entwicklungsprozeß zu entwickelnden und zu untersuchenden anwendungsorientierten Modelle lokal und nebeneinander. Es liegt eine funktionale und datenorientierte Unabhängigkeit der Werkzeuge vor, so daß dort ständig strukturell vollständige Planungsmodelle vorhanden sind, die eine konsistente Weiterentwicklung der Lösung ermöglichen.

In Abhängigkeit vom Verarbeitungskontext werden die relevanten Daten, die durch Schemata beschrieben sind, aus dem globalen Datenbereich für den Entwicklungsprozeß im Planungswerkzeug bereitgestellt. Die Beziehungen zu den relevanten Datenstrukturen sind den Werkzeugen durch die Projektion der werkzeugorientierten Sichten auf das AMS bekannt und erlauben so eine effiziente Nutzung und Konsistenzsicherung (vgl. Abbildung 3).

Die Darstellung des Planungs- bzw. Lösungsprozesses erfolgt durch eine modellbasierte Elaboration der abstrakten Modelle des gemeinsamen Datenbereiches, die den Problemlösungsprozeß dokumentiert. Ausgehend von einem initialen Modell (Problembeschreibung) werden Modelltransformationen durch die Verwendung der Planungsmethoden erzielt. Die Elaboration bildet ein Netzwerk von Modellen und charakterisiert die unterschiedlichen Beziehungen zwischen Modellen auf verschiedenen Entwicklungs- und Abstraktionsebenen.

Vorzüge des kooperativen Ansatzes bilden neben der weiterhin bestehenden funktionalen und datenorientierten Unabhängigkeit der Werkzeuge (Kapselung), die Bereitstellung problemadäquater Methoden zur Datenmanipulation, die Austauschbarkeit von Datenbanken und die Offenheit des Ansatzes, der eine Integration weiterer Werkzeuge gestattet.

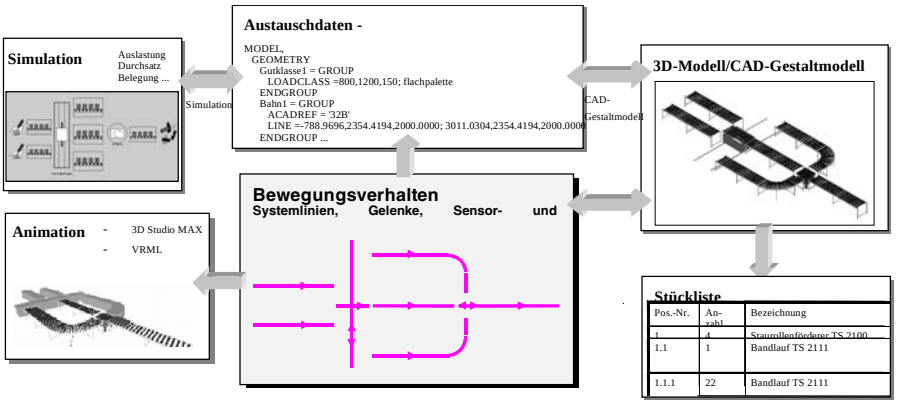


Abbildung 3: Modelldatenaustausch im kooperativen Entwicklungsprozeß

4 Zusammenfassung und Ausblick

Mit dem vorgestellten Konzept werden Möglichkeiten des Zusammenspiels zwischen CAD-Projektierung, Materialflußsimulation und -animation für den Einsatz bei mittelständischen Systemanbietern der Branche Materialflußtechnik untersucht. Dazu muß die Funktionalität der Planungswerkzeuge teilweise erheblich erweitert werden, um planungsmethodische Zusammenhänge abbilden zu können. Die realisierten Softwaretools für die CAD-Projektierung haben gezeigt, daß planungsspezifische CAE-Applikationen die Funktionalität heutiger CAD-Systeme erheblich verbessern und auch frühe Phasen des Planungsprozesses in hoher Qualität unterstützen können. Im Bereich Logistik der Otto-von-Guericke-Universität Magdeburg wird derzeit an der Weiterentwicklung von Methoden und Techniken zur ganzheitlichen Planung von Materialflußsystemen in der Angebotsprojektierung gearbeitet.

Erste Ansätze zur Integration heterogener Werkzeuge wie der CAD-Projektierung, Bewegungsanalyse und der Funktionsmodellierung werden gegenwärtig erarbeitet. Zukünftig sollen weitere planungsunterstützende Werkzeuge (z. B. Materialflußsimulation, VRML-Animation) und Office-Werkzeuge (z. B. Word, Excel) eingebunden werden. Weitere Arbeiten beschäftigen sich mit der Konzeption und Einbindung einer den Planungsvorgehensprozeß unterstützenden Komponente. Diese umfaßt Funktionen, die die Spezifikation von Planungsprozessen sowie die Führung der Benutzer im Vorgehensprozeß assistiert, wie es beipsielsweise bereits in Ansätzen des Designflow [NM97] zu finden ist. Dabei sind assistierende Problemlösungsmethoden nach [GKNO96] vorstellbar; Interpretation expliziten Problemlösungswissens, Elimination „fehlerhafter“ Planungsalternativen durch Prüfung von Konsistenzkriterien, Zuordnung spezifischer Bearbeitungsmethoden durch Zustandsanalyse der Lösung u. a.

Darüber hinaus werden Möglichkeiten des kooperativen Arbeitens im Anlagenentwicklungsprozeß, basierend auf Kommunikationsnetzwerken wie dem Internet erarbeitet.

5 Literatur

[Abe95] O. Abeln: CAD Referenzmodell. Stuttgart [u. a.]: Teubner, 1995.

[ADJP97] R. Anderl, B. Daum, H. John, C. Pütter: Architektur einer Konstruktionsumgebung für das rechnerunterstützte und kooperative Entwickeln umweltgerechter Produkte. In: Beiträge des Workshop Arbeitsplatzrechner-Integration zur Prozeßverbesserung im Rahmen der 27. Jahrestagung der Gesellschaft für Informatik, 23. September 1997 in Aachen. Softwaretechnik-Trends, 17(3):5-8, 1997.

[BBK97] K. Bender, K. Bindbeutel, A. Karcher: Integration von Rechnerwerkzeugen der Produktentwicklung mit Rahmensystemen. In: Beiträge des Workshop Arbeitsplatzrechner-Integration zur Prozeßverbesserung im Rahmen der 27. Jahrestagung der Gesellschaft für Informatik, 23. September 1997 in Aachen. Softwaretechnik-Trends, 17(3):9-12, 1997.

[BMG93] H.-J. Bullinger, F. Marcial, A. Gräble: Fertigungsgerecht konstruieren. In: ZwF Zeitung für wirtschaftlichen Fabrikbetrieb, 88(5):215-217, 1993.

[CD91] J.S. Colton, J.L. Dascanio: An Integrated, Intelligent Design Environment. In: Engineering with Computers, (7):11-22, 1997.

[ERW97] W. Eversheim, P. Ritz, M. Walz: Integration von Anwendungssystemen in einer Engineering-Infrastruktur. In: Beiträge des Workshop Arbeitsplatzrechnerintegration zur Prozeßverbesserung im Rahmen der 27. Jahrestagung der Gesellschaft für Informatik, 23. September 1997 in Aachen. Softwaretechnik-Trends, 17(3):35-38, 1997.

[ERW97] W. Eversheim, P. Ritz, M. Walz: Integration von Anwendungssystemen in einer Engineering-Infrastruktur. In: Beiträge des Workshop ArbeitsplatzrechnerIntegration zur Prozeßverbesserung im Rahmen der 27. Jahrestagung der Gesellschaft für Informatik, 23. September 1997 in Aachen. Softwaretechnik-Trends, 17(3):35-38, 1997.

[Fel89] J. Feldhusen: Systemkonzept zur durchgängigen und flexiblen Rechnerunterstützung in der Konstruktion. Berlin, Technische Universität, Dissertation, 1989.

[GAP93] H. Grabowski, R. Anderl, A. Polly: Integriertes Produktmodell. Berlin: Beuth, 1993.

[Gen95] M. Genderka: Objektorientierte Methode zur Entwicklung von Produktmodellen als Basis integrierter Ingenieursysteme. Aachen: Shaker, 1995.

[GHS96] J. Gausemeier, A. Hahn, W. Schneider: Kooperatives Modellieren auf Basis transienter Objekte. In: D. Ruhland, Hrsg., Verteilte und intelligente CAD-Systeme: Tagungsband CAD '96; Kaiserslautern, 7./8. März 1996. Bonn: Ges. für Informatik; Kaiserslautern: Dt. Forschungszentrum für Künstliche Intelligenz, 1996, S. 311-325.

[GKNO96] P. Ganghoff, A. Köhne, G. Näger, U. Osmers: KNOSPE Ein unterstützendes Planungssystem für die integrierte Montagesystemplanung. In: Informatik, Forsch Entw., 11(1):37-43, 1996.

[Göe93] J. Göers: Ein neues Konzept zur integrierten CIM-Informationsverwaltung und seine Realisierung am Beispiel eines CADund Objektbanksystems. Aachen: Shaker, 1993.

[Had95] S. Hader: Einsatz von Simulation beim Konfigurieren. In: A. Günter, Hrsg., Wissensbasiertes Konfigurieren: Ergebnisse aus dem Projekt PROKON. Sankt Augustin: Infix, 1995, S. 229-235.

[HH90] V. Horn, J. Hein: Komplexe Produktionssysteme planen. In: ZwF Zeitung für wirtschaftlichen Fabrikbetrieb, 85(6):300-304, 1990.

[JKQS96] U. Jasnoch, H. Kress, R. Quester, A. Stork: Eine plattformübergreifende Umgebung zur kooperativen Produktentwicklung. In: D. Ruhland, Hrsg., Verteilte und intelligente CAD-Systeme: Tagungsband CAD '96; Kaiserslautern, 7./8. März 1996. Bonn: Ges. für Informatik; Kaiserslautern: Dt. Forschungszentrum für Künstliche Intelligenz, 1996, S. 384-399.

[Joe97] G. Joeris: Characterization of Integrated Process and Product Management. In: Beiträge des Workshop Arbeitsplatzrechner-Integration zur Prozeßverbesserung im Rahmen der 27. Jahrestagung der Gesellschaft für Informatik, 23. September 1997 in Aachen. Softwaretechnik-Trends, 17(3):17-20, 1997.

[KCR+94] F.-L. Krause, M. Ciesla, E. Rieger, A. Ulbrich, M. Stephan: Features als semantische Objekte integrierter Prozeßketten. In: J. Gausemeier, Hrsg., Produktdatenmodellierung und Prozeßmodellierung als Grundlage neuer CAD-Systeme: GI-Fachtagung CAD '94, Paderborn, 17./18.3.1994. München [u. a.]: Hanser, 1994, S. 125-148.

[KVK96] J. Klussmann, M. Vöge, J. Krauth: Neutrales Produktdatenmodell zur Einbindung der Simulation in betriebliche Abläufe. In: Wt: Produktion und Management, 86(6):333-336, 1996.

[Lan93] V. Lange: Entwerfen von Fertigungsanlagen mit Modellund Erfahrungsunterstützung. Fortschr.-Ber. VDI Reihe 2 Nr. 302. Düsseldorf: VDI-Verl., 1993.

[MS94] J. Mohrmann, H.-J. Speck: Das Produktmodell als Integrationsplattform für Prozeßketten. In: J. Gausemeier, Hrsg., Produktdatenmodellierung und Prozeßmodellierung als Grundlage neuer CAD-Systeme: GI-Fachtagung CAD'94, Paderborn, 17. /18.3.1994. München [u. a.]: Hanser, 1994, S. 93-110.

[Pät91] B. Pätzold: Integration rechnerunterstützter Verfahren für die Konstruktion auf der Basis eines objektorientierten Produktmodellansatzes. Fortschr.-Ber. VDI Reihe 20 Nr. 53. Düsseldorf: VDI-Verl., 1991.

[PSK96] G. Paul, K.-U. Sattler, F. Kreutzmann: Eine Integragtionsarchitektur für Ingenieursysteme im CAD/CAM-Bereich. In: CAD-CAM-Report, 3(März):130-138, 1996.

[Rad95] M. Radtke: Konzept zur Gestaltung prozeß und integrationsgerechter Produktmodelle. Kaiserslautern, Universität, Dissertation, 1995.

[RM97] N. Ritter, B. Mitschang: Die Assistenzfunktion kooperativer Designflows. In: Informatik, Forsch Entw., 12(2):91-100, 1997.

[RS92] F. J. Rammig, B. Steinmüller: Frameworks und Entwurfsumgebungen. In: Informatik Spektrum, 15(1):33-43, 1992.

[Rud91] S. Rude: Rechnerunterstützte Gestaltfindung auf der Basis eines integrierten Produktmodells. Fortschr.-Ber. VDI Reihe 20 Nr. 52. Düsseldorf: VDI-Verl., 1991.

[Sch96] G. Scholz: Heterogene konzeptuelle Modelle und Interoperabilität im Elektronischen CAD. Fortschr.-Ber. VDI Reihe 20 Nr. 198. Düsseldorf: VDI-Verl., 1996.

[WRH97] Wolter, O.; Richter, K.; Höpner, C.: Prozeßfähige Virtual-Reality-Modelle der Materialflußtechnik. Fortschritte in der Materialflußtechnik: 11. Symposium Simulationstechnik (ASIM '97), Dortmund, 11.-14. November 1997. Kuhn, A.; Wenzel, S. (Hrsg.). Braunschweig [u.a.]: Vieweg, 1997, S. 563-568.

[ZRW96] Ziems, D.; Richter, K.; Wolter, O.: CAD-Modelle mit Planungs-Knowhow zur Konfigurierung von Materialflußsystemen. In: Verteilte und intelligente CAD-Systeme: Tagungsband CAD ‘96; Kaiserslautern, 7./8. März 1996. Hrsg. von Detlev Ruhland. Bonn: Ges. für Informatik; Kaiserslautern: Dt. Forschungszentrum für Künstliche Intelligenz, 1996, S. 135-149.

[ZLR1997] D. Ziems, P. Lehmann, K. Richter: Gestaltung funktionsfähiger virtueller Prototypen von Paketsortieranlagen. In: Informationsverarbeitung in der Konstruktion '97: Neue Generation von CAD/CAM-Systemen: erfüllte und enttäuschte Erwartungen; Tagung München, 28./29. Oktober 1997, (VDI-Berichte; 1357), VDI-Gesellschaft Entwicklung, Konstruktion, Vertrieb. Düsseldorf: VDI-Verl., 1997, S. 459-477.

[VDI96] VDI-Richtlinie 2222: Methodisches Entwickeln von Lösungsprinzipien. Entwurf Blatt 1, Ausgabe 7/96, 1996.

[Wei91] D. Weigel: Entwicklung einer modularen Systemarchitektur für die rechnerintegrierte Produktgestaltung. Braunschweig, Technische Universität, Dissertation, 1991.

Konfiguration auf Basis unsicherer Informationen im Produktentstehungsprozeß

Walter Eversheim, Heinrich Unbescheiden

Laboratorium für Werkzeugmaschinen und Betriebslehre

Lehrstuhl für Produktionssystematik

SFB 361 "Modelle und Methoden zur integrierten Produkt- und Prozeßgestaltung"

der Deutschen Forschungsgemeinschaft (DFG),

RWTH Aachen

Abstract: Aufgrund der Globalisierung der Märkte, des schnellen technologischen Wandels sowie veränderter Kundenanforderungen kommt der Produktinnovation eine entscheidende Bedeutung zu. Die Randbedingungen der Produktentstehung sind komplex, dynamisch und unsicher, wodurch die Entwicklung erheblich erschwert wird. Im Rahmen der Produktinnovation nimmt die Konstruktion eine herausragende Stellung ein, da hier Kosten, Qualität und Entwicklungszeit entscheidend bestimmt werden. Für erfolgreiche Produktentwicklungen ist es daher erforderlich, Konstrukteure mit effizienten Hilfsmitteln auszustatten, um die Lösung der Bearbeitungsaufgabe zu erleichtern.

Ein Ansatz ist die Unterstützung von Konstrukteuren mit Konfigurationswerkzeugen, die die Berücksichtigung von Restriktionen und die Sicherung der Konsistenz gefundener Lösungen ermöglichen. Diese Werkzeuge müssen für die speziellen Randbedingungen in der Produktentstehung geeignet sein. Hierfür wurde der Ansatz der Fuzzy Constraint Netzwerke entwickelt. Diese wurden zur Klärung und Präzisierung der Aufgabenstellung genutzt. Es wurde gezeigt, daß durch Konfiguration in den frühen Phasen der Produktinnovation unsichere Informationen handhabbar sowie die Komplexität und Dynamik in Entwicklungsprojekten reduziert werden können. Dadurch entsteht mehr Transparenz in der Produktinnovation.

1. Ausgangssituation

Die heutigen Anforderungen an die Unternehmen sind von der Nachfrage nach individuellen, qualitativ hochwertigen und preisgünstigen Produkten geprägt, die ständig dem Wandel der technologischen Entwicklung angepaßt werden müssen [Saretz, 1993]. Daher nimmt die Produktentwicklung wesentlichen Einfluß auf die Wettbewerbsfähigkeit von Unternehmen.

Entwicklungsprojekte gelten als komplex, unsicher und dynamisch [Clausing, 1994; Eversheim, Laufenberg, 1995; Zanger, Baier, Bierschenk, 1994]. Komplexität kann in die Komplexität von Produkten (z. B. durch Mechatronik), von Prozessen (z. B. durch integrierte Technologien) und von Projektstrukturen (z. B. durch steigenden Parallelisierungsgrad) unterteilt werden. Sie äußert sich im Entstehungsprozeß durch den Aufwand, der zur Einhaltung der Abhängigkeiten und Restriktionen erforderlich ist [Roggatz, 1997].

Unsicherheit ist in der Produktentwicklung durch den erforderlichen hohen Innovationsgrad und durch die weitreichenden Folgen von Entscheidungen in frühen Phasen bedingt. Um innovative Produkte auf den Markt zu bringen, müssen oftmals Lösungsansätze verfolgt werden, über die nur wenige Erfahrungen gesammelt wurden und mit denen Entwickler nicht vertraut sind. Die Entscheidungen über diese Lösungsansätze werden in frühen Phasen gefällt, während die Auswirkungen der Entscheidungen erst in späteren Phasen auftreten. Die Folge sind zahlreiche Änderungen, die zum einen die Kosten der Produktentwicklung in die Höhe treiben, zum anderen die Entwicklungszeit entscheidend verlängern [Roggatz, 1997].

Aufgrund der beschriebenen Probleme in der Produktentwicklung erwächst die Notwendigkeit, effiziente Hilfsmittel zur Bewältigung der Aufgaben zur Verfügung zu stellen. In diesem Beitrag wird die Möglichkeit der Unterstützung von Konstrukteuren mit einem Konfigurationshilfsmittel beschrieben, mit dem Komplexität und Unsicherheit handhabbar werden. Hierdurch wird die Transparenz in Produktentstehungsprozessen erhöht, und zeit- und kostenintensive Änderungen in der Produktentwicklung werden vermieden.

2. Konfiguration in Produktinnovation

Zur optimalen Unterstützung des Entwicklungsprozesses ist ein Einsatz von Konfigurationswerkzeugen in allen Phasen der Produktentstehung sinnvoll. Konfigurationswerkzeuge erleichtern die Berücksichtigung von Wirkzusammenhängen und Restriktionen und ermöglichen zudem die Überprüfung und Bewertung gefundener Lösungen hinsichtlich der Erfüllung der Anforderungen. Im Bereich der Konfiguration existieren zahlreiche Anwendungssysteme (CASS, SELCON, PLAKON, etc.), die allerdings nur für Aufgaben aus dem sogenannten Routine-Konfigurieren geeignet sind [Günter, 1993]. Unter Routine-Konfigurieren werden solche Problemstellungen verstanden, bei denen alle Domänenobjekte, ihre Eigenschaften und ihre kompositionelle Struktur vorgegeben sind und bei denen die Lösungsfindung auf einer bekannten Vorgehensweise basiert [Brown, Chandrasekaran, 1989]. Aufgrund der beschriebenen Probleme in der Produktentwicklung (Komplexität, Unsicherheit, Dynamik) sind solche Systeme für den Einsatz in diesem Anwendungsgebiet nur

wenig geeignet. Daher wurde diese Form der Konfigurierung erweitert und der Ansatz der Fuzzy Constraint Netzwerke entwickelt.

3. Fuzzy Constraint Netzwerke

Fuzzy Constraint Netzwerke sind eine Verbindung von Constraint Netzwerken mit der Fuzzy-Set-Theorie [Young, Giachetti, Ress, 1996]. Dieses Konfigurierungssystem erlaubt die Erweiterung des Routine-Konfigurierens um

- unscharfe, unvollständige und unsichere Zielspezifikationen,
- unscharfe, unvollständige und unsichere Definitionen von Domänenobjekten, Objekteigenschaften und Relationen sowie um
- funktionale Anforderungen [Young, Perrone, Eversheim, Roggatz, 1995].

Zudem kann bei der Verwendung dieses Ansatzes flexibel konfiguriert werden. Definiert werden Fuzzy Constraint Netzwerke wie folgt [Dechter, Pearl, 1988]:

Ein Fuzzy Constraint Netzwerk besteht aus einer Menge von *n* Variablen, $\tilde{X} = \{\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_n\}$, und einer Menge von *m* Constraints, C = {C_1, C_2, ... , C_m}. Jede unscharfe Variable $\tilde{\tilde{X}}_i$ gehört zu einem Domain Ω_i. Dieser Domain enthält alle Werte, die für die Variable zugelassen sind. Ein Constraint C_i ist eine Relation zwischen k Variablen einer Untermenge $\tilde{X}$, $\tilde{X}_i' = \{\tilde{X}_{i1}, \tilde{X}_{i2}, ..., \tilde{X}_{ik}\} \subseteq \tilde{X}$. Somit ist ein Constraint (z. B. $C_i(\tilde{X}_{i1}, \tilde{X}_{i2}, ..., \tilde{X}_{ik})$) eine Untermenge des kartesischen Produktes $\Omega_{i1} \times \Omega_{i2} \times ... \times \Omega_{ik}$ [Dechter, Beek, 1995; Mackworth, 1990].

Bei dieser Formulierung wird jeder Constraint zu einem bestimmten Grad $\mu_{Ci} \in [0,1]$ erfüllt. Das Netz gilt als gelöst, wenn $\mu_{Ci} \geq \alpha_S$ wobei α_S der Wahrheitswert ist. Der Anwender bestimmt, ab welchem Wahrheitswert er einen Constraint als ”eingehalten”, bzw. ab welchem Grenzwahrheitswert er Constraints als ”verletzt” betrachtet [Roggatz, 1997].

4. Einsatz von Fuzzy Constraint Netzwerken in der Konstruktion

Aufbauend auf den Arbeiten zur Entwicklung von Fuzzy Constraint Netzwerken wurden im Rahmen des SFB 361 an der RWTH Aachen Vorgehensweisen erarbeitet, um die Netzwerke zur Klärung und Präzisierung der Aufgabenstellung im Konstruktionsprozeß einzusetzen. Hierbei wurden sowohl Ansätze zur Strukturierung der Netzwerke untersucht als auch Möglichkeiten zur Nutzung häufig eingesetzter präventiver Qualitätstechniken

für den Aufbau der Netzwerke geprüft [Eversheim, Roggatz, Young, 1997]. Dabei hat sich die Verbindung der Netzwerke mit Quality Function Deployment (QFD) als besonders geeignet erwiesen. Diese Verbindung ermöglicht sowohl die Unterstützung von Anwendern bei der Formulierung von Constraints als auch die Bewertung der ermittelten Lösung im Hinblick auf die Erfüllung von Kundenanforderungen [Eversheim, Roggatz, Unbescheiden, 1997].

Bei dieser Vorgehensweise müssen mögliche Ausprägungen der in der QFD ermittelten Produktmerkmale bestimmt und fuzzifiziert werden. Im nächsten Schritt werden die Ausprägungen der Produktmerkmale kombiniert und Restriktionen abgebildet, die bei der Kombination der Ausprägungen berücksichtigt werden müssen. Bei der Formulierung der Restriktionen können aus der Korrelationsmatrix der QFD wichtige Hinweise über die Existenz und die Art von Abhängigkeiten entnommen werden.

Aufbauend auf die Abbildung der Merkmalsausprägungen und der Restriktionen können die realisierbaren Lösungen mit dem Konfigurationswerkzeug ermittelt werden. Für die quantitative Bewertung von Merkmalkombinationen muß eingangs für jedes Merkmal (M_i), das über einen Wertebereich variiert werden kann, eine Zielfunktion (Z_i) ausgewählt werden. Diese Zielfunktion gibt die Optimierung eines Merkmals über seiner Basisvariablen an. Der Wertebereich der Zielfunktion bewegt sich auf der Ordinate zwischen null (Ziel nicht erfüllt) und eins (Ziel voll erfüllt). Auf der Abszisse wird die Basisvariable aufgetragen. Hier besteht keine Einschränkung bezüglich des Wertebereichs. Als Zielfunktion eignet sich prinzipiell jede mathematische Funktion, deren Wertebereich zwischen ‘0’ und ‘1’ liegt. Stetigkeit und Differenzierbarkeit sind nicht zwingend notwendig. Im Anschluß an die Definition der Zielfunktion wird der Erfüllungsgrad (E_{ij}) der Zielfunktion durch die möglichen Ausprägungen (M_{ij}) des Produktmerkmals (M_i) ermittelt. Hierfür können verschiedene Fuzzy-Operatoren verwendet werden, für eine optimistische Abschätzung ist der Possibility-Operator besonders geeignet. Mit Hilfe des Possibility-Operators wird das Maximum der Schnittmenge zwischen einer Ausprägung und der Zielfunktion bestimmt. Dadurch erhält man eine Abschätzung des Erfüllungsgrads der Zielfunktion durch eine Ausprägung. Der Erfüllungsgrad der Zielfunktion durch die Ausprägung j des Merkmals i ergibt sich dadurch zu: $E_{ij} = Poss(Z_i \mid M_{ij})$. Der Erfüllungsgrad von zweiwertigen Merkmalen ist entweder ‘0’ (Merkmal nicht vorhanden) oder ‘1’ (Merkmal vorhanden). Durch Multiplikation des Erfüllungsgrads der Zielfunktion mit der in dem QFD ermittelten Bedeutung eines Merkmals bezüglich der Erfüllung aller Kundenanforderungen (B_i) erhält man die Wertigkeit der Ausprägung

im Hinblick auf die Kundenanforderungen: $W_i = E_i * B_i$. Die Gesamtwertigkeit einer Kombinationsmöglichkeit von Ausprägungen im Hinblick auf die Erfüllung aller Kundenanforderungen erhält man durch Summenbildung der Wertigkeiten der Ausprägungen aller Produktmerkmale: $W_{ges} = \sum_{i=1}^{n} (E_{ij} * B_i)$

bzw. $W_{ges} = \sum_{i=1}^{n} (Poss(Z_i \left| M_{ij} \right.) * B_i)$.

Zur Überprüfung der Konsistenz gefundener Lösungen können die Abhängigkeiten zwischen Aktivitäten modelliert werden. Durch Zuweisung von Werten zu den einzelnen Variablen des Fuzzy Constraint Netzwerks werden die Abhängigkeiten ermittelt, die bei der angestrebten Lösung nicht eingehalten werden können. Diese Überprüfung ist vor allem bei parallelen Aktivitäten in der Produktentwicklung sinnvoll [Eversheim, Roggatz, Young, 1997].

4. Beispiel zur Ermittlung und Bewertung von Merkmalsausprägungen

Die beschriebene Vorgehensweise zum Einsatz von Fuzzy Constraint Netzwerken in der Konstruktion wird im folgenden am Beispiel der Entwicklung eines Mobiltelefons verdeutlicht. In dieser Beispielanwendung werden die Constraint Netzwerke zur Definition des Produktentwicklungsprojektes herangezogen.

	sehr klein	klein	mittel	groß	sehr groß
Länge [mm]	[<110,110,125]	[110,125,140]	[125,140,155]	[140,155,170]	[155,170,>170]
Akku-Kapazität [mAh]	[<300,300,450]	[300,450,600]	[450,600,750]	[600,750,900]	[750,900,>900]

Tabelle 1: Unscharfe Ausprägungen von Länge und Akku-Kapazität eines Mobiltelefons

Wesentliche Kundenanforderungen an ein mobiles Telefon sind unter anderem die äußeren Abmessungen und die maximale Gesprächszeit. Unscharfe Ausprägungen der genannten Merkmale zeigt Tabelle 1. Aufbauend auf der Bestimmung der möglichen Ausprägungen der Produktmerkmale können deren Kombinationsmöglichkeiten zusammengestellt werden. Bei j Produktmerkmalen mit je n unterschiedlichen Ausprägungen ergeben sich j^n mögliche Zusammenstellungen. Diese können mit einem Fuzzy Constraint Netzwerk abgebildet werden. Um die technisch realisierbaren Kombinationen zu ermitteln ist es erforderlich, zusätzliche Constraints zu formulieren, die die Abhängigkeiten zwischen den Merkmalsausprägungen repräsentieren. Beispiele für solche Constraints sind, daß zur Realisierung einer Akkumulatorkapazität von über 550 mAh die Länge des Handys größer als 150 mm sein muß, bzw. daß für eine Akkumulatorkapazität

von über 700 mAh die Länge des Mobiltelefons mindestens 160 mm betragen muß. Die Anwendung derartiger Constraints auf das ursprüngliche Netzwerk liefert die technisch realisierbaren Kombinationen von Merkmalsausprägungen.

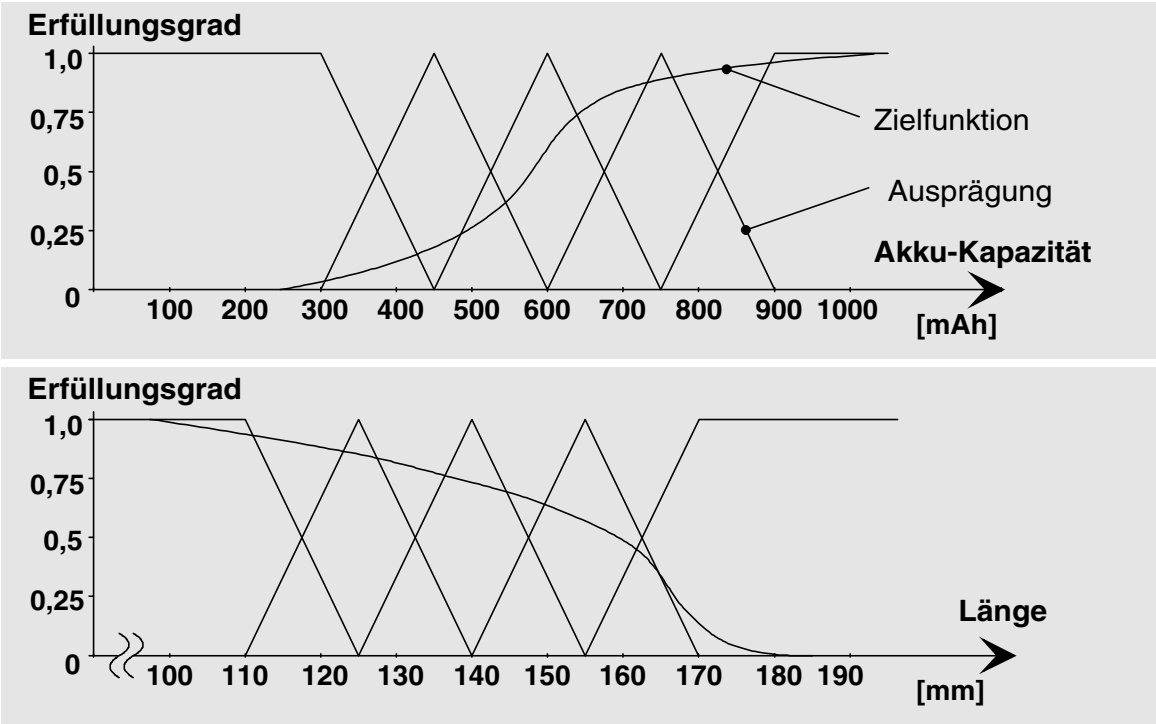


Bild 1: Zielfunktionen

Zur unscharfen Bewertung der Kombinationen müssen Zielfunktionen für die Produktmerkmale definiert werden. Zielfunktionen für Akkumulatorkapazität und Länge eines Mobiltelefons zeigt Bild 1. Im nächsten Schritt wir die Erfüllung der Zielfunktion durch die einzelnen Ausprägungen ermittelt. Zur Bestimmung dieses Erfüllungsgrads wird in diesem Beispiel der Fuzzy-Operator Possibility verwendet. Eine Akkumulatorkapazität zwischen 600 und 900 mAh ergibt somit einen Erfüllungsgrad der Zielfunktion von 0,8. Wurde in der QFD eine relative Bedeutung dieses Produktmerkmals von 117 ermittelt, ergibt sich die Wertigkeit der Ausprägung zu 0,8*117=93,6. Die Summe der Wertigkeiten aller Ausprägungen einer Kombinationsmöglichkeiten liefert die Gesamtwertigkeit. Zur besseren Übersichtlichkeit kann diese normiert werden. Tabelle 2 zeigt einen Auszug aus der Bewertung der Kombinationsmöglichkeiten von Länge und Akkumulatorkapazität des Mobiltelefons. Aus dieser Bewertung ist ersichtlich, daß ein Mobiltelefon mit einer Akkumulatorkapazität zwischen 800 und 900 mAh und einer Länger von über 155 mm die Kundenanforderungen am besten erfüllt und somit den größten Kundennutzen aufweist.

	Akku-Kapazität	Länge	Absolute Wertung	Relative Wertung
Handy 1	klein	klein	70,2	5,3
Handy 2	mittel	mittel	114,75	8,6
Handy 3	mittel	groß	111,15	8,3
Handy 4	groß	sehr groß	127,35	9,5

Tabelle 2: Auszug aus der Bewertung der Kombinationsmöglichkeiten

6. Nutzen des Einsatzes von Fuzzy Constraint Netzwerken bei der Projektdefinition

In der Vergangenheit konnte gezeigt werden, daß sich Fuzzy Constraint Netzwerke gut für den Einsatz in der Produktentstehung eignen. Am Beispiel der Entwicklung eines Mobiltelefons wurden frühe Phasen der Produktentstehung durchlaufen, wobei Markt- und Produktionsanforderungen sowie Zeit-, Kosten- und Qualitätsrestriktionen abgebildet wurden. Hierdurch konnte die Konsistenz gefundener Lösungen überprüft und somit die Dynamik in Produktentstehungsprozessen reduziert werden [Eversheim, Roggatz, Young, 1997; Giachetti, Young, Roggatz, Evershiem, Perrone, 1997]. In diesem Beitrag wird beschrieben, wie mit Hilfe der Netzwerke ebenfalls eine Bewertung der Ergebnisse hinsichtlich des Kundennutzens vorgenommen werden kann. Diese Bewertung der Ergebnisse erleichtert die Ermittlung der marktgerechtesten Lösungsalternative.

Insgesamt kann gesagt werden, daß durch unscharfe Konfiguration mit Hilfe von Fuzzy Constraint Netzwerken die Klärung und Lösung der Konstruktions- bzw. Entwicklungsaufgabe erheblich erleichtert wird.

7. Literatur

Brown, D.C.; Chandrasekaran, B., 1989: Design Problem Solving, Pitman.

Clausing, D., 1994: Total Quality Development, A Step-By-Step Guide to World-Class Concurrent Engineering, ASME Press - New York.

Dechter, R.; Beek, P. van, 1995: Local and Global Relation Consistency, Principals and Practise of Constraint Programming, Proceedings of the 1st International Conference on Constraint Programming, Cassis, Frankreich, Springer Verlag.

Dechter,R.; Pearl, J., 1988: Network-Based Heuristics for Constraint Satisfaction Problems, Artificial Intelligence, vol.34, S. 1-38.

Eversheim, W.; Laufenberg, L., 1995: Markterfolg ist planbar!, Integrierte Gestaltung von Simultaneous Engineering-Projekten, VDI-Z 1/2.

Eversheim, W.; Roggatz, A.; Unbescheiden, H., 1997: The Use of Uncertain Information for Premature and Customer-Oriented Assessment of Product Parameters, Proceedings of the IDEA, Aachen, S. 7-11.

Eversheim, W.; Roggatz, A.; Young, R. E., 1997: Ability of Fuzzy Constraint Networks to Reduce the Dynamic of Changes in Product Development, Reusch, B. (Hrsg.), Computational Intelligence - Theory and Applications, Springer Verlag, Berlin, S. 122-140.

Giachetti, R.; Young, R. E.; Roggatz, A.; Eversheim, W.; Perrone, G., 1996: A Methodology for the Reduction of Imprecision in the Engineering Design Process, erscheint in: European Journal of Operations Research.

Günter, A., 1993: Überblick über PROKON-Ziele und –Aktivitäten, Tagungsband zum Statusseminar KI des BMFT‘s, DLR, Berlin.

Machworth, A. K., 1990: Constraint Satisfaction; in: Encyclopedia of Aritficial Intelligence, John Wiley & Sons, Chichester, S.205-211.

Roggatz, A., 1997: Entscheidungsunterstützung für die frühen Phasen der integrierten Produkt- und Prozeßgestaltung, Dissertation an der RWTH Aachen.

Saretz, B., 1993: Entwicklung einer Methodik zur Parallelisierung von Planungsabläufen, Dissertation an der RWTH Aachen.

Young, R.; Giachetti, R.; Ress, D., A., 1996: A Fuzzy Constraint Satisfaction System for Design and Manufacturing, IEEE International Conference on Fuzzy Systems, New Orleans, S.: 8-10.

Young, R.; Perrone, G.; Eversheim, W.; Roggatz, A., 1995: Fuzzy Constraint Satisfaction for Simultaneous Engineering, Production Engineering Vol. II/2, S.: 181-184.

Zanger, C.; Baier, G. 1994: Computereinsatz zur Entscheidungsunterstützung für Führungskräfte, Management und Computer, Jg. 2, Heft 3, S.: 203-209.

Strategien zur Domänenanalyse

Ulrich Scholz
Fachbereich Informatik, Technische Universität Darmstadt
Alexanderstraße 10, 64283 Darmstadt, Germany
scholz@informatik.tu-darmstadt.de
http://kirmes.inferenzsysteme.informatik.tu-darmstadt.de/~scholz/

Abstract

Propositionales Planen ist ein hartes Problem, trotz rasanter Entwicklung stoßen Planer, bedingt durch die Weite des Suchraums, bei Problemen mittlerer Größe an Laufzeitgrenzen. Mit Informationen über die Struktur von Planungsdomänen kann der Suchraum drastisch eingeschränkt werden. Dieser Artikel stellt drei Strategien vor, die aus einer gegebenen Domäne Strukturinformationen extrahieren und sie einem Planer zur Verfügung stellen. Dabei ist es denkbar, diese Strategien als Vorverarbeitungsschritt unabhängig von einem konkreten Planungsproblem durchzuführen und die gewonnen Informationen für viele Anwendungen wiederzuverwenden.

Einleitung

Propositionales Planen ist ein hartes Problem, selbst im weniger allgemeinen Fall ist es NP-hart (Bylander 1994). Trotz rasanter Entwicklung stoßen Planer, bedingt durch die Weite des Suchraums, bei Problemen mittlerer Größe an Laufzeitgrenzen. Mit Informationen über die Struktur von Planungsdomänen kann der Suchraum drastisch eingeschränkt werden (Kautz & Selman 1998), allerdings wird dieses domänenspezifische Wissen dem Planer manuell zur Verfügung gestellt. Dies bedeutet nicht nur Aufwand und die Gefahr von Auslassungen und Fehlern. Es ist eine Einschränkung der Allgemeinheit des propositionalen Planens, da Planungsdomänen den Vorzug gegeben wird, die vom Benutzter speziell konstruiert oder bearbeitet wurden.

Dieser Artikel stellt drei Strategien vor, die aus einer gegebenen Domäne Strukturinformationen extrahieren und sie einem Planer zur Verfügung stellen. Dabei ist es denkbar, diese Strategien als Vorverarbeitungsschritt unabhängig von einem konkreten Planungsproblem durchzuführen und die gewonnen Informationen für viele Anwendungen wiederzuverwenden. Die Strategie MCP ermöglicht es Planeren, die total geordnete Pläne verwenden, ihren Suchraum einzuschränken. Im nächsten Abschnitt wird die Strategie RAS vorgestellt, die ersetzbare Aktionssequenzen ermittelt. Die Strategie SMP beweist für Teilmengen der Propositionen einer Domäne, daß nur einer ihrer Elemente zu einem Zeitpunkt aktiv sein kann. Im letzten Abschnitt werden Aufwand und Ergebnisse der Strategien besprochen.

Terminologie

Diese Arbeit beschäftigt sich mit Beziehungen zwischen Lösungen in gegebenen Domänen bzw. mit Eigenschaften, die für jede Lösung aller Planungsprobleme einer gegebenen Domäne gelten. Hierzu verwenden wir die folgende Terminologie.

Eine ***Aktion*** a besteht aus einer Menge von ***Vorbedingungen*** $pre(a)$, einer Menge von durch die Aktion ***aktivierten Effekten*** $add(a)$ und ***deaktivierten Effekten*** $del(a)$. Alle Vorbedingungen und Effekte sind positive Propositionen. Zum Beispiel bedeutet $p \in add(a_1) \cap pre(a_2)$, daß die Proposition p von a_1 aktiviert wird und daß sie Vorbedingung von a_2 ist. Eine ***Domäne*** $\mathcal{A}$ ist eine Menge von Aktionen, wobei Prob($\mathcal{A}$) die Menge aller in den Aktionen von $\mathcal{A}$ vorkommenden Propositionen bezeichnet.

Ein ***Planungsproblem*** $\mathcal{PP}$ ist ein Tripel $\prec\mathcal{A}, \Sigma, \Omega\succ$, wobei $\mathcal{A}$ eine Domäne und Σ bzw. Ω die ***Ausgangs-*** bzw. die ***Zielsituation*** bezeichnet. O. B. d. A. schränken wir Σ und Ω auf Teilmengen von Prob($\mathcal{A}$) ein.

Eine ***Zeitstufe*** ist eine Zahl $t \in \mathbb{N}_0$. Ein ***Prä-Plan*** der Domäne $\mathcal{A}$ ist eine Menge von Aktions/Zeitstufenpaaren a^t. Durch die Definition der Zeitstufen als natürliche Zahlen sind

die Aktionen eines Prä-Planes total geordnet. Ein ***serieller*** Plan hat maximal eine Aktion zu einer Zeitstufe, einem ***parallelen*** Plan fehlt diese Einschränkung. Ein Prä-Plan ist in allen seinen Eigenschaften endlich.

Ein serieller Prä-Plan P ist ***konfliktfrei***, wenn es für jedes Aktionspaar $a_1^{t_1}, a_2^{t_2} \in P$ mit $p \in del(a_1) \cap pre(a_2), t_1 < t_2$ eine Aktion $a_3^{t_3} \in P$ gibt, mit $p \in add(a_3)$ und $t_1 < t_3 < t_2$. Für einen konfliktfreien, parallelen Prä-Plan P muß zusätzlich gelten, daß für kein Aktionspaar $a_1^t, a_2^t \in P$ gilt: $add(a_1) \cap del(a_2) \neq \emptyset$ oder $pre(a_1) \cap del(a_2) \neq \emptyset$. Für einen Prä-Plan P ist $kf(P)$ wahr, falls P konfliktfrei ist.

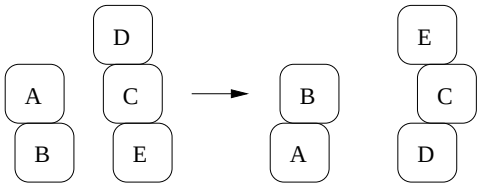
Ein ***Plan*** P ist die Lösung eines Planungsproblems $\prec \mathcal{A}, \Sigma, \Omega \succ$. Für P gilt:

- P ist ein konfliktfreier Prä-Plan.
- Die einzige Aktion in der Zeitstufe 0 ist die zusätzliche Aktion INIT, mit $pre(\text{INIT}) = \emptyset$, $add(\text{INIT}) = \Sigma$ und $del(\text{INIT}) = \text{Prob}(\mathcal{A}) \setminus \Sigma$. Dies ist die einzige Verwendung von INIT in P.
- Die einzige Aktion in der letzten Zeitstufe ist die zusätzliche Aktion GOAL, mit $pre(\text{GOAL}) = \Omega$ und $add(\text{GOAL}) = del(\text{GOAL}) = \emptyset$. Dies ist die einzige Verwendung von GOAL in P.

Mit diesen Definitionen können nun die drei Strategien vorgestellt werden.

Maximal komprimierte Pläne

Bei der Verwendung paralleler, total geordneter Pläne kann die Lösungsmenge eines Planungsproblems Pläne beinhalten, die sich nur durch die Reihenfolge von Aktionen unterscheiden. Wird die totale Ordnung durch die Zuweisung an bestimmte Zeitstufen erzwungen, können zusätzlich noch Lösungen enthalten sein, die sich nur durch diese Zuweisung unterscheiden, bei denen die Reihenfolge der Aktionen aber gleich ist. Dies kann den Lösungsmenge eines Planungsproblems aufblähen. Solche Plandarstellungen werden z. B. von Planern wie SATPLAN (Kautz & Selman 1996) oder GRAPHPLAN (Blum & Furst 1995) verwendet.



Beispiel für die Aufblähung des Lösungsraums durch explizite Zeitstufen in einer Blocks World Domäne.[1] Das Umdrehen des rechten Turms benötigt drei Zeitstufen. Die beiden Aktionen zur Vertauschung von Block A und Block B können auf drei verschiedene Weisen diesen Zeitstufen zugewiesen werden.

Die Strategie MCP (maximal compressed plans) ist auf dieses Problem zugeschnitten. Sie basiert auf dem Ausschluß von Prä-Planen, bei denen es Aktionen gibt, die eine Zeitstufe früher eingefügt sein könnten, ohne einen weiteren Konflikt zu erzeugen.

Definition 1. $a_1 \overset{mcp}{\rightarrow} a_2 \equiv$
$pre(a_1) \cap del(a_2) \neq \emptyset \vee del(a_1) \cap add(a_2) \neq \emptyset \vee$
$add(a_1) \cap del(a_2) \neq \emptyset \vee add(a_1) \cap pre(a_2) \neq \emptyset$

Die Relation $\overset{mcp}{\rightarrow}$ hält genau dann zwischen zwei Aktionen, falls sich zwischen ihnen bei Platzierung in der selben Zeitstufe ein Konflikt ergeben könnte. Nur falls alle Aktionen eines Prä-Planes P, außer den ersten, mit einer direkt vorher eingefügten Aktion in dieser Beziehung stehen, wollen wir P als Teil einer Lösung akzeptieren.

Definition 2. *Für einen Prä-Plan P sei $t_{min}(P)$ die kleinste besetzte Zeitstufe.*
$mcp(P) :=$
$\forall a_1^t \in P.\ t = t_{min}(P) \ \vee \ \exists a_2^{t-1} \in P.\ a_2 \overset{mcp}{\rightarrow} a_1$

Satz 3. *Für jeden konfliktfreien Prä-Plan P gibt es einen konfliktfreien Prä-Plan P' mit $mcp(P')$, der sich von P nur durch die Zuweisung der Aktionen zu den Zeitstufen unterscheidet.*

Beweis. Annahme: Sie P ein konfliktfreier Prä-Plan und $mcp(P)$ gilt nicht. Dann gibt es eine Aktion $a_1^t \in P$ mit $t \neq t_{\min}(P)$ und $\neg \exists a_2^{t-1} \in P.\ a_2 \overset{mcp}{\rightarrow} a_1$. Dann kann a_1 auch eine Zeitstufe früher eingefügt sein, also $P' = (P \setminus a_1^t) \cup a_1^{t-1}$ ist konfliktfrei und unterscheidet sich von P nur durch die Zuweisung der Aktionen zu den Zeitstufen.

Falls wiederum die Bedingung $mcp(P')$ nicht gilt, kann dieser Schritt wiederholt werden und wir erhalten ein neues P'. Da die Zahl

[1] Bei den in dieser Arbeit verwendeten Blocks World Domänen ist eine Aktion das Versetzen eines Blocks von der Ausgangsposition zur Endposition, ein Greifarm ist nicht Vorgesehen.

der Aktionen und Zeitstufen von Prä-Plänen endlich ist, erhält man zuguterletzt ein P' mit $mcp(P')$.

□

Satz 4. *Für jede Lösung P eines Planungsproblems gibt es genau eine Lösung P', mit $mcp(P')$, P' unterscheidet sich von P nur durch die Zuweisung der Aktionen zu den Zeitstufen und P' ist kürzer oder gleich lang wie P.*

Beweis. (Skizze). Der Graph G_P eines Planes P sei folgendermaßen definiert: Die Aktionen von P sind die Knoten und die Beziehungen $\overset{mcp}{\rightarrow}$ zwischen Aktionen an früheren Zeitpunkten zu Aktionen an späteren sind die Kanten. Wir bilden Äquivalenzklassen, indem wir Pläne mit gleichem Graphen gleich setzen. Als Repräsentanten jeder Klasse wählen wir denjenigen Plan P_R, der alle Aktionen $a \in P_R$ zum Zeitpunkt t_a = 'Länge des längsten Pfads von INIT zu a' eingefügt sind.

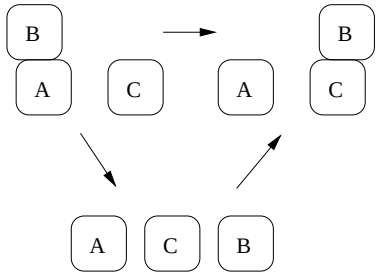
Es wird offensichtlich, daß P_R eindeutig ist, $mcp(P_R)$ gilt und daß P_R einer der kürzesten Mitglieder seiner Äquivalenzklasse ist.

□

Der Suchraum von Planungsproblemen wird nicht nur durch Vertauschungen sondern auch durch ersetzbare Aktionen vergrößert.

Ausschluß von ersetzbaren Aktionssequenzen

Viele Domänen haben die Eigenschaft, daß die direkte Hintereinanderausführung ihrer Aktionen (*Sequenzen*, $\circ$) durch eine einzelne Aktion ersetzt werden kann. Durch Ausschließen solcher Sequenzen kann der Suchraum eingeschränkt werden, wobei die verbleibenden Lösungen weniger Aktionen enthalten und kürzer sein können. Auf Grund der Komplexität der Berechnung ersetzbarer Sequenzen behandelt die RAS-Strategie (replacable action sequences) nur Sequenzen der Länge zwei.



Anstelle Block B erst auf den Tisch und dann auf Block C zu stellen, kann man ihn auch direkt dorthin bewegen.

Um für einen Plan sicherzugehen, daß eine Sequenz $a_1 \circ a_2$ durch eine Aktion a_3 ersetzt werden kann, muß garantiert werden, daß zu jeder Lösung P eine Lösung P' existiert, so daß P und P' sich nur durch die Ersetzung von $a_1 \circ a_2$ durch a_3 unterscheidet.

Für einen *linearen* Plan kann man Definieren:

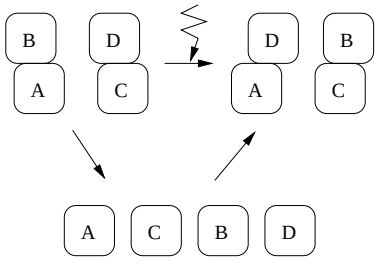
Definition 5. $a_1 \circ a_2 =_{lin} a_3 \equiv$
$pre(a_3) \subseteq pre(a_1) \cup (pre(a_2) \setminus add(a_1)) \wedge$
$(add(a_1) \setminus del(a_2)) \cup add(a_2) \subseteq add(a_3) \wedge$
$del(a_3) \subseteq (del(a_1) \setminus add(a_2)) \cup del(a_2) \wedge$
$del(a_1) \cap pre(a_2) = \emptyset$

Satz 6. *Sei*
$P = \{\text{INIT}^0, a^1_{i_1}, \ldots, a^t_1, a^{t+1}_2, a^{t+2}_{i_{t+2}}, \ldots, \text{GOAL}^n\}$
ein linearer Plan und $a_1 \circ a_2 =_{lin} a_3$. Dann ist
$P' = \{\text{INIT}, a^1_{i_1}, \ldots, a^t_3, a^{t+1}_{i_{t+2}}, \ldots, \text{GOAL}^{n-1}\}$
auch ein Plan.

Beweis. Trivial, da a_3 nach Definition höchstens die Vorbedingungen der Sequenz $a_1 \circ a_2$ hat, höchstens ihre deaktivierten Effekte deaktiviert und mindestens ihre aktivierten Effekte aktiviert. Es können also durch die Ersetzung keine Konflikte mit Aktionen entstehen, die Zeitstufen kleiner t oder größer $t+1$ zugewiesen sind.

□

Bei parallelen Plänen muß zusätzlich garantiert werden, daß die Ersetzung mit keiner weiteren Aktion eines Planes in Konflikt geraten kann.



Obwohl $=_{\text{lin}}$ zwischen den entsprechenden Aktionen gilt kann bei der Verwendung paralleler Pläne in diesem Beispiel keiner der Blöcke B oder D direkt auf das Ziel gesetzt werden.

Definition 7. $a_1 \circ a_2 =_{par} a_3 \equiv$
$a_1 \circ a_2 =_{lin} a_3 \wedge \forall a_4, a_5.$
$kf(\{a^t_1, a^{t+1}_2, a^t_4\}) \wedge kf(\{a^t_1, a^{t+1}_2, a^{t+1}_5\}) \rightarrow$
$kf(\{a^t_3, a^t_4\}) \wedge kf(\{a^t_3, a^{t+1}_5\}) \vee kf(\{a^t_4, a^{t+1}_3\}) \wedge$
$kf(\{a^{t+1}_3, a^{t+1}_5\})$

Satz 8. *Sei P ein Plan mit $a_1^t, a_2^{t+1} \in P$ und sei $a_1 \circ a_2 =_{par} a_3$. Dann ist auch $P' = (P \setminus \{a_1^t, a_2^{t+1}\}) \cup a_3^t$ oder $P'' = (P \setminus \{a_1^t, a_2^{t+1}\}) \cup a_3^{t+1}$ ein Plan.*

Beweis. Da $(a_1 \circ a_2 =_{\mathrm{par}} a_3) \rightarrow (a_1 \circ a_2 =_{\mathrm{lin}} a_3)$, kann die Ersetzung von $\{a_1^t, a_2^{t+1}\}$ keine Konflikte mit Aktionen in Zeitstufen kleiner t oder größer $t+1$ erzeugen, es bleiben nur mögliche Konflikte mit Aktionen in den zu verändernden Zeitstufe. Der zweite Teil der Definition schließt dies aus.

□

Durch eine Ersetzung in einem parallelen Plan wird dieser i. A. nicht verkürzt, da zu der Sequenz parallele Aktionen dies verhindern können.

Mengen von sich gegenseitig ausschließenden Propositionen

Bei der Betrachtung einer Blocks World Domäne $\mathcal{A}$ erwarten wir, daß ein Block sich nur an einer Stelle befindet bzw. daß nur ein Block auf einem anderen steht. Entsprechend ist von der Teilmenge $\{\mathrm{on(A,B)}, \mathrm{on(A,C)}, \ldots\}$ der Menge $\mathrm{Prob}(\mathcal{A})$ nur jeweils eine Proposition aktiv. Die Kenntnis und Verwendung von Teilmengen mit dieser Eigenschaft kann für das Planen von großem Nutzen sein.

Das manuelle Suchen und Definieren solcher *Mengen sich gegenseitig ausschließender Propositionen* SMPs (sets of mutually exclusive propositions) kann schwierig sein. Zum Beispiel hat die Flat Tire Domäne von Russell (Russell & Norwig 1995) bei nur 28 Aktionen und 33 Propositionen 15 SMPs der Größe zwei bis vier, die sich mehrfach überschneiden. Außer dem Arbeitsaufwand sie zu finden, muß noch bewiesen werden, daß für sie die oben genannte Eigenschaft gilt.

Die SMP-Strategie umgeht diese Probleme, indem sie es ermöglicht, einen großen Teil der SMPs einer gegebenen Domäne zu berechnen. Sie basiert auf der folgenden Tatsache: Aktiviert eine Aktion a ein inaktives Element p eines SMP M, so muß sie ein aktives Element $q \in M$ deaktivieren, falls es ein solches gibt. Wird dessen Existenz in jedem Fall durch genau eine Vorbedingung erzwungen, so kann die SMP-Strategie dieses SMP finden, es gilt $\{q\} = pre(a) \cap del(a) \cap M$. Die Zahl der aktiven Elemente eines SMPs mit dieser Eigenschaft kann sich also nicht erhöhen. Die Menge der konsumierten Vorbedingungen einer Aktion a, $pre(a) \cap del(a)$, wird im Weiteren durch $cons(a)$ abgekürzt. Im Folgenden wird der Algorithmus der SMP-Strategie skizziert.

Die SMP-Strategie konstruiert SMPs einer Domäne $\mathcal{A}$ rekursiv. Bei jeder Rekursion werden drei Mengen übergeben:

- Die Kandidatenmenge $K \subseteq \mathrm{Prob}(\mathcal{A})$, von der vermutet wird, daß sie Teil eines SMP ist: Weder ist K selbst ein SMP noch wurde gezeigt, daß K nicht Teil eines SMPs sein kann.

- Die Verbotsmenge $V \subseteq \mathrm{Prob}(\mathcal{A})$, von der gezeigt ist: Ist K Teilmenge eines SMPs M der Domäne $\mathcal{A}$, so gilt $V \cap M = \emptyset$.

- Die Menge I mit $\forall E \in I.\ E \subseteq \mathrm{Prob}(\mathcal{A})$. Die Elementmengen von I erklären sich folgendermaßen: Aktiviert eine Aktion a eine Proposition eines SMP M, so muß nach der oben genannten Bedingung $|cons(a) \cap M| = 1$ gelten. Dies wird zur Erweiterung von K ausgenutzt. Die Menge I enthält nun die Mengen $cons(a) \cap V$ von Aktionen a, die ein Element aus K aktivieren. Aus ihnen werden die Elemente von V gestrichen, da von diesen schon gezeigt wurde, daß sie keine Erweiterung von K sein können.

Bei dem rekursiven Aufruf wird eine Elementmenge $E \in I$ ausgewählt. Nacheinander wird K durch jedes der Propositionen $p \in E$ erweitert und V bzw. I entsprechend angepaßt: Aus I werden alle Mengen E' gestrichen, die p enthalten und $E' \setminus \{p\}$ erweitert V. Für alle Aktionen a mit $p \in add(a)$ erweitert $cons(a)$ die Menge I. Zuletzt werden alle Elemente von V aus allen Elementmengen von I gestrichen.

Falls I nun die leere Menge *ist*, d. h. alle Aktionen, die ein Element aus K aktivieren auch eines deaktivieren, so ist K ein SMP. Falls I die leere Menge *enthält*, so gibt es eine Aktion, die ein Element aus K aktiviert ohne ein Element aus einem SMP deaktivieren zu können, von der K eine Teilmenge sein könnte. Dies bedeutet, daß K keine Teilmenge eines SMPs sein kann.

Die Rekursion wird gestartet, indem man nacheinander für alle $p \in \mathrm{Prob}(\mathcal{A})$ die Menge $K = \{p\}$, die Menge $V = \{q | a \in \mathcal{A} \wedge \{p, q\} \subseteq add(a)\}$ und die Menge $I = \{cons(a) | a \in \mathcal{A} \wedge p \in add(a)\}$ setzt.

Die SMPs können nur dann uneingeschränkt zur Lösung eines Planungsproblems

eingesetzt werden, wenn die Bedingungen in dessen Ausgangssituation beachtet werden. Falls zwei oder mehr Propositionen eines SMPs in einer Ausgangssituation aktiv sind, so kann dies auch in weitern Zeitstufen eines Planes der Fall sein. Eine Möglichkeit dies zu Umgehen ist, solche Planungsprobleme zu verbieten. So ist die Blocks World Domäne, die für die Experimente in (Kautz & Selman 1996) verwendet wurde, manuell um SMP-Informationen erweitert worden und erlaubt nur das Stellen entsprechender Probleme.

Verschiedene SMPs haben oft gemeinsame Elemente. So ist die Proposition on(A,B) sowohl Teil des SMPs, das die Einzigartigkeit des Ortes von Block A beschreibt, als auch Teil des SMPs, daß nur einen anderen Block auf A zuläßt. Ein SMP kann daher nur schlecht als mehrwertige Variable gesehen werden, da in Domänen mit extrem verknüpften SMPs die Zahl der Werte dieser Variablen die Zahl der Propositionen übersteigen kann. Z. B. hat eine Blocks World Domäne mit n Blöcken n^2+2n Propositionen und die SMP-Strategie findet $2n$ SMPs, jeweils der Größe $n+1$. Die entsprechenden Variablen hätten zusammen $2n^2+2n$ Werte.

Die vorgestellte SMP-Strategie kann nicht alle SMPs einer Domäne finden. Ein Beispiel dafür sind die SMPs {on(X,Y), on(Y,X)}, wiederum aus der Blocks World Domäne, wobei X und Y zwei verschiedene Blöcke bezeichnen. Da Blöcke unabhängig voneinander platziert werden können, schwankt die Zahl der aktiven Elemente dieser SMPs zwischen Null und Eins, sie kann sich also erhöhen. Durch die Beschränkung auf konsumierende Aktionen wird dieser Fall nicht abgedeckt.

Ergebnisse

Um die Leistungsfähigkeit der vorgestellten Strategien zu demonstrieren, wurden einige Experimente in den Domänen Blocks World und in Logistics durchgeführt. Wie z. B. in (Kautz & Selman 1996) sind nur die reinen Suchzeiten in Sekunden angegeben, gemittelt über 50 Läufe. Die verwendeten Planer sind ProbaPla (Scholz 1997) und SatPlan.

Problem	mit SMP	ohne Strat.	SatPlan
bw_large.a	0.12	13.39	0.77
bw_large.b	0.84	350.43	33.95
bw_large.c	4.51	–	1104.15
bw_large.d	17.20	–	–

Beispiel für die Verwendung der SMP-Strategie.

Die Zeiten für SatPlan sind mit der original SatPlan-Distribution mit den Standardeinstellungen unter den selben Bedingungen wie für ProbaPla ermittelt worden. Die MCP- und die RAS-Strategien wurden nicht verwendet.

Problem	MCP & RAS	ohne MCP	ohne RAS
logistics6	0.03	5.55	0.36
logistics8	0.03	5.96	0.40

Beispiel für die Verwendung der MCP- und RAS-Strategie mit dem Planer ProbaPla.

Die Ergebnisse für die Probleme 'bw_large.a' bis 'bw_large.d' zeigen, daß die Verwendung von SMP-Informationen für das Planen nützlich ist. Ohne die SMP-Strategie kann ProbaPla nur die ersten beiden Probleme lösen. Mit der Strategie werden die Lösungen deutlich schneller gefunden als mit SatPlan, obwohl die mit SatPlan verwendete Domäne manuell hinzugefügte SMP-Beziehungen enthält. Die Experimente in der Logistics Domäne zeigen den Effekt der Strategien MCP und RAS.

Der Berechnungsaufwand für die einzelnen Strategien unterscheidet sich stark. Die Blocks World Domäne 'bw_large.d' hat bei 19 Blöcken 6498 Aktionen und 380 Fakten. Die Strategie SMP findet die zugehörigen 38 SMPs in 11 Sekunden während die MCP- und die RAS-Strategie für diese Domäne 131 bzw. 9497 Sekunden benötigen. Da die Ergebnisse von speziellen Planungsproblemen unabhängig sind, ist dieser Aufwand nur *einmal* für eine Domäne nötig. Auch können für ein Planungsproblem Ergebnisse von 'größeren' Domänen verwendet werden, also die aktuelle subsumieren. So können z. B. die Analyseergebnisse der Domäne für das Problem 'bw_large.d' für alle Blocks World Probleme mit bis zu 19 Blöcken verwendet werden.

Diskussion

Heuristiken, die den hier vorgestellten Strategien ähnlich sind, sind schon in verschiedenen Planern eingesetzt worden. Zum Beispiel wird in (Kautz & Selman 1998) beschrieben, daß in der Logistics Domäne ein Laster direkt nach dem Beladen in eine andere Stadt fahren sollte. Die Proposition 'Laster in X' ist eine Vorbedingung der Aktion, die den Laster in X beläd, als auch ein deaktivierter Effekt einer Aktion, die den Laster in eine andere Stadt bewegt. Zwischen Beladen und Fahren hält also die Relation $\overset{mcp}{\rightarrow}$. Die MCP-Strategie ist

aber keine künstlich hinzugefügte Heuristik, sondern garantiert das Finden aller solcher Beziehungen.

Das die Qualität von Plänen der Blocks World Domäne erhöht wird, wenn Züge über den Tisch durch direktere ersetzt werden, ist allgemein bekannt. Ein Beispiel ist die Planungsmethode 'Planning by Rewiting' (Ambite & Knoblock 1997), die u. a. auf der Ersetzung solcher Sequenzen basiert, die dem System manuell beigefügt wurden. Der RAS-Strategie ist es möglich, ersetzbare Sequenzen automatisch für eine gegebene Domäne zu berechnen.

Auch die SMP-Beziehung wird als Heuristik verwendent, z. B in der Blocks World Domäne, die für die Experimente in (Kautz & Selman 1996) verwendet wurde. Obwohl die SMP-Strategie nicht alle SMPs einer Domäne berechnen kann, ist sie eine effiziente Möglichkeit einen großen Teil dieser Strukturinformation automatisch zu errechnen und so die Allgemeinheit des Planers zu erhalten.

Die Ergebnisse zeigen, daß die Verwendung der vorgestellten Strategien die Leistung von Planern bedeutend erhöhen kann. Dies gibt Hoffnung auf weitere Erfolge der Arbeit auf diesem Gebiet. Neben dem Finden besser Algorithmen für die Strategien geht unser Interesse in drei Richtungen: das Entwickeln weiterer Strategien, die Verallgemeinerung auf ausdrucksstärkere Domänen und die Anwendung der gesammelten Informationen für eine grössere Gruppe von Planungsverfahren.

References

Ambite, J. L., and Knoblock, C. A. 1997. Planning by Rewriting: Efficiently Generating High-Quality Plans. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-97)*.

Blum, A. L., and Furst, M. L. 1995. Fast planning through planning graph analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1636–1642. San Mateo, CA: Morgan Kaufmann.

Bylander, T. 1994. The Computational Complexity of STRIPS Planning. *Artificial Intelligence Journal* 69:165–204.

Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-96)*, 1194–1201. Portland, OR: Morgan Kaufmann, San Mateo, CA.

Kautz, H., and Selman, B. 1998. The Role of Domain-Specific Knowledge in the Planning as Satisfiability Framework. In *Proceedings of AIPS-98*.

Russell, S., and Norwig, P. 1995. *Artificial Intelligence, A Modern Approach*. Prentice Hall.

Scholz, U. 1997. Planning by Local Search. Master's thesis, Technische Universität Darmstadt, Fachbereich Informatik, Alexanderstraße 10, 64283 Darmstadt, Germany.

Solving Resource-Constrained Planning Tasks

Jana Koehler
Institut für Informatik
Albert Ludwigs Universität
Am Flughafen 17
79110 Freiburg
koehler@informatik.uni-freiburg.de

Abstract

This paper outlines the basic principles underlying reasoning about resources in IPP, which is a classical planner based on planning graphs originally introduced with the GRAPHPLAN system. The main idea is to deal with resources in a strictly action-centered way, i.e., one specifies how each action consumes or produces resources, but no explicit temporal model is used. This avoids the computational problems of solving general constraint satisfaction problems by using instead interval arithmetics and propagation of resource requirements over time steps in the planning graph.

1 Actions that provide, produce, and consume Resources

The starting point for the language extension is the ADL subset that is available in IPP 3.0 [7]. It offers universally quantified and conditional effects, atomic negation, equality as well as quantified and conditional goals. To reason about resources, an action[1] description is extended in the following way:

1. Following the "ordinary preconditions" (which are logical facts) *resource requirements* can be specified.

2. Effect descriptions are extended to specify which of the resource variables is *provided*, *produced* or *consumed* by the action.

3. Database query schemata allow for a compact representation of resource requirements and resource effects.

As an example, consider the following two actions that are part of our version of the *airplane* example used by the ZENO planner [9].

```
fly(?x ?y:airport)
:p at-plane(?x) ($gas ≥ distance(?x ?y)/3)
:e ADD at-plane(?y) not at-plane(?x);
   ALL ?p passenger boarded(?p)
           => ADD at(?p ?y) not at(?p ?x);
   $gas  -= distance(?x ?y)/3;
   $time += distance(?x ?y) * (3/20) .

refuel(?x:location)
:p at-plane(?x)
:e ADD full-plane();
   $gas  := 750;
   $time += -0.08*$gas+60 | 60.
```

The action **fly** specifies what happens when an airplane flies from airport x to airport y. As a precondition it needs to be at airport x and as a resource requirement there must be enough gas in the tank to fly the distance between the two airports specified with `$gas ≥ distance(?x ?y) / 3`. We allow the distinguished binary predicates $=, \leq, \geq$, which mean that the current amount of the resource variable `$gas` specified as the first argument must be either *equal*, *lower-than*, or *greater-than* the arithmetic expression following as the second argument. An arithmetic expression can contain a database query schema such as `distance(?x ?y)` together with an arithmetic term built out of the four basic mathematical operations and real numbers. When the action gets instantiated, the query schema is instantiated into a database query such as `distance(Basel Paris)` for which the specific value is found in the associated database. Together with the arithmetic term it can be simplified to a single real number yielding `$gas ≥ 200` as the resource requirement of the action instance **fly(Basel Paris)**.[2]

[1] To avoid confusion, we speak of actions instead of planning operators. The term operator is reserved to denote a mathematical operator in an arithmetic expression.

[2] For each resource variable a unit measure is assumed that remains implicit in the planner.

As an effect of the action, `$gas` is consumed (indicated by the assignment operator `-=`) and `$time` is "produced" (indicated by `+=`), i.e., it advances on a discrete scale of time units when actions are executed. Producers (`+=`) and consumers (`-=`) increase or decrease a resource relative to its current value, e.g., if `$gas` is decreased by 200 by the action **fly(Basel Paris)** and its current value is 300 then the new value of `$gas` results with 100.

Two more kinds of effects are allowed and shown in the action **refuel**. First, it assigns to `$gas` the absolute value of 750 independently of its previous value (indicated with the assignment operator `:=`), i.e., **refuel** is a *provider* of `$gas`. Second, the time for refueling depends on the gas amount that is needed by the airplane. We allow for a limited form of interdependencies between resource variables where one variable can depend on atmost another one via a linear equation of the form $y = mx + b$. Effects without resource interdependencies are are referred to as *simple*.

In the first prototype, we do not allow interdependency chains where `$x` depends on `$y`, which again could depend on `$z` etc. The action also shows an expression of the form `|NUMBER`, which represents a *worst-case value*, i.e., `$time` will at most be increased by 60 minutes (if this is the assumed unit) when refilling the maximum amount of 750.

2 Planning Problems involving Resources

The specification of a planning problem includes the usual declaration of constants and their types, i.e.,

```
passenger:  Dan Ernie Scott;
location:   Paris Basel London;
```

a declaration [dmin,dmax] of the minimum and maximum values each resource variable can take

```
$time:  [0, ∞);
$gas:   [0,750];
```

and the database information (since no connection to a real database is implemented yet) such as

```
database:  distance(Paris Basel)    600
           distance(Paris London)   400
           distance(Basel Paris)    600
           distance(Basel London)   800.
```

The specification of the initial state specifies the usual logical facts and assigns values to *all* resource variables that are involved in the planning problem

```
initial:  $time=0 $gas=300
          checked-in(Ernie) checked-in(Scott)
          at-plane(Basel) at(Ernie Paris)
          at(Dan Paris) at(Scott Basel).
```

The goal specification contains resource requirements together with logical facts that have to hold in the goal state:

```
goal:  $time ≤ 330
       at(Ernie London) at(Scott London).
```

This states the planning problem to fly two passengers from two different locations to London in less than 330 minutes, i.e., $5\frac{1}{2}$ hours.

3 Planning Graphs with Resource Time Maps

To build planning graphs, all valid instances of applicable actions are determined. The instantiation of a database query schema is a specific query returning the value from the database. This means, an instantiated action can only require a specific numerical value as a resource requirement in its preconditions such as $\$time \leq 100$ or $\$gas = 50$. Resource effects are reduced to an operator followed by a numerical value or to an operator followed by a linear equation containing one resource variable.

The first layer in the planning graph is obtained from the logical facts in the initial state specification. Together with the graph, a *resource time map* RTM is built that records the possible interval boundaries [tmin(n,$r),tmax(n,$r)] for each resource variable $\$r$ at each time step n. The initialization of the intervals at time step 0 is based on the specification of the initial state. For example `>$money` $\geq$ `200` leads to [200, $+\infty$) and `$debts` $\leq$ `10, $debts` $\geq$ `5` leads to [5,10]. This way, we can also represent uncertainty about the exact amount of a resource that is initially available.

time step	$gas	$time
0	[300,300]	[0,0]

From each action instance a (in the following denoted with action), its specific requirements for resource $\$r$ (represented as an interval [rmin(a,$r),rmax(a,$r)]) and a computational instruction for each resource that it affects are determined. If no resource requirement is specified, the action is applicable if the resource takes an arbitrary value specified by $(-\infty, +\infty)$. For the example, we obtain

action	$gas	$time
fly(B P)	p: $[200, +\infty)$	$(-\infty, +\infty)$
	e: $-= 200$	$+= 90$
fly(B L)	p: $[266.67, +\infty)$	$(-\infty, +\infty)$
	e: $-= 266.67$	$+= 120$
fly(P L)	p: $[133.33, +\infty)$	$(-\infty, +\infty)$
	e: $-= 133.33$	$+= 60$
refuel(B)	p: $(-\infty, +\infty)$	$(-\infty, +\infty)$
	e: $:= 750$	$+= -0.08 * \$gas$
		$+60 \mid 60$
board(P)	p: $(-\infty, +\infty)$	$(-\infty, +\infty)$
board(B)	e:	$+= 30$

Providers can assign arbitrary values to resources, while producers and consumers are limited to positive values. For example, one cannot declare `$gas += -500`, i.e., that an action is a producer, but the instruction would in fact decrement the value of the resource. Linear equations are allowed for providers, producers, and consumers, but also have to return positive values for the latter two action types.

Definition 1 *Given a resource requirement* $[rmin(a, \$r), rmax(a, \$r)]$ *of an action* a *wrt. a resource* $\$r$ *and the RTM interval* $[tmin(n, \$r), tmax(n, \$r)]$ *for* $\$r$ *at* n*, the resource requirement is* possibly satisfiable *at* n *iff* $[rmin(a, \$r), rmax(a, \$r)] \cap [tmin(n, \$r), tmax(n, \$r)] \neq \emptyset$.

Definition 2 *A simple resource effect* $\$r$ OP N *is* impossible *iff*

$$tmin(n, \$r) + \mathsf{N} > dmax \text{ if } \mathsf{OP} \text{ is } +=$$
$$tmax(n, \$r) - \mathsf{N} < dmin \text{ if } \mathsf{OP} \text{ is } -=$$
$$\mathsf{N} \notin [dmax(\$r), dmin(\$r)] \text{ if } \mathsf{OP} \text{ is } :=$$

Definition 3 *An action* a *is* applicable *to a fact level* n *in the graph iff*

1. *its logical preconditions are non-exclusive in* n *and*
2. *all its resource requirements are* possibly satisfiable *in* n *and*
3. *none of its simple effects is impossible.*

The exclusivity relation over pairs of actions is extended in such a way that a parallel set of actions causes the same resource effects independently of a particular linearization of the actions. This restriction eliminates all possible resource conflicts within parallel actions and leads to a unique resulting state with respect to resources independently of the execution order of the actions.

Definition 4 *Two actions are marked as* exclusive *if one of the following holds*

1. *they are exclusive wrt. their logical effects or preconditions [7]*
2. *they belong to different action types, i.e., consumers, providers, and producers are exclusive*
3. *both actions are providers of the same resource*
4. *one affects a resource variable that occurs in the linear equation of the other's effects*
5. *their combined simple effects are impossible based on Definition 2.*
6. *both have contradictory resource requirements, i.e., the intersection of the corresponding intervals for the same resource is empty.*

The next fact level in the graph is built as usual, i.e., all effects whose effect conditions can probably be made true are added to the level and the appropriate ADD and DEL edges are drawn. Logical facts are marked as exclusive as before, see [7]. Resource effects update the resource time map at the next level based on the following rules:

Definition 5 *Given* $[tmin(n, \$r), tmax(n, \$r)]$ *as the interval for a resoure* $\$r$ *at time step* n *in the RTM, to compute the new interval* $[tmin(n + 1, \$r), tmax(n + 1, \$r)]$ *for* $\$r$ *at time step* $n + 1$ *we do the following:*
(1) If no action a *affects the resource* $\$r$ *at time step* n *then* $[tmin(n + 1, \$r), tmax(n + 1, \$r)] = [tmin(n, \$r), tmax(n, \$r)]$.
(2) Otherwise, for all actions a *that affect the resource* $\$r$*: If the effect of* a *on* $\$r$ *contains a linear equation, it is replaced by the worst-case value, i.e., each resource effect is simplified to* OP $\mathsf{N}_{(a)}$.
If $\$r$ *is the distinguished resource variable* `$time`*, which can only be "produced"*

$$tmin(n + 1, \$r) = tmin(n, \$r)$$
$$tmax(n + 1, \$r) = \mathbf{MAX}_{a} [tmax(n, \$r) + \mathsf{N}_{(a)}]$$

For all other resource variables $\$r$ *we compute the interval boundaries for maximal sets of non-exclusive actions that affect a resource, i.e., given an action the search algorithm adds to it all actions that are non-exclusive and whose combined simple resource effects are possible. This set construction is repeated for all actions that occur at an action level in the graph.*
For each set s *of* k *non-exclusive* producers *of* $\$r$ *a new upper interval boundary*

$$tmax(n + 1, \$r)_s = tmax(n, \$r) + \textstyle\sum_{a=1}^{k} \mathsf{N}_{(a)}$$

For each set s *of* k *non-exclusive* consumers *of* $\$r$ *a new lower interval boundary*

$$tmin(n + 1, \$r)_s = tmin(n, \$r) - \textstyle\sum_{a=1}^{k} \mathsf{N}_{(a)}$$

For all providers a *of* $\$r$

$$tmin(n+1,\$r)_s = \underset{a}{\mathbf{MIN}}\ \mathsf{N}_{(\mathsf{a})}$$
$$tmax(n+1,\$r)_s = \underset{a}{\mathbf{MAX}}\ \mathsf{N}_{(\mathsf{a})}$$

We obtain individual interval boundaries for each maximal set of non-exclusive producers and consumers and the largest absolute increase or decrease of a resource that can be achieved when a single provider is selected.

Furthermore, the interval boundaries at time step n are propagated to time step $n+1$ to yield another pair of interval boundaries

$$tmin(n+1,\$r)_s = tmin(n,\$r)$$
$$tmax(n+1,\$r)_s = tmax(n,\$r)$$

The new interval for the resource \$r at time step $n+1$ in the RTM is now obtained as

$$[tmin(n+1,\$r), tmax(n+1,\$r)]$$
$$= [dmin(\$r), dmax(\$r)] \cap$$
$$[\underset{s}{\mathbf{MIN}}\ tmin(n+1,\$r)_s, \underset{s}{\mathbf{MAX}}\ tmax(n+1,\$r)_s]$$

The intersection between an RTM interval and a predeclared interval is never empty if it was non-empty in the initial state because RTM intervals grow monotonically and an empty intersection in the initial state is interpreted as an inconsistent planning problem specification for which no planning takes place.

Let us consider the following example: In the initial state (fact level 0), the following actions are applicable: **fly(Basel Paris)**, **fly(Basel London)**, **refuel(Basel)**, **board(Basel)**. For `$time` we only have producers which add 30, 60, 90 or 120 minutes each, i.e., at fact level 1 we end up with the interval [0,120] taking the maximal increase. The amount of gas is set to 750 by refueling, which is a provider, but flying actions are consumers decreasing the variable by either 266.67, 200 or 133.33. The *flying* actions are marked as exclusive for two reasons: first, they interfere with respect to their logical effects and second, their combined simple resource effects fall below the valid interval minimum of 0 for `$gas`. Therefore, three individual new lower bounds are obtained – one for each flying action, of which the minimum is selected. This means we have $[tmin(1,\$gas), tmax(1,\$gas)] = [0,750] \cap$ $[\mathbf{MIN}(300-200, 300-133.33, 300-266.67, 750), 750] = [33.33, 750]$

RTM:

fact level	$gas	$time
0	[300,300]	[0,0]
1	[33.33,750]	[0,120]
2	[0,750]	[0,240]
3	[0,750]	[0,360]
4	[0,750]	[0,480]

This way, the graph is expanded until the logical goals are reached for the first time without being exclusive. For the resource requirements we test if the interval from the time map and the goal interval have a non-empty intersection. In the example, both tests are satisfied at fact level 3.

4 Finding a Valid Choice of Resource-constrained Parallel Actions

Recall that only a set of non-exclusive actions can be selected at each time step when we are searching from the goal level back to the initial state. For resources, this implies that at each time step a resource can either be consumed, produced, or provided and that never a dependency resource can be changed simultaneously with the depending resource.

The planning algorithm comprises two parts: The *candidate generation*, which searches the planning graph and RTM for a plan that is *possibly* a solution to the planning problem and the *symbolic execution* that proves that the generated candidate is indeed correct.

The generation algorithm searches the planning graph level by level starting from the goals. It works in two phases: In a first action selection phase, it chooses a minimal, non-exclusive, and conflict-free set of parallel actions to achieve the logical goals which leads to resource requirements that are consistent with the individual requirements of each action and the possible values of resources as reflected in the RTM. If a conflict occurs, additional actions are added to the minimal set in order to achieve a conflict resolution. In the following, we solely concentrate on the resource-side of each action selection, how the process proceeds for the logical goals is described in [7].

At each time step $n+1$, the algorithm is given the logical goals G_{n+1} and a resource goal $[gmin(n+1,\$r), gmax(n+1,\$r)]$ for each resource variable \$r that occurs in the planning problem. It is initially called starting at the *max* time step of the graph with the logical goals that have been specified in the planning problem. If any resource goals have been specified too, they are intersected with the corresponding RTM interval to yield the resource goals at time step *max*. If no specific value for a resource is required in the goal state, then the resource goal is initialized with the value of the resource as recorded in the RTM at time step *max*. Action selection proceeds as follows:

(1) We start with an initialization: the set of selected actions Δ_n is initially empty and all new goals at time step n for all resources are equal to the goals at time step $n+1$.

$\Delta_n = \emptyset$

$\forall\ \$r : [gmin(n, \$r), gmax(n, \$r)] :=$
$[gmin(n+1, \$r), gmax(n+1, \$r)]$

(2) An action a is selected that achieves a goal from G_{n+1} and that is non-exclusive to the actions already contained in Δ_n and $\Delta_n = \Delta_n \cup \{a\}$. If no selection is possible the search algorithm backtracks to action level $n+1$.

(3) For each resource \$r, the following tests are performed with the selected action a and the resource effects of a to update the resource goals:

if a does not affect \$r: do nothing
elsif a is provider of \$r
then if $gmin(n+1, \$r) \leq \mathsf{N} \leq gmax(n+1, \$r)$
then $gmin(n, \$r) := -\infty$
$gmax(n, \$r) := +\infty$
else backtrack to select a new action
else
a is simple-effect producer/consumer
and $\$r \neq \$time$:
$gmin(n, \$r) := gmin(n, \$r)\ \overline{\mathsf{OP}}\ \mathsf{N}_{(a)}$
$gmax(n, \$r) := gmax(n, \$r)\ \overline{\mathsf{OP}}\ \mathsf{N}_{(a)}$

a is simple-effect producer and $\$r = \$time$:
$gmin(n, \$r) := gmin(n+1, \$r) - \mathbf{MAX}_{a \in \Delta_n}\ \mathsf{N}_{(a)}$
$gmax(n, \$r) := gmax(n+1, \$r) - \mathbf{MAX}_{a \in \Delta_n}\ \mathsf{N}_{(a)}$
a is not a simple-effect producer/consumer (effect contains linear equation): do nothing

with $\overline{\mathsf{OP}}$ being the inverse operator to OP, i.e., we use $+$ instead of $-$ and vice versa. $\mathsf{N}_{(a)}$ is the numerical value from the simple effect. For example, if the old goal was $[100, +\infty)$ and the effect is -= 50 then we obtain $[100 + 50, +\infty + 50)$ which yields the new goal interval $[150, +\infty)$.

(4) This way, actions are selected until a minimal action set is found that achieves all logical goals. The updated resource goals for the current choice of actions are tested against the resource requirements of each action and the RTM intervals at time step n. A test against the predeclared interval boundaries is not needed as these are already reflected in the RTM intervals.

Test (1):
$\exists\ \$r, a \in \Delta_n : [gmin(n, \$r), gmax(n, \$r)] \cap [rmin(a, \$r), rmax(a, \$r)] = \emptyset$

Test (2):
$\exists\ \$r : [gmin(n, \$r), gmax(n, \$r)] \cap [tmin(n, \$r), tmax(n, \$r)] = \emptyset$

If the tests fail, i.e., all intersections are non-empty, the final new resource goals for this minimal action set are obtained by actually determining the intersection of the goals with all resource requirement intervals of all actions in Δ_n and the RTM interval:

$[gmin(n, \$r), gmax(n, \$r)] :=$
$[gmin(n, \$r), gmax(n, \$r)] \cap$
$\bigcap_{a \in \Delta_n} [rmin(a, \$r), rmax(a, \$r)] \cap$
$[tmin(n, \$r), tmax(n, \$r)]$

Together with the new logical goals resulting from the preconditions and effect conditions of selected actions, these resource goals are forwarded to the next level of the planning graph.

The algorithm terminates successfully in the initial state if the interval that specifies how much of a resource is initially available is a subinterval of the resource goal for the initial state, i.e., if $[tmin(0, \$r), tmax(0, \$r)] \subseteq [gmin(0, \$r), gmax(0, \$r)]$ for all $\$r$.

In the case of linear equations, the resource effect of an action is ignored because it is unclear what the specific impact on the resource would be until a complete plan is found and the specific value of the dependency variable in a particular state is fixed. Only its resource requirements are considered by the planner because these must be satisfied by any solution plan. The planner could use the worst-case estimate, but this would make the planning process much more complicated if completeness needs to be retained.

If any of the tests succeeds for a minimal set of actions, the planner needs to resolve resource conflicts by adding more actions such that the new goals of the final action set will fall within predeclared interval boundaries and the recorded RTM intervals, and are consistent with the resource requirements of each action. Depending on why a test was successful, a conflict resolution policy is determined:

(A) If $gmax(n, \$r) < T$ where T stands for the interval against which the goals are tested, then a consumer or provider of \$r is added to Δ_n in the *next* selection.

(B) If $gmin(n, \$r) > T$ then a producer or provider of \$r is added to Δ_n in the *next* selection.

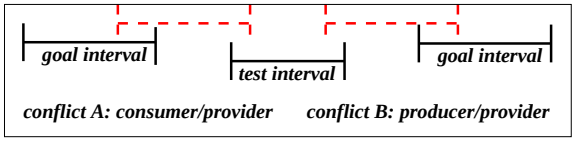


Figure 1: Pushing goal intervals to obtain a non-empty intersection with the test interval.

This means that it is no longer sufficient to restrict the search space of the planner to action sets

that are minimal wrt. the logical goals. It is easy to see that there can be actions that must solely be selected to achieve resource goals, but are not necessary at all to achieve the logical goals. If actions exist in the graph with only resource effects and without logical effects, several copies of the same action can be selected for parallel execution. For example if there is an action

get-sheet-of-paper
```
:p at-paper-tray
:e $sheets  += 1.
```

taking 10 sheets can be done in parallel. This leads to more parallelism in plans and in the case that temporal restrictions have to be met, it can be the only way how a solution to a planning problem becomes possible.

To achieve completeness, all possible minimal action sets that achieve a set of logical goals are generated and for each of the sets all possibilities to add conflict resolvers have to be explored. The algorithm returns the first conflict-free set of actions that it finds, which does not need to be the smallest set in terms of the number of actions. Furthermore, if a conflict cannot be resolved at time step n then the planner has to explore if it can prevent the conflict at levels $i \geq n$, i.e., a backtracking process tries if adding actions at later levels in the plan and then searching forward again resolves the conflict. The conflicting resource variable is stored in a separate data structure together with the policy to resolve it (i.e., the type of action that has to be considered). When the planner returns to an action level, it does not immediately consider a new set of actions, but tries to extend the current set by one action such that at least one conflict from the agenda is reduced or resolved. Then it searches forward again. This way, all maximal possible sets of non-exclusive actions are considered for a given goal, before another minimal action set is considered.

The necessity to explore all possibilities for conflict resolution by addition of actions at various levels in the graph leads to a large search space that is only limited by the rather strict exclusivity relations that we formulated over actions, the number of actions at each time step in the planning graph, and the depth of the graph - more precisely the number of actions levels between max and the level n where the conflict was detected. If no linear equations involving resource dependencies occur in the choosen actions then this algorithm is a sound and complete planner. Otherwise, the symbolic execution phase needs to verify that a generated plan is indeed a solution.

If the complete search over a graph of depth t has failed in generating valid plans, the planning graph and RTM are extended to depth $t+1$ and the planner searches again the extended graph.

In the airplane example, action selection proceeds as follows: We start with the goal intervals for `$time` $(0, 330]$ and `$gas` $(0, 750)$ at fact level 4 (the planner has unsuccessfully tried to search a plan starting from level 3, i.e., the planning graph and RTM were extended by one more level). To achieve the goals `at(Ernie London)` and `at(Scott London)` the action **fly(Paris London)** is selected at time step 3, which leads to the following goals at fact level 3:

f 3: `$time` $\in$ `[0,270] $gas` $\in$ `[133.33,750] boarded(Ernie) boarded(Scott) at-plane(Paris)`

For `$time` the planner computes $(0 - 60, 330 - 60]$ intersected with the resource requirements $(-\infty, +\infty)$ and the RTM interval at fact level 3 $[0, 360]$. For `$gas` the planner computes $(0 + 133.33, 750 + 133.33)$, which gets intersected with the resource requirements $[133.33, +\infty)$ and the RTM interval $[0, 750]$.

To fly Ernie and Scott from Paris to London both have to be on board the aircraft being at Paris. Flying, boarding, and refueling are exclusive, but refueling and boarding can be done in parallel. Let us assume that the planner selects the action **board(Paris)** to have Ernie entering the aircraft, while `boarded(Scott)` and `at-plane(Paris)` are added by a no-op. Boarding takes 30 minutes and consumes no gas, i.e., the new time goal is $[0 - 30, 270 - 30]$ intersected with $(-\infty, +\infty)$ and $[0, 240]$.

f 2: `$time` $\in$ `[0,240] $gas` $\in$ `[133.33,750] checked-in(Ernie) at(Ernie Paris) at-plane(Paris) boarded(Scott)`

No-ops are selected to achieve all subgoals except `at-plane(Paris)` that is achieved by the action **fly(Basel Paris)** which requires 200 amounts of gas and takes 90 minutes time, i.e., the planner gets for `$time` the intersection of $[0 - 90, 240 - 90]$ with $(-\infty, +\infty)$ and $[0, 120]$. For `$gas` it intersects $[133.33 + 200, 750 + 200]$ with $[200, +\infty)$ and $[33.33, 750]$ which yields $[333.33, 750]$.

f 1: `$time` $\in$ `[0, 120] $gas` $\in$ `[333.33,750] checked-in(Ernie) at(Ernie Paris) at-plane(Basel) boarded(Scott)`

Finally, Scott needs to board in Basel before the airplane takes off for Paris, i.e., the planner selects **board(Basel)** as the action that will be executed in the initial state. For `$time` the new goal interval in the initial state is $[0 - 30, 120 - 30]$ intersected with $(-\infty, +\infty)$ and $[0, 0]$, which yields

$[0,0]$. For `$gas` the planner intersects $[333.33, 750]$ with $[300, 300]$, which is empty. This alerts the planner that a resource conflict has occurred on `$gas`. Since the goal interval lies "right" of the RTM interval, a producer or provider has to be considered for conflict resolution. The refueling action is the only provider of `$gas` and fortunately, it is non-exclusive of boarding and satisfies the test $333.33 \leq 750 \leq 750$. The new goal for this resource is recomputed as $[-\infty, +\infty]$, which is intersected with $[300, 300]$ to yield the resource goal in the initial state. Obviously, this goal interval coincides perfectly with the amount of gas that is available in the initial state.

For `$time` we do not need to recompute the new goal because the new producer changes it depending on a linear equation. All goals at time step 0 are supersets of the resource intervals in the initial state and the planner terminates with a plan candidate. But since a non-simple resource effect is used by the plan, this candidate has to be verified by a subsequent symbolic execution phase.

fl 0: `$time[0,0] $gas[300,300]`
`checked-in(Ernie) at(Ernie Paris)`
`at-plane(Basel) checked-in(Scott)`

5 Verification of Plan Candidates through Symbolic Execution

During graph search and action selection, effects based on linear equations are ignorered, because the specific impact of the effect can only be correctly determined after the exact value of the dependency variable is known. But this becomes possible only after a complete plan has been generated, i.e., as soon as the planner selects an action with a non-simple resource effect, soundness can be affected and a verification of the plan candidate becomes necessary.

To avoid the symbolic execution one had to generate a plan that is sound wrt. best-case and worst-case estimates, i.e., the linear equations had to be equipped with two estimates that have to be considered during planning. For some practical applications this could be useful, but in the general case it would rule out too many solutions.

The symbolic execution takes as input a plan and the initial and goal specification for resource variables and returns true if and only if the plan is a solution to the given planning problem.

In the initial state, an interval is specified for each resource, which can also be infinite. If no interval is given for a resource, it is initialized with the predeclared interval boundaries. Furthermore, the planner tests if all initial intervals are subintervals of the predeclared intervals, otherwise, the planning problem is rejected as inconsistent.

The plan candidate determines which actions are to be executed in each state and with that we know which resource effects will occur because all actions are executable - their logical preconditions are guaranteed to be satisfied in the state and resource effects are unconditional. The resource effects in each action specify which resource variable they affect and a computational instruction says how the resource variable will change. Furthermore, each action has atmost one effect per resource.

Since parallel action sets have to be non-exclusive, each possible linearization has the same impact on resources except on $time for which we assume true parallel execution. This means, times do not sum up, but only the maximal use of time determines the total advance on the time scale that is caused by a parallel set of actions.

The symbolic execution algorithm takes the intervals from the current state and the resource effects of all actions in a parallel action set as input and determines the new resulting state. It is based on the semantics of resources as defined in [6] and tests if each new state is valid and if the final resulting state satisfies the goals.

The symbolic execution itself causes only a marginal computational overhead, but the necessity to construct a complete plan before execution can start leads to a much larger search space. Therefore, resource dependencies should be used rather sparsely in actions whenever possible.

As an example, let us consider the airplane plan with the tighter time requirement `$time` $\leq$ `215` as a goal. This would lead to the following resource goals at each time step:

time	time	gas	actions
0:	$[0, 0]$	$[300, 300]$	board(B) refuel(B)
1:	$[0, 35]$	$[333.33, 750]$	fly(B P)
2:	$[0, 125]$	$[133.33, 750]$	board(P)
3:	$[0, 155]$	$[133.33, 750]$	fly(P L)
4:	$[0, 215]$	$[0, 750]$	

This plan ignores the increase in time caused by the refuel action and is therefore incorrect. Symbolic execution reveals this problem because refueling requires 36 minutes and therefore a violation of the time requirements at fact level 1 and all subsequent levels occurs.

time	time	gas	actions
0:	0	300	board(B) refuel(B)
1:	36	750	fly(B P)
2:	126	550	board(P)
3:	156	550	fly(P L)
4:	216	416.67	

Failures during execution could be analyzed to derive information that guides the subsequent search process for a new plan candidate. However, such failures can have many possible sources and offer a huge range of solution possibilities. It is therefore not obvious how to derive control information. In the example, one can only conclude that refueling this specific amount of gas violates all time requirements in subsequent states. Thus, one could try to reduce gas consumption, or try to refuel in a different way, or relax temporal constraints caused by *flying* actions, etc. All these possibilities are explored during the systematic search of the planning graph and it seems to be impossible to decide in advance which of them should be favored.

6 Conclusion

This paper describes the first attempt of dealing with resources in IPP. The approach is strictly action-centered and follows the classical planning paradigm, i.e., a logical goal is achieved by constructing a valid plan, but resources are no longer unlimited. In this formalism, we cannot specify that a resource has a certain value at a specific time point because no explicit time model is available as for example in [1, 10, 9, 3, 4, 8, 2]. We decided to adopt this limitation in order to avoid the necessity to solve general constraint satisfaction problems during planning since the available algorithms are excellent satisfiability checkers, but usually fail in their generative capabilities, i.e., in actually constructing solution plans. First empirical test with the current prototype show quite good runtime behavior of our algorithm. For example, the airplane problem as described in this paper is solved by IPP in 0.02 seconds (without any user interaction) compared to 2-3 minutes for ZENO with user-guided goal selection.

We allow for a limited form of temporal reasoning by treating time as a distinguished resource. Actions produce time, but they are still treated as instantaneous in the planning graphs. It has to be noted that this can be adequate only for some applications. We consider it as a preliminary step towards an extension of planning graphs to model action duration.

Currently, we are concentrating on the development of an efficient termination test for unsolvable problems and techniques to further speed up the planner.

References

[1] J. Allen, H. Kautz, R. Pelavin, and J. Tenenberg. *Reasoning about Plans*. Morgan Kaufmann, USA, 1991.

[2] B. Drabble and A. Tate. The use of optimistic and pessimistic resource profiles to inform search in an activity based planner. In Hammond [5], pages 243–248.

[3] A. El-Kholy and Barry Richards. Temporal and resource reasoning in planning: the parcPLAN approch. In W. Wahlster, editor, *Proceedings of the 12th European Conference on Artificial Intelligence*, pages 614–618. John Wiley & Sons, Chichester, New York, 1996.

[4] M. Ghallab and H. Laruelle. Representation and control in IxTeT, a temporal planner. In Hammond [5], pages 61–67.

[5] K. Hammond, editor. *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems*. AAAI Press, Menlo Park, 1994.

[6] J. Koehler. Planning under resource constraints in IPP. Technical report, University of Freiburg, 1998. forthcoming.

[7] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an ADL subset. In S. Steel, editor, *Proceedings of the 4th European Conference on Planning*, volume 1348 of *LNAI*, pages 273–285. Springer, 1997.

[8] P. Laborie and M. Ghallab. Planning with sharable resource constraints. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1643–1649. Morgan Kaufmann, San Francisco, CA, 1995.

[9] J. Penberthy and D. Weld. Temporal planning with continuous change. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence*, pages 1010–1015. AAAI Press, MIT Press, 1994.

[10] D. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, San Francisco, 1988.

Modellierung paralleler kontinuierlicher Aktionen von autonomen mobilen Robotern

Henrik Grosskreutz und Michael Beetz
Institut für Informatik III, Universität Bonn,
Römerstr. 164, 53117 Bonn,
email: {grosskre,beetz}@cs.uni-bonn.de

Abstract

Die Fähigkeit von Servicerobotern, das eigene Verhalten beim Ausführen von Plänen vorherzusagen, ist eine wesentliche Grundlage für die Entwicklung von autonomen, ihr eigenes Vorgehen planenden Robotern. Wir beschreiben ein Verfahren zur automatischen Generierung von kausalen Modellen des Verhaltens des mobilen Roboters RHINO bei der Ausführung von Bürobotenaufträgen. Die Modelle werden aus einer sehr kompakten Beschreibung des dynamischen Systems, das Roboter und Umgebung bilden, generiert. Sie berücksichtigen die zeitliche Struktur der kontinuierlichen Prozesse, mögliche wechselseitige Beeinflußung zwischen kontinuierlichen Prozessen, sowie Unsicherheit über den Zustand der Welt. Dies ermöglicht es dem transformationellen Planer XFRM(McD92; BM94), Fehlverhalten wie das Überschreiten von Deadlines vorherzusagen und durch Plantransformation zu vermeiden.

Einleitung

Viele autonome Roboter operieren in nicht vollständig bekannten, sich verändernden Umgebungen. Sie arbeiten dabei nicht mit perfekten Sensoren und Effektoren, und müssen mit nicht vorhersehbaren Veränderungen ihrer Umgebung zurechtkommen. Um trotz unvollständiger Information und nichtdeterministischen Effekten zuverlässig und effektiv zu operieren, müssen die Steuerungssysteme eines Roboters, der in einer solchen Umgebung arbeitet, flexibel auf asynchron eintreffende Sensorinformationen reagieren und kleinere Ausführungsfehler lokal beheben. So muß ein Navigationsprozess auf kleinere Hindernisse reagieren, indem er um diese herumfährt, anstatt mit einem Fehler abzubrechen.

Um solche Steuerungsprogramme (siehe (BBFC98)) effektiv generieren und revidieren zu können müssen KI-Planungssysteme berücksichtigen, daß Roboteraktionen keine deterministischen Effekte haben, daß die Wirkung einer Aktion oft nicht unmittelbar eintritt und daß Prozesse Auswirkungen auf andere, parallel ablaufende Prozesse haben können. Die Verwendung eines Modells, das auf atomaren Aktionen ohne zeitliche Ausdehnung und mit deterministischen Effekten basiert und Pläne als partiell geordnete Menge von atomaren Aktionen darstellt, ist daher für derartige Umgebungen oft nicht adäquat. Modelle, die diesen Eigenschaften Rechnung tragen, müssen zeitliche Strukturen, kontinuierliche Aktionen, exogene Ereignisse, gegenseitige Beeinflussung von Prozesssen, unvollständige Weltbeschreibungen und passive Sensoren berücksichtigen.

Ziel unserer Forschung ist die Modellierung der Steuerung von autonomen mobilen Robotern am Beispiel von RHINO, einem RWI B21 Roboter. Aufgabe von RHINO ist es, in einer Büroumgebung (siehe Abb. 1) als Bürobotenroboter Briefe zwischen verschiedenen Orten zu transportieren. Dabei kann RHINO über EMails sowohl neue Aufträge entgegennehmen, als auch Informationen über Veränderungen der Umgebung (z.B. das Schließen einer Tür) erhalten. Unser Ziel ist es, aussagekräftige Vorhersagen über das Verhalten von RHINO zu erhalten. Mit Methoden des transformationellen Planens kann dann in einem weiteren Schritt, den wir hier nicht behandeln, gegebenenfalls eine Korrektur des Plans vorgenommen werden. Anstatt von atomaren Aktionen auszugehen, verwenden wir in unserem Modell kontinuierliche Prozesse wie Navigation, Positions- und Türöffnungswinkelbestimmung als Basis der Interaktion mit der Umgebung. Diese Prozesse haben eine zeitliche Ausdehnung und können zu einer kontinuierlichen Veränderung von bestimmten Zustandsvariablen, wie z.B. der Roboterposition, führen. Wir unterscheiden dabei zwischen kontinuierlichen Prozessen, die Effektoren - wie z.B. Servomotoren - steuern, und kontinuierlichen Prozessen, die Sensoren auslesen und die gelesenen Daten interpretieren. Diese nennen wir im folgenden Perzeptions-Prozesse. Die kontinuierlichen Prozesse

werden durch ein strukturiertes reaktives Programm (SRP) parametrisiert, gestartet und beendet.

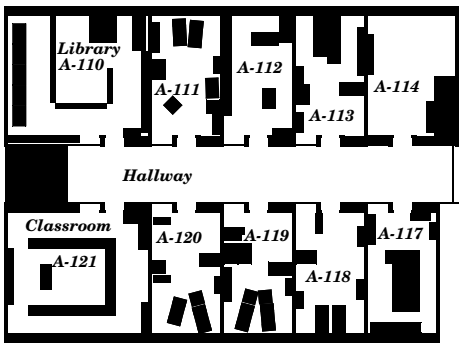


Abb. 1: Die RHINO-Umgebung

Nebenläufige Prozesse

Betrachten wir folgendes Beispiel für eine Situation, in der es zu gegenseitiger Beeinflussung durch nebenläufige Prozesse kommen kann. RHINO erhält den Auftrag, Briefe der Reihe nach zu verteilen. Die Empfänger befinden sich in den Räumen A-118, A-117, A-111 und A-120. Der Brief nach A-111 soll bis 13:30 abgeliefert werden. Zunächst wird ein SRP generiert, bestehend aus Routineplänen zum Anfahren der Räume sowie einem Überwachungsprozess, der prüft, ob die Annahmen über den Zustand der Welt noch gelten. Eine Standardannahme der Routinepläne ist, daß alle Türen offen sind. Ist dem Planungssystem bekannt, daß die Tür zu Raum A-118 geschlossen ist, so wird der Plan modifiziert. Er besteht nun aus drei parallel auszuführenden Unterplänen. Der erste sorgt dafür, daß die Räume A-117, A-111 und A-120 angefahren werden. Der zweite zeigt nur Wirkung, wenn erkannt wird, daß A-118 offen ist. In diesem Fall sorgt er dafür, daß in Raum A-118 gefahren und der entsprechende Brief dort abgelegt wird. Dieser zweite Prozess hat eine höhere Priorität als der erste, er kann den ersten Prozess gegebenenfalls unterbrechen. Zudem sorgt ein dritter Prozess dafür, daß Veränderungen der Öffnungswinkel der nahegelegenen Türen registriert werden. Das SRP des Roboters kann durch folgenden Pseudocode beschrieben werden:

```
execute concurrently
    whenever (near door)
        estimate door angle every 10 seconds
    with-high-priority
        as soon as known (door A-118 open)
            deliver mail to A-118
    execute-in-order
        deliver mail to A-117
        deliver mail to A-111 before deadline 13:30
        deliver mail to A-120
```

Abb. 2 zeigt die Trajektorie, die der Roboter bei der Ausführung des Plans abfährt (durchgängige Linie), und die relevanten Ereignisse, die bei der Projektion durch das Planungssystem mit Hilfe der nachfolgend vorgestellten kausalen Modelle vorhergesagt wurden (Kreise mit Zahlen). Hierbei beginnt RHINO gegen 13:20 mit der Ausführung, und die Tür zu A-118 wird um 13:25 wieder geöffnet. Nachdem RHINO den Raum A-117 (Punkt 2) angefahren hat, steuert er als nächstes A-111 an. Auf dem Weg dorthin stellt er fest, daß die Tür zu A-118 offen ist (Punkt 3), und liefert zuerst den Brief nach A-118 ab (Punkt 4), da dieser eine höhere Priorität hat. Erst danach wird der Brief an A-111 abgeliefert (Punkt 5). Dies führt zum Überschreiten der Deadline. Dieses Fehlverhalten sollte das Planungssystem prognostizieren, wenn es die Information erhält, daß die Tür zu A-118 demnächst geöffnet wird.

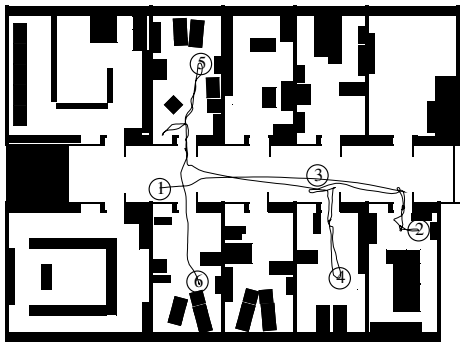


Abb. 2: Die Trajektorie des Roboters

Wie wir aus dem Bild erkennen können, stimmt die symbolische Vorhersage und die Ausführung stark überein. Ferner enthält die symbolische Projektion alle Ereignisse, wie das Passieren von Türen, verlassen von Räumen etc. die auf die Ausführung des Plans einen Einfluß haben.

Ein anderes Beispiel ist der Erhalt der Nachricht: “Es steht ein Tisch hinter der Tür zu A-120”. Dies sollte dazu führen, daß ein möglicher Zusammenstoß

mit dem Tisch vorhergesagt wird, weil RHINO seine Sonarsensoren vor Türen abschaltet. Um derartiges vorherzusagen, muß ein Modell des Systems kontinuierliche Aktionen, exogene Ereignisse, gegenseitige Beeinflussung von Prozesssen, und passive Sensoren beinhalten.

Modellierungsprinzipien

Das Verhalten des Roboters, seiner Kontrollprogramme und der Umgebung kann modelliert werden als ein Kontroll-Prozess, der ein dynamisches System, die Umgebung, kontrolliert (siehe Abb. 3). Der Zustand des Systems wird von Perzeptions-Prozessen über verschiedene Sensoren geschätzt. Der Roboter kann über seine Effektoren Zustandsvariablen verändern. Die Kommunikation zwischen Perzeptions-Prozessen und SRP geschieht über sogenannte Fluenten. Dies sind Variablen, die eine Veränderung ihres Wertes signalisieren, und dadurch zur Prozesssynchronisation und -kommunikation verwendet werden können.

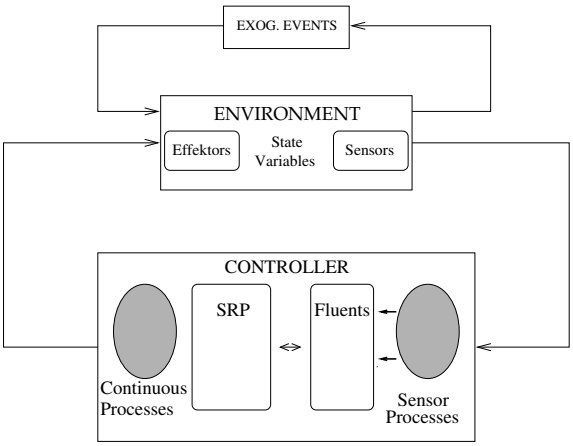


Abb. 3: Das allgemeine Prozess-Modell

Damit der Planer in der Lage ist, Fehlverhalten des Roboters vorherzusehen und rechtzeitig Korrekturen am Plan vorzunehmen, müssen die Vorhersagen über das Verhalten des Roboters effizient und kompakt generiert werden. Die Kompaktheit impliziert, daß nur die wesentlichen Ereignisse vorhergesehen werden, um den Suchraum nicht unnötig zu vergrössern, den ein Planer, der diese Vorhersage erhält, betrachtet. Damit ist es insbesondere bei kontinuierlichen Prozessen nicht möglich, die Auswirkungen zu jedem Zeitpunkt zu vermerken. Dennoch kann nicht ohne weiteres ein einfacheres Modell der Umgebung verwendet werden, da sonst die Wechselwirkung zwischen den Prozessen nicht mehr erfaßt würde. Wir machen daher einige Annahmen über das System sowie über den Roboter-Controller, die es uns ermöglichen nur wenige relevante, qualitative Veränderungen zu projizieren.

1. Die Welt wird außer durch die Effektoren des Roboters nur durch das Eintreten bekannter exogener Ereignisse verändert. Wir betrachten bislang nur Klassen von nicht-kontinuierlichen exogenen Ereignissen.

2. Kontinuierliche Veränderungen, die die Zustandsvariablen SV_1 bis SV_n verändern, induzieren eine stetige Kurve im Raum $\bigotimes_{i=1}^{n} SV_i$. Diese Kurve kann approximiert werden durch einen Polygonzug. Die Eckpunkte dieses Polygonzugs nennen wir $steps$.

3. Eine kontinuierliche Veränderung führt dann und nur dann zu einer Veränderung der Konfiguration der kontinuierlichen Prozesse, wenn während des Übergangs von $step_i$ zu $step_{i+1}$ Fluenten getriggert werden.

4. Alle Veränderungen des Systems, auf die das SRP reagiert, werden von Sensor-Prozessen festgestellt und dem SRP über Fluenten mitgeteilt.

Die Bedeutung dieser Annahmen wollen wir nochmals kurz hervorheben:

- Die Synchronisation von Prozessen darf nur über Fluenten ablaufen. Ein aktives Warten auf eine Bedingung wird nicht berücksichtigt.

- Wir modellieren nicht sowohl die Sensoren als auch die davon abhängenden Perzeptions-Prozesse jeweils getrennt, sondern verwenden nur ein Modell, das die Fluent-Ausgabe des logischen Sensors in Abhängigkeit der Zustandsvariablen beschreibt.

Unter diesen Annahmen ist es nun möglich, das System und den Controller auf einem recht hohem Abstraktionsniveau zu spezifizieren. Dazu sind folgende Angaben nötig:

1. Die Menge der Zustandsvariablen des Systems.

2. Zu jedem Perzeptionsprozess die gemessenen Zustandsvariablen, die Fluenten die der Prozess verändert sowie ein Sensormodell, daß die Schätzfunktion von den Zustandsvariablen auf die Fluenten sowie die Umkehrabbildung der Schätzfunktion beinhaltet.

3. Zu jedem kontinuierlichem Prozess die Menge der veränderten Zustandsvariablen, sowie eine Funktion, die eine Parametrisierung des Prozesses auf einen Polygonzug im Raum der Zustandsvariablen abbildet.

Modell des RHINO-Systems

Betrachten wir ein konkretes Beispiel. Wir wollen einen Teil des Navigationssystems eines Bürobotenroboters beschreiben, das auf RHINO realisiert ist. Das System besitzt einen Kontrollprozess zur Navigation. Diesem wird eine Folge von Zielpunkten übergeben, wobei der letzte Punkt das tatsächliche Ziel ist, während die anderen Punkte lediglich Zwischenpunkte sind. Die Navigation zwischen diesen Punkten erfolgt mit Hilfe eines Pfadplaners (TBB+98). Das System benutzt ein Verfahren zur passiven Schätzung der Roboterposition in der Umgebung (BFHS96; BFH97). Außerdem besitzt es einen probabilistischen Schätzer für den Öffnungswinkel von Türen, der immer dann aktiviert wird, wenn RHINO an einer Tür vorbeikommt. Das Ziel ist, ein kausales Modell zu generieren, das qualitativ vorhersagen kann was passiert, wenn der Roboter Pläne wie den obigen ausführt. Das konkrete Systemmodell für dieses Steuerungssystem zeigt Abb. 4.

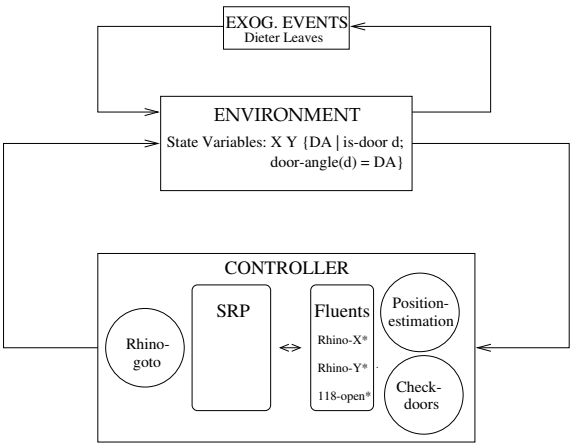


Abb. 4: Das konkrete Prozess-Modell

Unser Planungssystem erwartet eine Spezifikation des dynamischen Systems sowie der verschiedenen zugehörigen Prozesse. Diese erfolgt in diesem Beispiel folgendermaßen:

```
(def-dynamical-system Rhino-Umgebung
    (:state-vars (x "X-Position des Roboters")
                 (y "Y-Position des Roboters")
                 (:static map "Karte der Umgebung")
                 (:initializer rhino-sv-init))
    (:state-functions
                 (door-angle "Tür-Öffnungswinkel")
                 (:initializer rhino-sf-init))
    (:sensor-processes
                 position-estimation
                 door-angle-estimation)
    (:control-processes rhino-goto)
    (:exogeneous-events person-arrives))
```

Die Spezifikation enthält die Systemvariablen sowie Angaben welche kontinuierlichen Prozesse und Perzeptionsprozesse das SRP verwendet. Um die Angaben zum System möglichst kompakt zu halten, verwenden wir zudem *state-functions*. Dies sind Funktionen, die Bezeichner, in unserer Umgebung beispielsweise Türnamen, auf nicht direkt benannte Zustandsvariablen abbilden. Außerdem wird der Wert der Zustandsvariablen zu Beginn der Projektion über bestimmte Funktionen, genannt *initializer*, berechnet.

Neben dem Gesamtsystem müssen noch die einzelnen Prozesse spezifiziert werden. Der Steuerprozess RHINO-GOTO wird wie folgt spezifiziert:

```
(def-continuous-process rhino-goto (dest-descr)
    (:modifies x y)
    (:polyline compute-goto-steps
               "Abb: Zielbeschreibung → Trajektorie")
    (:start-esc start-nav)
    (:stop-esc stop-nav))
```

Der Prozess RHINO-GOTO erhält als Argument eine Beschreibung des anzufahrenden Ziels. Zum Erreichen dieses Ziels führt der Prozess eine kontinuierliche Veränderung der Zustandsvariablen x und y durch. Den Polygonzug, den diese Veränderung im Raum $x \times y$ beschreibt, liefert die Funktion compute-goto-steps, parametrisiert durch die Zielbeschreibung. Zudem werden zu Beginn und am Ende des Prozesses die Funktionen start-nav und stop-nav wegen ihrer Seiteneffekte aufgerufen.

Diese Spezifikation wird in eine Menge von ca. 12 Projektionsregeln (siehe (McD94)) transformiert, die dafür sorgen, daß nur an den Eckpunkten (bzw. $steps$) des Polygonzugs Ereignisse generiert werden. Diese führen eine Aktualisierungen der Fluenten auf der Grundlage der neuen Werte der Zustandsvariablen durch. Zwischen solchen $steps$ wird lediglich die Ableitung der Zustandsvariablen nach der Zeit vermerkt, so daß deren zwischenzeit-

licher Wert gegebenenfalls rekonstruiert werden kann.

Perzeptionsprozesse wie der Positionsschätzprozess POSITION-ESTIMATION werden folgendermaßend spezifiziert:

```
(def-sensor-process position-estimation
    (:output-fluents (rhino-x* measures x)
                     (rhino-y* measures y))
    (:sensor-estimation loc-estimation)
    (:sensor-trigger loc-trigger))
```

POSITION-ESTIMATION schätzt die Zustandsvariablen x und y, und liefert seine Ergebnisse über die Fluenten RHINO-X* und RHINO-Y* an das SRP. Die tatsächliche Schätzfunktion, angewendet auf die evtl. fehlerhaften Sensordaten, wird durch die Funktion loc-estimation approximiert. loc-trigger ist die Umkehrkunktion von loc-estimation.

Auch hier wird die Spezifikation in eine Menge von Projektionsregeln transformiert (ca. 20). Diese führen im wesentlichen zu folgendem Verhalten:

1. Immer wenn Threads blockieren, weil sie auf einen Fluenten warten, werden die Bedingungen des Fluenten an die Zustandsvariablen extrahiert. Dazu wird zunächst Bestimmt, von welchen Zustandsvariablen $sv_1, ..., sv_m$ der Fluent abhängt. Dann werden diejenigen Regionen im Raum $\bigotimes_{i=1}^{m} SV_i$ berechnet, in denen der Fluent seinen Wert ändert. Diese werden als *trigger-areas* vermerkt.

2. Vor jeder kontinuierlichen Veränderung von $step_i$ zu $step_{i+1}$ wird bestimmt, ob es zu einem Schnitt von einer *trigger-area* mit der durch die $steps$ induzierten Strecke durch $\bigotimes_{i=1}^{m} SV_i$ kommt. Wenn ja, so wird am Schnittpunkt ein *passive-sensor-update* Ereignis generiert. Dieses sorgt dafür, daß die Fluenten aktualisiert werden, was zur Befreiung von blockierten Threads führt.

Die so generierten Projektionsregeln bilden dynamisch in Abhängigkeit der Prozess-Konfiguration ein kausales Modell der kontinuierlichen Prozesse. Das kausale Modell beachtet nur die Zeitpunkte, an denen sich die Ableitung der Zustandsvariablen ändert, oder an denen die Prozesskonfiguration sich durch Blockieren oder Befreien von Threads verändern kann. Die Zeiträume zwischen solchen Punkten werden übersprungen. Diese Modelle liefern damit Vorhersagen, die nur qualitative Veränderungen des Systems und des Controllers liefern, und somit relativ kompakt bleiben.

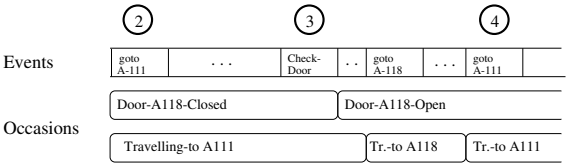


Abb. 5: Die Timeline

Realisierung und Evaluierung

Unsere Modelle bauen wir im Rahmen des transformationellen Planers XFRM(McD92; McD94) auf. Dieser bietet eine sehr mächtige Modellierungssprache für dynamische, unvollständig bekannte Umgebungen. Die SRP sind in RPL (McD91) geschrieben, einer Lisp-ähnlichen Roboter-Kontroll-Sprache, die Schleifen, bedingtes Verhalten, Variablen, prozedurale Abstraktion, parallele Prozesse sowie Synchronisationsmethoden zwischen diesen bereitstellt.

Unser Ansatz ermöglicht es, das dynamische System, das Roboter und Umgebung bilden, auf einer recht hohen Abstraktionsebene zu beschreiben. Ausgehend von dieser Beschreibung werden kontextabhängige XFRM-Modelle des Systems generiert (BG98). Diese berücksichtigen Unsicherheit im Weltwissen, kontinuierliche Veränderung des Systems durch kontinuierliche Prozesse, passive Sensoren und die Wechselwirkungen zwischen Perzeptions-Prozessen, SRP's und kontinuierlichen Prozessen.

Diese Modelle ermöglichen es, Projektionen der SRP zu generieren die wesentliche Ereignisse, die bei der Ausführung des SRP eintreten, vorhersehen (siehe Abb. 5). Die Projektionen bilden eine ausreichend genaue Vorhersage des Verhaltens des Systems um Fehler wie Deadlineüberschreitungen oder Zusammenstöße mit Hindernissen nach dem Abschalten der Sonarsensoren vorhersagen zu können, und ermöglichen es damit dem Planungssystem, diese Fehler zu beheben (BM94). Uns sind keine anderen KI-Planungsysteme bekannt, die diese Aussagekraft haben. Unsere weitere Forschung im Bereich der Modellierung autonomer mobiler Roboter wird sich auf die Verallgemeinerung der hier vorgestellten Ansätze, die Modellierung weiterer Prozesse sowie das Lernen von Parametern des Systems konzentrieren.

References

M. Beetz, W. Burgard, D. Fox, and A. Cremers. Integrating active localization into high-level control systems. *Robo-*

tics and Autonomous Systems, 1998.

Wolfram Burgard, Dieter Fox, and Daniel Hennig. Fast grid-based position tracking for mobile robots. In *Proceedings of the 21th German Conference on Artificial Intelligence (KI 97), Freiburg, Germany*. Springer Verlag, 1997.

Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Position tracking with position probability grids. In *Proceedings of the First Euromicro Workshop on Advanced Mobile Robots (EUROMICRO '96)*, pages 2–9. IEEE Computer Society Press, 1996.

M. Beetz and H. Grosskreutz. Causal models of mobile service robot behavior. In R. Simmons, M. Veloso, and S. Smith, editors, *to appear in Fourth International Conference on AI Planning Systems*, Morgan Kaufmann, 1998.

M. Beetz and D. McDermott. Improving robot plans during their execution. In Kris Hammond, editor, *Second International Conference on AI Planning Systems*, pages 3–12, Morgan Kaufmann, 1994.

D. McDermott. A reactive plan language. Research Report YALEU/DCS/RR-864, Yale University, 1991.

D. McDermott. Transformational planning of reactive behavior. Research Report YALEU/DCS/RR-941, Yale University, 1992.

D. McDermott. An algorithm for probabilistic, totally-ordered temporal projection. Research Report YALEU/DCS/RR-1014, Yale University, 1994.

S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998. to appear.

Generating, Executing and Revising Schedules for Autonomous Robot Office Couriers

Maren Bennewitz and Michael Beetz
University of Bonn, Dept. of Computer Science III,
Roemerstr. 164, D-53117 Bonn, Germany,
email: {maren, beetz}@cs.uni-bonn.de

Abstract

Scheduling the tasks of an autonomous robot office courier and carrying out the scheduled tasks reliably and efficiently pose challenging problems for autonomous robot control. To carry out their jobs reliably and efficiently many autonomous mobile service robots acting in human working environments have to view their jobs as everyday activity: they should accomplish long-term efficiency rather than optimize problem-solving episodes. They should also exploit opportunities and avoid problems flexibly because often robots are forced to generate schedules based on partial information.

We propose to implement the controller for scheduled activity by employing concurrent reactive plans that reschedule the course of action whenever necessary and while performing their actions. The plans are represented modularly and transparently to allow for easy transformation. Scheduling and schedule repair methods are implemented as plan transformation rules.

Introduction

To carry out their jobs reliably and efficiently many autonomous mobile service robots acting in human working environments have to view their jobs as everyday activity. We consider a particular instance of everyday activity: performing office courier service. Thus, scheduling everyday activity differs from many other scheduling tasks such as job scheduling (FS84), space shuttle scheduling (DSB94), or transportation scheduling in the following aspects:

- ***Long-term efficiency.*** The goal of scheduling everyday activity is the optimization of long-term efficiency rather than problem-solving episodes. In certain situations, for instance, a competent office courier distributes empty envelopes according to an expected consumption profile while performing its delivery jobs. Distributing the envelopes beforehand decreases the chances that the robot has to pickup empty envelopes before delivering a letter and therefore misses a deadline. Such preparation actions make the performance of individual jobs slower but they can be expected to improve the overall performance significantly.
- ***Robustness and Flexibility.*** Schedules are to be generated based on partial information about the environment and the tasks. For instance, incomplete task specifications like "pick up the letter from Wolfram's desk and deliver it," lack proper descriptions of the envelope as well as the destination of the letter. In such a case, the robot has to postpone the execution until more information is available. Acting flexibily requires the robot courier to watch out for opportunities and exploit them as well as to detect and avoid problems while executing scheduled activity.
- ***Experience.*** Information acquired through extended experience is exploited to compute more appropriate schedules. For instance, knowledge about the time needed by individual users to upload or unload items can be exploited to estimate the required overall time more accurately.

It is important that the scheduler of an autonomous robot office courier is able to interleave delivery jobs, reschedule when problems are detected, and exploit opportunities. The scheduler also has to be able to predict whether exploiting an opportunity that has just been detected might cause failures in other activity threads such as missing deadlines (BG98). What seems less important is the computation of schedules that guarantee minimal path length because loading and unloading takes significant amount of time.

We propose to implement the controller for scheduled activity by employing concurrent reactive plans that reschedule the course of action whenever necessary and while performing their actions. The plans are represented modularly and transparently to allow for easy transformation. Scheduling and schedule repair methods are implemented as plan transformation rules.

Our research on scheduling everyday activity is carried out in the context of FAXBOT, a structured reactive controller (SRC) (Bee98) that is designed for robust and efficient execution of delivery plans on the autonomous mobile robot RHINO (see Fig. 1), an RWI B21 robot.

Fig. 1: RHINO

The main contributions of this paper are that we show (1) the installation of modular and transparent schedules in complex concurrent and reactive robot control programs; (2) explain the design of schedules and controllers that allow for opportunistic and robust execution of scheduled activity; and (3) describe novel plan transformation techniques for scheduling and rescheduling everyday activity.

FAXBOT's Scheduling Methods at Work

FAXBOT's delivery routines are implemented in RPL (Reactive Plan Language) (McD91). RPL provides conditionals, loops, program variables, processes, and subroutines. RPL also places high-level constructs (interrupts, monitors) to synchronize parallel physical actions and make plans reactive and robust by incorporating sensing and monitoring actions, and reactions triggered by observed events at the programmer's disposal. The RPL constructs used to specify scheduled activity are the PLAN-, WITH-POLICY-, WHENEVER-, and WAIT-FOR-statements; but see (McD91) for a complete description.

The FAXBOT controller carries out two kinds of subplans: *primary activities*, actions taken to accomplish the robot's mission, and *policies*, which monitor and maintain the conditions necessary for the successful and efficient execution of the primary activities. WITH-POLICY *P B* means "execute the primary activity *B* such that the execution satisfies the policy *P*." Policies are concurrent processes that run while the primary activity is active and interrupt the primary if necessary. Primary activities must handle interrupts and, due to the possible side-effects of policies, have to make suitable preparations for their successful continuation after reactivation.

The primary activities are separated into the *opportunistic* primaries and the *active* primaries. The *active* primaries are the ones that the robot is able to accomplish without help. The order in which the subplans of the active primaries are executed is given by the *order constraints* that specify a (partial) order on the navigation tasks contained in the active primary tasks. The *opportunistic* primaries are the ones that robot cannot accomplish autonomously. To complete them it has to wait for enabling conditions. For example, because FAXBOT has no action for opening doors it might have to wait for doors to open in order to complete its deliveries. The open door might be an opportunity to complete a user command.

Consider the following experiment that is carried out on RHINO using FAXBOT's scheduling capabilities (details on the scheduling and plan transformation methods can be found in (BB98)). RHINO receives two commands: "put the red letter on the meeting table in room A-111 onto the desk in room A-120" and "deliver the green book from the librarian's desk in room A-110 to the desk in room A-114" (see Fig. 2).

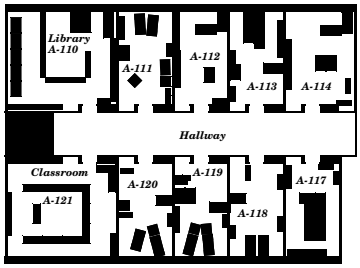
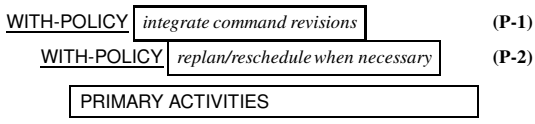
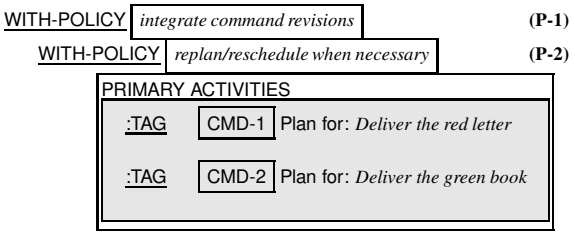


Fig. 2: Environment of the office courier

In the beginning, FAXBOT carries out no primary activities. Its outermost policy ensures that new commands are received and processed.



Upon receiving the two commands the policy **P-1** puts plans for the commands into the active primary activities of the SRC.



The insertion of the commands triggers the scheduler of the policy **P-2** that orders the navigation tasks in the primary activities.

Figure 3 shows the formalization of the transformation rule that schedules office delivery tasks. The transformation rule revises the primary activities by adding another policy that hat generates a bug whenever an assumption underlying the current schedule is detected as violated. The rule also revises the active primaries by

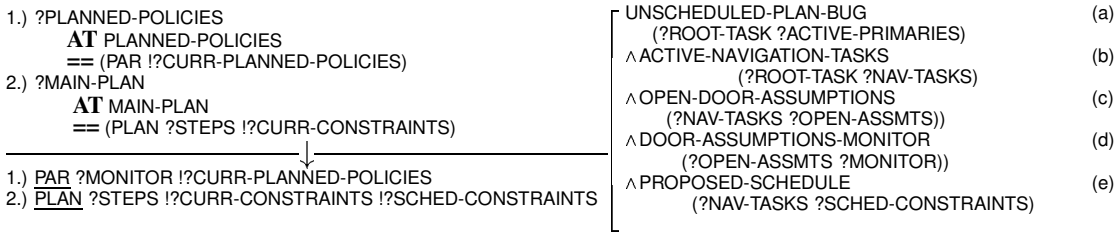


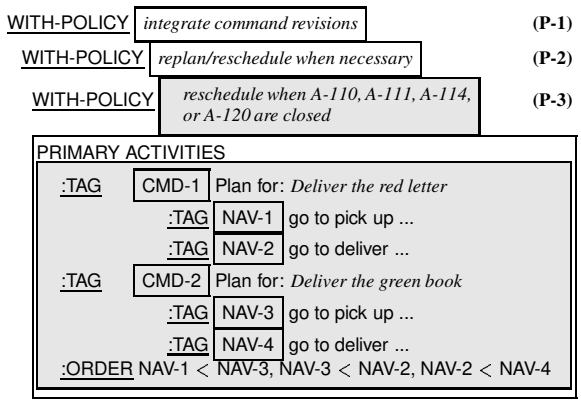
Figure 3: Plan revision rule for the the installation of task schedules.

adding the ordering contraints of the schedule to the constraints of the primary activities. The scheduling rule is applicable under a set of conditions specifying that (a) There is a bug of the category "unscheduled plan"; (b) The navigation tasks contained in the active primaries are ?NAV-TASKS; (c) ?NAV-TASKS can be accomplished if ?OPEN-ASSMPTS are satisfied; (d) ?SCHED-CONSTRAINTS are ordering constraints on ?NAV-TASKS such that any order which satisfies ?SCHED-CONSTRAINTS will accomplish the active primary tasks fast and avoid deadline violations and overloading problems. The rule is applied whenever the set of user commands changes.

The algorithm for ordering the navigation tasks sorts the destinations on one side of the hallway (see Fig. 2) in ascending and the others in descending order. After this initial sort, the scheduler iteratively resolves problems such as missed deadlines or confusions caused by carrying objects that look identical.

The scheduling routine is implemented as a plan transformation rule that can be applied to FAXBOT's overall plan while the plan is executed (BM97; BM94).

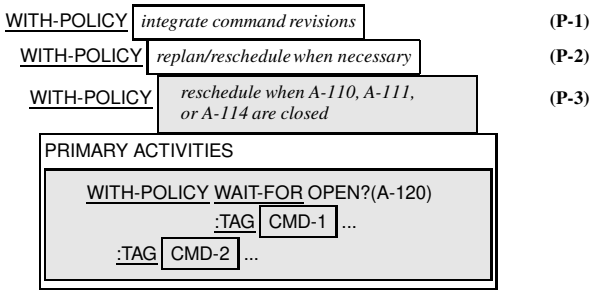
In our example, the application of the revision rule yields:



After FAXBOT has picked up the red letter from the meeting table and left room A-111, it notices that the door of room A-120 has been closed in the meantime. Because FAXBOT cannot complete the delivery of the red letter the corresponding command fails. This failure triggers the replanning policy **P-3**.

Each violated scheduling assumption triggers the application of the closed-door-transformation rule shown in Figure 4.

The revision rule of the FAXBOT controller that is triggered by closed doors (see Figure 4) causes the corresponding user command to fail. The rule deletes the failed plan for the user command from the active primary activities and adds the plan to the opportunistic primary activities. This is done by adding another policy that watches out for the door of A-120 to be opened again.



Thus, as soon as FAXBOT notices room A-120 to be open it interrupts its current mission, completes the delivery of the red letter, and continues with the remaining missions after the red letter has been successfully delivered.

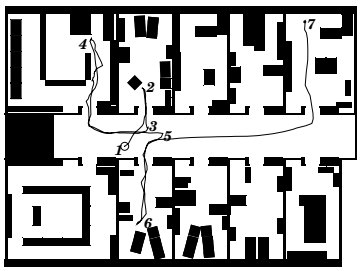


Fig. 5: Complete trajectory for the two deliveries

Figure 5 pictures RHINO's trajectory during the accomplishment of the two delivery jobs. FAXBOT starts with the delivery of the red letter and heads to the meeting table in A-111 where the letter is loaded (step 2). At

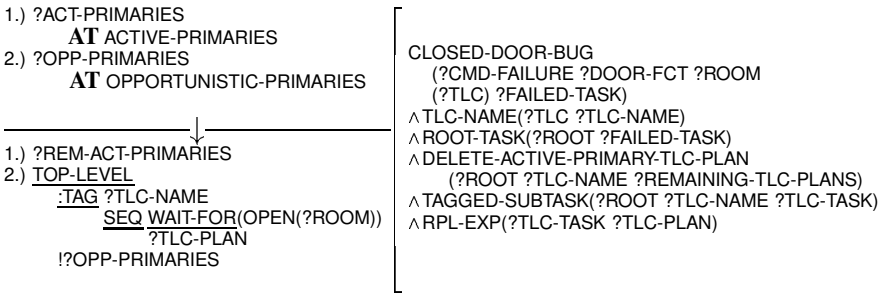


Figure 4: Plan revision rule for delivery tasks that cannot be completed because of closed doors.

this moment the door of A-120 is closed. Thus, when FAXBOT enters the hallway to deliver the red letter at Michael's desk, it estimates the opening angle of the door of room A-120. At this moment FAXBOT detects that the door has been closed and that it cannot complete the delivery (step 3). Thus FAXBOT navigates to the librarian's desk in A-110 to pick up the green book to room A-114 (step 4). At this moment room A-120 is opened again. As FAXBOT heads towards A-114 to deliver the green book it passes room A-120 (step 5). At this point the door estimation process signals an opportunity: A-120 is open! Therefore, FAXBOT interrupts its current delivery to complete the delivery of the red letter. After the delivery of the red letter is completed (step 6), FAXBOT continues the delivery of the green book (step 7).

The behavior generated by FAXBOT if all doors stay open is shown in Fig. 6 and the one if A-120 is closed but not opened again in Fig. 7.

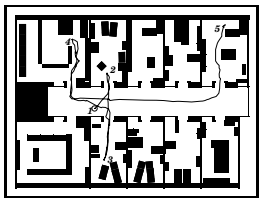


Fig. 6. Trajectory if A-120 stays open.

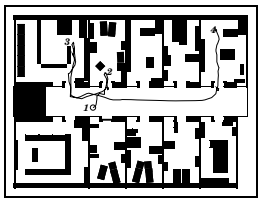


Fig. 7. Trajectory if A-120 is closed again.

Discussion

We have mainly focussed on the application of scheduling techniques to plans that control an autonomous mobile service robot. As such the contributions of this paper lie mainly in the representation of complex concurrent and reactive plans that facilitate scheduling operations, the specification of plan revision methods by means of plan transformation rules, and the application of these scheduling methods while the robot carries out the scheduled activity.

FAXBOT accomplishes its jobs successfully because its subplans are made interruptable and restartable using high-level control structures that specify synchronized concurrent reactive behavior. FAXBOT achieves adaptivity through plan revision and scheduling processes, implemented as policies, that detect opportunities, contingent situations, and invalid assumptions. Plan revision techniques are able to perform the required adaptations because of the modular and transparent specification of concurrent and reactive behavior. In particular the distinction of policies and primary activities increases the modularity significantly. Policies enable FAXBOT to specify opportunistic behavior and to achieve reliable operation while making simplifying assumptions.

Our research is still at an early stage. So far we have only performed some simple experiments to validate that the FAXBOT controller and its scheduler work; that is that it can reliably monitor scheduling assumptions and schedule delivery jobs during their execution.

There are many open issues that we would like to investigate more carefully in the near future. These issues include the development of more sophisticated scheduling methods (ZF94), the application of learning techniques to acquire useful information that can be exploited by heuristic scheduling methods (HC92b; HV98a; HV98b; ZDD+92), and a thorough experimental investigation on the effects of different scheduling techniques on the behavior exhibited by autonomous service robots (HC92a).

References

M. Beetz and M. Bennewitz. Planning, scheduling, and plan execution for autonomous robot office couriers. In *submitted for publication*, 1998.

M. Beetz. Structured reactive controllers. In *submitted for publication*, 1998.

M. Beetz and H. Grosskreutz. Causal models of mobile service robot behavior. In R. Simmons, M. Veloso, and S. Smith, editors, *to appear in Fourth International Conference on AI Planning Systems*, Morgan Kaufmann, 1998.

M. Beetz and D. McDermott. Improving robot plans during their execution. In Kris Hammond, editor, *Second International Conference on AI Planning Systems*, pages 3–12, Morgan Kaufmann, 1994.

M. Beetz and D. McDermott. Expressing transformations of structured reactive plans. In *Recent Advances in AI Planning. Proceedings of the 1997 European Conference on Planning*, pages 64–76. Springer Publishers, 1997.

M. Drummond, K. Swanson, and J. Bresina. Scheduling and execution for automatic telescopes. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*, pages 341–369. Morgan Kaufmann Publishers, 1994.

M. S. Fox and S. Smith. Isis - a knowledge-based system for factory scheduling. *Expert systems*, 1(1):25–49, 1984.

D. Hart and P. Cohen. Predicting and explaining success and task duration in the phoenix planner. In J. Hendler, editor, *AIPS-92: Proc. of the First International Conference on Artificial Intelligence Planning Systems*, pages 106–115. Kaufmann, San Mateo, CA, 1992.

A. Howe and P. Cohen. Isolating dependencies on failure by analyzing execution traces. In J. Hendler, editor, *AIPS-92: Proc. of the First International Conference on Artificial Intelligence Planning Systems*, pages 277–278. Kaufmann, San Mateo, CA, 1992.

K. Haigh and M. Veloso. Learning situation-dependent costs: Improving planning from probabilistic robot execution. In *To appear in Autonomous Agents 98*, 1998.

K. Haigh and M. Veloso. Planning, execution and learning in a robotic agent. In *To appear in AIPS-98*, 1998.

D. McDermott. A reactive plan language. Research Report YALEU/DCS/RR-864, Yale University, 1991.

M. Zweben, E. Davis, B. Daun, E. Drascher, M. Deale, and M. Eskey. Learning to improve constraint-based scheduling. *Artificial Intelligence*, 58:271–296, 1992.

M. Zweben and M. S. Fox. *Intelligent Scheduling*. Morgan Kaufmann, 1994.

Problemzerlegung beim Konfigurieren molekularer Fehlordnung

Karsten Knorr, Fritz Mädler

Hahn-Meitner-Institut, 14091 Berlin, maedler@hmi.de

Zusammenfassung

Die Steuerung maschineller Inferenz ist nach wie vor ein schwieriges Problem und in realen Anwendungen eine kritische Hürde. Wir berichten, wie wir die Modellierung von molekularen Einschlüssen in mikroporösen Materialien durch einen wissensbasierten Konfigurierungsansatz vorantreiben konnten [5, 6] und wie sich die unverzichtbare Problemzerlegung auch beim Konfigurieren als Anwendung des Nadelöhrprinzips [7, 10] deuten läßt.

1 Anwendungshintergrund

Zunehmend werden kristalline Materialien interessant, die molekulare Substanzen aufnehmen können. Sie lassen sich in der chemischen Industrie als mikroporöse Filter einsetzen oder als Katalysatoren zur Abgasreinigung in Kraftfahrzeugen und Fabriken. Durch geeignete Einschlüsse kann man die Brechungseigenschaften optischer Materialien gezielt beeinflussen, ebenso Reinigungs- und Umwelteigenschaften der weltweit verbreiteten Waschmittel.

Abbildung 1 zeigt den Prototypen einer einfachen "Wirt-Gast-Struktur": Eckenverknüpfte Tetraeder aus vier Sauerstoff- und einem Siliziumatom in der Mitte bilden ein Silica-Sodalith-Gerüst, das ein Dioxolan-Ringmolekül $(CH_2)_3O_2$ einzuschließen vermag. Dabei nimmt das eingelagerte Molekül im Wirtsgitter nicht nur eine Lage ein. Wegen der Symmetrien bei Wirt und Gast besitzt es Freiheiten hinsichtlich einer diskreten Anzahl von Raumpositionen, Orientierungen und Konformationen, das heißt hinsichtlich der verschiedenen energetisch stabilen Anordnungen seiner Atome im Raum. Im einfachsten, statischen Fall nimmt das Gastmolekül in jeder Wirtszelle eine der möglichen Lagen ein und behält sie während der Beobachtung bei. Im anderen Extrem, dem rein dynamischen Fall, wechselt es schnell zwischen den zulässigen Lagen und Konformationen hin und her. Man bezeichnet diese statischen und dynamischen Unbestimmtheiten des Moleküls (samt ihrer Zwischenstufen) als *Fehlordnung* des Gastes im Wirt. Ihre Erforschung trägt zum Verständnis von Wirt-Gast-Wechselwirkungen bei und dient letztlich der gezielten Synthese von Wirt-Gast-Substanzen mit erstrebenswerten, industriell nutzbaren Eigenschaften.

Man untersucht das Phänomen zum Beispiel durch Beugung hochauflösender Synchrotronstrahlen an pulverisierten Proben. Aus dem detektierten Streumuster läßt sich mit Hilfe einer Maximum-Entropie-Methode die räumliche Elektronendichte-Verteilung der Wirt-Gast-Struktur rekonstruieren [5]. Abbildung 2 zeigt für unsere Prototyp-Substanz eine Iso-Fläche dieser Dichteverteilung (in diesem Fall die Fläche, die alle Raumpunkte zum Wert von 1.2 Elektronen pro Kubik-Angström verbindet).

Allerdings zeigt Abbildung 2 nicht die Elektronendichte eines einzelnen Gastes in einer einzelnen Wirtszelle; solche Experimente können nur ein zeitlich und räumlich gemitteltes Bild aus vielen Zellen mit fehlgeordneten Molekülen in ihren unterschiedlichen Lagen und Konformationen liefern. Es stellt sich die Frage, wie letztere gemeinsam in statistischer Superposition die gemessenen Streumuster erklären. In erster Linie ist dies das Problem einer verträglichen Einbettung einzelner Dioxolan-Moleküle in die rekonstruierte Dichteverteilung, das wir mit einem wissensbasierten, hierarchisch gegliederten Konfigurierungsansatz gelöst haben.

2 Wissensbasierte, hierarchische Konfigurierung

Auch wenn die Dichteverteilung nur zeitlich und räumlich gemittelte Information liefert, so steckt in ihr doch verborgenes Wissen über das gesuchte Einzelmolekül, das es für seine Einbettung in die Dichteverteilung zu nutzen gilt. Neben allgemeinem Expertenwissen aus Chemie, räumlicher Geometrie etc., das zur Konstruktion des Moleküls benötigt wird, dient diese in den Daten verborgene Information über das Einzelmolekül als die maßgebende Wissensquelle für eine physikalisch sinnvolle Verbindung des gesuchten Moleküls mit den Meßergebnissen. Die Leistung des Konfiguratorprogramms besteht nicht zuletzt darin, sich diese Quelle zu erschließen und in akzeptabler Zeit Moleküle zu liefern, die in ihrer Gesamtheit und in der Fülle ihrer Erscheinungen eine modellhafte Erklärung der Experimentdaten gestatten.

Bei unserer Prototypsubstanz liefern die Maxima

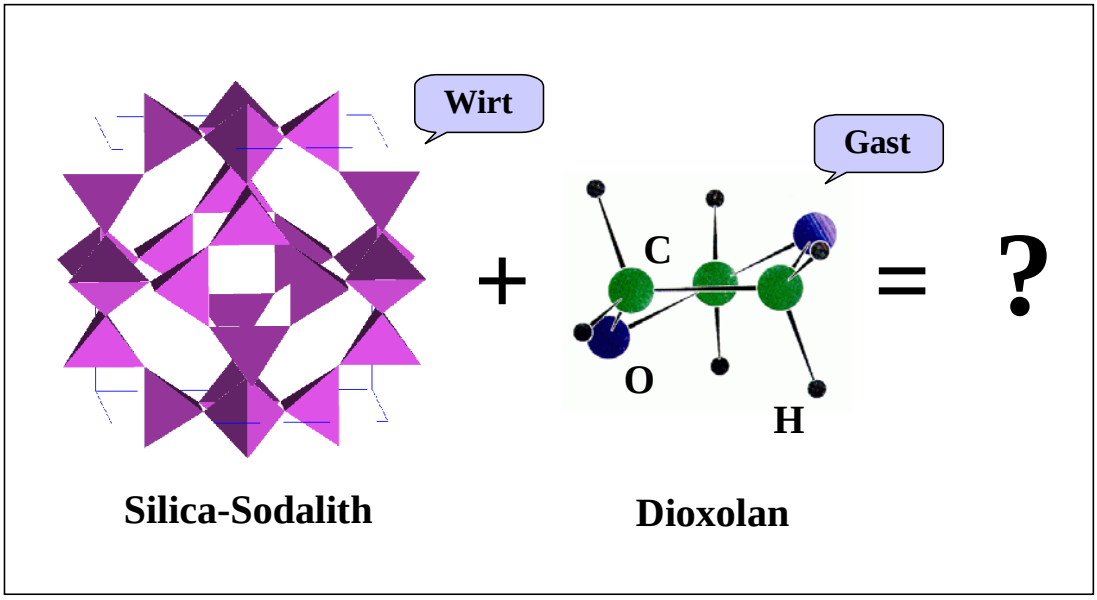


Abbildung 1: Silica-Sodalith-Gerüst und Dioxolan-Molekül als Prototyp einer Wirt-Gast-Struktur

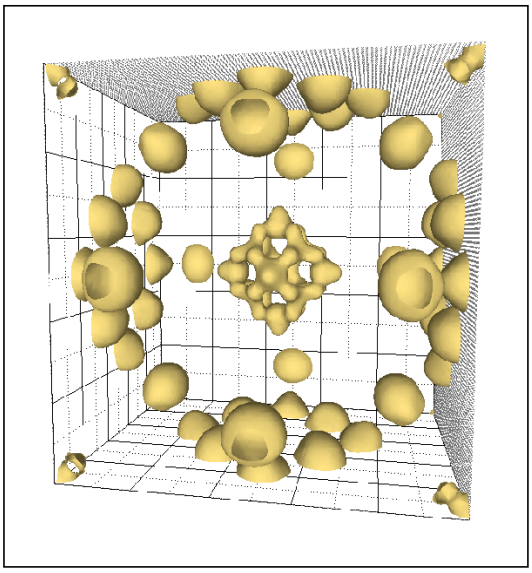


Abbildung 2: Iso-Fläche der Elektronendichte

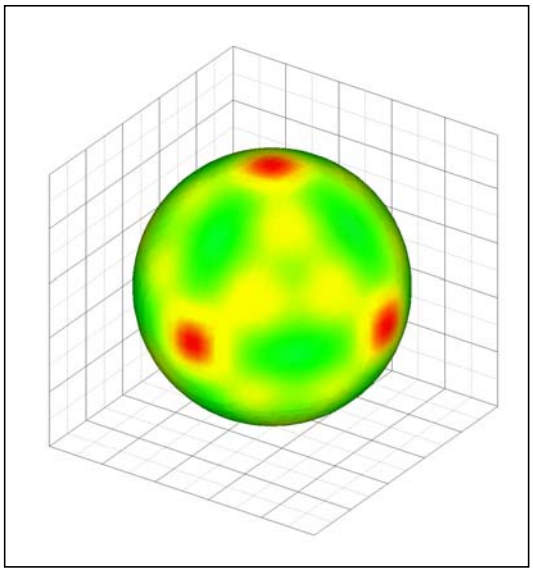


Abbildung 3: Sphärische Maxima -Verteilung

der Dichteverteilung einen Ansatzpunkt. Der zentrale Bereich in Abbildung 2 gehört zu dem Teil der Elektronendichte, der die zeitliche und räumliche Mittelung beim Gastmolekül wiedergibt. In Abbildung 3 ist eine Schnittkugel so im Datenvolumen plaziert, daß die - in diesem Fall - sphärische und symmetrische Anordnung der Dichte-Extrema sichtbar wird: Die Verteilung besitzt hier sechs globale Maxima (dunkle Stellen), von denen jedes von vier lokalen Maxima umgeben ist (helle Stellen). Keine Auswahl dieser Extremstellen kommt jedoch als Aufenthaltsort für die fünf Atome eines Dioxolan-Ringes in Frage: Selbst die ähnlichsten Fünf-Ringe aus Maxima (einen zeigt Abbildung 4) sind in ihren räumlichen Abmessungen - etwa in ihren Kantenlängen oder in ihrer räumlichen Verfaltung[1] - zu weit von den stereochemischen Parametern des Dioxolan-Ringes entfernt, um selbst schon als gesuchtes Gastmolekül gelten zu dürfen.

Solche dem Dioxolan ähnlichen Maxima-Ringe können allerdings durch Dioxolan-Ansiedlung in ihrer Nähe erzeugt sein, im allgemeinen in mehrdeutiger Weise. Beispielsweise können die Gastmoleküle verbleibenden Raum im Wirt gleichverteilt ausfüllen, oder sie treten normalverteilt auf bei thermischer Schwingung um die Extremstellen. In beiden Fällen sind letztere nicht Aufenthaltsort von Atomen, sondern Ergebnis einer statistischen Überlagerung vieler Einzelmoleküle.

Im vorliegenden (einfachen) Fall hat es genügt, "Nähe" durch Varianzkugeln zu modellieren, das heißt durch Kugeln mit chemisch begründbaren Radien um die globalen und lokalen Dichte-Maxima. Auf diese Weise wird der gesuchte Aufenthaltsort der Atome - und damit Lage und Gestalt des Gastmoleküls - sinnvoll und vor allem komplexitätsenkend eingeschränkt, ohne die Fehlordnungstendenzen zu beschneiden. Abbildung 5 zeigt einen Maxima-Ring mit fünf zugehörig gewählten Varianzkugeln, durch die das zu konfigurierende Molekül während seiner geometrischen Konstruktion "hindurchgefädelt" werden muß, und zwar unter Einhaltung seiner charakteristischen Parameter wie Atomabfolge, Bindungslängen, räumliche Verfaltung und Pseudo-Rotationsphase[2] des Moleküls etc. Dabei werden verbleibende Konstruktionsfreiheiten gemäß einer vorab gewählten Statistik verteilt.

Insgesamt hat dieser hierarchisch gegliederte Sachverhalt - die Existenz Dioxolan-ähnlicher Ringe aus Extremstellen, in deren Nähe dann die Dioxolan-Charakteristika erfüllt werden können - in natürlicher Weise auf das folgende, zweistufig zerlegte **Konfigurierungsverfahren** geführt:

Ebene 1

Suche Maxima-Ring-Kandidaten mit folgenden Eigenschaften:

- die Ringkantenlängen liegen in der Größenordnung der Dioxolan-Bindungen
- der Ring besitzt eine räumliche Verfaltung ähnlich der des Dioxolan-Moleküls
- die Konformation des Ringes ist mit Dioxolan-Konformationen vergleichbar

Ebene 2

Wähle zum Ring-Kandidaten aus Ebene 1:

- eine Atomfolge mit korrekten Bindungslängen
- Varianzkugelradien in Abhängigkeit von Ringecken und Atomen
- Schranken für Faltung und Rotationsphase des Moleküls
- eine Statistik für die Verteilung der Atome in den Varianzkugeln

und konstruiere ein gültiges Dioxolan-Molekül auf Ebene 2 unter Nutzung der Backtrack-Möglichkeiten auf Ebene 1.

Auf beiden Ebenen steht Expertenwissen für den Test der verlangten Eigenschaften zur Verfügung. Allerdings ist für Begriffe wie "Größenordnung", "ähnlich" etc. notgedrungen nur eine ungenaue quantitative Festlegung durch grobe Beschränkung der stereochemischen Molekülparameter möglich. Auch fehlt zunächst alles explizite Wissen über die Verbindung des konkreten Einzelmoleküls mit dem gemessenen Streumuster bzw. mit der rekonstruierten Elektronendichte-Verteilung; allein aus der Kenntnis der 30 Dichte-Maxima und mit dem sonstigen explizit überlassenen Wissen könnte noch

[1] Bei Fünf-Ringen repräsentiert man ihre räumliche Verfaltung

$$q = \sqrt{\sum_{i=1}^{5} z_i^2}$$

mit Hilfe der Auslenkungen $z_i = z_i(E^\star(S))$ der Ringecken aus der Mittelebene $E^\star = E^\star(S)$: Unter allen Ebenen durch den Schwerpunkt S der Ecken ist diese Mittelebene durch die "non-rotation"-Bedingungen

$$\sum_1^5 z_i \cos[2\pi(i-1)/5] = 0, \quad \sum_1^5 z_i \sin[2\pi(i-1)/5] = 0$$

eindeutig bestimmt [2].

[2] Die Pseudo-Rotationsphase Φ eines räumlichen Fünf-Ringes wird durch die Bedingungen

$$z_i = q\sqrt{2/5}\,\cos[4\pi(i-5)+\Phi], \quad i = 1,\ldots,5.$$

eindeutig festgelegt. Zusammengenommen erfaßt man mit der (q,Φ)-Repräsentation sowohl die verschiedenen Konformationen des Moleküls als auch seine Rotationsisomerien, die durch bloße In-sich-Rotation des Auslenkungsvektors $z = (z_1,\ldots,z_5)$ zustande kommen, unter Beibehaltung der Atomabfolge. Beide Phänomene sind für eine umfassende Modellierung molekularer Fehlordnung unverzichtbar.

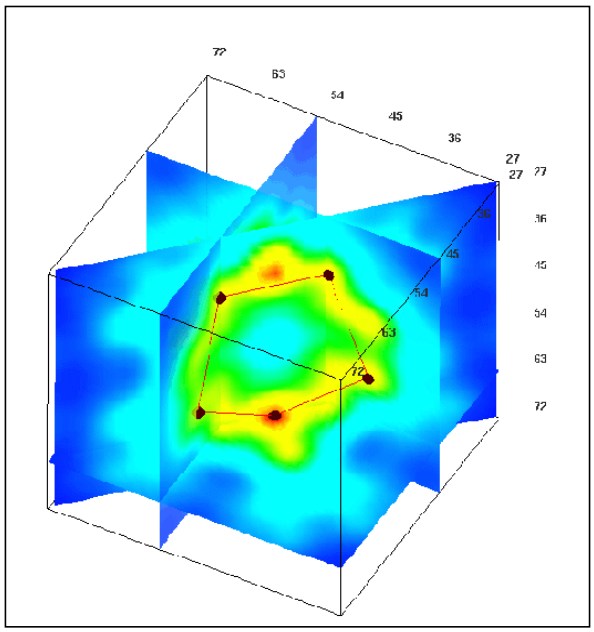


Abbildung 4: Ein Fünf-Ring aus Dichte-Maxima

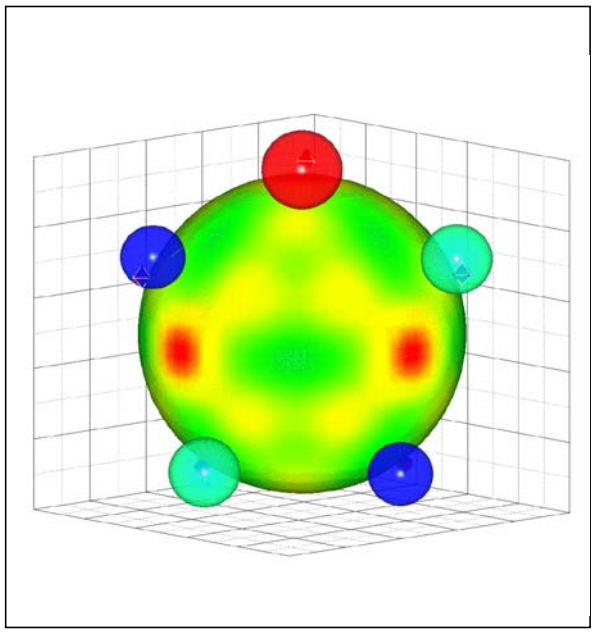


Abbildung 5: Varianzkugeln zum Ring aus Abb. 4

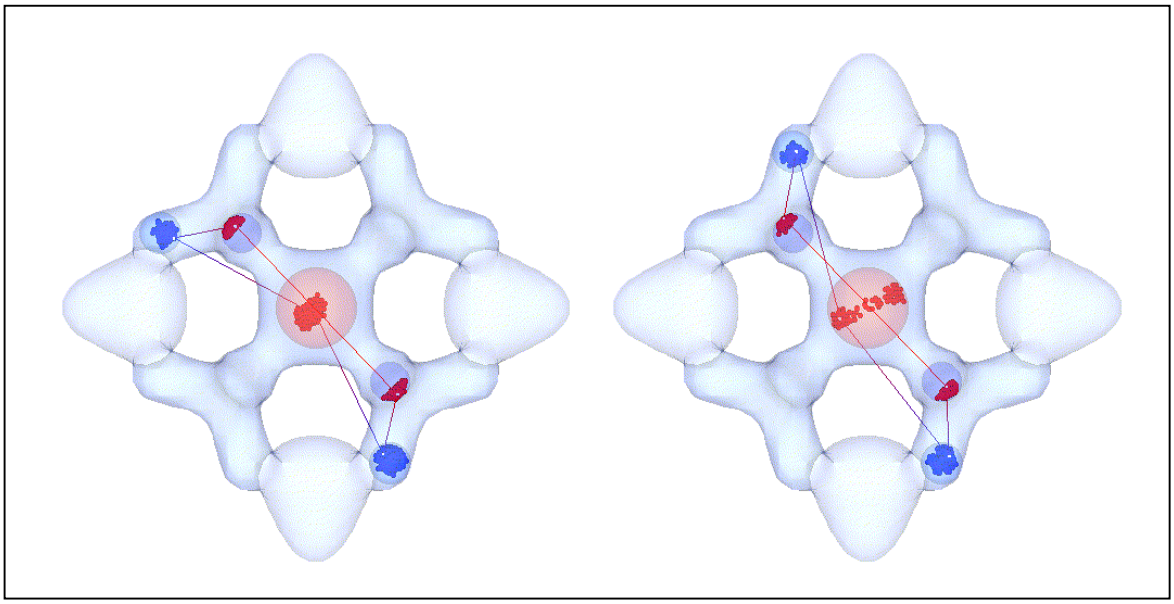


Abbildung 6: Je 300 Envelope- (links) und Twist-Moleküle (rechts) bei Dioxolan in Silica-Sodalith

kein menschlicher Experte modellhaft Moleküle zur Erklärung der Dichte-Gestalt angeben, jedenfalls nicht in statistischer Verteilung und unter systematischer Aufdeckung aller Möglichkeiten bei Konformation und Isomerie. Problemzerlegung und grobe Beschränkung senken aber die kombinatorische Vielfalt so drastisch, daß auf dem Rechner das Konfigurieren vieler Gastmoleküle in akzeptabler Zeit gelingt. Dabei erschließt sich der Konfigurator fehlendes Wissen, indem er die in der Hierarchie verbliebene Restkombinatorik systematisch nach sinnvollen Lösungen durchsucht, das heißt nach Kombinationsergebnissen, die aus dem explizit repräsentierten Wissensteil heraus beweisbar sind.

So gesehen stellt der Konfigurator ein implizites, rechner-gestütztes Modell zur Erklärung der Fehlordnung dar. Beispielsweise zeigt Abbildung 6 für die beiden gefundenen Dioxolan-Konformationen je 300 Gastmoleküle zu zwei einander sehr ähnlichen Maxima-Ringen (in Draufsicht entlang der z-Achse): Die beiden Konformationen unterscheiden sich nur durch den Wechsel eines Dioxolan-Ringatoms von einem der Nebenmaxima zu einem benachbarten (im oberen Bildteil), ein Ergebnis, das bislang noch nicht gesehen worden war.

Dieses wissensbasierte Konfigurierungsverfahren ist allgemein und wird uns auch bei anderen Wirt-Gast-Strukturen als implizites Modell dienen, vor allem, wenn in symmetrie-ärmeren oder dynamischeren Fällen kaum Hoffnung auf ein explizites Modell in Gestalt einer geschlossenen Formel besteht.

3 Deutung der Zerlegung als Nadelöhr-Kette

Nicht von ungefähr sind *Konfigurieren* und *Planen* zu einer gemeinsamen Problemklasse zusammengefaßt worden [3, 11]. Die Gemeinsamkeiten äußern sich nicht zuletzt darin, daß bei einer geeigneten Formalisierung von *Zustand* und *Handlung* für beide Aufgabentypen ein sehr ähnlicher Lösungsgraph auftritt. Für Planungsaufgaben wurden in [7, 10] die besonderen Eigenschaften dieses Graphen zur axiomatischen Ausstattung eines allgemeinen "Nadelöhrprinzips" benutzt, mit dem sich über einem Handlungsmodell Kontrollwissen zur Steuerung der Planung ableiten und repräsentieren läßt. Ohne den Formalismus hier auszubreiten, skizzieren wir in diesem Abschnitt, in welchem Sinn auch die obige Steuerung der Konfigurierung - das "Hindurchfädeln" des Moleküls durch eine die Kombinatorik beschneidende Folge von Varianzkugeln - als Kontrollwissen im Sinne einer Nadelöhr-Kette aufgefaßt werden kann. Dazu zitieren wir zunächst ein Beispiel für eine Nadelöhrmenge im Kontext optimaler Handlungsplanung[3] und beschreiben dann das graphische Prinzip für das obige Konfigurierungsverfahren in Analogie.

Eine Handlung, beispielsweise

$$A = \textbf{action(}\ Actor, S_0, S_1\ \textbf{)}$$

bewegt ein "Stellglied" namens A_{name}

$$Actor = \textbf{actor(}\ A_{name}, A_{old}, A_{new}\ \textbf{)}$$

aus seiner Stellung A_{old} in die neue Stellung A_{new} und überführt auf diese Weise den aktuellen Istzustand S_0 der repräsentierten Domäne in den Nachfolgezustand S_1. Dabei bestehen Ist- und Sollzustand aus Merkmalswerten $X_1, \ldots, X_m$ von Stellgliedern und aus Werten $X_{m+1}, \ldots, X_n$ von "abgeleiteten Parametern"[4], das heißt

$$S = \textbf{state(}\ X_1, \ldots, X_m; X_{m+}1, \ldots, X_n\ \textbf{)}$$

In Prolog kann der Zustandsübergang $S_1 = A(S_0)$ in der Form

```
action(Actor,
       S0,
       S1
      )
  <=
       update_1( Actor,..., Xi,..... ),
       ...
       update_k( Actor,....., Xj,... ).
```

repräsentiert werden, wobei die Handlung gelingt, wenn sie über der Wissensbasis mit Hilfe der Rumpfprädikate **update_1** bis **update_k** für die Merkmalswerte X_i, X_j von Ist- und Nachfolgezustand bewiesen werden kann.

Im Graphen der Abbildung 7 stellt jeder numerierte Knoten einen Zustand S_0 dar, jede abwärts gerichtete Kante A_1 zwischen zwei Knoten S_0 und S_1 einen Zustandsübergang $S_1 = A_1(S_0)$ und jeder Pfad aus solchen Kanten einen kürzesten, "optimalen" sequentiellen Handlungsplan

$$L = (A_1, \ldots, A_k),$$

auch "Linearplan" genannt, der den Startzustand S_0 in den Zielzustand $S_k = L(S_0)$ bringt und dabei durch die Zustände

$$S_i = A_i(S_{i-1}),\ i = 1, \ldots, k$$

führt. Die Existenz mehrerer, permutierter Linearpläne $L_j = (A_1, \ldots, A_k)$ zum selben Zustandspaar

[3] Das Beispiel stammt aus einer realen Anwendung hier im Hahn-Meitner-Institut, bei der aufwendige und komplizierte Spül- und Reinigungsprozeduren für eine Plasmadepositionsanlage zu planen waren [7].

[4] Im Beispiel etwa Gas- und Druckzustände in Depositionskammern und Rohrleitungen; sie hängen indirekt von den Stellgliedern ab und stellen kritische Werte dar, die über die Legalität von Handlungen entscheiden und genau wie alle anderen Merkmalswerte Zielvorgaben bilden können.

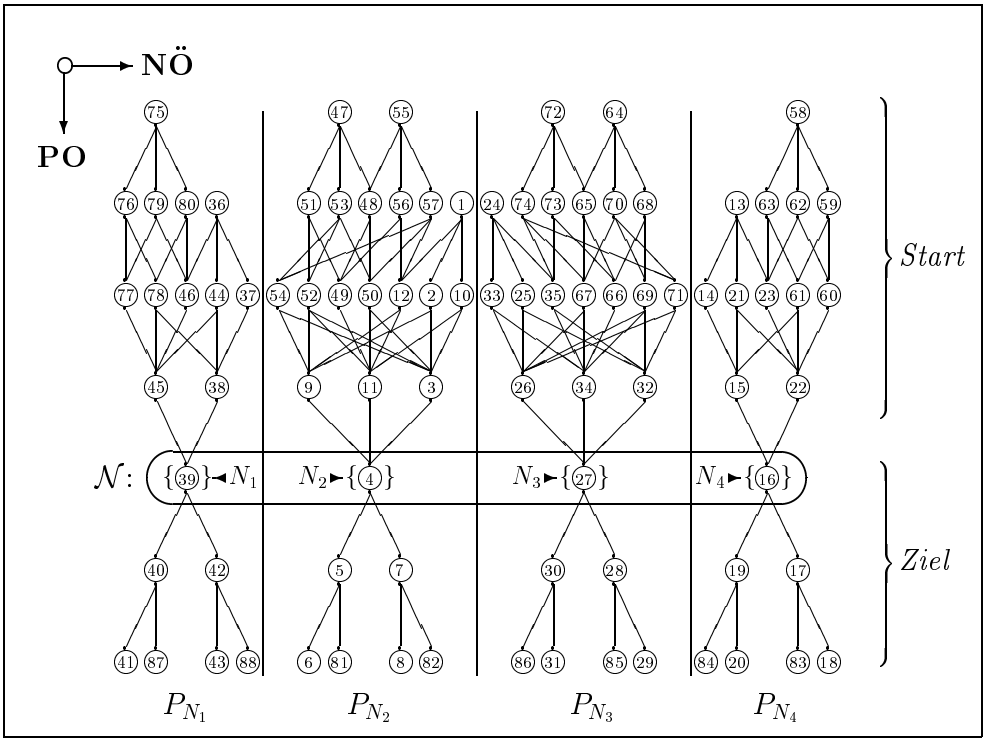


Abbildung 7: Eine Nadelöhrmenge $\mathcal{N}$ als extensionale Form eines Planungsoperators

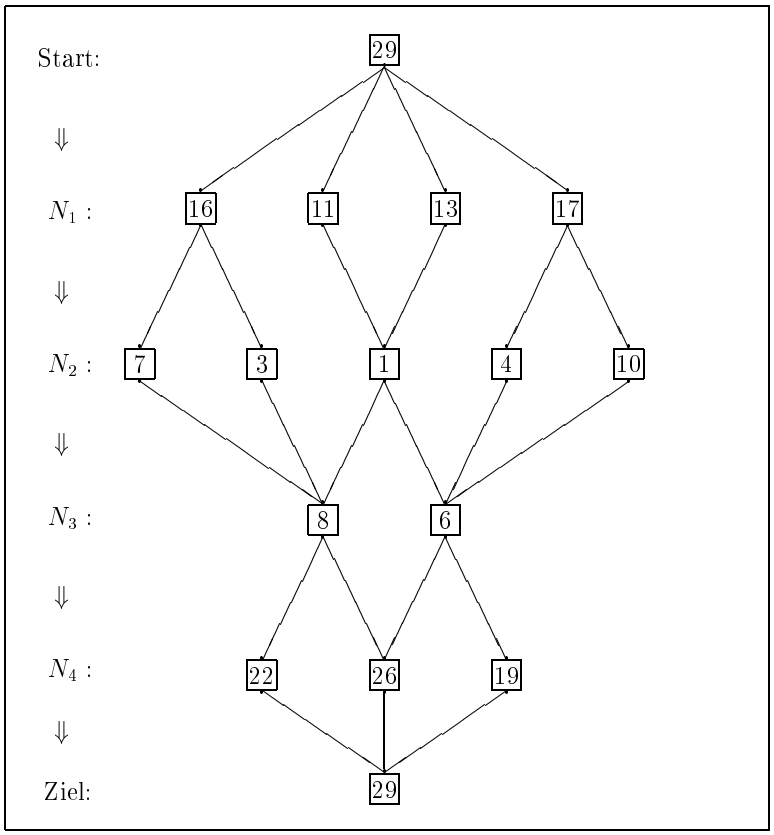


Abbildung 8: Eine Nadelöhr-Kette zur Steuerung beim Konfigurieren

S_0 und $S_k = L_j(S_0)$ und zur selben Handlungsmenge $H = \{A_1, \ldots, A_k\}$ ist eine notwendige Bedingung für die Existenz eines parallelisierten Planes P in Form einer seriell-parallelen partiellen Ordnung von H, deren Linearisierungen dann eine Teilmenge aller Linearpläne mit $L_i(S_0) = S_k$ bilden [7, 8].

Die besondere Gestalt des Graphen beschränkt sich nicht auf unser Beispiel, sie ist dem Konzept des linearen wie parallelen Handelns immanent und kommt zum Vorschein, sobald man die Frage nach einer vollständigen Beschreibung der optimalen Handlungsmöglichkeiten stellt. Jede solche Handlungsrepräsentation ist im folgenden Sinne verkettungsabgeschlossen: Mit je zwei optimalen Linearplänen

$$(S_0, \ldots, S_i^\star, \ldots, S_k), \quad (S'_0, \ldots, S_i^\star, \ldots, S'_{k'}),$$

die einen gemeinsamen Zustand $S_i^\star$ besitzen, sind auch ihre überkreuzten Verkettungen

$$(S_0, \ldots, S_{i-1}, S_i^\star, S'_{i+1}, \ldots\ldots, S'_{k'})$$

und

$$(S'_0, \ldots\ldots, S'_{i-1}, S_i^\star, S_{i+1}, \ldots, S_k)$$

optimale Pläne, und zwar für die Planungsaufgaben $(S_0, S'_{k'})$ und (S'_0, S_k). Anschaulich bedeutet dies, daß die Zustände in der Handlungsrepräsentation schichtweise auftreten bzw. daß bei optimalen Linearplänen im Graphen keine Knotenschicht übersprungen wird. Axiomatisch betrachtet zeigt Abbildung 7 das Diagramm einer weiteren partiellen Ordnung, nämlich einer Jordan-Dedekind-Ordnung mit Tiefenfunktion (etwa [1]), bei der der Abstand der Elemente von ihren Vorgängern und Nachfolgern nicht vom betrachteten Pfad abhängt.

In einer solchen Struktur kann man in wohldefinierter Weise Problemzerlegung betreiben: Zunächst läßt sich in einer partiellen Ordnung $(\mathcal{P}, \prec)$ für jedes Element präzise ausdrücken, was *vor* und was *hinter* ihm liegt. Im Idealfall - wie etwa beim Zustand des Knotens 39 in der linken Spalte in Abbildung 7 - zerlegt das Element im Ordnungsdiagramm bereits vollständig die Teilstruktur, die von seinen Vorgängern und Nachfolgern gebildet wird. Nimmt man erstere als Startzustände S_0 und letztere als Zielzustände S_g, so ist der Zustand 39 ein unvermeidbarer *Nadelöhrzustand*, der bei der Lösung aller Planungsaufgaben (S_0, S_g) mit $S_0 \prec 39 \prec S_g$ als komplexitätsenkendes Zwischenziel fungieren kann. Im weniger "engen" Fall - etwa beim Zustand 45 - würde die Menge $N := \{45, 38\}$ aus zwei Nadelöhrzuständen das Gewünschte leisten. Stets gibt es auf der Schicht eines Zustands S eine minimal erweiterte Zustandsmenge $N \subset \mathcal{P}$ mit $S \in N$, die sich als *Nadelöhr* zur Zerlegung heranziehen läßt. Darüberhinaus kann man Nadelöhre N_j so zu einer *Nadelöhrmenge* $\mathcal{N} = \{N_1, \ldots, N_m\}$ zusammenstellen, daß die gesamte Handlungsrepräsentation vollständig und disjunkt partitioniert wird, wie es zum Beispiel die abgebildete Nadelöhrmenge

$$\mathcal{N} = \{\, \{39\}, \{4\}, \{27\}, \{16\} \,\}$$

leistet. Dazu müssen die Nadelöhre N_j nicht derselben Schicht angehören; auch Mengensysteme wie

$$\mathcal{N} = \{\, \{45, 38\}, \{4\}, \{30, 28\}, \{16\} \,\}$$

oder

$$\mathcal{N} = \{\, \{45, 38\}, \{4\}, \{30\}, \{28\}, \{16\} \,\}$$

wären vom Nadelöhr-Formalismus erfaßt, notfalls sogar die Zerlegung in einem einzigen Nadelöhr aus den Zuständen einer ganzen Schicht.

Der abgebildete Graph repräsentiert das Kontrollwissen zunächst nur in einer *extensionalen* Form, indem er eine Vielzahl von Planungsaufgaben (S_0, S_g) und ihre Nadelöhre N_j aufzählt, in denen sich diese Aufgaben zerlegen lassen. Man kann aber maschinelle Lernverfahren anwenden und durch "strukturelle Abstraktion" eine *intensionale* Beschreibung aus Domänen-Merkmalen erzeugen [9]: Wenn man die mit dem Formalismus verbundenen Klasseneinteilungen nutzt - zum einen die vertikale Einteilung entlang der partiellen Ordnung nach Start- und Zielklasse, zum anderen die horizontale Partitionierung entlang der Nadelöhrmenge - erhält man nach zweimaliger Induktion einen klassischen abstrakten Planungsoperator (im Sinne etwa von [4, 12]) mit Vor- und Nachbedingungen samt Expansionsvorschrift in Form von Zwischenzielen.

Für eine analoge Zerlegung beim Konfigurieren gehen wir von der "Handlungsrepräsentation" des Verfahrens von Abschnitt 2 aus. Unter Beschränkung auf das Wesentliche läßt sich im vorliegenden Fall ein Konfigurationszustand - das heißt eine Belegung der Atome

$$A_i = \mathbf{atom}(C_i, P_i, M_i, R_i), \quad i = 1, \ldots, 5,$$

mit einer Atomart C_i, einer räumlichen Position

$$P_i = P_i(x, y, z),$$

einem Varianzkugelzentrum

$$M_i = M_i(x, y, z)$$

und einem zugehörigen Varianzkugelradius R_i -, mit Hilfe eines Terms

$$\mathbf{state}(\ \ldots, \mathbf{atom}(\ C_i, P_i, M_i, R_i\ ), \ldots\ )$$

repräsentieren. Ein Konfigurierungsschritt - etwa die Konstruktion der zweiten Atomposition P_2 bei schon erfolgter erster Belegung von P_1 - hat jetzt im wesentlichen die Form[5]

[5] Prolog gemäß tritt der Zustandsterm **state** nur einmal auf; die Atompositionen $P_i(x, y, z)$ sind am Anfang uninstanziert und nach gelungener Konfigurierung sämtlich belegt.

```
c_step(
   state(
         atom( C1, P1, M1, R1 ),
         atom( C2, P2, M2, R2 ),
         A3,
         A4,
         A5
        )
      )
      <=
         bond_length( C1, C2 ),
         atom_position(C1, C2, P1, P2, M2, R2).
```

Die Verkettung der Konfigurierungsschritte entlang eines Maxima-Ringes $M_1 M_2 \cdots M_5 M_1$ aus Ebene 1 liefert dann auf Ebene 2 einen Molekül-Kandidaten $A_1 A_2 \cdots A_5 A_1$ in der Nähe dieser Maxima.

Für die Darstellung eines Beispielgraphen haben wir die Maxima durchnumeriert. In Abbildung 8 ist die Gesamtheit von fehlgeordneten Molekül-Kandidaten repräsentiert, die mit einem ihrer C- oder O-Atome beim Maximum M_1 = "29" liegen und mit den restlichen vier Atomen entlang der Pfade auf die Varianzkugeln anderer Maxima verteilt sind. Die gerahmten Knoten stehen dabei sowohl für Maxima-Positionen, als auch für Atom-Positionen benachbarter Molekül-Kandidaten; insofern repräsentiert ein Knoten hier keine Einzelzustände wie zuvor beim Planungsbeispiel, sondern die (nicht mehr endliche) Menge der bis zum Knoten erreichbaren Zwischenzustände der verlangten Konfiguration. Mit Blick auf diesen Graphen erfolgt die Steuerung bei der Konfigurierung des Moleküls auf Ebene 2, indem man entlang der Pfade nacheinander die Schichten besucht und dort den Konstruktionsschritt entweder erfolgreich ausführt oder ungenügende Kandidaten frühzeitig und unter Backtracking wissensbasiert aussondert.

Auch dieser Kandidatengraph ist verkettungsabgeschlossen und stellt wie im Falle der Planung das Diagramm einer Jordan-Dedekind-Ordnung dar. Dies war die entscheidende Voraussetzung für die Definitionen und Sätze des Nadelöhr-Formalismus. Indem man sich beim Konfigurieren schrittweise von einer Varianzkugel zur nächsten bewegt, also im Graphen von Schicht zu Schicht, folgt man genau betrachtet einer Kette aus Nadelöhren $N_1, \ldots, N_4$; denn in der Sprache des Nadelöhr-Konzepts bewegt man sich durch vier einelementige Nadelöhrmengen $\mathcal{N}_j = \{N_j\}$ aus Zustandsmengen N_j, die in diesem Fall aus allen Zuständen ihrer Schicht bestehen. Man beachte, daß die Schichtbreiten des Graphen und die dichte Aufeinanderfolge der Zerlegungsstellen der intuitiven Vorstellung vom Nadelöhr nur scheinbar widersprechen: Außerhalb der Varianzkugeln ist reichlich Raum für "kombinatorische Hyper-Explosion".

Die Problemzerlegung mit Hilfe von Varianzkugeln hat sich in natürlicher Weise aus der kristallographischen Problemstellung herleiten lassen; man muß nicht bewußt den Nadelöhr-Formalismus heranziehen, um hier ausreichende Komplexitätssenkung zu erreichen. Aber die Interpretation des hierarchischen Verfahrens anhand des Kandidatengraphen und seiner Besonderheiten zeigt, daß wir im Grunde einem strukturgegebenen, in seiner Abstraktion hoch angesiedelten Zerlegungsprinzip gefolgt sind; in seiner Allgemeingültigkeit und Wirksamkeit weist das Nadelöhrprinzip über die hier geschilderten Anwendungsfälle hinaus.

Literatur

[1] G. Birkhoff: *Lattice Theory.* Am. Math. Soc., Colloquium Publications, Vol. XXV, 1961

[2] G. Gilli: *Molecules and molecular crystals.* In: C. Giacovazzo (ed.), Fundaments of Crystallography. Oxford Univ. Press, 1994

[3] A. Günter: *Flexible Kontrolle in Expertensystemen zur Planung und Konfigurierung in technischen Domänen.* DISKI Bd. 3, infix, St. Augustin, 1992

[4] J. Hertzberg: *Planen - Einführung in die Planerstellungsmethoden der Künstlichen Intelligenz.* BI Wissenschaftsverlag, Reihe Informatik, Bd. 65, Mannheim, 1989

[5] K. Knorr, F. Mädler, R. Papoular: *Model-free Density Reconstruction of Host/Guest-Compounds from High-Resolution Powder Diffraction Data.* J. Microporous Materials, Elsevier (im Druck).

[6] K. Knorr, F. Mädler: *Knowledge-Based Configuration of Molecule Representations.* Web-Version, http://www.hmi.de/people/maedler

[7] F. Mädler: *Problemzerlegung durch Nadelöhrmengen - Ein modellbasierter Ansatz zur Akquisition von Kontrollwissen für Planungssysteme.* DISKI Bd. 74, infix, St. Augustin, 1994

[8] F. Mädler: *Seriell-parallele Ordnung als Lösungsbegriff für Planungsaufgaben.* In: A. Horz (Hrsg.), 7. Workshop Planen und Konfigurieren. Arbeitspapiere der GMD, Nr. 723, S. 74-85, Inst. f. Ang. Informationstechnologie, St. Augustin, 1993

[9] F. Mädler: *Towards Structural Abstraction.* In: J. Hendler (ed.), Proc. First Int. Conf. on AI Planning Systems (AIPS-92, Maryland): 163-171. Morgan Kaufmann, San Mateo, Ca, 1992

[10] F. Mädler: *Problemzerlegung als optimalitätserhaltende Operatorabstraktion.* In: Th. Christaller (Hrsg.), Proc. GWAI-91: 74-83. IFB 285, Springer, 1991. (Auch in Zeitschrift KI 2/92: 37-41.)

[11] F. Puppe: *Problemlösungsmethoden in Expertensystemen.* Studienreihe Informatik, Springer, 1990

[12] D. E. Wilkins: *Practical Planning.* Morgan Kaufmann, San Mateo, Ca, 1988

Reduktion von Entwicklungskosten durch wissensbasierte Konfiguration (Ein Fallbeispiel)

Gerhard Fleischanderl [x], Gerhard E. Friedrich [*x], Alois Haselböck [x],
Herwig Schreiner [x], Markus Stumptner [+]

[x]Siemens AG Österreich
Entwicklungszentrum für Elektronik
Erdberger Lände 26, A-1030 Wien, Austria
e-mail: firstname.{middle initial.}lastname@siemens.at

[*]Universität Klagenfurt
Institut für Informationstechnologie
Lehrstuhl für Produktionsinformatik
Universitätsstrasse 65-67, A-9020 Klagenfurt, Austria
e-mail: friedrich@ifi.uni-klu.ac.at

[+]Technische Universität Wien
Institut für Informationssysteme
Paniglgasse 16, A-1040 Wien, Austria
e-mail: mst@dbai.tuwien.ac.at

Abstract

Die wissensbasierte Konfiguration ist mit der Zielsetzung angetreten, die Entwicklung und Wartung von Konfigurationssystemen zu beschleunigen. Gleichzeitig sollen die Kosten dieser Aufgaben gesenkt und die Qualität der Benutzerinteraktion im Konfigurationsprozeß durch erhöhte Flexibilität und Erklärungsfähigkeit gesteigert werden.

Aus dem Blickwinkel von Entwicklungsabteilungen steht die Methode der wissensbasierten Konfiguration in Konkurrenz zu etablierten Software-Entwicklungsmethoden vor allem aus den Bereichen der objektorientierten Programmierung und der Datenbank-Applikationssprachen.

Für den technischen Entscheidungsträger stellt sich daher die Frage, welcher Ansatz in einem konkreten Entwicklungsprojekt zu bevorzugen ist. Wir berichten anhand von im produktiven Einsatz befindlichen Konfigurationssystemen, daß der wissensbasierte Ansatz für umfangreiche Konfigurationsprobleme tragfähig ist und gewinnbringend eingesetzt werden kann. Die Verwendung dieser Methode ermöglicht erhebliche Kostenreduktionen der Entwicklungsaufwendungen sowie eine gleichzeitige Erhöhung der Funktionalität und Flexibilität.

1 Motivation aus betriebswirtschaftlicher Sicht

Unternehmungen stehen von zwei Seiten unter steigendem Druck. Einerseits ist auch bei vielen technischen Produkten ein zunehmender Preisverfall festzustellen, andererseits sind erhebliche Kostensteigerungen zu verzeichnen. Als eine mögliche Gegenstrategie wird eine verstärkte Ausrichtung auf die Kundenzufriedenheit vorgeschlagen, ein Vorgehen, das vermehrt zur Entwicklung von anpassungsfähigen Produkten führt. Um konkurrenzfähige Preise erzielen zu können, muß diese Varianz der Produkte aber im gesamten Geschäftsprozeß (von Verkauf über Produktion bis zur Wartung) beherrscht werden. Die Konfiguration von Produkten nimmt hier eine Schlüsselposition ein. Automatisierung von Konfigurationsaufgaben, die aufgrund der geforderten Komplexität als unrentabel galten, beginnen sich unter den geänderten Rahmenbedingungen zu rechnen.

Diese Zunahme von Varianz und Komplexität der Aufgabe führt jedoch zu einer Steigerung der Entwicklungsaufwendungen von Konfiguratoren. Besonders in Bereichen, in denen Großsysteme zu konfigurieren sind, die Bereiche Kommunikationstechnik und Eisenbahnsicherungstechnik seien hier stellvertretend erwähnt, nehmen die Aufwendungen zur Entwicklung von Konfiguratoren signifikante

Größen an. Bezogen auf die wissensbasierte Konfiguration müssen im wesentlichen zwei Fragen beantwortet werden. Können wir auch für Großsysteme wissensbasierte Ansätze einsetzen, und ist es uns möglich, durch diesen Einsatz tatsächlich nennenswerte Einsparungen im Entwicklungsprozeß zu erreichen? Im folgenden werden wir anhand eines Fallbeispieles aus dem Bereich Telefonvermittlungssysteme diese Fragen beantworten.

2 Die Herausforderung: Konfiguration von Telefonvermittlungssystemen

Die Basisaufgabe eines Telefonvermittlungssystems wie EWSD (Elektronisches Wählsystem Digital) ist es, Telefonverbindungen durchzuschalten. Es werden jedoch zahlreiche zusätzliche Services geboten, wie z.B. ISDN, Online- und Broadband-Services, Videotelefonie, und Videokonferenzschaltungen. Lokale EWSD-Zentralen versorgen bis zu 600.000 Teilnehmer.

Eine große EWSD-Konfiguration besteht aus ungefähr 200 Racks (Schränken), 1.000 Rahmen, 30.000 Modulen, 10.000 Kabeln und 1.000 anderen Einheiten. In der Wissensbasis sind ca. 20 Typen von Racks, 50 Typen von Rahmen, 200 Typen von Modulen, 50 Typen von Kabeln und 50 Typen von anderen „organisatorischen" Einheiten abgebildet. Diese geringe Anzahl an unterschiedlichen Typen konnte dadurch erreicht werden, daß Komponenten durch das Einstellen von Parametern im Konfigurationsprozeß selbst adaptiert werden können.

Module werden in Baugruppenrahmen gesteckt, welche wiederum in Racks montiert sind. Die Module und Rahmen sind durch Kabel miteinander verbunden und bilden eine Netzwerktopologie, die auf der hierarchischen physikalischen Struktur aufgesetzt ist. Zusätzlich gibt es (externe) PCs, Ventilatoren und andere Einrichtungen, die montiert und mit dem Hauptteil des Systems verbunden werden müssen.

Die Spezifikation von gültigen Konfigurationen wird durch Konfigurationsconstraints bestimmt. Typische Beispiele sind:

- Der erste Anschluß jedes Rahmens ist für Stromversorgungsmodule vorgesehen.
- Der Energieverbrauch aller in einem Rahmen montierten Module muß kleiner oder gleich der maximalen Energieabgabe des entsprechenden Stromversorgungsmoduls sein.
- Analoge und digitale Module dürfen in einem Rahmen nicht gemischt werden.
- Ist ein Durchschaltemodul in einem Rahmen gesteckt, dann muß mindestens einer der Anschlüsse contr1 oder contr2 auch mit einer Komponente des Typs DigitalController verbunden sein.

Basierend auf der Komponentenbibliothek und den Konfigurationsconstraints werden gültige Konfigurationen während eines Konfigurationsprozesses generiert. Dieser Prozeß, den wir in der Langversion näher beschreiben, wird durch den Konfigurator LAVA (Lokale Aufbauplanung und Verkabelung für AOSA) unterstützt und automatisiert. LAVA ist auf Basis des domänen-unabhängigen Konfigurationswerkzeugs COCOS (Configuration by Constraint Satisfaction [SHF94, FFH+95]) implementiert, welches die effiziente Entwicklung von wissensbasierten Konfiguratoren für komponentenorientierte Systeme ermöglicht.

3 Wissensrepräsentation

Die Beispielconstraints in Abschnitt 2 zeigen wichtige Anforderungen an die Form von Wissensbasen, z. B. gibt es eine Oder-Verknüpfung im Folgerungsteil der Implikation. Weiters werden Quantoren und Gleichheit in der Formulierung der Konfigurationsconstraints von den Konfigurationsexperten verwendet. Geht man davon aus, daß dieses Wissen möglichst explizit modelliert werden soll, um Wartungsfreundlichkeit und Erklärungsfähigkeit zu gewährleisten, so müssen diese Ausdrucksmittel in der Wissensrepräsentationssprache zur Verfügung gestellt werden. Wissensrepräsentation und Schlußfolgern in LAVA entsprechen im wesentlichen den Ansätzen wie sie in [SHF94, FFH+95] beschrieben wurden und stützen sich auf verschiedene Konfigurationsprojekte in den Bereichen Tonstudiosysteme, Kommunikationstechnik und Eisenbahnsicherungsanlagen.

Unser Ziel war es daher, um breite Anwendbarkeit zu erreichen, ein Wissensrepräsentationsschema zu entwickeln, das so allgemein ist, daß es für verschiedenste Anwendungsdomänen verwendet werden kann und eine explizite und direkte Modellierung dieser Domänen xuläßt. Das Ergebnis dieses Prozesses ist die Sprache ConTalk[1], welche auf der Basis unserer Erfahrung bei der Entwicklung verschiedenster Konfigurationsanwendungen in den Bereichen Telekommunikation, Kontrollsysteme und Audiostudios entstanden ist (insgesamt mehr als 100 Personenjahre).

[1] ConTalk ist eine Weiterentwicklung von LCON [SHF94], die entwickelt wurde, um den Anforderungen einer Produktivumgebung gerecht zu werden. Da ConTalk als Smalltalk-Sprachschnittstelle realisiert ist, basiert die Syntax auf der von Smalltalk.

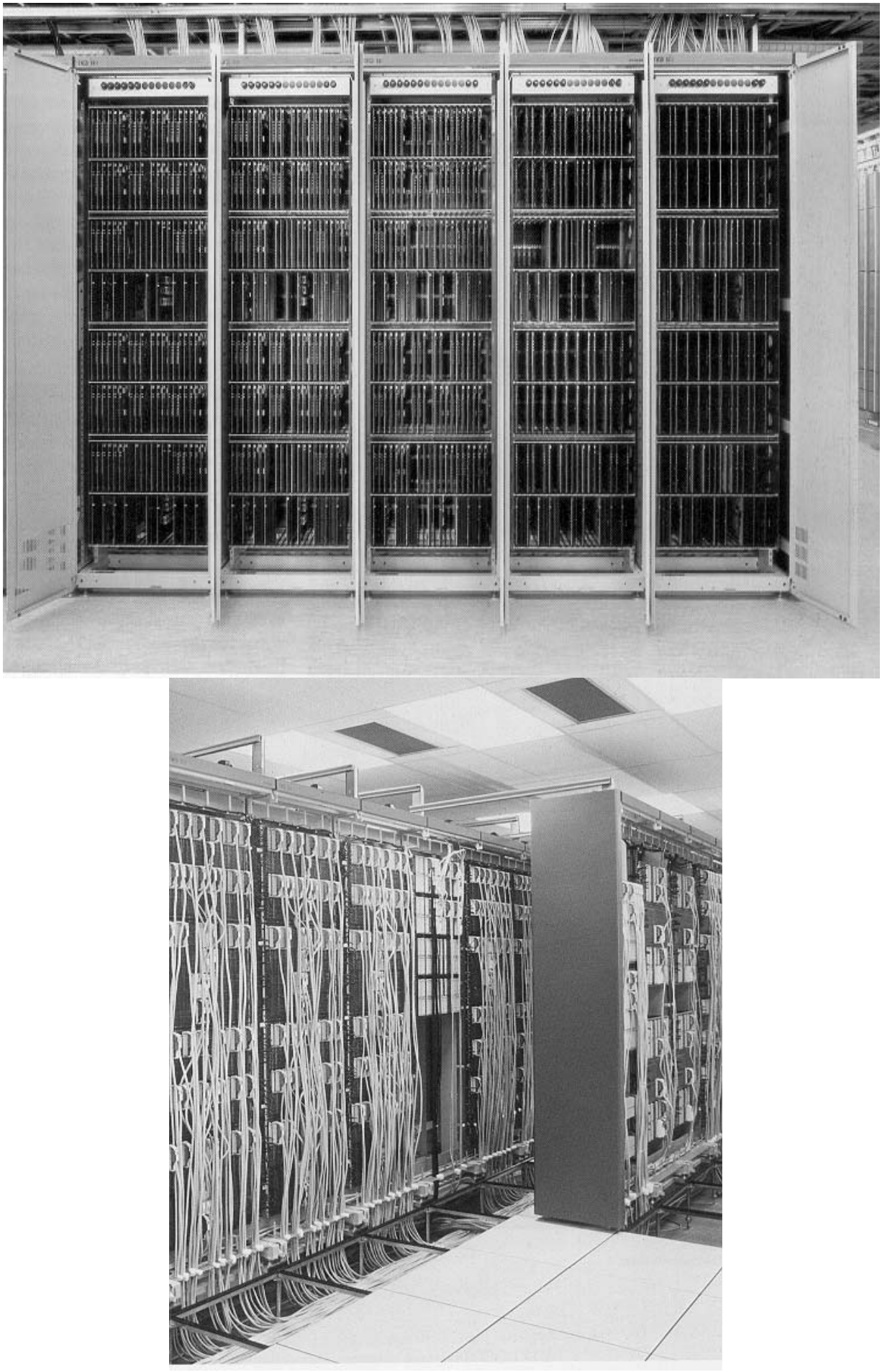


Abbildung 1: Vorder- und Rückansicht eines EWSD-Telefonvermittlungssystems

Komponentenbibliothek

Alle diese Problembereiche haben gemeinsam, daß die Komponententypen, welche zum Bau einer Konfiguration verwendet werden könnten, in der Komponentenbibliothek beschrieben sind. Da im voraus nicht bekannt ist, wieviele Komponenten für eine bestimmte Konfiguration benötigt werden, müssen die Komponenten während des Konfigurationsprozesses nach Bedarf generiert werden.

Die Komponententypen sind in einer Vererbungshierarchie organisiert. Die Wurzel der Hierarchie ist der Typ *Component*. Die Blätter des Baumes sind die konkreten Komponententypen, die für die zu konfigurierenden Systeme erhältlich sind. Eigenschaften und Verhalten eines Komponententyps werden durch seine Attribute, Ports und Constraints definiert.

Attribute

Jeder Komponententyp besitzt eine Menge von *Attribut*definitionen (die Namen, Typ und wahlweise einen Defaultwert festlegen). Beispiele sind:

```
Frame attribute:'name' type: String const: 'F_DLU_A'.
Frame attribute: 'width' type: Number.
```

Ports

Ports dienen zum Herstellen von Verbindungen zwischen zwei Komponenten. Eine Portdefinition für eine Komponente besteht aus dem Namen des Ports, seinem Typ[2] und seinem Wertebereich (der Menge von Komponententypen, die an diesen Port angeschlossen werden könnten). Beispiele von Portdefinitionen sind:

```
Rack port: 's01' type: Port mustConn: #(F_DLU_A).
Rack port: 's02' type: Port mayConn: #(Frame).
```

In diesem Beispiel hat ein Rack zwei Ports: s01, an den ein Rahmen des Typs F_DLU_A angeschlossen sein muß, und s02, an den ein beliebiger Rahmen (F_DLU_A oder F_DLU_B) angeschlossen sein muß. Ports sind immer paarweise miteinander verbunden (oder werden explizit auf 'unverbunden' gesetzt).

Constraints

Constraints stellen ein gebräuchliches Repräsentationsschema für Konfigurationsprobleme dar. Sie sind ein natürlicher Weg, um Kompatibilitätswissen zwischen Komponenten auszudrücken. Außerdem erleichtern ihre Eigenschaften die Wartung von Wissensbasen:

- Derselbe Constraint kann sowohl für das Generieren als auch für das Prüfen einer Konfiguration verwendet werden.
- Constraints bieten reiche Repräsentations- und Argumentationsmöglichkeiten, die so erweitert werden können, daß komplexeres Konfigurationswissen natürlich auszudrücken ist.
- Einfachheit und Deklarativität des zugrunde liegenden Constraintmodells erlauben die Definition einer klaren Semantik.
- Constraints unterstützen eine klare Trennung zwischen strategischem Wissen und Problemwissen sowie Präferenzen, da Strategien und Präferenzen mittels Ordnungen von Variablen und Werten ausgedrückt werden können (z.B. einen Rahmen von links nach rechts auffüllen).

In COCOS werden Constraints immer lokal bei einer einzigen Komponente definiert, von wo aus sie mit Nachbarkomponenten in Beziehung treten und dadurch durch die gesamte Konfiguration navigiert werden kann. Hier ein Beispiel eines Constraints, der mehr als eine Komponente betrifft:

> „Die zwei an den Ports s01 und s02 eines jeden Racks angeschlossenen Rahmens müssen die gleiche Weite haben."
>
> ```
> (Rack connComp: 's01') width
> eq: (Rack connComp: 's02') width.
> ```

aComp connComp: aPort ist ein „Navigationsausdruck" und referenziert die mit dem Port aPort der Komponente aComp verbundene Komponente.

Bestimmte Spezifikationen haben die Form von Requirements. Solche „Ressourcenconstraints" können in ConTalk durch das Definieren von Constraints auf Variablenmengen formuliert werden, wie dies im folgenden Beispiel dargestellt wird:

> „Der Stromverbrauch aller Module in einem Rahmen (dies sind alle Module, die mit den Portset allSwitchingSlots verbunden sind), muß kleiner gleich dem Stromoutput des entsprechenden Stromversorgungsmoduls sein. Beachten Sie, daß das Stromversorgungsmodul in jedem Rahmen am ersten Anschluß slot1 angebracht ist."
>
> ((Frame ports: 'allSwitchingSlots')[3]

[2] Ports in COCOS sind selbst Objekte, die Attribute haben können und in eine Typenhierarchie eingebettet sind.

[3] allSwitchingSlots steht für ein Set von Portnamen,

```
Sum: [ :aPort | aPort connComp
        powerConsumed ][4] )
  Ie: (Frame connComp: 'slot1')
        powerSupplied.
```

Eine andere wichtige Art von Ausdrücken in ConTalk sind Quantoren, die z.B. eine Bedingung definieren, die sich über eine Menge von Ports erstreckt. Grundsätzlich gibt es den All- und den Existenzquantor, wobei der letztere im folgenden Beispiel demonstriert wird:

> *„Gibt es einen Anschluß für Schaltmodule in einem Rahmen, der mit einem Schaltmodul verbunden ist, dann muß mindestens einer der Anschlüsse contr1 und contr2 auch mit einer Komponente des Typs DigitalController verbunden sein.“*

```
((Frame ports: 'allSwitchingSlots')
        Exists: [ :aPort | aPort isConn ])
  Then: [ (Frame ports: #(contr1 contr2))
        Exists: [ :ctrPort | ctrPort isConn
        And:[ctrPort connComp isA:
              DigitalController ] ].
```

Um ConTalk-Constraints leichter lesbar zu machen, kann man anwenderdefinierte Prädikate (eine Art Makros) spezifizieren, welche die Definition eines domänenspezifischen Vokabulars erlauben. Diese anwenderdefinierten Prädikate können in jedem Constraint verwendet werden, um die Formeln zu vereinfachen.

Ein wesentlicher Unterschied zwischen ConTalk und herkömmlichen Constraintsystemen ist die Darstellung des Konfigurationsproblems als *Generatives* CSP (GCSP). Ein generatives CSP erweitert das Constraintnetzwerk während der Suche um zusätzliche Variablen und Constraints, indem bei Bedarf neue Komponenten erzeugt werden. Auf diese Weise können große Systeme konfiguriert werden, ohne künstlich im voraus alle zu erwartenden Komponenten spezifizieren zu müssen.

4 Resultate

LAVA ist nun seit mehr als einem Jahr in Anwendung und ist vollständig in die EWSD-Logistikkette integriert, was die Implementierung verschiedener Interfaces erforderte, angefangen von Schnittstellen zu Anwenderprogrammen, die auf Großrechnersystemen laufen, bis hin zu Excel-Sheets. LAVA selbst läuft derzeit unter Windows NT 4.0. Da wir Smalltalk (Visualworks) als Implementierungssprache verwenden, sind COCOS und LAVA unmittelbar auch für alle Systeme verfügbar, für die Visualworks erhältlich ist, z.B. Sun Solaris und viele Unix-Plattformen.

Beim Anwender von LAVA arbeiten im Schnitt sechs Personen an Konfigurierungsaufgaben. Diese Benutzer widmen ungefähr 20% Prozent ihrer Arbeitszeit dem Konfigurieren von Anlagen. (Andere Tätigkeiten sind Organisationsaufgaben, Planungen für Ressourcen, die die Abteilung disponiert, und Besprechungen bei den Endkunden der Anlagen.)

Die Arbeitsweise mit dem Konfigurator ist interaktiv, also kein Batchdurchlauf. Die Analyse der Arbeitsprozesse ergab, daß diese Arbeitsweise für die Modularität der Anlagen und für den Arbeitsfortschritt am geeignetsten ist, weil die Benutzer zwischen den einzelnen Teilschritten Überprüfungen durchführen und Korrekturen anbringen können. In einem repräsentativen Jahr wurden ungefähr 90 (vorwiegend kleine) Anlagen neu konfiguriert, 60 (vorwiegend große) Anlagen erweitert und 140 Anlagen umgebaut. In den genannten Zahlen sind nachträgliche Umkonfigurierungen aufgrund geänderter Anforderungen nicht enthalten. Bei der Bearbeitung einer Anlage sind des öfteren noch Klärungen erforderlich, wodurch sich die Bearbeitungszeit ausdehnt.

Eine durchschnittliche EWSD-Konfiguration besteht aus ungefähr 25 Racks, 125 Rahmen, 3.500 Modulen, 1,500 Kabeln und 250 anderen Einheiten.

4.1 Einsparungen im Projektierungsprozeß

Der finanzielle Aspekt eines Entwicklungsprojekts ist für die Bewertung des Erfolges entscheidend. Im Falle von LAVA machte sich die gesamte Investition bereits innerhalb des ersten Betriebsjahres bezahlt.

Zusätzlich zu den Vorteilen, die im Konfigurationsprozeß realisiert werden konnten, wurde die Qualität der Konfigurationsergebnisse verbessert und damit die Kosten in jenen Abteilungen reduziert, welche die Ergebnisse dieses Prozesses nutzen. Während des Ladens alter EWSD-Beschreibungen wurden zahlreiche Fehler entdeckt, z.B. falsche Längencodes von Kabeln, was zu Problemen in der Montage und Produktion führte. Diese aus den Qualitätsverbesserungen des Konfigurationsprozesses resultierenden Kostenreduktionen sind in der vorhin beschriebenen Kalkulation der Kosteneinsparung noch nicht enthalten und würden die Beurteilung des Erfolgs von LAVA noch verbessern.

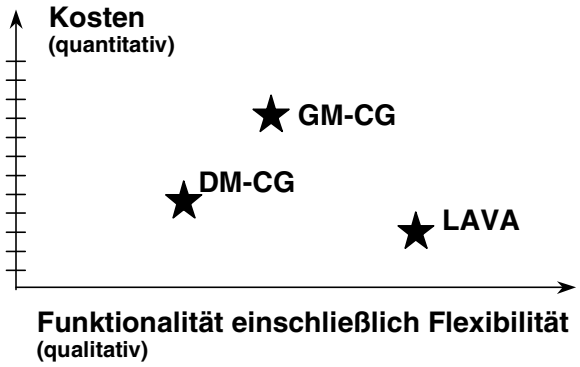
---

die alle jene Anschlüsse bezeichnen, wo Schaltmodule angeschlossen werden können.

[4] In ConTalk definieren die sogenannten, in eckige Klammern gefaßten *blocks*, namenlose Ausdrücke, ähnlich den Lambda-Expressions in LISP. Z.B. evaluiert der Ausdruck

```
aSet Sum:  [ :x | <block body>]
```

den <block body> für alle Elemente der Menge aSet; die Laufvariable x wird der Reihe nach an alle Elemente von aSet gebunden.



Zusätzliche Funktionalität von LAVA im Vergleich zu GM-CG:

- **Interaktive Konfiguration**
- **Graphisches Benutzerinterface**
- **Erklärungen**
- **Automatische Kabelkonfiguration**
- **Wiederverwendung von Baugruppen**
- **Ausdrucksstarke Beschreibungssprache**
- **Erweiterung von bestehenden Konfigurationen**
- **Unterstützung des Urladens von Altbeständen**

DM- & GM-CG: Konfiguratoren, tabellenbasiert, prozedurales Wissen
LAVA: Konfigurator, constraintbasiert, deklaratives Wissen

Abbildung 2: Vergleich von Konfiguratoren

4.2 Ersparnis in der Entwicklung

Obwohl für den Erfolg eines Projekts der Gewinn ein dominanter Faktor ist, ist für die zukünftige Entwicklung von Konfiguratoren die interessantere Frage, ob die verwendete Technologie Vorteile verglichen mit anderen Ansätzen mit sich bringt. Wir können auf eine Historie zurückblicken, in der LAVA die dritte Anwendung war, die im Bereich der automatisierten Konfiguration von EWSD-Systemen entwickelt wurde.

Abbildung 2 zeigt einen Vergleich zwischen LAVA und den beiden früheren Konfiguratoren DM-CG und GM-CG in Bezug auf Kosten und Funktionalität (einschließlich der Flexibilität für den Anwender während des Konfigurationsvorganges).

Im Gegensatz zum Kostenvergleich ist ein aussagekräftiger Vergleich von Funktionalität in unserem Fall nur qualitativ möglich. DM-CG vs. GM-CG waren spezielle EWSD-Konfiguratoren (im Gegensatz zum generellen Konfigurationswerkzeug COCOS), wobei GM-CG eine bedeutend höhere Flexibilität in der Wartung von Konfigurationswissen bietet. GM-CG erlaubt die Modellierung von Konfigurationswissen mittels propositionaler Logik erweitert um Defaulteigenschaften. Komplexe Constraints, die Verbindungen, Attributwerte, Prädikate und Quantifizierung verwenden, konnten jedoch nicht formuliert werden.

Die Hauptvorteile von LAVA gegenüber GM-CG sind in Abbildung 2 aufgelistet. Ein zusätzliches Ergebnis des LAVA-Projekts ist ein erweitertes Konfigurationswerkzeug, das für die Realisierung weiterer Konfiguratoren verwendbar ist.

Wenn man alle diese zusätzlichen Ergebnisse betrachtet, ist es bemerkenswert, daß verglichen mit GM-CG eine Kostenreduktion von 66% erzielt werden konnte. Selbst wenn wir auch den für das vorhergehende COCOS Forschungsprojekt verwendeten Aufwand mit einbeziehen (einschließlich der Forschung in theoretischen Themen der Constraint Satisfaction), erzielen wir noch immer eine Kostenreduktion von ungefähr 50%.

Im Vergleich zu DM-CG ist die Funktionalität von DM-CG ein Bruchteil von jener von LAVA, nichtsdestotrotz sind die Kosten von LAVA um 30% geringer als jene von DM-CG.

4.3 Vorteile in der Wartung

Die Wartungskosten hängen von der Häufigkeit und vom Umfang der Änderungen in der Komponentenbibliothek ab. Zum gegenwärtigen Zeitpunkt ist eine solide quantitative Metrik für das Messen dieser Menge nicht verfügbar. Im Gegensatz zu direkten Veränderungen der Komponentenbibliothek kann die Funktionalität des Systems auch durch andere Anforderungen verändert werden, z.B. Unterstützung neuer Fileformate. Wir betrachten jedoch nur Veränderungen der Komponentenbibliothek, da diese Veränderungen die für uns interessanten sind, sowohl aus wirtschaftlicher Sicht als auch für die Arbeiten auf dem Gebiet der Konfigurationstechniken.

Der Vergleich anhand konkreter Änderungswünsche zeigt, daß bei DM-CG und GM-CG ein gewisser Prozentsatz an kleineren Änderungswünschen zwar

mittels Anpassung der Tabelleninformation befriedigt werden konnten, daß aber zahlreiche Veränderungen des Produktkataloges Änderungen des DM-CG- und des GM-CG-Programmcodes auslösten, was zu größeren Wartungsaufwendungen führte. In LAVA konnten alle diese Wünsche durch kleinere Veränderungen in der Wissensbasis erfüllt werden. Außerdem ist die LAVA-Wissensbasis kompakter. Änderungen in der Komponentenbibliothek, welche eine Änderung an vielen Punkten in DM-CG und GM-CG auslösen, verursachen nur einen einzigen Änderungspunkt in LAVA.

Weiters verglichen wir die Fähigkeiten von COCOS mit einem anderen wissensbasierten Konfigurationssystem, das sich im Zustand der praktischen Anwendung befindet. Dieses System verfolgt den regelbasierten Ansatz [BO89], wobei durch framebasierte Repräsentationsfähigkeiten Verbesserungen eingebracht wurden. Wiederum zeigte unsere Bewertung ähnliche Ergebnisse. Einfache Konfigurationsconstraints neigen zu umfangreichen Beschreibungen in der Regelbasis. Durch die Repräsentationsformalismen von COCOS konnte eine mehr als fünffache Reduktion für Teile der Wissensbasis erzielt werden. Veränderungen in der Komponentenbibliothek lösen häufig viele Veränderungen in der Wissensbasis des regelbasierten Systems aus, was für die untersuchten Fälle in COCOS vermieden werden könnte.

4.4 Organisatorische Vorteile

LAVA macht das Wissen über EWSD-Konfiguration explizit. Daher können neue Mitarbeiter im Vertrieb die essentiellen Eigenschaften von EWSD schneller und in einer besser strukturierten Form lernen, und das Konfigurationswissen bleibt erhalten, so daß zu einem späteren Zeitpunkt direkt darauf zugegriffen werden kann.

5 Verwandte Arbeiten

Es gibt zahlreiche vielversprechende Vorschläge (welche teilweise in wirtschaftlich sehr erfolgreichen Werkzeugen integriert sind), wie Entwicklung, Wartung und Funktionalität von Konfiguratoren verbessert werden können. Obwohl solche Werkzeuge in verschiedenen Konfigurationsdomänen anwendbar sind, hat die tatsächliche Projekterfahrung gezeigt, daß sie in Bezug auf Ausdrucksstärke und Effizienz verbessert werden müssen: Dies gilt auch für weniger komplexen Aufgaben, als wir im EWSD-Bereich antreffen.

Im industriellen Sektor sehen wir derzeit drei gängige Ansätze zum Lösen von Konfigurationsproblemen. Neben dem constraintbasierten Ansatz ist der ressourcenbasierte Ansatz [HJ91] ein verbreitetes Konzept, um verschiedenste Formen von Konfigurationswissen auszudrücken. In unserem Ansatz gehen wir über dieses Konzept hinaus, so daß komplexe Strukturbeziehungen entsprechend der Sicht des Fachexperten dargestellt werden können. Ein weiterer vielversprechender Ansatz zur Konfiguration sind Description Logics, welche von einer umfangreichen Grundlage theoretischer Forschung profitieren können. Ihre Anwendung ist in [WWB+93] für die Konfiguration von Telekommunikationssystemen beschrieben, die jedoch bedeutend kleiner als unsere Anwendungsdomäne sind. Auch bei diesem Ansatz stellt sich die Frage, wie komplexe Strukturbeziehungen aus Sicht des Anwenders möglichst einfach dargestellt werden können.

6 Resümee

Unsere Anwendung zeigt, daß ausdrucksstarke Wissensrepräsentationssprachen in umfangreichen Domänen erfolgreich verwendet werden können. Für diese Domänen trägt eine ausdrucksvolle Wissensrepräsentation wesentlich zur Verbesserung der Wartung von Konfiguratoren bei. Durch die Verwendung eines reichhaltigen Repräsentationsformalismus kann das Konfigurationswissen direkt, ohne aufgrund von fehlenden Spracheigenschaften notwendige Umwege, spezifiziert werden.

Eine wichtige Frage ist natürlich, wie zufriedenstellende Effizienz erreicht werden kann. In unserem Fall war es ausreichend, Constraint-Satisfaction-Algorithmen und eine Aufteilung nach funktionalen Gesichtspunkten auszunutzen sowie genügend Aufwand bezüglich der Verfeinerung von Algorithmen und Datenstrukturen zu betreiben. Sehr einfache Heuristiken (wie z.B. Port- und Komponententyp-Ordnungen) reichen, um die Suche zu steuern. Die Rolle der Constraints ist es, den Suchprozeß durch das Eingrenzen der Anzahl von Auswahlmöglichkeiten zu unterstützen, wobei anwendungsunabhängige Heuristiken (wie z.B. Forward checking) benutzt werden.

Wie unsere Erfahrung zeigt, gilt dies auch für andere Domänen, an denen wir arbeiten. Jedoch wäre sowohl aus anwendungsorientierter als auch aus wissenschaftlicher Sicht von Bedeutung, ein besseres Verständnis der Eigenschaften von Konfigurationsdomänen zu erlangen, die solch ein günstiges Verhalten zeigen.

Aus dem Blickwinkel der wissenschaftlichen KI-Gemeinschaft glauben wir, einen Schritt vorwärts gemacht zu haben, indem wir gezeigt haben, daß Constraint Satisfaction eine vielversprechende Technik für reale umfangreiche Konfigurationsanwendungen ist.

Aus dem Blickwinkel von Software-Entwicklern, die persönlich für das Erreichen von Kosten-, Zeit- und Qualitätsanforderungen verantwortlich sind, ist es notwendig, bereits vom Beginn des Projekts an die

richtigen Technologien und Werkzeuge zu verwenden, da Kosten- und Zeitbeschränkungen typischerweise zu stark sind, um die Technologie während eines Projektes zu ändern. Aus diesem Grund werden in Entwicklungsprojekten neue Technologien nur mit größter Vorsicht angenommen. Wir haben gezeigt, daß constraintbasierte Techniken erfolgreich in produktiven Systemen angewendet werden können, wobei wir Systeme konfigurieren, die zu den komplexesten derzeit in Betrieb stehenden elektronischen Systemen gehören.

Die Anwendung dieser Technik aus dem Bereich der wissensbasierten Konfiguration hat sich in Bezug auf die Senkung von Kosten und Durchlaufzeiten in der Softwareentwicklung und Wartung positiv ausgewirkt. Gleichzeitig konnte die Funktionalität von Konfiguratoren erhöht werden. Wir haben damit zum Ziel beigetragen, eine in der KI anerkannte Technologie wie die Constraint Satisfaction auf eine breitere Anwendungsbasis in der Softwareentwicklung zu stellen.

Bibliographie

BO89 Virginia E. Barker and Dennis E. O'Connor. Expert systems for configuration at Digital: XCON and beyond. *Comm. ACM*, 32 (3): 298-318, 1989.

FFH+95 Gerhard Fleischanderl, Gerhard Friedrich, Alois Haselböck, and Markus Stumptner. Knowledge-based Configuration of Switching Systems. In *Proc. 15th Intl. Switching Symposium (ISS-95)*, pp. 158-162, Berlin, Germany, April 1995.

HJ91 M. Heinrich and E.W. Jüngst. A resource-based paradigm for the configuring of technical systems from modular components. In *Proc. 7th CAIA*, pages 257-264, February 1991.

SHF94 M. Stumptner, A. Haselböck, and G. Friedrich. COCOS - A Tool for Constraint-Based, Dynamic Configuration. In *Proceedings of the 10th IEEE Conference on Artificial Intelligence Applications (CAIA)*, pages 373-380, San Antonio, Texas, March 1994.

WWB+93 Jon R. Wright, Elia S. Weixelbaum, Karen Brown, Gregg T. Vesonder, Stephen R. Palmer, Jay I. Berman, and Harry G. Moore. A Knowledge-Based Configurator that Supports Sales, Engineering, and Manufacturing at AT&T Network Systems. In *Proc. 5th IAAI*, AAAI Press, 1993.

Erfahrungen beim Transfer von Konfigurierungsmethoden

Andreas Günter
Technologie-Zentrum Informatik
Universität Bremen
Postfach 330440
28334 Bremen
guenter@tzi.de

Christian Kühn
Daimler-Benz AG
Forschung und Technologie
HPC T721
70546 Stuttgart
christian.kuehn@dbag.stg.daimlerbenz.com

Kurzfassung:
Für den Transfer von Forschungsergebnissen zur Lösung von Konfigurierungsproblemen werden einige wesentliche Aspekte behandelt. Dazu zählen neben der Analyse des Anwendungsproblems, Machbarkeits- und Wirtschaftlichkeitsüberlegungen u.a. auch die Auswahl von Software-Tools, Strategien zur schrittweisen Einführung und die Integration in die bestehende Organisationsstruktur und vorhandene Informationsverarbeitung.

1 Einleitung

Es gibt kein Patentrezept für den Transfer von Forschungsergebnissen in Anwendungen, dies gilt auch für Konfigurierungsaufgaben. In einer allgemeinen Betrachtung werden wir auf die einzelnen Aspekte des Transfers von Informatiktechniken eingehen: der Analyse des Anwendungsproblems, der Verwendung von Software-Tools, Personen und deren Wissen und Einführungsstrategien. Wir lassen uns dabei durch ein kürzlich bearbeitetes Beispiel leiten. Dieses befaßt sich mit der Auslegung und Parametrierung von Antriebssystemen und ist ein komplexes Konfigurierungsproblem, für das es zur Zeit keine Standardlösungen gibt. Desweiteren verfügen wir über langjährige Erfahrung aus der Analyse und Modellierung diverser Konfigurierungsprobleme aus dem technischen Bereich.

2 Schritte bei der Konzeption, Entwicklung und der Einführung von Anwendungssystemen

Bei der Entwicklung eines integrierten Softwaresystems mit einem wissensbasierten Konfigurierungssystem als wesentliche Komponente sind die Grundsätze des „klassischen“ Software-Engineering zu befolgen, auf die wir hier nicht weiter eingehen. Daneben existieren für wissensbasierte Systeme eigene Vorgehensmodelle, die meist auf den zwei grundsätzlich verschiedenen Ansätzen des prototypbasierten und modellbasierten Vorgehens aufbauen (vgl. [Curth et al. 1991; Lenz 1991]). Eine Vielzahl von Vorgehensmodellen für die Expertensystementwicklung versucht beide Ansätze zu vereinen. Beispiele für entsprechende Vorgehensweisen sind zu finden in: [Curth et al. 1991; Guida u. Tasso 1989; Harmon u. King 1989; Lebsanft u. Gill 1987].

Die folgenden Schritte sind nicht in jedem Anwendungsbereich notwendig, müssen nicht in der strengen Reihenfolge bearbeitet werden und haben auch zahlreiche Rückkopplungen. Trotzdem erscheint uns dies als eine sinnvolle Strukturierung (vgl. [Karbach u. Linster 1990]):

- Analyse des Anwendungsproblems
- Definition eines Leitbeispiels
- Wissensakquisition
- Machbarkeitsuntersuchung

- Wirtschaftlichkeitsbetrachtungen
- Toolauswahl
- Demonstrationsprototyp
- Spezifikation und Realisierung
- Stufenweise Einführung
- Integration

2.1 Analyse des Anwendungsbereiches

Dieser Punkt zieht sich durch den gesamten Prozeß. Gerade zu Beginn ist die Analyse des Anwendungsbereiches und seines Umfeldes von zentraler Bedeutung. Notwendig sind hier Gespräche mit Beteiligten sowie das Auswerten von Unterlagen und der zugrundeliegenden Geschäftsprozesse. Aus Sicht der Konfigurierung ist ein zentrales Ziel der Analyse die Klassifikation und Einordnung des Problems sowie die Abbildung auf abstrakte Konfigurierungskriterien. Im Beispiel ist dies u.a. die Erkenntnis, daß hier ein komplexes Parametrierungsproblem vorliegt, bei dem die Konfigurierung der Komponenten hingegen relativ einfach erscheint.

2.2 Leitbeispiel

Anwender und Konfigurierer kommen aus unterschiedlichen „Welten". Die Diskussion über ein geeignetes Beispiel kann die Kommunikation deutlich verbessern. Das Leitbeispiel dient als Standard-Szenario, welches jeder der Beteiligten kennt und an dem leicht etwas aufgezeigt werden kann. Dabei dient das Leitbeispiel auch zur beispielhaften Darstellung der Lösungsmöglichkeiten. Das Leitbeispiel kann/sollte auch in Form eines Demonstrationsprototypen realisiert werden.

2.3 Wissensakquisition

Für die Wissensakquisition bestehen eine Reihe von Konzepten, hierzu sei auf [Karbach u. Linster 1990; Lenz 1991; Meyer-Fujara et al. 1995; VDI 1992] verwiesen. Wichtig ist, daß der Prozeß der Wissensakqusition nicht als einmalige Tätigkeit verstanden wird, sondern kontinuierlich durchgeführt werden kann, um der Dynamik des abzubildenden Wissens im Unternehmen gerecht zu werden.

Dementsprechend sollte anstelle eines ausschließlich indirekten Wissenserwerbs (bei dem die Wissensakquisition durch einen Knowledge Engineer auf der Basis von Experteninterviews durchgeführt wird) ein direkter Wissenserwerb angestrebt werden, bei dem die Fachleute selbst – also insbesondere auch „Nicht-Informatiker" – ihr Wissen auf das System übertragen und das gespeicherte Wissen insbesondere auch warten können. Dieses ist ein Aspekt der Wissensakquisition, der sich über die gesamte Lebensdauer eines Systems bezieht. Wichtig sind hierzu geeignete Wissensakquisitions-Schnittstellen. Auch sollte – soweit es möglich ist – eine Kopplung bzw. ein automatischer Abgleich von vorhandenen Datenbanken und Wissensbasis möglich sein, um Datenredundanz und mehrfache Datenpflege zu vermeiden.

2.4 Machbarkeitsuntersuchung

Ist das Problem technisch lösbar? Dies zu klären ist Aufgabe einer Machbarkeitsstudie. Hier wird das Problem klassifiziert, bekannte Methoden aus der Literatur zu dessen Lösung vorgestellt und ggf. aufgezeigt, an welchen Stellen noch Defizite vorliegen.

2.5 Wirtschaftlichkeitsbetrachtungen

Ist das Problem unter wirtschaftlichen Gesichtspunkten sinnvoll lösbar? Hierbei müssen die Vorteile für ein Unternehmen den Kosten gegenübergestellt werden. Im Beispiel sind die wesentlichen wirtschaftlichen Ziele: Senkung der Entwicklungs- und Angebotskosten, Reduzierung der Fehlerraten, Einbeziehung bereits erfolgreich gelöster Anwendungen, Unterstützung beim schnellen Wechseln von Produkten und Produktgenerationen, Sicherung des firmeneigenen Wissens und Standardisierung der Anwendungslösungen.

2.6 Toolauswahl

Gibt es auf dem Markt Software-Werkzeuge, mit denen das Anwendungssystem oder wesentliche Teilprobleme bereits gelöst werden können? Falls

ja, dann sollte möglichst auf diese zurückgegriffen werden.

2.7 Demonstrationsprototyp

Dieser dient zur beispielhaften Realisierung von ausgewählten Aspekten, z.B. dem Leitbeispiel. Ein Prototyp hat oftmals eine ganz andere Wirkung als eine Machbarkeitstudie auf Papier. Dabei kann unterschieden werden zwischen einem Prototyp zur Demonstration der späteren Arbeit mit dem System (Benutzungsoberfläche) oder der Kernfunktionalität (Konfigurierungstechniken).

Neben Demonstrations- und Forschungsprototypen können in späteren Phasen Feldprototypen für den Test in der Anwendungsumgebung und Produktionsprototypen zur Realisierung korrekten und effizienten Systemverhaltens folgen. Eine Übersicht über die verschiedenen Stadien von Prototypen liefern z.B. [Lenz 1991; Watermann 1986].

2.8 Spezifikation und Realisierung

Der „klassische“ Teil der Software-Entwicklung, hierauf möchten wir nicht eingehen.

2.9 Stufenweise Einführung

Die Entwicklung eines komplexen integrierten Systems braucht seine Zeit. Ziel einer Einführung muß es sein, bereits frühzeitig geeigente Teilversionen dem Anwender zur Verfügung zu stellen. Eine derartige stufenweise Einführung erhöht deutlich die Akzeptanz des einzuführenden Systems. Besondere Aspekte sind hier u.a. die Schulung und der Ersteinsatz bei ausgewählten Personen.

2.10 Integration

Wichtig für eine erfolgreiche Einführung in die betriebliche Praxis ist die Realisierung einer integrierten Informationsverarbeitung. Dabei sind eine Reihe unterschiedlicher Erfordernisse zu beachten: In der VDI-Richtlinie *Expertensysteme in betriebswirtschaftlichen Anwendungen* werden drei „Dimensionen“ der Integration unterschieden (vgl. [VDI 1992]):

- *Technische Integration:*
  Einbindung in die bestehende DV-Welt, also das Zusammenspiel mit der bisher vorhandenen Hard- und Software im Unternehmen.

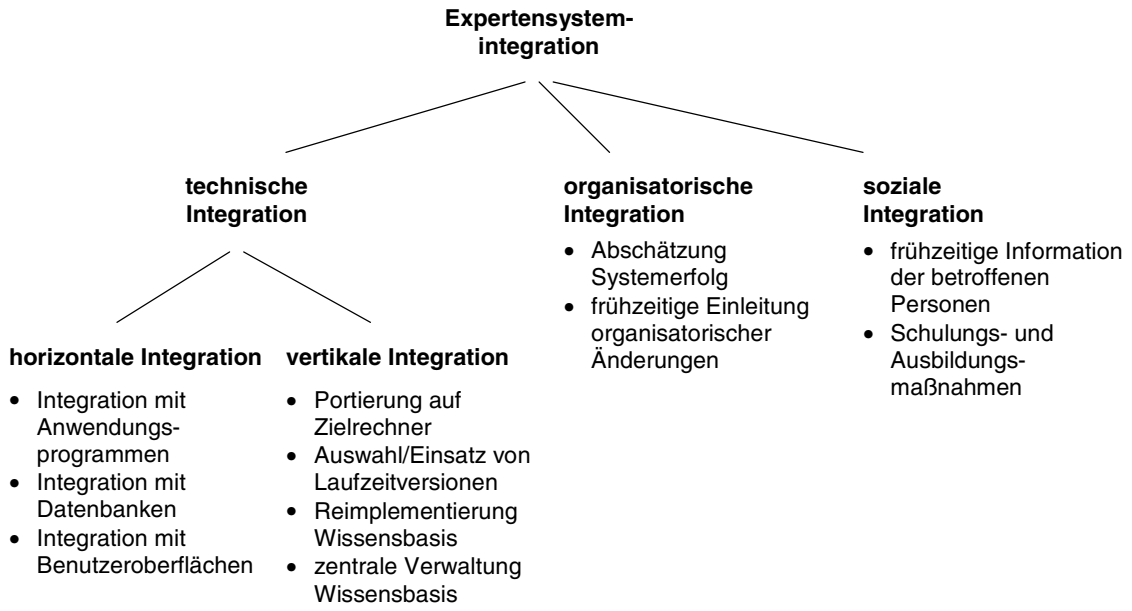


Abb. 1: Integration eines wissensbasierten Systems in die betriebswirtschaftliche Praxis nach [Timm 1994; VDI 1992]

- *Organisatorische Integration:*
  Auswirkungen der Systemeinführung auf die Abläufe im Unternehmen: die Mitarbeiter, die Geschäftsprozesse und die Organisationsstruktur des Unternehmens.

- *Soziale Integration:*
  Aufzeigen der Vorteile und transparente Darstellung des Systems zur Beseitigung von Befürchtungen und Barrieren der betroffenen Mitarbeiter.

Die technische Integration wird weiter unterschieden in eine horizontale und eine vertikale Integration. Mit der horizontalen Integration ist die Anbindung an Anwendungsprogramme (durch Client/Server-Konstellationen, Interprozeß-Kommunikation, Unterprogrammaufrufe, etc.), an Datenbanken (physische Integration oder Datenbank-Downloads) sowie der Benutzeroberflächen (Einbindung in ein vorhandenes Oberflächenkonzept) gemeint. Bei der vertikalen Integration wird die Distribution des wissensbasierten Systems an die Benutzer betrachtet (vgl. Abbildung 1).

3 Erfahrungen und Zusammenfassung

Nachfolgend eine zusammenfassende Kommentierung unser Erfahrungen:

- *Schnelle Analyse*
  Bereits im ersten Gespräch muß man bereits eine erste Einordnung des Problems treffen können und einige Aussagen zu Lösungsmöglichkeiten machen. Es ist zu diesem Zeitpunkt beiden Seiten klar, daß noch nichts Verbindliches gesagt werden kann. Der erste Eindruck und die Reaktion darauf ist aber wichtig. Dies bedingt, daß man Erfahrungen in der Lösung von Konfigurierungsproblemen und der Analyse von Anwendungen hat. Als Informatiker hat man häufig die Aufgabe, die Probleme „anderer Leute“ zu lösen. Dies setzt Kommunikationsfähigkeit und eine Dienstleistungsorientiertheit voraus. Beides spielt in der universitären Ausbildung von Informatikern kaum eine Rolle.

- *Technisch machbar*
  Nicht alles, was technisch machbar ist, sollte auch realisiert werden. Was realisiert wird, sollte sich an wirtschaftlichen Überlegungen, dem betrieblichen Umfeld und der Einführungsstrategie orientierten.

- *Tools vs. Individualsoftware*
  Individualsoftware ist kostenintensiv, sowohl in der Entwicklung als auch der Wartung. Falls möglich sollte man Software-Tools verwenden. Dies ist nicht gerade eine neue Erkenntnis, wird aber häufig nicht berücksichtigt.

- *Bedeutung von (universitären) Prototypen*
  Im Beispiel haben wir mit KONWERK [Günter 95] einen Prototypen realisiert, wobei klar war, daß dieser keinen Eingang in das Anwendungssystem finden würde. Trotzdem ist ein derartiger Prototyp eine überzeugende Argumentationshilfe. Dies gilt unser Meinung nach auch dann, wenn der Prototyp auf einem universitären Werkzeug basiert. Software-Werkzeuge sind für den Technologietransfer ein entscheidendes Hilfsmittel (vgl. [Neumann 1995]).

- *Frühzeitige Einbeziehung der Benutzer*
  Was erwarten die späteren Benutzer? Eine Frage, die oftmals erst zu spät gestellt wird. Eine frühzeitige Diskussion mit den Benutzern verbessert das System und insbesondere auch die Akzeptanz.

- *Software-ergonomische Betrachtungen*
  Die Benutzbarkeit statt der Funktionalität steht im Vordergrund der Ergonomie. Auch durch die Einbeziehung der Benutzer in die frühen Phasen der Systementwicklung können deren Anforderungen wesentlich besser realisiert werden.

- *Integration*
  Konfigurierungssysteme werden in einem Kontext angewendet. Eine Anbindung von Datenbanken, CAS-Systemen oder Internet/ Intranet ist oftmals notwendig und muß bereits im Vorfeld eingeplant werden.

4 Literatur

[Curth et al. 1991] M. Curth, A. Bölscher, B. Raschke. *Entwicklung von Expertenssystemen*, Carl Hanser Verlag, München/Wien, 1991

[Guida u. Tasso 1989] G. Guida, C. Tasso. *Building Expert Systems: From Lifecycle to Development Methodologies and Tools*, S. 3-24, Amsterdam, 1989

[Günter 95] A. Günter (Hrsg.). *Wissensbasiertes Konfigurieren – Ergebnisse aus dem Projekt PROKON*, infix Verlag, St. Augustin, 1995

[Harmon u. King 1989] P. Harmon, D. King. *Expertensysteme in der Praxis – Perspektiven, Werkzeuge, Erfahrungen*, 3., akt. u. erg. Aufl., München/Wien, 1989

[Karbach u. Linster 1990] W. Karbach, M. Linster. *Wissensakquisition für Expertensysteme – Techniken, Modelle und Softwarewerkzeuge*, Carl Hanser Verlag, München/Wien, 1990

[Lebsanft u. Gill 1987] W. Lebsanft, U. Gill. *Expertensysteme in der Praxis: Kriterien für die Verwendung von Expertensystemen zur Problemlösung*, in: S. E. Savory (Hrsg.): *Expertensysteme: Nutzen für Ihr Unternehemen – Ein Leitfaden für Entscheidungsträger*, S. 135-149, München/Wien, 1987

[Lenz 1991] A. Lenz. *Knowledge Engineering für betriebliche Expertensysteme: Erhebung, Analyse und Modellierung von Wissen*, Dissertation, Universität Köln, erschienen im Deutschen Universitäts-Verlag, Wiesbaden, 1991

[Meyer-Fujara et al. 1995] J. Meyer-Fujara, F. Puppe, I. Wachsmuth. *Expertensysteme und Wissensmodellierung*, in: G. Görz (Hrsg.): *Einführung in die künstliche Intelligenz*, 2. Auflage, Addison-Wesley Publishing, S. 705-753, Berlin, 1995

[Neumann 1995] B. Neumann. *KONWERK – Technologietransfer mit einem Werkzeugsystem*, in: A. Günter (Hrsg.). *Wissensbasiertes Konfigurieren – Ergebnisse aus dem Projekt PROKON*, infix Verlag, St. Augustin, S. 61-67, 1995

[Timm 1994] H. Timm. *Gestaltungsprinzipien zur Erhöhung des Einsatznutzens von Expertensystemen in der betrieblichen Praxis*, Dissertation, Universität Duisburg, erschienen im Europäischen Verlag der Wissenschaften, Frankfurt am Main, 1994

[VDI 1992] Verein Deutscher Ingenieure: *Expertensysteme in betriebswirtschaftlichen Anwendungen (VDI 5006 Entwurf)*, VDI-Richtlinie „Bürokommunikation“, 1992

[Watermann 1986] D. A. Watermann. *A Guide to Expert Systems*, Reading, Massachusetts, 1986

ON RESOURCE-BASED CONFIGURATION—RENDERING COMPONENT-PROPERTY GRAPHS

Oliver Niggemann* **Benno Stein*** **Michael Suermann***

* *Department of Mathematics and Computer Science / Knowledge-based Systems*
University of Paderborn, D–33095 Paderborn, Germany
email: {murray / stein / michel}@uni-paderborn.de

Abstract: In a broader sense, this paper is on knowledge acquisition support for configuration problems. Actually it concentrates on resource-based configuration—precisely: on the visualization of component-property graphs. A component property graph is a directed graph in first place, which defines nodes (components, functionalities) and directed edges (relations between components and functionalities).
Each resource-based configuration problem defines such a component-property graph. From a knowledge engineering viewpoint, the visualization of this graph is of great value: It can illustrate functional cause-effect chains within complex technical systems, exhibit crucial points in the modeling, or organize the knowledge base of a large system into useful parts.
This paper provides an answer to the following question: Given a knowledge base containing a resource-based component model—in which way and to which quality can the underlying component-property graph be drawn automatically?

Keywords: configuration, graph visualization, clustering, heuristic search

1. INTRODUCTION

Configuration is the process of composing a technical system from a predefined set of objects. This process relies on a particular component model, which is a useful abstraction of the domain and the technical system to be composed (Tong, 1987), (Stein, 1995).

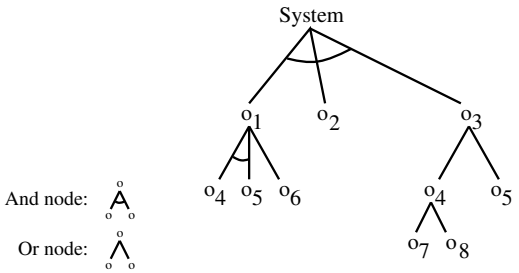


Figure 1: *And-Or tree of some technical system.*

In this context the visualization of a configuration knowledge base of a given configuration problem actually means to visualize the underlying component model. Such a visualization should not happen by a universal approach but exploit meta knowledge concerning the type of component model under investigation. E.g., hierarchical component models, which form the basis of skeletal configuration problems, are often visualized in the form of And-Or trees. Figure 1 gives an example.

Also resource-based component models can be visualized as graphs, so-called component-property graphs; see Figure 2 for an example. Component-property graphs will be discussed in greater detail below.

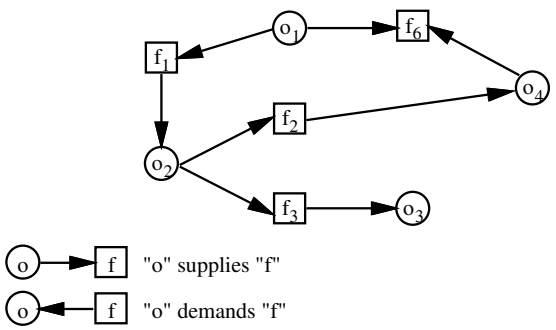


Figure 2: *Component-property graph of a technical system.*

Matter of the paper in hand is the visualization of resource-based component models, so to speak, of the related component-property graphs. Why is such a vi-

sualization useful? The main reasons lie in acquisition and maintenance purposes:

Resource-based component models are defined locally, component by component, and normally they are formulated textually. A consequence is that the global view, which a knowledge engineer has developed in his mind, will go lost sooner or later in the course of the acquisition process. Moreover, knowledge bases are maintained by several persons, and, modifications and revisions of a knowledge base may occur a long time after its initial building.

In this connection a visualization of the component-property graph is of great value. It can illustrate functional cause-effect chains within complex technical systems, exhibit crucial points in the modeling, or organize the knowledge base of a large system into useful parts. This wish immediately raises the question: Given a knowledge base containing a resource-based component model—in which way and to which quality can the underlying component-property graph be drawn automatically?

This paper gives answers to these questions; it is organized as follows. Section 2 provides a short introduction to the resource-based configuration approach. Section 3 presents some generic issues respecting graph visualization, while section 4 shows how component-property graphs, implicitly defined in a knowledge base, can be rendered properly. Section 5 compares our visualization results to those of existing tools.

2. RESOURCE-BASED CONFIGURATION

This section provides a short introduction to resource-based configuration.

2.1 *The Resource-Based Component Model*

The resource-based component model establishes an abstraction level that reduces a domain to a set of components described by simple functionality-value-pairs. More precisely, all technical properties that are relevant for the configuration process form a set of resources (functionalities), which are supplied or demanded by the components.

E. g. when configuring a small technical device such as a computer, one property of power supply units could be their power output, and one property of plug-in cards could be the cards' power consumption. Both properties are reflected by the resource "power": A power supply unit *supplies* some power value, while a plug-in card *demands* some power value. Figure 3 depicts some resource-based descriptions of computer components.

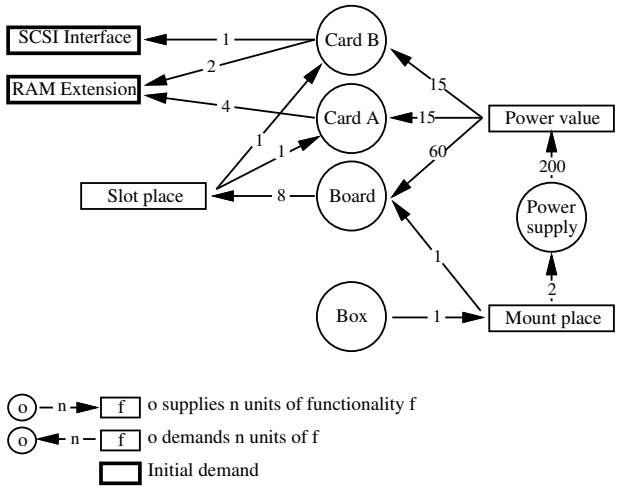


Figure 3: *Resource-based component descriptions*

Note that a component-property graph as shown in Figure 3 represents a simplified functional model of the domain. Actually, resource-based configuration means the instantiation and simulation of such a functional model.

The resource-based component model is suitable for a configuration problem if the following conditions are fulfilled: (*i*) structural information plays only a secondary role, (*ii*) the components can be characterized by resources that are supplied or demanded, and (*iii*) the components' properties are combined in order to provide the system's entire functionality.

A precise specification of the resource-based configuration problem and its solution can be found in (Stein, 1995).

2.2 *The Configuration Process*

Main purpose of this subsection is to round off the introduction of resource-based configuration; it is of secondary importance with respect to visualization aspects.

If there exists a configuration C that solves a resource-based configuration problem, C can be determined by means of the balance algorithm. This algorithm operationalizes a generate-and-test strategy and has been operationalized in the configuration systems COSMOS, CCSC, AKON, and MOKON (Heinrich and Jüngst, 1991), (Laußermair and Starkmann, 1992),

(Weiner, 1991), (Stein and Weiner, 1991). The generate part, controlled by propose-and-revise heuristics or simply by backtracking, is responsible for selecting both an unsatisfied functionality f and a set of objects that supply f. The test part simulates a virtual balance. A functionality (resource) is called unsatisfied, if its supplied amount x is smaller than its demanded amount y.

Basically, configuration works as follows. First, the virtual balance is initialized with all demanded functionalities, and C is set to the empty set. Second, with respect to some unsatisfied f, an object set is formed; the related supplies and demands are added to the corresponding functionalities of the balance, and C is updated by the object set. Third, it is checked whether all functionalities are satisfied. If so, C establishes a solution of the configuration problem. Otherwise, the configuration process is continued with the second step.

3. VISUALIZING GRAPHS

3.1 *Generic Concepts*

When visualizing graphs the question about a definiton of a good or lucid layout plays a key role. Important criteria for such a layout are:

- No two vertices should intersect, and
- neither should an edge and a vertice.
- Having placed all vertices, edges should be drawn by straight lines.
- The angle between two intersecting edges should be obtuse.
- Parallel edges should not be too close.
- Clusters in the graph should be identifiable as such.

Note, however, that both the criteria and their importance depend on the actual design problem. Various approaches for visualizing graphs exist; some visualize graphs by constructing just one layout, others construct several possible layouts and choose the best. The algorithms also vary in their usage of domain knowledge: some are general graph visualization tools, but many, like ours, exploit features of the domain.

3.2 *More on Component-Property Graphs*

In component-property graphs, components point only to functionalities and vice versa. As a consequence, component-property graphs are bipartite. Figure 4 shows a different layout of the graph of Figure 2 that emphasizes this characteristic.

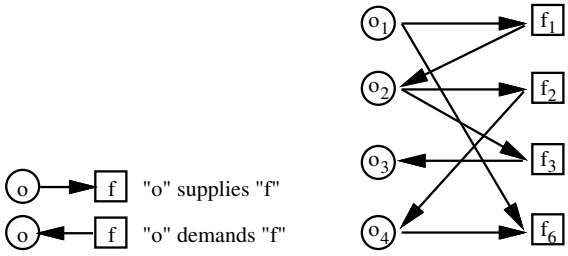


Figure 4: *The component-property graph of Figure 2.*

This bipartite-characteristic can be exploited when constructing a layout for component-property graphs: If the nodes of a graph are arranged in several layers, no edges must be drawn *within* a layer, if all nodes in the layer are of the same type (cf. *The Leveling Step* in the next section).

Recall that component-property graphs represent a functional model of a technical system. Technical systems are composed from subsystems or modules, which finally are built up with components. Although in a configuration knowledge-base these submodules are not labelled as such, they can often be identified by a smart clustering algorithm (cf. *The Clustering Step* in the next section).

Such a clustering algorithm exploits the fact that the components within a module are closely connected, while the modules themselves are connected via rather narrow interfaces. The automatic identification of modules is a particularity when visualizing component-property graphs; note that standard approaches for graph visualizations may consider user-defined clusters but do not try to identify clusters on their own.

It is quite evident that a clustering that resembles the technical setup of a system is a benefit from the knowledge-engineering standpoint.

4. VISUALIZING COMPONENT-PROPERTY GRAPHS

We now present a visualization algorithm exploiting the already mentioned features of the graph and considering our criteria of a lucid layout.

4.1 *Strategy*

Three major steps to a good layout can be identified. In a first step the clusters must be found. The next

task consists of finding a good arrangement of the clusters on the page, finally we seek positions for the nodes within a cluster.

4.2 *The Clustering Step*

How can a cluster be identified? As mentioned above clusters relate to modules or subsystems of a technical system, therefore we expect less edges between clusters than within a cluster: Modules are combined by only a few edges, while the components within a cluster are highly connected. A clustering divides the vertices V of a Graph $G = (V, E)$ into disjunct sets $C_1, \ldots, C_n, n \in N$.

Definition 4.1 (Clustering). A clustering $\mathcal{C}$ of a graph $G = (V, E)$ is defined as follows:
$\mathcal{C} = \{C | C \subseteq V\}$, and $\forall C_i, C_j \in \mathcal{C} : C_i \cap C_j = \emptyset$.

The cut of two disjunct sets of vertices is defined by the number of edges between these sets:

Definition 4.2 (Cut). The cut $cut : V^2 \rightarrow N$ is defined as follows:
$\forall C_1, C_2 \subseteq V, C_1 \cap C_2 = \emptyset$, and $cut(C_1, C_2) = |\{(v_1, v_2) | (v_1, v_2) \in E, v_1 \in C_1, v_2 \in C_2\}|$.

How can the quality of a clustering be measured? We want a small interface between clusters, i.e. the cut between clusters should be as small as possible. On the other hand, defining just one cluster $C_1 = V$ can not be called a great solution. Hence we demand the number of clusters to be maximized. However, we also want the clusters to be as dense as possible.

Definition 4.3 (Density of a cluster). The density $d : \mathcal{P}(V) \rightarrow \mathbf{R}$ of a cluster C_i is defined as follows:
$d(C_i) = \frac{|\{(v_1, v_2) | (v_1, v_2) \in E, v_1, v_2 \in C_i\}|}{|C_i|}$.

Definition 4.4 (Density of a clustering). The density d_2 of a Clustering $\mathcal{C}$ is defined as follows:
$d_C(\mathcal{C}) = \frac{\sum_{C_i \in \mathcal{C}} |\{(v_1, v_2) | (v_1, v_2) \in E, v_1, v_2 \in C_i\}|}{\sum_{C_i \in \mathcal{C}} |C_i|}$.

Definition 4.5 (Cut of a clustering). The Cut cut_C of a Clustering $\mathcal{C}$ is defined as follows:
$cut_C(\mathcal{C}) = \sum_{C_i, C_j \in \mathcal{C}, i<j} cut(C_i, C_j)$.

Definition 4.6 (Connectivity of a clustering). The Connectivity con of a Clustering $\mathcal{C}$ is defined as
$con(\mathcal{C}) = \frac{cut_C(\mathcal{C})}{|\mathcal{C}|}$

Obviously we can rewrite d_C as $d_C(\mathcal{C}) = \frac{|E| - cut_C(\mathcal{C})}{|V|}$.

Definition 4.7 (Quality of a clustering). The quality q of a clustering $\mathcal{C}$ is defined as $q(\mathcal{C}) = \frac{d_C(\mathcal{C})}{con(\mathcal{C})}$.

How can a cluster be found? Here we present a solution based on a greedy strategy. Starting with an arbitrary vertex within a cluster, the algorithm enters the cluster as deep as possible, i.e. it increases the cut of the respective cluster. In a second phase the algorithm walks through the cluster while the cut remains constant. In a last phase the cut is allowed to decrease, i.e. the algorithm tries to find the border of the cluster. Figure 5 illustrates the situation.

A special feature of component-property graphs supports the termination of the clustering: clusters are normally connected to a functionality or component with a small degree (see vertex v_2 in Figure 5). The algorithm does not cross this special vertex, therefore the clustering stops at the interface of components.

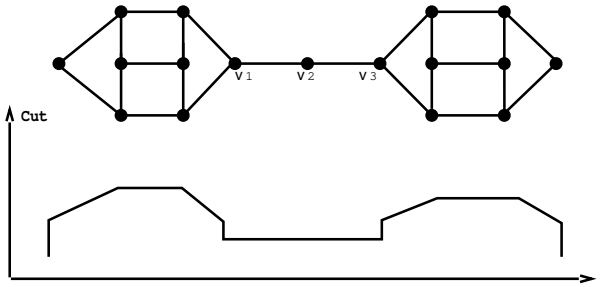


Figure 5: *Cut and cluster connectivity.*

A division of the graph by a vertical borderline would result in a cut as shown in the diagram below the graph. An algorithm intended to find clusters could exploit the "hill-appearance" of the clusters in the cut-diagram.

Note that not all vertices are suitable as start vertices for the clustering algorithm. The vertex should be within a cluster and not be part of an interface between components (in the example above v_2 is not suitable). So vertices with in-degree = out-degree = 1 are not used as starting points.

For some knowledge-bases a preprocessing step is useful: In some graphs there exist a few functionalities or components with an exceptional high degree, e.g. a functionality "power consumption", which is demanded by almost all components. This functionality could feign a non-existing module and therefore make a cluster detection difficult. To overcome this problem vertices with an exceptional high degree should be deleted.

4.3 *The Cluster Arrangement Step*

Having identified the clusters, the width and height necessary for their graphical representation is estimated. Resting on empirical evaluations we developed heuristics for this task. We call this graphical representation of a cluster a box; two boxes are connected iff there exists at least one edge between vertices from the corresponding clusters.

Definition 4.8 (Box representation of a Clustering). Let $\mathcal{C}$ be a clustering of the graph $G = (V, E)$. The box representation $\mathcal{B}(C) = (V_B, E_B)$ of $\mathcal{C}$ consists of boxes $V_E = \{b_1, \ldots, b_n\}$ (each relating to a cluster) and of connections $E_B \subseteq V_B^2$, where $(b_i, b_j) \in E_B \Leftrightarrow \exists v_a \in b_i, v_b \in b_j$ and $((v_i, v_j) \in E \vee (v_j, v_i) \in E)$.

To arrange the boxes on the pane a local optimization strategy is applied: Starting from reasonable initial positions we allow random position changes, and if these moves result in a better layout, we restart the process with the new arrangement. Of course, in order to optimize the layout the notion "lucid layout" must be defined in mathematical terms. We do not want boxes to intersect; we also would like to avoid a box being between two connected boxes. Finally, connected boxes should be as close as possible. The following optimization function f formalizes the desired conditions:

Definition 4.9 (Quality of a box arrangement). Let $\mathcal{A}$ be an arrangement of boxes on the pane, then the quality f of the arrangement is defined as follows:

$$f(\mathcal{A}) = \begin{cases} \infty, & \text{if two boxes intersect—otherwise:} \\ \displaystyle\sum_{(b_i,b_j)\in E_B} dist(b_i, b_j) + \lambda \cdot \sum_{(b_i,b_j)\in E_B} num(b_i, b_j) \end{cases}$$

where $dist(b_i, b_j)$ refers to the distance between two boxes, λ is a constant (we use $\lambda = \frac{\text{screen size}}{5}$) and $num(b_i, b_j)$ relates to the number of boxes between b_i and b_j. In Figure 6 and 7 the area between two boxes is shown hatched.

Initially we place the boxes on a circular arc (cf. Figure 6), in the course of the optimization process we decrease the step size of the movements.

Figure 7 shows the result of the optimization process.

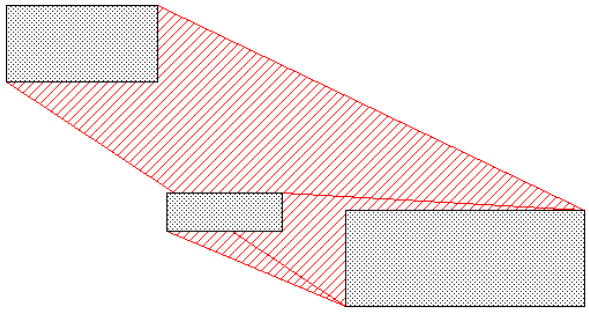


Figure 6: *Initial arrangement of the clusters.*

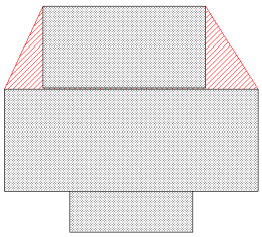


Figure 7: *After the cluster arrangement step.*

4.4 *The Leveling Step*

The last step of our visualization process determines the positions for the vertices within a box. We apply a modified leveling approach (Eades and Wormald, 1994; Sander, 1996). At first, the vertices of a cluster are divided into layers; here it is desired having an edge only between neighbored layers. Then the vertices of a layer are ordered, minimizing the number of edge intersections. Both the division into layers and the ordering within a layer take the global arrangement of boxes into consideration: A vertex v_1 in box b_1 connected to another vertex v_2 in a different box b_2 should have a position within b_1 close to b_2. Ghost vertices must be added for edges crossing layers. Finally the ordering of vertices of a layer is mapped onto Euclidean positions.

How can the division into layers be found? For acyclic graphs a topological sorting is sufficient, but for our mainly cyclic graphs the NP-complete feedback edge set problem needs to be solved. We overcome this problem by using a heuristic. For our domain it makes sense to allow only one type of vertex (functionality or component) being in a layer. Thus we find the components or functionalities with the smallest in-degree and choose one type to be placed as first layer. As a decision criterion the accessibleness of all other vertices starting from the vertices of the first layer is examined: The more vertices can be reached, the better. Unaccessible vertices are simply put into the appropriate of the first two layers. All other layers are derived canonically from the first layer. Vertices with connections to vertices in other boxes above (below)

the actual box are also moved into the appropriate of the first (last) two layers.

How can the vertices within a layer be ordered? Here, the well-known barycenter heuristic is employed. In a first run, starting with the second layer, each vertex v obtains a number $bc(v) = \frac{1}{|N_p|} \cdot \sum_{u \in N_p} bc(u)$, where N_p are the predecessors of v belonging to the same cluster. The bc values for the first layer are initialized in a reasonable way. In a second run, going from the second last layer up to the first layer, each vertex obtains a new bc value: $bc(v) = \frac{1}{|N_s|} \cdot \sum_{u \in N_s} bc(u)$, where N_s denotes the successors of v belonging to the same cluster. This standard procedure is slightly modified; a vertex v_1 in box b_1 connected to another vertex v_2 in a different box b_2 left (right) of b_1 is attracted to the left (right) side of its layer. This is achieved by giving v_2 a virtual bc value of 0 or max_x, where max_x is the maximum number of vertices in a layer.

For each edge crossing a layer an additional ghost vertex is added that prevents edges from crossing vertices. These ghost vertices are treated like normal vertices.

The vertices in a layer are mapped onto real positions on the pane as follows. We place all vertices on a given grid, the y-position on the grid follows from the division of the vertices into layers, we also use the bc value as a initial value for the x-position and optimize the x-positions later. Our optimization criteria is the overall edge length in a cluster. The process starts at the first layer. In a first run, we go through the vertices of a layer from the left to the right. If moving a vertex v horizontally improves the overall edge length, we restart this process with the new x-position for v. After going through the layer from the left to the right, we rerun the procedure going from the right to the left. This way, free space created by the first run can be used to improve the positioning in the second run. Figure 8 shows the result.

4.5 *Realization*

Our approach has been implemented under Windows NT using Allegro Common Lisp. Note that recognizing the clusters and their interfaces is quite easy. Nevertheless, for a "creative" arrangement of the cluster much more time needs to be spent. This can be considered as acceptable since the process of visualization does happen only once when importing a textual knowledge base.

Since the runtime behavior is dominated by the cluster arrangement step, it can be scaled quite easily:

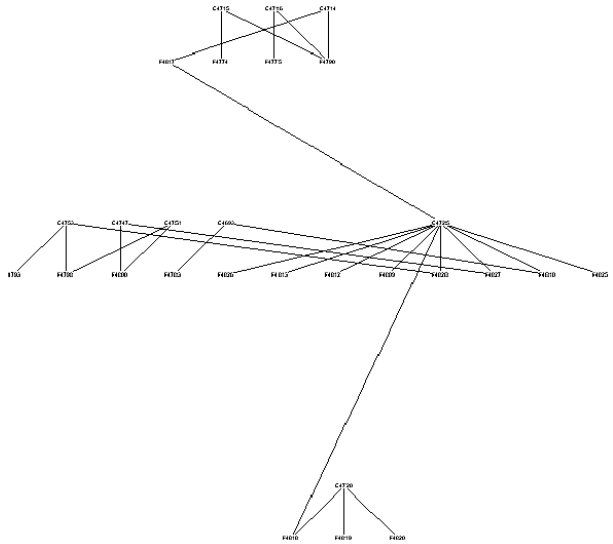


Figure 8: *After the cluster arrangement step.*

The better the cluster layout shall be, the more time must be spent.

The knowledge-bases that have been used to evaluate our approach stem from real-world examples. These examples where analyzed respecting the distribution of the node types, the node degrees, the components' supply-demand overhangs, and the functionalities' supply-demand ratios. Using this statistical information new knowledge bases have been generated from the existing ones by merging, deletion, and upsizing rules.

5. EXISTING VISUALIZATION TOOLS

The next two subsections show layout results that have been produced with VCG and Da Vinci for the component-property graph in Figure 9. VCG and Da Vinci are universal graph visualization tools, and therefore they may exploit less domain knowledge about resource-based configuration graphs. Nevertheless, a comparison will give some insight into graph visualization. The last subsection, 5.3, is on AKON, a resource-based configuration tool that provides a graphical acquisition mode for component-property graphs.

5.1 *VCG*

The VCG tool was developed at the University of Saarbrücken for the purpose of visualizing compiler graphs. It is based on a leveling approach and can take

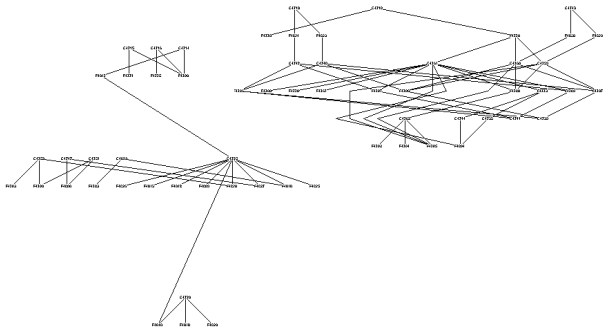


Figure 9: *Graph generated with our approach.*

cluster information into consideration; these cluster information must be part of the input.

Opposed to our cluster arrangement step, the cluster visualization of VCG is intertwined with the leveling algorithm. This results in a fast runtime behavior, but confines the freedom for arranging the clusters.

Figure 10 shows the visualization of our example. For further information, please see (Sander, 1994) and (Sander, 1996).

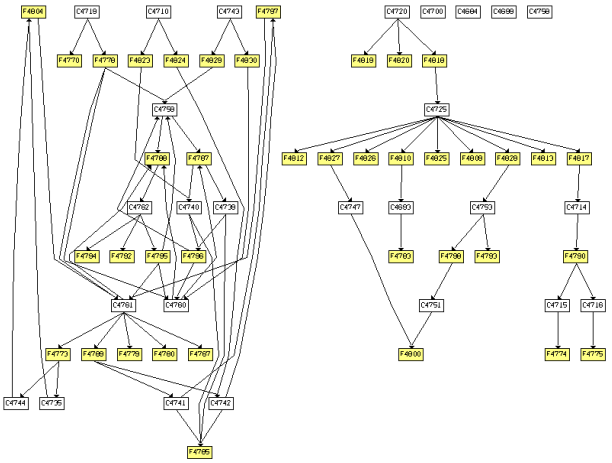


Figure 10: *Component-property graph generated by VCG.*

5.2 *Da Vinci*

Another popular tool for graph visualization is Da Vinci. Written at the university of Bremen, it is another realization of the leveling approach. Figure 11 shows the graph of Figure 10 and Figure 9 respectively, visualized by Da Vinci. Detailed information on Da Vinci can be found in (M. Fröhlich, M. Werner, 1994).

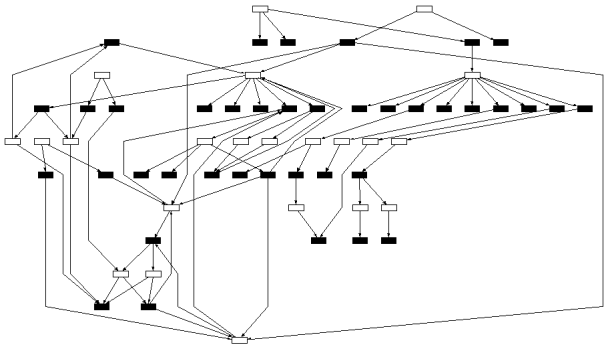


Figure 11: *Graph generated by Da Vinci.*

5.3 Akon

Akon is not graph visualization tool but a fully-fledged configuration system that operationalizes the resource-based configuration approach (Kleine Büning *et al.*, 1994).

Within Akon, configuration knowledge cannot be entered in textual form but is completely graphically specified, by means of drag-and-drop operations. Different types of lines indicate supply and demand relations respectively. A supply (demand) connection between a functionality icon, ▭, and a component icon, ▭, is established by dragging ▭ over ▭ (▭ over ▭). In this way, complex knowledge bases, say component-property graphs, can be formulated efficiently while avoiding any syntactical mistake. Figure 12 depicts a section of a telecommunication knowledge base formulated using Akon.

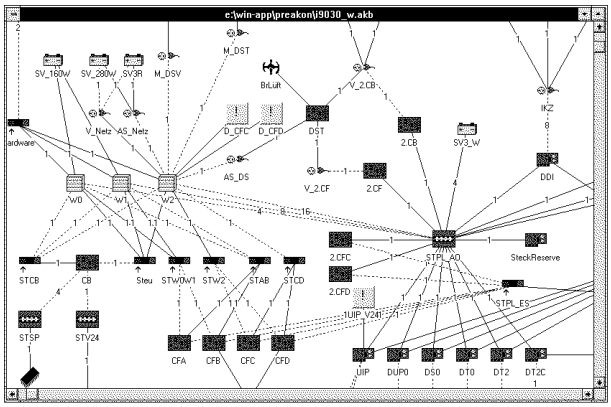


Figure 12: *A telecommunication knowledge base in* Akon.

Being in Akon's acquisition mode, the layout of the component-property graph can be organized reflecting both configurational aspects and the knowledge engineer's flavor. As a matter of concept, the component-property graph is not generated algorithmically but by the human sense for design and usefulness.

To us, the knowledge bases in Akon define a measurement respecting the quality of our automatically

generated layouts. However, if all knowledge bases for resource-based configuration problems were acquired by direct building the component-property graph, there would be no need for their graphical post-processing.

6. CONCLUSION

Components with supplied and demanded functionalities form the backbone of each resource-based configuration problem. This backbone can be illustrated by means of a so-called component-property graph, where the set of nodes is formed by the components and functionalities, and the edges stand for supply and demand relations.

For different acquisition and maintenance tasks relating configuration knowledge-bases, a smart visualization of component-property graphs is very useful. The paper showed in which way and to which quality such a visualization can be performed automatically. The developed approach consists of three basic steps—cluster detection, cluster arrangement, and inside-cluster leveling.

Within the configuration domain, the clusters play a special role: they represent subsystems or modules of a larger technical system. Their detection and visualization thus gives insight into the technical structure of the system defined in the knowledge-base.

Our approach has been realized and compared to the graph visualization tools VCG and da Vinci. These visualization tools generate clear layouts but require the definition of clusters by the user, and the cluster arrangement is restricted to hierarchical layouts only.

REFERENCES

Eades, Peter and Nicholas C. Wormald (1994). Edge Crossing in Drawing of Bipartite Graphs. In: *Mathematica 1994*. Springer Verlag.

M. Fröhlich, M. Werner (1994). The Graph Visualization System daVinci - A User Interface for Applications. Technical Report No. 5/94, Dept. of Computer Science, University of Bremen.

Heinrich, M. and E. W. Jüngst (1991). A Resource-based Paradigm for the Configuring of Technical Systems for Modular Components. In: *Proc. CAIA '91*. pp. 257–264.

Kleine Büning, Hans, Daniel Curatolo and Benno Stein (1994). Knowledge-Based Support within Configuration and Design Tasks. In: *Proc. ESDA '94, London*. pp. 435–441.

Laußermair, T. and K. Starkmann (1992). Konfigurierung basierend auf einem Bilanzverfahren. In: *6. Workshop "Planen und Konfigurieren", München*. FORWISS, FR-1992-001.

Sander, Georg (1994). Graph Layout through the VCG Tool. Technical Report A/03/94.

Sander, Georg (1996). Layout of Compound Directed Graphs. Technical Report A/03/96.

Stein, Benno (1995). Functional Models in Configuration Systems. Dissertation. University of Paderborn, Department of Mathematics and Computer Science.

Stein, Benno and Jürgen Weiner (1991). Model-based Configuration. In: *OEGAI '91, Workshop for Model-based Reasoning*.

Tong, Christopher (1987). Towards an Engineering Science of Knowledge-based Design. *Artificial Intelligence in Engineering* **2**(3), 133–166.

Weiner, Jürgen (1991). Aspekte der Konfigurierung technischer Anlagen. Dissertation. Gerhard-Mercator-Universität - GH Duisburg, FB 11 Mathematik / Informatik.

Personaleinsatzplanung von Zugpersonal als mehrstufiges Zuordnungsproblem

Stefan Klingenbeck
sd&m
Postfach 830851, 81708 München
Stefan.Klingenbeck@sdm.de

Frank Puppe
Universität Würzburg
Am Hubland, 97074 Würzburg
puppe@informatik.uni-wuerzburg.de

1 Einleitung

Ziel dieses Beitrages ist die Analyse eines komplexen praktischen Anwendungsproblems mit der Fragestellung, ob sich das Problem als Instanz bekannter Problemklassen mit wiederverwendbaren Problemlösungsmethoden und Werkzeugen verstehen läßt.

Das Problem entsteht im Planungsprozeß eines großen europäischen Eisenbahnunternehmens. Die Planung für die Personen- und Güterzüge des Unternehmens erfolgt jeweils für etwa ein Halbjahr im voraus (Sommerfahrplan oder Winterfahrplan). Die wichtigsten Teilaufgaben des Planungsprozesses befassen sich mit der Trassenbelegung, den Umläufen der Wagen und der Triebfahrzeuge sowie dem Einsatz des fahrenden und stationären Personals. Diese Planungsaufgaben sind wechselweise voneinander abhängig und werden von verschiedenen Organisationseinheiten teilautomatisiert oder manuell gelöst.

In dieser Problemanalyse beschränken wir uns auf Aspekte der Regeldienstplanung des fahrenden Personals, d.h. auf die Zuordnung von Personal zu den Aufgaben, die an Bord von regelmäßig verkehrenden Zügen auftreten. Nicht Gegenstand der Problemanalyse ist der Prozeß der Planänderungen im laufenden Betrieb (z.B. wegen Zugverspätungen, Erkrankungen des Personals, Einsatz von Sonderzügen, etc.).

Das Unternehmen unterhält einen Zugbetrieb von vielen tausend Zügen pro Tag mit einem einzuplanenden Bordpersonal von bis zu ca. 6 Personen pro Zug. Da die Züge nach Verkehrstagen (täglich, Mo-Fr, etc.) verkehren, können wir hier annehmen, daß z.B. alle Montage des betrachteten Halbjahres den gleichen Zugverkehr aufweisen.

Da die Gesamtaufgabe darin besteht, Personal zu Zugläufen zuzuordnen, geben wir in Kap. 2 eine kurze Charakterisierung der diskreten und kontinuierlichen Zuordnung an und gehen in Kap. 3 darauf ein, wie sich das Gesamtproblem in mehrere, hintereinander ausführbare Zuordnungsprobleme zerlegen läßt.

2 Diskrete und kontinuierliche Zuordnungsprobleme

Diskrete Zuordnungsprobleme sind durch zwei Mengen charakterisiert, wobei jedes Element der Nachfragemenge auf ein Element der Angebotsmenge unter Einhaltung vorgegebener Randbedingungen abgebildet werden soll. Ein typisches Beispiel ist die Schulstundenplanung, bei der Unterrichtseinheiten auf Zeitslots von Räumen, Klassen und Lehrern abgebildet werden. Die Komplexität von Zuordnungsproblemen hängt dabei entscheidend von der Art der Randbedingungen ab: Gewöhnlich hat man neben harten, die Konsistenz gewährleistenden Randbedingungen auch weiche Randbedingungen, die als eine zu optimierende Kostenfunktion zusammengefaßt werden. Wenn nur lokale Randbedingungen (lineare Kostenfunktion) vorliegen, gibt es effiziente Verfahren wie den Busacker-Goven-Algorithmus, die in polynomialer Laufzeit eine optimale Lösung finden. Bei globalen Randbedingungen (nicht-lineare Kostenfunktion) gibt es keine effizienten Verfahren, die eine optimale Lösung liefern. Die vielversprechensten Verfahren beruhen daher auf einer heuristischen Suche, die nach [Poeck 95] auf folgendem Grundalgorithmus beruhen:

1. Generiere Anfangslösung
   Solange die Zuordnung nicht vollständig
   1.1 Wähle ein freies Nachfrageobjekt N.
   1.2 Wähle zu N ein Angebotsobjekt A.
   1.3 Teste die Randbedingungen.
   1.4 Falls Randbedingungen verletzt, suche nach Verbesserungen.
2. Optimiere Anfangslösung

Abbildung 1: Basis-Algorithmus für Zuordnung durch heuristische Suche

Ein konkreter Algorithmus ist die Vorschlagen-und-Vertauschen-Strategie [Poeck & Puppe 92], die im Zuordnungs-Shell-Baukasten COKE [Poeck 95] realisiert wurde.

Bei vielen Problemen muß eine Menge von Nachfrageobjekten auf Angebotsobjekte mit einer kontinuierlichen Zeitskala abgebildet werden, z.B. wenn bei der Schulstundenplanung die Unterrichtsstunden zu beliebigen Zeiten beginnen dürfen. Obwohl der Suchraum dann theoretisch unendlich groß wird, läßt sich der Algorithmus aus Abbildung 1 mit einer leichten Modifikation weiterbenutzen: Im Schritt 1.2 wird ein Angebotsobjekt nicht ausgewählt, sondern sein passendes Zeitintervall berechnet. Diese Erweiterung liegt dem Werkzeug zur Generierung von Resourcenbelegungsplänen WIZARD [Hestermann et al. 97] zugrunde, die COKE um Mechanismen zur Behandlung kontinuierlicher Zuordnungsprobleme erweitert.

3 Personaleinsatzplanung als mehrstufiges Zuordnungsproblem

Bei der Personaleinsatzplanung müssen alle Zugläufe auf Zeitintervalle des Personals abgebildet werden. Die besondere Schwierigkeit liegt in der Berücksichtigung räumlicher Constraints: Im Normalfall soll das Personal eine Schicht am selben Ort beenden, wo es sie begonnen hat. Daher muß eine Person mindestens zwischen zwei Zügen wechseln, z.B. in einer Schicht von A nach B fahren und mit dem nächsten Zug wieder zurück. Daher müssen nicht ganze Züge sondern Zugabschnitte zu Personen zugeordnet werden - mit der Randbedingung, daß eine Person nur dann in einen anderen Zug umsteigen kann, wenn der innerhalb des passenden Zeitintervalls an dem richtigen Ort hält.

Die Nachfragen sind daher im wesentlichen[1] von folgendem Typ:

Führe die Tätigkeit X zwischen den Halten A_i und A_{i+1} am Wochentag Z auf dem Zug Y aus. Ein Halt ist durch den Ort sowie die Ankunfts- und Abfahrtszeit des Zuges gekennzeichnet.

Die Angebote sind das vorhandene Personal mit den wichtigsten Merkmalen

- Qualifikation (kann Tätigkeit X ausführen) und
- Einsatzstelle (der Ort an dem der Tageseinsatz beginnen und enden sollte)

Wegen der hohen Komplexität des Zuordnungsproblems wird es in der Praxis in drei aufeinander aufbauende Schritte zerlegt:

1. Zusammenfassung der Zugläufe zu Planungseinheiten
2. Für jede Planungseinheit: Einteilung der Zugläufe in Schichten
3. Für jede Schicht: Zuordnung zu Dienstplänen des Personals

3.1 Zusammenfassung der Zugläufe zu Planungseinheiten

Eine Planungseinheit faßt typischerweise solche Zugläufe zusammen, die im Wochentag, in der erforderlichen Qualifikation und „weitgehend“ hinsichtlich der Orte übereinstimmen, in denen die Züge halten, z.B. alle Fernzüge auf der Route A-B-C in beiden Richtungen. Abbildung 2 zeigt einen Ausschnitt aus einer schematischen Darstellung einer Planungseinheit.

Die gebildeten Planungseinheiten ändern sich selten und werden von verschiedenen Organisationsstrukturen gemäß deren regionaler und fachlicher Zuständigkeit weiterbearbeitet.

Die Zusammenstellung von Nachfragen zu Planungseinheiten beeinflußt das Ergebnis der Gesamtplanung erheblich. Aufgrund der engen Verflechtungen der Zuordnung zu Planungseinheiten mit den Organisationsstrukturen des Unternehmens und den seltenen Änderungen ist der Bedarf nach einem (teil)automatisierten Verfahren für dieses Teilproblem gering. Die Zusammenfassung der Nachfragen zu Planungseinheiten wird daher als gegeben vorausgesetzt.

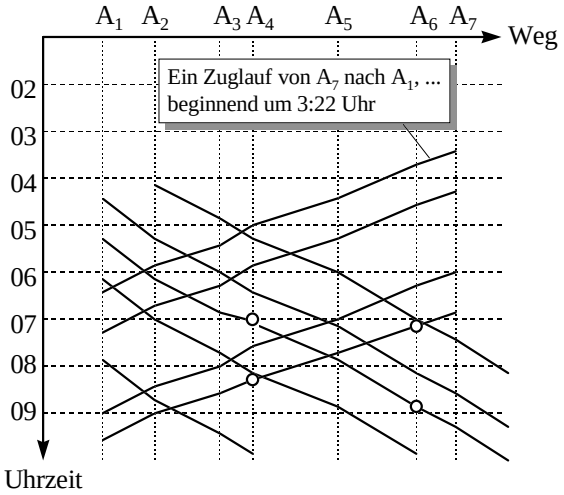


Abbildung 2: Ausschnitt aus einem Weg-Zeit-Diagramm zur Darstellung der zu einer Planungseinheit gehörigen Züge. Für die Haltepunkte A_1,...,A_7 gibt es in der Regel eine „natürliche“ Anordnung. Sofern ein Zug nicht an einem Ort hält oder nicht durch diesen Ort fährt, ist dies durch einen Kreis gekennzeichnet.

3.2 Bilden von Schichten

Im nächsten Schritt werden die in einer Planungseinheit enthaltenen Zugläufe zu sogenannten Schichten zusammengestellt. Unter einer Schicht wird dabei

[1] Hier wird natürlich eine vereinfachte Version der Aufgabenstellung beschrieben. Es gibt z.B. für das fahrende Personal noch eine Reihe von Aufgaben, die sich nicht in obiges Schema pressen lassen.

eine ableistbare Aneinanderreihung $T_1,...T_m$ von Tätigkeiten aufgefaßt; d.h. jede Tätigkeit T_i darf erst nach dem Ende ihrer Vorgängertätigkeit beginnen und sie wird dort aufgenommen, wo ihre Vorgängertätigkeit endet. Die Tätigkeit T_m sollte dort enden, wo T_1 beginnt. Ist zwischen zwei Tätigkeiten ein Wechsel der Züge erforderlich, so sind die notwendigen Umsteigezeiten zu berücksichtigen. Daneben sind einige weitere Randbedingungen, wie z.B. minimale und maximale Dauer einer Schicht, einzuhaltende Pausenzeiten und maximale Lenkzeiten bei der Führung von Triebfahrzeugen zu beachten. In Abbildung 3 ist eine Schicht als fettgedrucktes Polygon dargestellt.

Der entscheidende Kostenfaktor der Personaleinsatzplanung sind die Umsteigezeiten. In diesen Zeiten kann das Personal in der Regel nicht nutzbringend eingesetzt werden.

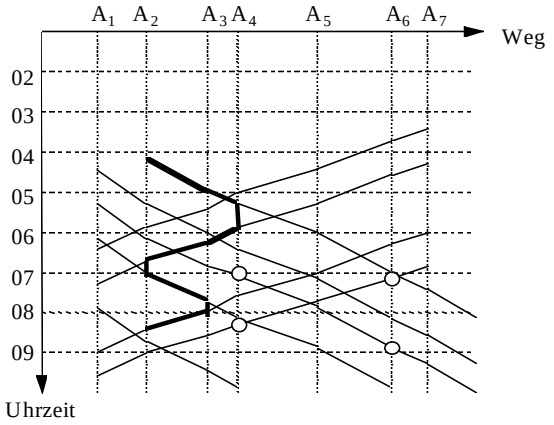


Abbildung 3: Schicht (fettgedruckt) in einem Weg-Zeit-Diagramm, die am Ort A_2 um 4:17 Uhr beginnt und am selben Ort um 8.26 Uhr endet. Die Schicht erfordert Umseigen in A_4, A_2 und A_3.

3.3 Zuordnen von Schichten zu Dienstplänen

Sind alle Nachfragen durch Schichten abgedeckt, werden diese Schichten zu Dienstplänen zusammengestellt, die jeweils die Tätigkeiten einer Person für eine Fahrplanperiode festlegen. Dieser Schritt stellt sich wiederum als Zuordnungsproblem von Schichten zu Dienstplänen dar. Beachtet werden müssen hierbei insbesondere Randbedingungen hinsichtlich der Gewährung freier Tage, nächtlicher Ruhezeiten, der Häufigkeit von Nachtschichten und der maximalen Wochenarbeitszeit. Die für das Unternehmen entstehenden Kosten des Personaleinsatzes werden im wesentlichen bei der Schichtbildung festgelegt. Die Dienstplanung kann daher kaum mehr Kosten einsparen. Ihre Zielfunktion ergibt sich daraus, die Arbeitszeit und Belastungen, wie Nacht- und Wochenendschichten möglichst sozialverträglich auf die einzelnen Dienstpläne zu verteilen.

Die einschlägigen Regelungen zum Personaleinsatz sehen etwa 80 zu beachtende Randbedingungen vor. Wir geben an dieser Stelle drei Beispiele für Randbedingungen:

- Schichten, die länger als 6 Stunden dauern, müssen mindestens 2 Pausen von jeweils mindestens 15 Minuten Dauer enthalten.
- In keinem 7-Tage-Zeitraum darf die Arbeitszeit 55 Stunden überschreiten.
- Der Abstand zwischen zwei freien Tagen darf maximal 144 Stunden betragen.

4 Diskussion

Die Zerlegung der außerordentlich komplexen Gesamtaufgabe hat sowohl Vor- als auch Nachteile: Einerseits entstehen handliche Teilaufgaben, andererseits können Wechselwirkungen zwischen den Teilaufgaben bei der Feinoptimierung nicht berücksichtigt werden. Betrachtet man die Teilaufgaben im einzelnen, ist die erste Teilaufgabe (Zusammenfassung der Zugläufe zu Planungseinheiten) am untypischsten für einen Zuordnungsalgorithmus, da es sich eher um ein Clustering-Verfahren zur Modularisierung der Gesamtaufgabe handelt. Die zweite Teilaufgabe (Einteilung der Zugläufe in Schichten) besitzt einen relativ großen Suchraum, da jeder Streckenabschnitt ein Nachfrageobjekt darstellt. Die Angebotsobjekte, d.h. die Schichten entstehen dynamisch während der Planung. Obwohl das kein echtes Zuordnungsproblem darstellt, kann es mit ähnlichen Mechanismen wie bei der kontinuierlichen Zuordnung behandelt werden. Hierzu sind jedoch gute Heuristiken erforderlich, die sich teilweise aus der typischen Dauer einer Schicht ergeben. Die dritte Teilaufgabe (Zuordnung der Schichten zu Dienstplänen) kann dagegen als eine klassische Zuordnungsaufgabe betrachtet werden, da hier die Nachfrageobjekte Schichten den Angebotsobjekten Personal zugeordnet werden.

5 Literatur

Hestermann, C., Wolber, M., Wellner, J.: Intelligentes, kooperatives Assistenzsystem zur Dispositionsunterstützung in Produktionsplanungs- und Steuerungs-Systemen, in Proceedings zum 11.Workshop 'Planen und Konfigurieren' (PuK-97); FR-1997-001, Februar 1997, FORWISS, Erlangen.

Poeck, K.: Konfigurierbare Problemlösungsmethoden am Beispiel der Problemklassen Zuordnung und Diagnostik, infix, diski 86, 1995.

Poeck, K. and Puppe, F.: COKE: Efficient Solving of Complex Assignment Problems with the Propose-and-Exchange Method, Proceedings der IEEE Conference on Tools with Artificial Intelligence, 136-143, 1992.

Interaktive, automatische Stundenplanung mittels constraintlogischer Programmierung

Hans-Joachim Goltz
GMD – Forschungszentrum Informationstechnik GmbH
Forschungsinstitut für Rechnerarchitektur und Softwaretechnik
GMD-FIRST, Rudower Chaussee 5, 12489 Berlin
e-mail: goltz@first.gmd.de

1 Einleitung

Ein Stundenplan ist eine Zuordnung von Ereignissen zu Zeitpunkten bzw. Zeitintervallen, so daß die geforderten Bedingungen (Constraints) erfüllt sind. Neben den Constraints, die alle erfüllt sein müssen und harte Constraints genannt werden, existieren auch Constraints, die möglichst erfüllt sein sollen und als weiche Constraints bezeichnet werden. Seit langem ist bekannt, daß das Stundenplanungsproblem zu der Klasse der NP-vollständigen Problemen gehört (siehe z.B. [5]). Existierende Softwareprodukte für die Stundenplanung schränken die möglichen erzeugbaren Pläne zu stark ein. Außerdem sind sie nicht flexibel genug, um abweichende Anforderungen integrieren zu können und Besonderheiten zu berücksichtigen, die es bei fast jedem Anwender gibt (siehe z.B. [3]).

Ein Ziel unserer Forschung ist die Entwicklung von Methoden, Verfahren und Konzepten für eine interaktive, automatische Stundenplanung, die in Universitäten und Schulen angewendet werden können. 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. Deswegen sind für uns die Entwicklung von Konzepten und Methoden der Modellierung von Problemen der Stundenplanung und Untersuchungen zum Einfluß verschiedener Problemmodellierungen auf die Lösungssuche wichtige Forschungsschwerpunkte.

Die Systeme der Stundenplanung sollen so flexibel sein, daß Besonderheiten der Anwender berücksichtigt und Constraints leicht geändert werden können, falls keine grundsätzliche konzeptionelle Änderung der Stundenplanung erforderlich ist. Wichtiger Bestandteil wird eine automatische heuristische Lösungssuche sein, in der der Anwender interaktiv Einfluß nehmen kann. Der Anwender kann einen Plan bzw. Teilplan aber nur so modifizieren, daß keine harten Constraints verletzt sind.

Die automatische Stundeplanung ist ein aktuelles Forschungsgebiet. Im letzten Jahr fand die 2. Internationale Konferenz zu diesem Thema statt. Für die automatische Stundenplanung existieren verschiedene grundlegende Softwaretechniken und Ansätze (siehe z.B. [10]). Zur Verwirklichung unserer Forschungsziele verwenden wird die constraintlogische Programmierung (CLP). Verschiedene CLP-Ansätze werden beispielsweise in [1, 2, 7, 9] diskutiert. Ein anderer constraintbasierter Ansatz wird in [8] beschrieben.

Im Rahmen unserer Forschungen entwickeln wir ein System für die interaktive, automatische Stundenplanung an der Medizinischen Fakultät Charité der Humboldt Universität zu Berlin. Als Implementationssprache wurde die constraintlogische Programmiersprache CHIP gewählt. Insbesondere haben sich die globalen Constraints, die in CHIP enthalten sind, als sehr vorteilhaft für die Modellierung der Probleme erwiesen. In dieser Arbeit berichten wir über diese Systementwicklung und einigen Forschungsergebnissen, die damit im Zusammenhang stehen.

2 Problembeschreibung

Für die Medizinische Fakultät Charité der Humboldt-Universität war bisher die Erzeugung von Stundenplänen für den klinischen Studienabschnitt (sechs Semester) problemreich und mit großem manuellen Aufwand verbunden. In der ersten Projektphase wurde deshalb für diesen Studienabschnitt ein automatisches Planungsssytem entwickelt.

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 bis zu 20 Studenten durchgeführt werden. Dazu werden die Studenten eines Semesters in Seminargruppen unterteilt. Die zwei Standorte der Fakultät befinden sich in verschiedenen Stadtbezirken. Einige Lehrveranstaltungen werden auch in anderen Berliner Kliniken und Krankenhäuser durchgeführt. Weitere wichtige Bedingungen dieses Problems der Stundenplanung sind:

1. Die Lehrveranstaltungen können zu jeder Viertelstunde beginnen und können unterschiedliche, vorgegebene Längen haben.
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.
6. Vorlesungen eines Semesters bzw. Lehrveranstaltungen einer Seminargruppe dürfen sich zeitlich nicht überlappen.
7. Für jede Seminargruppe sind die Wege zwischen den möglicherweise verschiedenen Veranstaltungsorten der Lehrveranstaltungen zu berücksichtigen.
8. Den Vorlesungen sind im Stundenplan auch Räume konfliktfrei zuzuordnen. Viele Vorlesungen können dabei nur in bestimmten Räumen stattfinden.
9. Die Anzahl der Seminare, Untersuchungskurse und Praktika, die eine Klinik gleichzeitig durchführen kann, ist beschränkt, wobei die Schranke auch zeitabhängig sein kann.
10. Die Wunschzeiten und Raumwünsche für Vorlesungen sind möglichst zu berücksichtigen.

3 Modellierung

Der Constraintlöser über endlichen Domänen von CHIP bildet die Grundlage unserer Problemmodellierung. Die gewählte Modellierung eines Problems der Stundenplanung besitzt einen großen Einfluß auf die Propagationseigenschaften des Constraintlösers und damit auf die Effizienz der Lösungssuche. Als Beispiel betrachten wir die Modellierung der 3. Bedingung *"zwei Vorlesungen eines Faches sind an verschiedenen Tagen einzuplanen"*. Wir setzen dabei voraus, daß ein Plan für 5 Tage zu erstellen ist und daß jeder Tag in 48 Zeiteinheiten (4×12 Viertelstunden) unterteilt ist. In [7], in der auch CHIP als Implementationssprache verwendet wurde, wird die Modellierung dieser Bedingung wie folgt vorgeschlagen. Für jede Vorlesung werden die folgenden Domänenvariablen für den Vorlesungsbeginn definiert: `Day in 1..5`, `Hour in 1..48` und `Qh in 1..240` (für jede mögliche Viertelstunde der Woche). Die Beziehungen zwischen diesen Variablen wird durch das Constraint `Qh = 48*(Day-1) + Hour` repräsentiert. Um die genannte Bedingung für zwei Vorlesungen eines Faches zu sichern, wird ein weiteres Constraint verwendet, das ausgedrückt, daß die Werte, die den beiden `Day`-Variablen zugeordnet werden, verschieden sein müssen. Wenn die verschiedenen Variablen nur durch die angegebene Gleichung verbunden sind, ist die Propagation zwischen diesen Variablen einer Vorlesung nicht ausreichend, da in Gleichungen nur die Veränderung von Minimum oder Maximum der möglichen Werte propagiert werden. Falls beispielsweise die eine Vorlesung am Tag 3 eingeplant wird, wird zwar in der `Day`-Variable der anderen Vorlesung dieser Wert gestrichen, aber durch das Gleichungsconstraint wird kein Wert aus der Domäne der entsprechenden `Qh`-Variable entfernt.

In der von uns gewählten Modellierung verwenden wir nur die `Qh`-Variable für den Vorlesungsbeginn. Zwei Vorlesungen eines Faches unterscheiden sich höchstens durch die Dauer. Deswegen legen wir

fest, welche der beiden Vorlesungen mindestens einen Tag vor der anderen einzuplanen ist. Falls die Vorlesungsdauer verschieden ist (Werte `d1` und `d2`), wird die Dauer jeweils als Domänenvariable mit den beiden möglichen Werten betrachtet (`D1 in [d1,d2]` , `D2 in [d1,d2]`) und die Gleichung `D1 + D2 = d1 + d2` als Constraint erzeugt. Um auszudrücken, daß die Vorlesung mit der Startzeitvariable `Qh1` mindestens einen Tag vor der Vorlesung mit der Startzeitvariable `Qh2` einzuplanen ist, wird eine Domänenvariable `X in [48,96,144,192]` definiert und die Constraints `Qh1 < X` und `X =< Qh2` erzeugt. Wenn nun die eine Vorlesung am Tag 3 eingeplant wird, erfolgt auch sofort eine Einschränkung der Startzeitvariable für die andere Vorlesung. Durch diese Modellierung der 3. Bedingung konnte eine bessere Propagation gegenüber der in [7] diskutierten Modellierung erzielt werden.

Die Problemmodellierung ist auch abhängig von den Möglichkeiten und Propagationseigenschaften des gewählten Constraintlösers. Die in CHIP enthaltenen globalen Constraints `cumulative` und `diffn` haben sich für unsere Modellierung als sehr wertvoll erwiesen. Für eine Erläuterung des Constraints `cumulative` betrachten wir die folgenden Bedingungen:

> *Die Ereignisse $E_1, E_2, \ldots, E_n$ benötigen gleichartige Ressourcen, wobei zu jedem relevanten Zeitpunkt insgesamt L Ressourcen zur Verfügung stehen; jedes Ereignis E_i benötigt H_i Ressourcen und dauert D_i Zeiteinheiten; gesucht sind die Startzeiten S_i der Ereignisse, so daß zu keinenm Zeitpunkt nicht mehr Ressourcen benötigt werden als zur Verfügung stehen.*

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 $S_j \leq k \leq S_j + D_j - 1$

Diese komplexe Bedingung der Ressourcenbeschränkung kann durch das globale Constraint `cumulative` wie folgt ausgedrückt werden:

$$\texttt{cumulative}([S_1, S_2, \ldots, S_n], [D_1, D_2, \ldots, D_n], [H_1, H_2, \ldots, H_n], L)$$

Die Bedingung 9 der Problembeschreibung kann beispielsweise mit diesem Constraint leicht modelliert werde. Seien `S1,S2,...,Sn` die Domänenvariablen für die Startzeiten der Lehrveranstaltungen, die eine Klinik durchführen muß, `D1,D2,...,Dn` die jeweilige Dauer dieser Lehrveranstaltungen und `Max` die maximale Anzahl von Lehrveranstaltungen, die diese Klinik gleichzeitig durchführen kann. Dann kann die Bedingung 9 durch das folgende Constraint ausgedrückt werden:

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

Falls `Max` nicht für alle Zeitpunkte gilt, können für die entsprechenden Zeiten "Dummy"-Einheiten hinzugefügt werden, die gerade die nicht vorhandenen Kapazitäten binden.

Durch das globale Constraint `diffn` können abstrakte Bedingungen der folgenden Art direkt modelliert werden:

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

In einem üblichen Stundenplanungsproblems einer Schule kann beispielsweise durch ein solches Constraint ausgedrückt werden, daß alle Unterrichtsstunden bezüglich der drei "Dimensionen" *Zeit, Raum, Lehrer* überlappungsfrei sein müssen.

Mit Ausnahme der letzten Bedingung der Problembeschreibung können alle Bedingungen direkt als Constraints formuliert werden und vor der Lösungssuche erzeugt werden. Aus der letzten Bedingung ergeben sich weiche Constraints. Innerhalb der Lösungssuche wird versucht, diese weichen Constraints möglichst zu erfüllen.

Der Beginn der Lehrveranstaltungen wird jeweils als Domänenvariable definiert, wobei sich zuerst die Werte der Domänen aus der Transformation der relevanten Zeiten in Einheiten von Viertelstunden ergeben (Bedingung 1). Unter Einbeziehung der Dauer der jeweiligen Lehrveranstaltungen werden diese Domänen entsprechend der 2. Bedingung eingeschränkt. Die Modellierung der Bedingungen 3 und 9 wurde bereits oben diskutiert.

Neben der Zeit werden für jede Lehrveranstaltung auch die Wochennummer und die Parallitätsnummer als Größen betrachtet, die unbekannt sein können und dann als Domänenvariable zu definieren sind (Bedingungen 4 und 5). Für jedes Semester und für jede Gruppe kann dann mittels des globalen Constraints `diffn` die Bedingung 6 formuliert werden, wobei die unbekannten Größen die Dimensionen festlegen. Für das Problem der

Modellierung der Berücksichtigung der verschiedenen Mindestabständen zwischen zwei Lehrveranstaltungen in Abhängigkeit des Veranstaltungsortes (Bedingung 7) wurde zusätzlich das symbolische Constraint `element` verwendet und die verschiedenen Wege als spezielle "Unterrichtseinheiten" modelliert.

Um den Vorlesungen Räume zuzuordnen (Bedingung 8), wird als weitere Unbekannte für jede Vorlesung der Raum als Domänenvariable definiert, wobei eine mögliche Einschränkung der Raumzuordnung bereits in der Definition der Domäne integriert wird. Die konfliktfreie Raumzuordnung wird durch Verwendung des Constraints `diffn` gesichert.

4 Lösungssuche

Constraintlöser über endlichen Domänen sind nicht vollständig. Für die Erzeugung einer Lösung ist i. allg. Suche erforderlich, die oft durch *"Labelling"* realisiert wird. Dabei werden schrittweise den Constraintvariablen Werte aus ihren Domänen zugeordnet. Als Modifizierung dieser Suchmethode haben wir in [6] vorgeschlagen, daß die Wertzuweisung durch eine Einschränkung der Domäne der ausgewählten Variable ersetzt wird. Als Grundalgorithmus ergibt sich dann:

```
reducing( [], _ ).
reducing(VarList, InfoList) :-
   select_var(Var,VarList,NewVarList),
   reducing_domain(Var, InfoList),
   reducing(NewVarList, InfoList).
```

Durch `select_var/3` wird die Variable aus der Liste der relevanten Variablen ausgewählt, deren Domäne als nächstes durch `reducing_domain/2` zu reduzieren ist, wobei die Art und Weise der Einschränkung auch von den Domänen der anderen Variablen und weiteren Komponenten (`InfoList`) abhängen kann. Im Falle eines Backtrackings wird die mengentheoretische Differenz zwischen der alten Domäne und der reduzierten Domäne als neue Domäne der entsprechenden Variable betrachtet und wieder die Prozedur `reducing_domain/1` bezüglich dieser Variable angewendet. Nachdem die Domänen aller relevanten Variablen reduziert wurden, muß die Suche noch nicht abgeschlossen sein, obwohl der größte Teil der Suche durch eine solche Folge von Einschränkungen der Domänen erreicht werden kann. Diese Idee der backtrackbaren Domänenreduzierung wurde auch hier verwendet. An vielen Beispielen konnte der signifikante Vorteil dieser Methode nachgewiesen werden.

In der Lösungssuche sind zwei Arten von Nichtdeterminismus enthalten: Auswahl einer Domänenvariable und Auswahl der Domäneneinschränkung für die ausgewählte Variable. Natürlich ist es i. allg. 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 Domänenreduzierung ist die Berücksichtigung der Wünsche integriert. Folglich wird im ersten Schritt versucht, eine Domäne so einzuschränken, daß die Wünsche enthalten sind.

In vielen Fällen unserer Versuche 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 Domänenreduzierung verwendet. Somit wird ein Backtracking über verschiedene Heuristiken durchgeführt.

Neben der vollständigen automatischen Planerzeugung kann die Lösungssuche über eine graphische Repräsentation interaktiv beeinflußt werden. 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 geplante Lehrveranstaltungen automatisch einplanen.

Bei der interaktiven Beeinflussung der Lösungssuche erfolgt kein "Backtracking" über Entscheidungen des menschlichen Planers. Dadurch ist natürlich möglich, daß der Anwender bei der interaktiven Planung keinen widerspruchsfreien Plan findet, obwohl die automatische Planung eine Lösung erzeugen kann.

5 Implementation

Als Implementationssprache wurde die constraintlogische Programmiersprache CHIP (Version 5.0) der Firma COSYTEC aus Frankreich gewählt. Auch die graphischen Schnittstellen wurden bzw. werden in CHIP implementiert. Neben den bereits erwähnten globalen Constraints erwies sich die objektorientierte Komponente von CHIP besonders vorteilhaft für die Implementation.

Die Problemrepräsentation erfolgt in drei Stufen: Problemdefinition, interne relationale Repräsentation, interne objektorientierte Repräsentation. Für die Problemdefinition wurde eine deklarative Problembeschreibungssprache entwickelt. Alle Komponenten eines Problems der Stundenplanung können dadurch auch ohne graphische Schnittstelle leicht definiert werden. Als Zwischenschritt werden die Problemdefinitionen intern in eine relationale Repräsentation transformiert, wobei auch die Daten aus den Definitionen aufbereitet werden (z.B. Umwandlung der Zeitangaben). Im nächsten Transformationsschritt wird dann aus der relationalen Repräsentation die objektorientierte Repräsentation erzeugt. Diese Repräsentation wird für die eigentliche Lösungssuche und die graphischen Darstellungen verwendet. Der Zwischenschritt der Transformation ist wegen der Verarbeitung der möglichen Querbezüge zwischen den Definitionen erforderlich. Die Repräsentation des Zwischenschrittes ist außerdem für die Erzeugung verschiedener Lösungsvarianten vorteilhaft. Wenn verschiedene Varianten betrachtet werden sollen, müssen für jede Variante die entsprechenden Objekte erzeugt werden. Für diese Erzugung kann jeweils das Ergebnis des Zwischenschrittes verwendet werden.

Die Ausgabe der erzeugten Stundenpläne erfolgt als HTML-Dateien. Dadurch sind die Ergebnisse der Planung universell weiter verwendbar.

6 Ergebnisse

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.first.gmd.de/plan/charite` zu finden.

Seitens der Fakultät wurde die automatisierte Stundenplanung sehr positiv beurteilt. Die Vorteile einer interaktiven, automatischen Stundenplanerzeugung konnten eindeutig nachgewiesen werden. Gegenüber der Situation in den Vorjahren 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.

7 Ausblick

Der ersten Testeinsatz unseres Systems hat bewiesen, daß wir die richtigen Methoden verwenden. 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 weiteren Schritte der Systementwicklung beinhalten u.a.: Stundenplanung für alle Semester, Raumzuordnung für alle Lehrveranstaltungen, Erweiterung und Modifizierung der Problembeschreibungsbesprache, Modifikationen der erforderlichen Transformationen, Verbesserungen der interaktiven Suche und Vervollständigung der graphischen Schnittstellen. Um die vorgesehene Systementwicklung erfolgreich abzuschließen sind weitere Forschungsarbeiten erforderlich. Zu den Schwerpunkten dieser Forschung gehören Untersuchungen zur Verbesserung der Problemmodellierung und Constraintpropation, und die Entwicklung und der Vergleich verschiedener Methoden und Verfahren der heuristischen Lösungssuche.

Literatur

[1] F. Azevedo and P. Barahona. Timetabling in constraint logic programming. In *Proc. World Congress on Expert Systems*, 1994.

[2] P. Boizumault, Y. Delon, and L. Peridy. Planning exams using constraint logic programming. In *[11]*, pages 79–93, 1994.

[3] E. Burke, D. Eiiliman, P. Ford, and R. Weare. Examination timetabling in British universities: A survey. In *[4]*, pages 76–90, 1996.

[4] E. Burke and P. Ross, editors. *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, New York, 1996.

[5] T. B. Cooper and J. H. Kingston. The complexity of timetable construction problems. In *[4]*, pages 183–295, 1996.

[6] H.-J. Goltz. Reducing domains for search in CLP(FD) and its application to job-shop scheduling. In U. Montanari and F. Rossi, editors, *Principles and Practice of Constraint Programming – CP'95*, volume 976 of *Lecture Notes in Computer Science*, pages 549–562, Berlin, Heidelberg, 1995. Springer-Verlag.

[7] C. Guéret, N. Jussien, P. Boizumault, and C. Prins. Building university timetables using constraint logic programming. In *[4]*, pages 130–145, 1996.

[8] M. Henz and J. Würtz. Using Oz for college timetabling. In *[4]*, pages 162–177, 1996.

[9] G. Lajos. Complete university modular timetabling using constraint logic programming. In *[4]*, pages 146–161, 1996.

[10] A. Schaerf. A survey of automated timetabling. Technical Report CS-R9567, Centrum voor Wiskunde en Informatica, 1995.

[11] L. Sterling, editor. *Proc. Internat. Conf. on the Practical Application of Prolog*. London, 1994.

Constraintbasierte Stundenplanung für Universitäten

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

Zusammenfassung

Stundenplanung für das Institut für Informatik der Universität München erfordert die Verarbeitung harter und weicher Constraints. Harte Constraints sind unbedingt zu erfüllende Bedingungen, weiche Constraints dürfen zwar verletzt werden, sollten aber so gut wie möglich erfüllt werden. Dieses Papier zeigt die Modellierung unseres Stundenplanungsproblems als Partial Constraint Satisfaction Problem und stellt einen kurzen und deklarativen Löser für endliche Bereiche vor, der auch weiche Constraints propagiert und damit für diese die Möglichkeit eröffnet, eine aktive Rolle im Lösungsprozess einzunehmen.

1 Einleitung

Constraints sind vordefinierte Relationen, die es erlauben, in deklarativer Weise Probleme durch Angabe von Nebenbedingungen zu formulieren, die im Vergleich zu herkömmlichen Programmiersprachen die Suche nach geeigneten Lösungen beschleunigen [vH89].

Das wichtigste Gebiet der kommerziellen Anwendung von Constraints ist das Lösen von kombinatorischen Problemen. Sie sind Bestandteil vieler industrieller Anwendungen, wie zum Beispiel Scheduling, Konfiguration oder Stundenplanung. Aus einer großen Menge von möglichen Wertekombinationen gilt es diejenigen zu bestimmen, die gewisse Constraints erfüllen (oder besonders gut erfüllen). Klassische Ansätze zur Lösung solcher Probleme leiden oft an mangelnder Flexibilität und Erweiterbarkeit. Der Erfolg der Constraintprogrammierung bei der Lösung solcher Optimierungsprobleme beruht auf der deklarativen Modellierung, der Constraintpropagierung und der guten Kombinierbarkeit mit Such- und Optimierungsverfahren.

Meistens werden Stundenpläne von Hand erstellt, wobei verschiedene „harte“ Bedingungen unbedingt erfüllt werden müssen, und „weiche“ Bedingungen nach Möglichkeit. Die meisten existierenden Stundenplaner sind mit Systemen implementiert, die von Hause aus keine weichen Constraints behandeln können. Einige Systeme behandeln dann nur harte Constraints [AB94]. Andere weisen jedem weichen Constraint des Problems fiktive, mit der Verletzung des Constraints verbundene Kosten zu, suchen eine Lösung, die alle harten Constraints erfüllt, und optimieren diese mit Branch-And-Bound [FHS95, HW95]. Ein weiterer Ansatz ist die Anwendung von Constraintverfahren, die sowohl harte als auch weiche Constraints behandeln [Mey97]. Harte Constraints werden unmittelbar zur Reduktion der betroffenen Wertebereiche verwendet. Weiche Constraints hingegen werden zur Ermittlung einer Bewertung der Elemente der Wertebereiche eingesetzt. Diese Bewertungen steuern die Suche. Dadurch erzielt man, daß die erste Lösung brauchbar ist, d.h. nicht nur alle harten Constraints erfüllt, sondern auch viele weiche Constraints. Dieser Ansatz wurde für die Dienstplanung in Krankenhäusern mit C++ implementiert [Mey97].

Unser Ziel war der Einsatz dieser Technik für die Stundenplanung in unserem Institut. Dafür haben wir einen Constraintlöser mit der deklarativen Constraintsprache *Constraint Handling Rules* (CHR) [Frü95] implementiert. Mit CHR konnte die Behandlung harter und weicher Constraints in einem Löser kurz und deklarativ spezifiziert werden. Basierend auf diesem Löser

ahmt unser System, der IfI-Planer, die manuelle Planung nach. Er erstellt für unser Institut, jeweils ausgehend vom Stundenplan des Vorjahres, einen neuen Plan für das kommende Semester. Dabei stellt die Kontinuität in der Stundenplanung eines der Optimalitätskriterien dar.

Dieses Papier ist wie folgt gegliedert: Abschnitt 2 beschreibt unser Planungsproblem und Abschnitt 3 dessen Modellierung als Partial Constraint Satisfaction Problem. In Abschnitt 4 wird dann beschrieben, wie dieses Planungsproblem mit CHR gelöst wurde. Abschnitt 5 ist Zusammenfassung und Ausblick.

2 Das Problem

Unser Ziel war die Generierung eines Stundenplans für eine gegebene Menge von Angeboten unter Berücksichtigung der Dozentenwünsche und des Vorjahresplans. Unser System sollte Teil des Planungsprozesses werden, der wie folgt abläuft.

Nach Sammlung von Dozentenwünschen und Informationen über neue Angebote wird ein erster Vorschlag basierend auf dem Vorjahresplan entwickelt. Durch nicht mehr stattfindende Angebote freigewordene Plätze werden dabei zuerst wiederverwendet, und zwar bevorzugt für neue Angebote desselben Lehrstuhls. Dozentenwünsche sind dabei aber wichtiger als der Vorjahresplan. Nach Verteilung des Vorschlags an alle Beteiligten werden Bewertungen und neue Wünsche gesammelt. Mit dem aktuellen Vorschlag als Ausgangspunkt wird dann ein nächster Vorschlag entwickelt, der diese Reaktionen berücksichtigt, wobei aber wieder sowenig wie möglich geändert wird, usw. Dieses konservative Vorgehen reduziert nicht nur den Aufwand für die Erstellung eines neuen Stundenplans, sondern sichert auch dessen Aktzeptanz durch Beibehaltung der gewohnten wöchentlichen Abläufe.

Die Anforderungen an den Stundenplan sind wie folgt.

- Veranstaltungen eines Dozenten dürfen sich nicht überschneiden. Zwischen Veranstaltungen eines Dozenten sollte mindestens eine Stunde Pause sein.
- Veranstaltungen eines Lehrstuhls dürfen sich nicht mit anderen Veranstaltungen desselben Lehrstuhls überschneiden.
- Einige Dozenten bevorzugen bestimmte Zeiten oder Tage für ihre Veranstaltungen. Montag nachmittag ist reserviert für die Vorstandssitzung der Professoren.
- Ein Angebot besteht typischerweise aus zwei Vorlesungen und einer Übung pro Woche. Zwischen den beiden Vorlesungen sollte mindestens ein Tag liegen. Eine Übung sollte nicht mit einer der Vorlesungen am gleichen Tag liegen. Veranstaltungen sollten zwischen neun und achtzehn Uhr stattfinden, Vorlesungen aber nicht am Freitag nachmittag und Übungen nicht am späten Freitag nachmittag.
- Veranstaltungen eines Semesters des Grundstudiums dürfen sich nicht überschneiden. Veranstaltungen des Hauptstudiums sollten sich nicht überschneiden.

Erste Untersuchungen ergaben, daß bestehende Stundenpläne die gegebenen Forderungen nicht erfüllen: Veranstaltungen eines Lehrstuhls oder Veranstaltungen des Hauptstudiums überschneiden sich. Vorlesung und Übung eines Angebots finden am gleichen Tag statt. Außerdem ergibt schon die Betrachtung der Anzahl der angebotenen Veranstaltungen des Hauptstudiums, daß es zuwenig Platz gibt, um alle diese Veranstaltungen überschneidungsfrei zu plazieren. Im Grundstudium tritt dieses Problem nicht auf, da dort verschiedene Semester klar unterschieden werden, was das Überschneiden der Veranstaltungen verschiedener Semester erlaubt. Wir waren also mit zwei Problemen konfrontiert:

- Inkrementelle Planung auf Basis des Vorjahresplans erforderte die Verarbeitung von Stundenplänen, die die angegebenen Bedingungen nicht erfüllen.
- Keine klare Struktur im Hauptstudium kostet Freiheiten bei der Planung und hat inkonsistente Stundenplanspezifikationen zur Folge.

Ein Auflösen des zweiten Problems durch das gezielte Zulassen von Überschneidungen gestaltete sich mühevoll und zeitaufwendig. Außerdem reagierten so entstandene konsistente Spezifikationen sehr empfindlich schon auf kleine Änderungen, die oftmals wieder Inkonsistenz zur Folge hatten. Die Klassifikation von Angeboten des Hauptstudiums nach Inhalt

und typischer Größe des Publikums eröffnete die Möglichkeit, Überschneidungen zwischen Veranstaltungen verschiedener Klassen zuzulassen. Diese Maßnahme konnte Freiheiten gewinnen, aber die Konflikte nicht völlig auflösen. Die Notwendigkeit einer Art gewichteter Constraints wurde deutlich.

3 Modellierung als PCSP

Ein *Partial Constraint Satisfaction Problem* (PCSP) [FW92] besteht aus einer Menge von Variablen und einer Menge von Relationen zwischen den Variablen (Constraints), wobei jedes Constraint mit einem Gewicht versehen ist. Gewichte erlauben die Unterscheidung harter und weicher Constraints. Harte Constraints müssen erfüllt werden. Die Verletzung weicher Constraints ist zwar zulässig, sollte aber doch weitgehend vermieden werden. Harte Constraints sind durch ein unendliches Gewicht gekennzeichnet. Die endlichen Gewichte weicher Constraints ermöglichen die Spezifikation von Präferenzen innerhalb der Menge der weichen Constraints. Jeder Variable ist außerdem ein endlicher Wertebereich zugeordnet. Die Lösung eines PCSP bildet jede Variable unter Erfüllung aller harten Constraints auf einen Wert aus ihrem Wertebereich ab, sodaß die Summe der Gewichte der verletzten weichen Constraints minimal ist.

Zur Modellierung unseres Stundenplanungsproblems als PCSP führen wir für jede zu plazierende Veranstaltung eine Variable ein. Der anfängliche Wertebereich jeder Variablen ist die ganze Woche, wobei für jeden Tag 24 Zeitpunkte vorgesehen sind. Die anwendungsspezifischen Anforderungen und die Dozentenwünsche können durch folgende Constraints ausgedrückt werden:

- Ein *Überschneidungsconstraint* verlangt, daß zwei Veranstaltungen sich nicht überschneiden dürfen.
- Ein *Zeitconstraint* bringt entweder einen Dozentenwunsch zum Ausdruck oder fordert, daß eine Veranstaltung zu bestimmten Zeitpunkten stattfinden muß bzw. zu diesen Zeitpunkten nicht stattfinden darf.
- Ein *Verteilungsconstraint* stellt sicher, daß mindestens ein Tag (eine Stunde) zwischen zwei Veranstaltungen liegt oder daß zwei Veranstaltungen an verschiedenen Tagen stattfinden.
- Ein *Anschlußconstraint* stellt sicher, daß eine Veranstaltung direkt nach einer anderen Veranstaltung stattfindet.

Bei den Gewichten für die weichen Constraints unterscheiden wir die drei Abstufungen „schwach bevorzugt“, „bevorzugt“ und „stark bevorzugt“, die in die ganzzahligen Gewichte 1, 3 bzw. 9 übersetzt werden.

4 Lösung mit CHR

Constraint Handling Rules (CHR) [Frü95] ist eine deklarative, auf einem Regelformalismus basierende Sprache zur Implementierung von Constraintlösern, mit der eine beliebige Basissprache, z.B. Prolog oder Lisp, um benutzerdefinierte Constraints erweitert werden kann. Man unterscheidet zwei Arten von Regeln: Simplifikationsregeln beschreiben, wie Constraints sich zu neuen Constraints vereinfachen (z.B. `X > Y, Y > X <=> false`) und Propagationsregeln beschreiben, wie Constraints neue Constraints propagieren (z.B. `X > Y, Y > Z ==> X > Z`). Aus Platzgründen kann hier keine formale Definition der Syntax und Semantik von CHR gegeben werden. Wir verweisen auf [Frü95] und [Abd97].

Um den Suchaufwand zur Lösung kombinatorischer Probleme zu reduzieren, werden in der Künstlichen Intelligenz *Konsistenzverfahren* [Mac92, Kum92] eingesetzt. Die Idee dieser Verfahren besteht darin, Werte aus den Wertebereichen der Problemvariablen zu entfernen, die nicht Bestandteil einer Lösung sein können. Die beiden Constraints $x \in \{2,3,\ldots,6\}$ und $x \in \{4,5,\ldots,9\}$ können z.B. durch das neue Constraint $x \in \{4,5,6\}$ ersetzt werden.

Dieses Verfahren kann leicht mit CHR implementiert werden [FA97], aber dieses Schema ist nicht ausreichend für unsere Bedürfnisse: Da weiche Constraints verletzt werden dürfen, ist die Entfernung von Werten aus Wertebereichen, deren Auswahl die Verletzung weicher Constraints bedeuten würde, nicht möglich. Außerdem benötigen wir für die Wertauswahl während der Suche eine Basis für die Entscheidung, ob ein Wert hinsichtlich der Erfüllung weicher Constraints eine gute Wahl darstellt oder nicht. Wir versehen deswegen

jeden Wert mit einer Einschätzung und repräsentieren einen Wertebereich als eine Liste von Paaren (Wert, Einschätzung). Sei z.B. `[(3, 0), (4, 1), (5, -1)]` der Wertebereich von `X`. Dann darf `X` einen der Werte `3`, `4` und `5` annehmen, wobei `4` mit Einschätzung `1` gegenüber `3` und `5` mit den Einschätzungen `0` bzw. `-1` bevorzugt wird.

Die Propagation eines weichen Constraints bedeutet dann die Veränderung der Einschätzungen für die mit dem Constraint unvereinbaren Werte. Betrachten wir z.B. ein Zeitconstraint, das mit Gewicht `2` den Wert `3` für `X` ablehnt. Dann müssen wir die Einschätzung für `3` um `2` nach unten korrigieren, womit wir den neuen Wertebereich `[(3, -2), (4, 1), (5, -1)]` für `X` erhalten. Die Propagation eines harten Constraints bedeutet aber weiterhin die Entfernung von Werten aus den Wertebereichen der betroffenen Variablen.

Der Constraintlöser basiert auf drei Arten von Constraints:

- `domain(C, S, D)` bedeutet, daß die Veranstaltung `C` nur zu einem Zeitpunkt `S` stattfinden darf, der im Wertebereich `D` enthalten ist, wobei `D` eine Liste von Paaren (Zeitpunkt, Einschätzung) ist.
- `in(C, L, W)`: Wenn `W = infinite`, d.h. wenn das Constraint hart ist, dann darf die Veranstaltung `C` nicht zu einem Zeitpunkt stattfinden, der nicht in `L` enthalten ist. Wenn `W` eine Zahl ist, d.h. wenn das Constraint weich ist, dann sollen die Einschätzungen für die Zeitpunkte, die in `L` vorkommen, um `W` erhöht werden.
- `notin(C, L, W)`: Wenn das Constraint hart ist, dann darf die Veranstaltung `C` nicht zu einem Zeitpunkt stattfinden, der in `L` enthalten ist. Wenn das Constraint weich ist, dann sollen die Einschätzungen für die Zeitpunkte in `L` um `W` verkleinert werden.

Wir stellen nun den Kern unseres Constraintlösers vor. Dieser besteht aus zwei Regeln zur Verarbeitung von `in-` und `notin`-Constraints. Weitere acht Regeln dienen der Übersetzung der anwendungsspezifischen Constraints.

Die Verarbeitung eines `in`-Constraints bedeutet entweder das Entfernen von Werten aus dem Wertebereich oder die Erhöhung der Einschätzung für die angegebenen Zeitpunkte. Dies kann in CHR mit der folgenden Simplifikationsregel ausgedrückt werden.

```
domain(C, S, D), in(C, L, W) <=>
  (
     W = infinite
  -> domain_intersection(D, L, D1),
  ;  increase_assessment(W, L, D, D1)
  ),
  domain(C, S, D1).
```

Wenn ein neues `in`-Constraint vorliegt, dann sucht die Simplifikationsregel nach dem zugehörigen `domain`-Constraint und ersetzt beide durch ein neues `domain`-Constraint mit einem veränderten Wertebereich. Für ein hartes `in`-Constraint, d.h. für `W = infinite`, ist das der Schnitt des Wertebereichs `D` mit der Liste von Zeitpunkten `L`. Für ein weiches `in`-Constraint bleiben alle Werte erhalten, lediglich die Einschätzungen für die Werte, die in `L` vorkommen, werden um `W` erhöht. `notin`-Constraints können durch eine weitere Simplifikationsregel in analoger Art und Weise behandelt werden.

Wenn ein Wertebereich leer wird, dann ist die Spezifikation inkonsistent. Eine Veranstaltung kann nicht ohne Verletzung harter Constraints plaziert werden. Diesen Fall erledigt folgende Simplifikationsregel:

```
domain(_, _, []) <=> fail.
```

Bis jetzt haben wir uns nur mit den grundlegenden, unären Constraints unseres PCSP-Lösers für endliche Bereiche beschäftigt. Im folgenden stellen wir exemplarisch die Realisierung von anwendungsspezifischen n-ären Constraints mittels `in-` und `notin`-Constraints dar.

`no_clash(W, Cs)` bedeutet in Abhängigkeit vom Gewicht `W`, daß sich die Veranstaltungen der Liste `Cs` nicht überschneiden dürfen oder sollten. Es wird in `notin`-Constraints übersetzt, wobei die Übersetzung datengetrieben ist: Wenn eine der Veranstaltungen in `Cs` plaziert wurde, dann wird der gewählte Platz durch die folgende Regel für die anderen Veranstaltungen verboten oder abgelehnt.

```
no_clash(W, Cs) <=>
  Cs \= [_],
  select_ground_var(Cs, X, CsRest)
  |
  post_notin_constraints(W, X, CsRest),
  no_clash(W, CsRest).
```

Der Wächter stellt zunächst sicher, daß die Liste `Cs` der Veranstaltungen mindestens zwei Elemente enthält. Dann wählt er eine instanziierte Variable `X` aus `Cs` aus und merkt sich die anderen Veranstaltungen in `CsRest`. Gibt es keine instanziierte Variable in `Cs`, dann scheitert `select_ground_var`. Ist der Wächter erfüllt, dann wird `no_clash(W, Cs)` ersetzt durch

- von `post_notin_constraints` erzeugte `notin`-Constraints, eines für jede Veranstaltung in `CsRest`, die den Platz `X` verbieten oder ablehnen und
- einem `no_clash`-Constraint, das fordert, daß sich die Veranstaltungen in `CsRest` nicht überschneiden dürfen oder sollten.

`post_notin_constraints` scheitert, falls `CsRest` den Wert von `X` enthält.

Eine einelementige Veranstaltungsliste bedeutet, daß es nichts mehr zu tun gibt. Diesen Fall erledigt die folgende Regel.

```
no_clash(_, [_]) <=> true.
```

Der hier vorgestellte PCSP-Löser für endliche Bereiche implementiert Kantenkonsistenz für harte binäre Überschneidungs-, Verteilungs- und Anschlußconstraints. Für n-äre Constraints wird inkrementell ein Constraintnetzwerk aus binären Constraints aufgespannt. Für harte Überschneidungsconstraints ist dieser Ansatz äquivalent zu der in CHIP [vH89] für das `all_distinct`-Constraint implementierten Methode. Wir wissen um die schlechte Propagationsleistung eines Netzwerks binärer Constraints für ein n-äres Constraint [Reg94], aber dieser Ansatz ist einfach zu implementieren und erwies sich als ausreichend zur Lösung unseres Problems.

Die Propagation weicher Constraints berechnet Einschätzungen für die Werte, die noch nicht durch Propagation harter Constraints entfernt wurden. Diese Einschätzungen können zur Information der Suchprozedur über vielversprechende Werte verwendet werden, wodurch weiche Constraints zu einem aktiven Bestandteil des Lösungsprozesses werden. Wenig Propagation für weiche n-äre Überschneidungsconstraints kann zur Folge haben, daß die erste Lösung viele weiche Constraints verletzt. In unserem Fall trat dieses Problem aber nicht auf.

Die Plangenerierung läuft wie folgt ab: Zuerst wird jeder Veranstaltung ein `domain`-Constraint zugeordnet. Die anfängliche Einschätzung ist `0` für jeden Zeitpunkt, d.h. kein Zeitpunkt wird gegenüber einem anderen bevorzugt. Dann werden die Bedingungen, die der Stundenplan erfüllen soll, in `in`- und `notin`-Constraints übersetzt. Die Suchprozedur verwendet zur Variablenauswahl das *first-fail*-Prinzip, d.h. eine der Variablen mit dem kleinsten Wertebereich wird gewählt. Zur Wertauswahl wird die *best-fit*-Strategie benutzt, d.h. einer der Werte mit der besten Einschätzung wird gewählt. Optimistisch betrachtet wird das einer der Zeitpunkte sein, für den die Gewichtssumme der verletzten weichen Constraints minimal ist, aber die Einschätzung kann wegen der Unvollständigkeit des Lösers für n-äre Überschneidungsconstraints zu gut sein. Da weiche Constraints vom Constraintlöser und vom Suchverfahren berücksichtigt werden, war die erste Lösung meistens sehr zufriedenstellend. Der IfI-Planer sucht deswegen nicht nach optimalen Lösungen. Durch Aufsummieren der Einschätzungen der für eine Lösung verwendeten Werte erhält man aber ein Qualitätsmerkmal, daß in Branch-And-Bound zur Verbesserung der ersten Lösung eingesetzt werden kann.

Der IfI-Planer benötigt etwa fünf Minuten zur Erstellung eines neuen Plans, der 89 Veranstaltungen innerhalb von 42 Stunden plaziert. Legt man den Vorjahresplan zugrunde, dann reduziert sich der Aufwand auf etwa zweieinhalb Minuten.

5 Zusammenfassung und Ausblick

Das Papier zeigt, daß Constraint Handling Rules zur Implementierung eines PCSP-Lösers für endliche Bereiche gut geeignet ist. Der Löser ist kurz und deklarativ, propagiert harte und weiche Constraints und ist leistungsfähig genug, um als Kern eines Stundenplaners für unser Institut zu dienen.

Unser Planer läuft auf ECL^{i}PS^e [ACD+94] ergänzt um die CHR-Bibliothek [BFL+94]. Dozenten können neue Wünsche und Angebote selbst über das WWW in die Spezifikation eintragen. Die WWW-Schnittstelle basiert auf HTML, die Seiten werden ebenfalls von einem Prolog-Programm generiert. Die Entwicklung der WWW-Schnittstelle dauerte zwei Wochen,

die Entwicklung des Planers nahm weitere drei Wochen in Anspruch.

Unser Prototyp wird mittlerweile im Institut für Informatik an der Universität München eingesetzt. Der Stundenplan für das Sommersemester 98 ist mit dem IfI-Planer erstellt worden. Um die Qualität der erzeugten Stundenpläne weiter zu verbessern, möchten wir das Constraintlösen mit Optimierungsverfahren (z.B. Branch-And-Bound) kombinieren.

Der in dieser Anwendung vorgestellte Ansatz wird jetzt zur Erstellung eines Stundenplans für ein Gymnasium untersucht. Diese Anwendung unterscheidet sich vom IfI-Planer dadurch, daß auf den Lehrplan des vorherigen Jahres nicht zurückgegriffen werden kann. Außerdem ist das Problem komplexer wegen der größeren Anzahl zu plazierender Stunden und den vielfältigen Qualitätskriterien, die der Plan erfüllen sollte.

Literatur

[AB94] F. Azevedo und P. Barahona. Timetabling in Constraint Logic Programming. In *Proceedings of 2nd World Congress on Expert Systems*, 1994.

[Abd97] S. Abdennadher. Operational Semantics and Confluence of Constraint Propagation Rules. In *Third International Conference on Principles and Practice of Constraint Programming, CP'97*, LNCS 1330. Springer, 1997.

[ACD+94] A. Aggoun, D. Chan, P. Dufrense, E. Falvey, H. Grant, A. Herold, G. MacCartney, M. Maier, D. Miller, B. Perez, E. van Rossum, J. Schimpf, P. Tsahageas und D. de Villeneuve. *ECL^{i}PS^e 3.4 User Manual.* ECRC Munich, 1994.

[BFL+94] P. Brisset, T. Frühwirth, P. Lim, M. Meier, T. Le Provost, J. Schimpf und M. Wallace. *ECL^{i}PS^e 3.4 Extensions User Manual.* ECRC Munich, 1994.

[FA97] T. Frühwirth und S. Abdennadher. *Constraint-Programmierung: Grundlagen und Anwendungen.* Springer, 1997.

[FHS95] H. Frangouli, V. Harmandas und P. Stamatopoulos. UTSE: Construction of Optimum Timetables for University Courses — A CLP Based Approach. In *Proceedings of the Third International Conference on the Practical Applications of Prolog (PAP'95).* Alinmead Software Ltd, 1995.

[Frü95] T. Frühwirth. Constraint Handling Rules. In A. Podelski, Hrsg., *Constraint Programming: Basics and Trends*, LNCS 910. Springer, 1995.

[FW92] E. C. Freuder und R. J. Wallace. Partial Constraint Satisfaction. *Artificial Intelligence*, 58(1-3):21–70, 1992.

[HW95] M. Henz und J. Wurtz. Using Oz for college time tabling. In *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95)*, 1995.

[Kum92] V. Kumar. Algorithms for Constraint-Satisfaction Problems: A Survey. *AI Magazine*, 13(1), 1992.

[Mac92] A. Mackworth. Constraint Satisfaction. In Stuart C. Shapiro, Hrsg., *Encyclopedia of Artificial Intelligence*, Band 1. Wiley, 1992.

[Mey97] H. Meyer auf'm Hofe. ConPlan/SIEDAplan: Personnel Assignment as a Problem of Hierarchical Constraint Satisfaction. In *Proceedings of the 3rd International Conference on the Practical Application of Constraint Technology*, Blackpool, 1997. The Practical Application Ltd.

[Reg94] J.-C. Regin. A filtering algorithm for constraints of difference in CSPs. In *Proc. 12th Conf. American Assoc. Artificial Intelligence*, 1994.

[vH89] P. van Hentenryck. *Constraint Satisfaction in Logic Programming.* MIT Press, Cambridge, Massachusetts, 1989.

Ablaufplanung mit Softcomputing Methoden

Jürgen Sauer

Carl von Ossietzky Universität Oldenburg, FB Informatik
Escherweg 2, D-26121 Oldenburg
sauer@informatik.uni-oldenburg.de

Abstract

Bei der Ablaufplanung sind sowohl prädiktive und reaktive als auch globale und lokale Sichtweisen auf die Problemstellung des Erstellens und Anpassens von Plänen zu finden. Insbesondere beim Multi-Site Scheduling wird die Ablaufplanung eines Betriebes mit mehreren Produktionsstandorten betrachtet. Hier muß auf zwei Ebenen global und lokal geplant werden. In der globalen Ablaufplanung werden auf Basis von externen Aufträgen Vorgaben für die lokalen Betriebe in Form eines globalen Ablaufplans ermittelt, der dann in den lokalen Ablaufplanungssystemen für die einzelnen Betriebe in konkrete maschinenbezogene Pläne für die einzelnen Zwischenprodukte umgesetzt werden muß. Auf beiden Ebenen sind prädiktive und reaktive Planungsaufgaben zu erfüllen. Die Lösung dieser Planungsaufgaben mit Hilfe von Methoden des Softcomputing wird im vorgestellten Projekt untersucht.

1. Ablaufplanung auf mehreren Ebenen

In der Ablaufplanung (Scheduling) [Sau97] werden Probleme der zeitlichen Zuordnung von Aktivitäten zu limitierten Ressourcen betrachtet, wobei unterschiedliche Nebenbedingungen zu berücksichtigen sind und bestimmte Ziele erreicht bzw. optimiert werden sollen. Unterschiedliche Anwendungsbereiche weisen Ablaufplanungsprobleme auf.
Zur Lösung von Ablaufplanungsproblemen werden drei miteinander verbundene Aufgabenkomplexe unterschieden. Bei der *prädiktiven* Ablaufplanung wird der Plan vorausschauend für einen bestimmten Zeitabschnitt unter Annahme einer statischen Planungsumgebung erstellt. In der *reaktiven* Ablaufplanung steht die Anpassung des Plans an neue Situationen im Mittelpunkt, wobei möglichst viel vom bestehenden Plan erhalten bleiben soll. Schließlich wird in der *interaktiven* Ablaufplanung der Endbenutzer des Planungssystems mit einbezogen. Dieser kann den Planungsprozeß steuern, indem er selbst bestimmte Planungsentscheidungen trifft oder geeignete prädiktive bzw. reaktive Verfahren auswählt.
Probleme der Ablaufplanung werden im allgemeinen nur für einzelne Produktionsstätten oder Aufgabenbereiche betrachtet [Zwe94]. Neue Aufgabenstellungen kommen hinzu, wenn man die heute übliche Verflechtung von Produktionsstätten, die sich z.B. Zwischenprodukte zuliefern, untersucht [BS95], Bild 1 zeigt das zugrundeliegende Szenario. Dies ist der Ausgangspunkt für das Projekt MUST (Multi Site Scheduling System), in dem ein System zur Ablaufplanung für mehrere Produktionsstandorte entwickelt wird.
Bei der Verteilung der Produktion auf mehrere Betriebe müssen einige spezifische Probleme gelöst werden, die bedingt sind durch

- komplexe Abhängigkeiten zwischen den Produktionsprozessen in verschiedenen Betrieben, u.a. zeitliche Abhängigkeiten, Kostenabhängigkeiten, Transportabhängigkeiten.
- die Art der zur Verteilungsplanung verwendeten Informationen, da man i.allg. nicht mit präzisen sondern verallgemeinerten Daten arbeitet, u.a. Kapazitätsinformationen auf Basis von Maschinengruppen, geschätzte Fertigungsdauern.
- die Integration bestehender (lokaler) Ablaufplanungssysteme für einzelne Betriebe.
- die Notwendigkeit der Koordination der Aktivitäten, da mehrere Planungsebenen mit jeweils unterschiedlichen Planungsaufgaben zusammenarbeiten müssen.

Diese Problemstellungen bedingen zusätzliche Aufgaben, die von den Planungssystemen der verschiedenen Ebenen erledigt werden müssen.
Ein Großteil der Aufgaben wird in der Globalen Ablaufplanung behandelt. Diese Planungsebene wird für den Lösungsansatz zusätzlich eingeführt (ist aber in der Realität zumindest informell auch schon vorhanden). Auf dieser Ebene muß sowohl die Erstellung einer Vorgabe für die lokalen Betriebe (der globale Plan), als auch die reaktive Korrektur dieses globalen Plans erfolgen. Als Teilaufgaben ergeben sich:

- *Global Prädiktive Planung*:
  Hier wird eine Vorgabe mit einer Verteilung der internen Aufträge auf die lokalen Betriebe erzeugt. Dabei sind u.a. auch Transporte zwischen den Betrieben zu berücksichtigen.
- *Global Reaktive Planung*:
  Können Probleme nicht lokal behoben werden, oder beeinflußt der geänderte lokale Plan andere lokale Planungen, wird die globale Ebene wieder eingeschaltet. Hier wird dann z.B. eine Umverteilung der internen Aufträge als Korrekturmaßnahme für den globalen Plan ausgearbeitet.

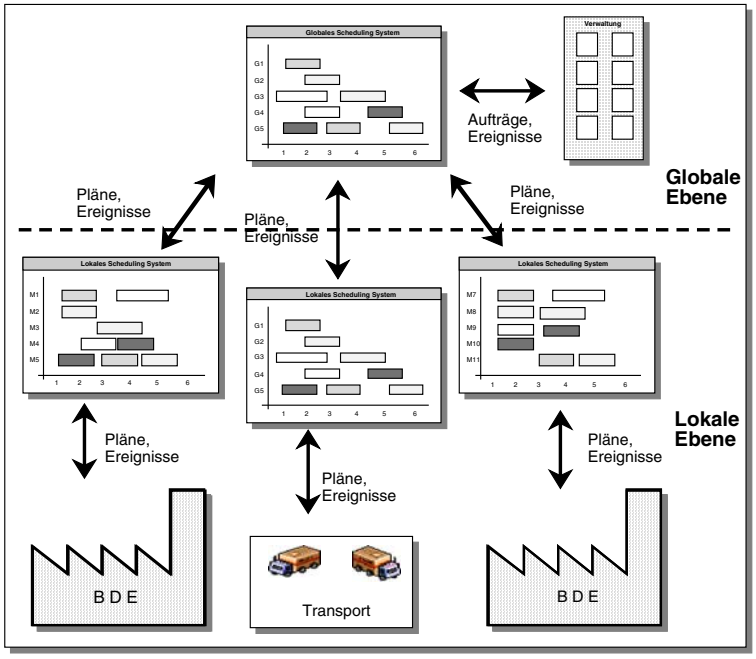


Bild 1: Multi-Site Scheduling

Die lokale Planungsebene existiert in der Regel schon. Hier sind (bereits existierende) Ansätze zur Lokalen/ Reaktiven Planung [Smi92, KS95, ZF94] angesiedelt, die integriert werden können. In der *Lokal Prädiktiven Planung* wird ausgehend von den globalen Vorgaben ein lokaler Plan erstellt. Störungen auf der lokalen Ebene werden zunächst durch die *Lokal Reaktive Planung* behandelt. Haben diese Störungen auch Auswirkungen auf andere lokale Systeme oder die Herstellung des Gesamtprodukts, falls z.B. Zwischenprodukte, die an anderem Ort benötigt werden, nicht rechtzeitig fertig werden, so muß die globale Ebene eingeschaltet und dort entsprechend auf die veränderte Situation reagiert werden. Die lokalen Systeme müssen dementsprechend um die Kommunikationsmöglichkeiten und eine entsprechende Ereignisbehandlung erweitert werden.
Der Kommunikation zwischen den einzelnen Ebenen fällt eine besondere Bedeutung zu, da nur so die Koordination der verschiedenen Planungsaktivitäten möglich ist. Dabei werden von der globalen zur lokalen Ebene mindestens folgende Informationen weitergegeben:

- der globale Plan bestehend aus den internen Aufträgen, den zugehörigen Zwischenprodukten, den zu benutzenden Maschinengruppen, den (möglichst) einzuhaltenden Zeitfenstern und den benötigten Mengen der Zwischenprodukte,
- Ereignisse mit Bedeutung für die lokale Ebene (z.B. Ausfall eines Auftrags).

Von der lokalen zur globalen Ebene sind es mindestens:

- die lokale Realisierung der globalen Vorgaben mit den internen Aufträgen, den zugehörigen Zwischenprodukten, jeweils Anfang und Ende der lokalen Einplanung und verwendete Maschinengruppen,
- aufgetretene Fehler,
- Vorschläge für eine mögliche lokale Umplanung.

Im Rahmen der Koordination muß auch eine Konsistenzsicherung bzgl. des Datenbestandes durchgeführt werden, damit alle beteiligten Planungssysteme auf dem gleichen aktuellen Datenbestand arbeiten können. Dazu muß eine Ereignisverarbeitung realisiert werden, die alle auftretenden Ereignisse auch für die entsprechenden Systeme verfügbar macht und sie in der richtigen Reihenfolge verarbeiten kann.

Entsprechend der beschriebenen Aufgaben wurde die in Bild 2 dargestellte Architektur des MUST-Systems gewählt.
Lokale Systeme bestehen im wesentlichen aus bereits bekannten und realisierten Komponenten, z.B. [SB97]. Zusätzlich muß eine Komponente eingefügt werden, die die Kommunikation zwischen globaler und lokaler Ebene realisiert.

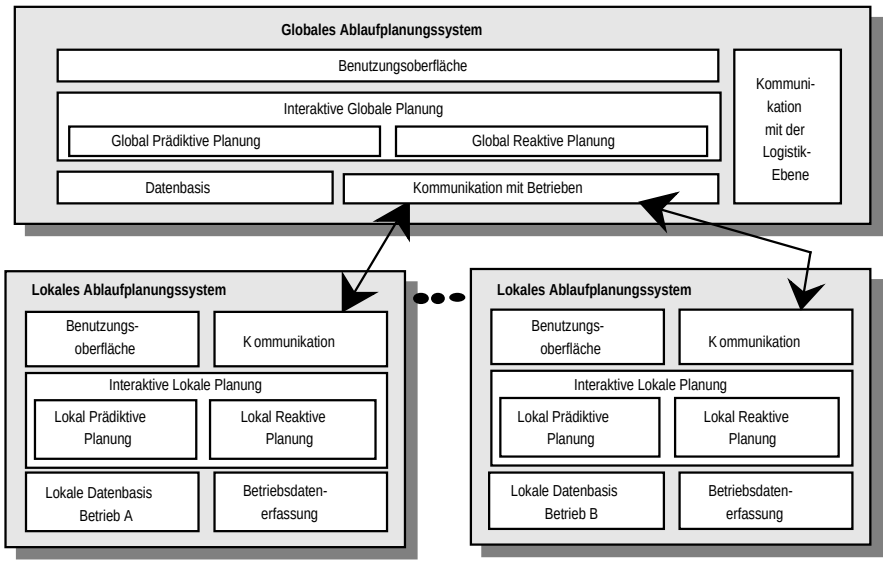


Bild 2: Architektur des MUST-Systems

Auf der globalen Ebene gibt es ein Globales Ablaufplanungssystem, das die Aufgaben der Planung und Koordination übernimmt. Das User-Interface stellt dem Benutzer einen globalen Leitstand zur Verfügung, der den globalen Ablaufplan und die wichtigen Informationen für die Durchführung der globalen Aktivitäten präsentiert. In den Komponenten für die Global Reaktive und Global prädiktive Planung werden unterschiedliche wissensbasierte Verfahren zur Generierung eines Globalen Vorgabeplans, z.B. über Heuristiken, und zur Reaktion auf Ereignisse der lokalen Ebene eingesetzt. Die globale Datenbasis enthält das Wissen über die Objekte der Ablaufplanung, z.B. Endprodukte mit ihren Zwischenprodukten sowie benötigte Maschinengruppen und Zeiten zur Herstellung. Wichtig sind auch hier die Komponenten zur Kommunikation, zum einen mit den Planungssystemen der lokalen Ebene und zum anderen mit der Logistik-Ebene, um die Entscheidungen berücksichtigen zu können.

Die Kommunikation zwischen den einzelnen Ebenen kann auf unterschiedliche Art realisiert werden. Implementiert ist ein Blackboard-Ansatz, auf dessen Basis weitere Untersuchungen folgen sollen.
Im Rahmen des Projekts wurden verschiedene Ansätze auf ihre Anwendbarkeit in den einzelnen Aufgabenbereichen untersucht. Tabelle 1 zeigt die Zuordnung, die aufgrund der Struktur der Ablaufplanungsprobleme und der prinzipiellen Eignung der Lösungsverfahren getroffen wurde.
Im folgenden sollen besonders die Fuzzy-Logik, Neuronale Netze und Genetische Algorithmen betrachtet werden.

Problembereich der Ablaufplanung	**Lösungsverfahren**
global prädiktiv	Heuristiken, Constraints, Fuzzy-Logik
global reaktiv	Heuristiken, Constraints, interaktive Planung mit Konsistenzprüfung
lokal prädiktiv	Heuristiken, Constraints, Genetische Algorithmen, OR-Verfahren, Neuronale Netze
lokal reaktiv	Heuristiken, Constraints, interaktive Planung mit Konsistenzprüfung, Multi-Agenten Systeme.

Tabelle 1: Problembereiche und Lösungsverfahren

2. Global Prädiktive Ablaufplanung mit FUZZY-Konzepten

Die globale Planung läßt sich entsprechend obiger Beschreibung konkret charakterisieren durch:

- Maschinengruppen als Ressourcen mit unterschiedlichen kumulierten Kapazitätsangaben sowie Transportzeiten und -kosten und die Menge aller Materialien in der Fertigungsstätte.
- Endprodukte, die jeweils in mehreren Varianten produziert werden können. Jede Variante besteht aus mehreren Zwischenprodukten, die auf verschiedenen Maschinengruppen mit unterschiedlicher Kapazitätsbelastung und Dauer mit bestimmten Materialien gefertigt werden können.
- externe Aufträge zur Herstellung jeweils eines Endprodukts mit einer Mengenangabe, dem Starttermin, dem Endtermin und einer Wichtung des externen Auftrages.
- die globalen Hard Constraints, u.a.:

- Alle externen Aufträge sind auszuführen.
- Genau eine Variante mit allen Zwischenprodukten ist für jeden externen Auftrag zu verwenden.
- Die Reihenfolge entsprechend der Präzedenzrelation der Zwischenprodukte ist einzuhalten.

• die globalen Soft Constraints, u.a.:
- Der vorgegebene Endtermin eines Endproduktes sollte von keinem seiner Zwischenprodukte überschritten werden.
- Die Transportzeiten und -kosten sind minimal zu halten.
- Die Fertigungsstätten sind gleichmäßig zu belasten.
- Die Lösung des globalen Ablaufplanungsproblems sollte möglichst robust sein, d.h. genügend Planungsspielraum für den lokalen Planer lassen.

Als Lösung des globalen Ablaufplanungsproblems ergibt sich ein globaler Vorgabeplan für die lokale Planung, der die zeitliche Zuordnung von Zwischenprodukten zu Maschinengruppen darstellt.
Das unvollständige Wissen bzw. die kumulierten Angaben der globalen Ablaufplanung, wie z.B. die Zusammenfassung der Maschinen zu Maschinengruppen mit kumulierten Kapazitätsaussagen, lassen sich recht gut mit Fuzzy-Logik modellieren und verarbeiten. Dazu werden die charakteristischen Informationen durch linguistische Variablen beschrieben, die mit Hilfe von Fuzzy-Regeln zu neuen Informationen verknüpft werden.
Im realisierten Ansatz für die globale Ablaufplanung [ASS97] werden die Planungsobjekte durch folgende ausgewählte Merkmale beschrieben:

- Ressourcen: durch Apparategruppenkapazität, Auslastung, unscharfe Transportkosten und -zeiten, Materialverbrauch und Materialverschleiß
- Produkte: durch Produktkapazität und Produktzeitverbrauch
- externe Aufträge: durch Zeitbedarf, Priorität, Termin und Wichtigkeit
- Hard-/ Soft Constraints: jeweils durch den Erfüllungsgrad des Constraints.

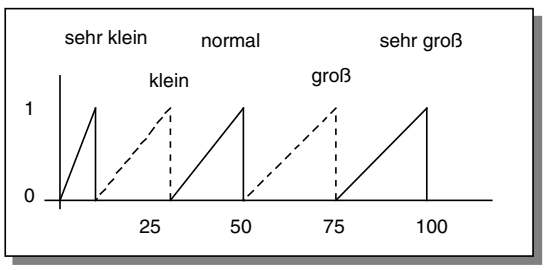


Bild 3: Zugehörigkeitsfunktionen

Alle anderen Merkmale werden durch scharfe Werte beschrieben. Dabei wird z.B. mit den Werten {sehr klein, klein, normal, groß, sehr groß} für die Apparategruppenkapazität zu jeder Zeiteinheit eine freie Kapazität beschrieben, die besagt, inwieweit die Apparategruppe noch belastet werden kann. Die entsprechenden Zugehörigkeitsfunktionen sind in Bild 3 dargestellt und zeigen die linguistischen Variablen für eine Prozentangabe an freier Kapazität.
Die Fuzzy-Regeln verwenden die aktuellen Werte der angegebenen Variablen und erlauben eine Einplanungsentscheidung. Für die Ermittlung einer Einplanungsreihenfolge werden die Aufträge entsprechend ihrer „Wichtigkeit“ sortiert. Diese Wichtigkeit wird u.a. durch Regeln wie

```
/* Regeln zur Ermittlung der Wichtigkeit
   von Aufträgen */
IF Zeitbedarf(sehr klein) FUZZY_AND Priori-
   tät(normal) FUZZY_AND Termin(bald)
   THEN Wichtigkeit(normal)
IF Termin(noch nicht)
   THEN Wichtigkeit(unwichtig)
IF Zeitbedarf(sehr klein) FUZZY_AND Priori-
   tät(hoch) FUZZY_AND Termin(bereits)
   THEN Wichtigkeit(sehr wichtig)
```

ermittelt. Dabei sind die gewünschten Zielsetzungen implizit in den verwendeten Regeln enthalten. Die Regelverarbeitung ist dabei zweistufig angelegt. In der ersten Stufe wird eine Strategie beschrieben, z.B. die Abfolge der Schritte „ermittle Planungsreihenfolge“, „ermittle zugehörige Ressource pro Zeiteinheit“, „weise Material zu“. Zu jedem Strategieschritt ist dann eine Regelmenge vorhanden, die die entsprechenden Entscheidungen ermöglicht. Für das MUST-System wurde zunächst ein allgemeiner Ansatz realisiert, der über einen Editor die Definition der Fuzzy-Mengen, Variablen und der zugehörigen Regelmengen erlaubt [ASS97].

Die so erstellte globale Planvorgabe kann interaktiv weiterverarbeitet werden. Die Ergebnisse der Planung lassen sich durchaus mit denen heuristischer Verfahren vergleichen. Positiv sind vor allem die besseren Modellierungs- und Verarbeitungsmöglichkeiten von ungenauen Angaben zu bewerten, so daß Fuzzy-basierte Komponenten in Planungssystemen ihren Platz finden werden.

3. Lokal Prädiktive Ablaufplanung mit Genetischen Algorithmen und Neuronalen Netzen

Auf der lokalen Ebene werden die Vorgaben des globalen Plans in geeignete lokale Ablaufpläne umgesetzt. Auch hier muß natürlich auf die Ereignisse der lokalen Fertigungsumgebung, die u.a. durch die Betriebsdatenerfassung ermittelt werden, reagiert werden. Der lokale Bereich läßt sich charakterisieren durch

- Maschinen als Ressourcen mit ihren technischen Restriktionen sowie weitere Ressourcen.
- eine Menge von Zwischenprodukten, die jeweils in mehreren Varianten produziert werden können. Jede Variante besteht aus mehreren Schrit-

ten, die auf verschiedenen Maschinen gefertigt werden können.
- Aufträge für Zwischenprodukte mit Angaben zu Menge, Start, Ende und Priorität.
- die lokalen Hard Constraints, u.a.:
  - Genau eine Variante mit allen Schritten ist für jeden Auftrag zu verwenden.
  - Die Reihenfolge entsprechend der Präzedenzrelation ist einzuhalten.
- die lokalen Soft Constraints, u.a.:
  - Die Fertigungstermine sollten eingehalten werden.
  - Die Ressourcen sollten gleichmäßig ausgelastet werden.
  - Die Durchlaufzeiten sollten minimal sein.

Hier wurden neben heuristischen Verfahren auch ein Ansatz auf Basis genetischer Algorithmen sowie auf Basis neuronaler Netze untersucht. Beide betrachten Ablaufplanung als kombinatorisches Optimierungsproblem und versuchen eine Zielfunktion zu optimieren.

3.1 Lokal Prädiktive Ablaufplanung mit Genetischen Algorithmen

Genetische Algorithmen zählen zu den iterativen Verbesserungsverfahren [Dor95], d.h. daß ausgehend von einer (oder einer Menge von) Anfangslösung(en) so lange neue (möglichst bessere) Lösungen gesucht werden, bis ein bestimmtes Abbruchkriterium erfüllt ist. Die Vorgehensweise und die Begriffswelt sind bei genetischen Algorithmen an die biologische Evolution angelehnt. Ausgehend von einer Menge von Individuen (Lösungen, hier Plänen), der Anfangspopulation, werden die Schritte „Selektion“ (Auswahl einer Teilmenge zur Rekombination), „Crossover“ (Erzeugung neuer Lösungen durch Rekombination von Individuen), „Mutation“ (Veränderung einzelner Individuen) so lange durchlaufen, bis ein bestimmtes Kriterium erfüllt ist, z.B. bis ein bekanntes Optimum gefunden oder eine Anzahl von Iterationen durchlaufen ist. Meist wird dabei ein Optimierungsproblem betrachtet, für das eine entsprechende Evaluierungsfunktion angegeben wird.
Der für das System MUST untersuchte genetische Algorithmus zur lokal prädiktiven Ablaufplanung [Bru93, Bru96] enthält spezifische Erweiterungen, die für die Lösung des beschriebenen lokalen Ablaufplanungsproblems entwickelt wurden.
Für die komplexe Problemstellung mit Varianten und Alternativmaschinen reicht eine einfache, indirekte Repräsentation des Plans, z.B. durch einen Bitstring i. allg. nicht aus. Daher wurde eine direkte Repräsentation des Planungsproblems gewählt, d.h. einzelne Individuen repräsentieren eindeutige Pläne, und nicht nur Reihenfolgen von Aufträgen, die erst noch zu gültigen Plänen umgerechnet werden müssen. Ein zulässiger Ablaufplan wird durch eine Liste von Operation/ Variante/ Maschine/ Produktionsintervall-Tupeln repräsentiert und basiert im Unterschied zu anderen GA-Ansätzen auf einer komplexen Datenstruktur, die sämtliche Information enthält, die für die eindeutige Beschreibung eines Ablaufplanes notwendig ist.
Passend zu dieser Repräsentation wurden spezielle Operatoren für Selektion, Crossover und Mutation entworfen, die die Repräsentation und gleichzeitig bestimmte Constraints der Planungsumgebung berücksichtigen. Die Operatoren machen intensiven Gebrauch von heuristischem Problemwissen und haben die Funktionalität von wissensbasierten Planungsalgorithmen, womit auch die Zulässigkeit der generierten Nachkommenpläne garantiert wird. Ziel des Verfahrens ist dabei die Minimierung der Verzüge, was ebenfalls bei der Gestaltung der Operatoren berücksichtigt wurde. Ausgehend von einer proportionalen Selektion werden beim Crossover die Variablen auftragsweise zusammengefaßt und zunächst alle nicht verspäteten Aufträge komplett aus einem Elternteil extrahiert. Für die verspäteten Aufträge werden aus dem zweiten Elternteil die Reihenfolgen und Maschinenbelegungen übernommen und konsistent mit den frühest möglichen Startzeiten versehen. Somit werden jeweils positive Eigenschaften aus mindestens einem Elternteil vererbt. Bei der Mutation entstehen für zufällig gewählte Variablen neue Werte, indem einer Variantenvariablen zufällig eine neue Variante, einer Startvariablen der früheste (kleinste) zulässige Startzeitwert und einer Maschinenvariablen zufällig eine neue Maschine, die zum gewählten Startzeitwert verfügbar ist, zugewiesen werden.
Die Ergebnisse [Bru96] zeigen, daß teilweise erhebliche qualitative Verbesserungen der Pläne erreicht werden können, allerdings führen die hohen Laufzeiten und die Struktur der Algorithmen dazu, daß genetische Algorithmen i.allg. nur für prädiktive Aufgabenstellungen mit klaren Bewertungsfunktionen einsetzbar scheinen.

3.2 Lokal Prädiktive Ablaufplanung mit Neuronalen Netzen

Ziel des hier entwickelten Ansatzes war es, mit Neuronalen Netzen realistische Planungsszenarien hoher Komplexität bearbeiten zu können. Dabei sollen vor allem größere Auftragsmengen, verschiedene Fertigungsvarianten und Alternativmaschinen modelliert und verarbeitet werden können.
Der verwendete Ansatz [MS97] löst das lokale Planungsproblem durch Kombination einer Heuristik zur Problemzerlegung mit Neuronalen Netzen. Das Planungsproblem wird dabei in Teilprobleme zerlegt, die dann jeweils durch ein neuronales Netz gelöst werden. Die heuristische Zerlegung der Ablaufplanung führt zu folgendem Planungsvorgehen:

1. Auswahl einer Fertigungsvariante für jeden Auftrag;
2. Auswahl einer Maschine für jede Operation der gewählten Variante;
3. zeitliche Festlegung der Operationen.

Die Heuristik (auftragsbasierte Problemzerlegung) beruht auf Erfahrungen aus der Praxis, daß eine sinnvolle Wahl von Varianten bzw. Maschinen möglich ist, bevor die Bearbeitung der Aufträge bzw. der Operationen zeitlich fixiert wurde. Dies gilt, weil die Auslastung von Maschinen(gruppen) anhand der Produktionsvorschriften in Verbindung mit den frühesten Start- und spätesten Endterminen abgeschätzt werden kann. Zudem ist die Varianten- und Maschinenauswahl weitgehend unabhängig von der Zielfunktion für den zu erstellenden Plan. Für die drei Teilaufgaben werden dann jeweils Netzttypen verwendet, die die Lösung von Optimierungsproblemen unterstützen, da auch hierbei die Minimierung der Verzüge als Zielfunktion gewählt wurde. Sie geht vor allem in die dritte Planungsstufe (Zeitplanung) ein.

Im Gegensatz zur sequentielle Aufgabenzerlegung der Heuristik wenden Neuronale Netze implizit eine parallele Problemzerlegung an, bei der in jeder Planungsstufe alle Aufträge bzw. Operationen, alle Maschinen bzw. Maschinengruppen sowie alle Punkte der Zeitachse gleichzeitig betrachtet werden. Auf diese Weise entfallen einige der Einschränkungen, die sich aus einer sequentiellen Bearbeitung ergeben, z. B. wird damit ein größerer Problemraum betrachtet.

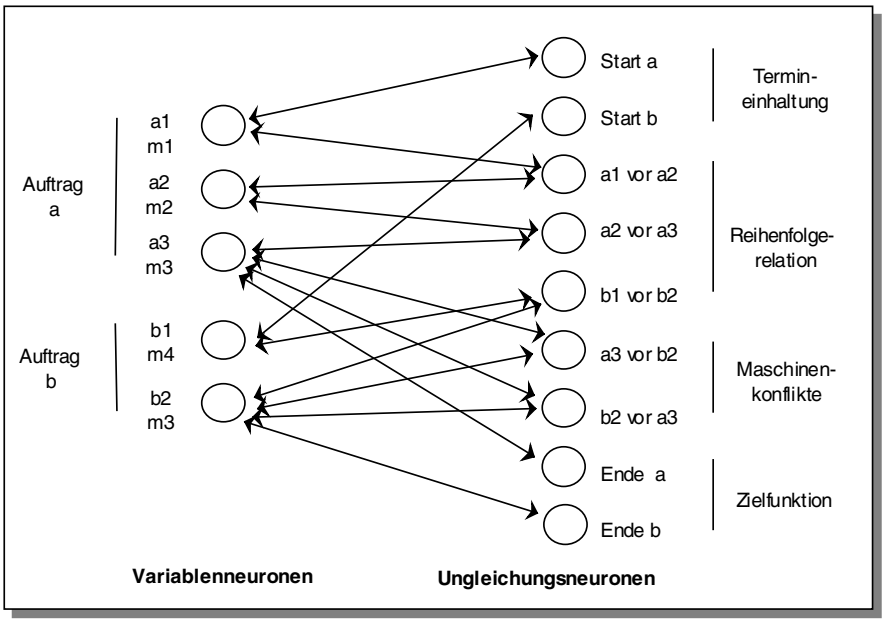


Bild 4: Beispiel für LP-Netz

In der ersten Planungsstufe wird ein modifiziertes Hopfield-Tank-Netz eingesetzt, um Varianten für die betrachtete Menge von Aufträgen auszuwählen. Dabei sollen die zur Verfügung stehenden Maschinen möglichst gleichmäßig ausgelastet werden. Das Netz besteht aus vollständig untereinander vernetzten Neuronen, die jeweils die Zuordnung Auftrag/Herstellungsvariante repräsentieren. Eine Gewichtsfunktion wurde entworfen, die eine zulässige Lösung ermittelt, bei der die Maschinenkonflikte minimal und die Verteilung der Belastung gleichmäßig sind.

Im zweiten Schritt findet die Maschinenauswahl ebenfalls mit Hilfe eines Hopfield-Tank-Netzes statt. Hier repräsentieren die Neuronen die Zuordnung von Operationen zu Alternativmaschinen. Auch hier werden zulässige Lösungen ermittelt, die die Maschinenkonflikte minimal halten.

Die dritte Planungsstufe wird mit einem LP-Netz bearbeitet (Bild 4). Als Variablen für die LP-Aufgabe werden die Startzeitpunkte der einzelnen Operationen gewählt. Die Constraints zur Termineinhaltung, zur Ausführungsreihenfolge und zur Vermeidung von Doppelbelegungen werden als Ungleichungen formuliert und jeweils durch ein Ungleichungsneuron repräsentiert. Die Neuronenwerte der Variablen werden dann solange verändert, bis sie alle Bedingungen in den Ungleichungsneuronen erfüllen. Durch die Definition entsprechender Schwellwerte wird erreicht, daß alle Aufträge möglichst früh beendet werden und damit das gewünschte Kriterium optimiert wird.

Wie bei den genetischen Algorithmen zeigt sich auch hier, daß die errechneten Pläne von guter Qualität sind, die Laufzeit und der Modellierungsaufwand aber Gründe gegen die Verwendung von Neuronalen Netzen für solche komplexen Ablaufplanungsprobleme sind.

4. Zusammenfassung

Einige der Probleme des Multi-Site Scheduling, bei dem die Ablaufplanung für Unternehmen mit mehreren Produktionsstandorten auf zwei Ebenen koordiniert durchgeführt werden muß wurden im vorliegenden Papier beschrieben. Dazu wurden drei verschiedene Ansätze aus dem Bereich des Softcomputing zur Lösung von Teilproblemen des

Multi-Site Scheduling vorgestellt. Genetische Algorithmen und Neuronale Netze werden hier für die Lösung kombinatorischer Optimierungsprobleme bei der Erstellung von lokal prädiktiven Ablaufplänen eingesetzt, Fuzzy-Konzepte zur Modellierung und Verarbeitung von unvollständigem Wissen auf der global prädiktiven Ebene zur Erstellung eines globalen Vorgabeplans. Insgesamt bieten alle drei Ansätze qualitativ gute Lösungen, lassen sich aber nicht für alle Planungsaufgaben einsetzen, vor allem nicht für reaktive Szenarien.

Literatur

[ASS97] H.-J. Appelrath, J. Sauer, and G. Suelmann. Globale Ablaufplanung mit Fuzzy-Konzepten. In: J. Biethahn, A. Höhnerloh, J. Kuhl, and V. Nissen, editors, *FuzzySet-Theorie in betriebswirtschaftlichen Anwendungen*. Verlag Vahlen, München, 1997.

[Bru93] R. Bruns. Direct Chromosome Representation and Advanced Genetic Operators for Production Scheduling. In: S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[Bru96] R. Bruns. *Wissensbasierte Genetische Algorithmen*. Infix-Verlag, 1996.

[BS95] R. Bruns and J. Sauer. Knowledge-Based Multi-Site Coordination and Scheduling. In: R. D. Schraft, editor, *Flexible Automation and Intelligent Manufacturing 1995*, Stuttgart, 1995. Begell House.

[Dor95] J. Dorn. Iterative Improvement Methods for Knowledge-Based Scheduling, *AICOM*, Vol. 8, No. 1, March, 1995.

[KS95] R.M. Kerr, E. Szelke. *Artificial Intelligence in Reactive Scheduling*. Chapman & Hall, 1995.

[MS98] H. Märtens and J. Sauer. Ein Ablaufplanungssystem auf Basis neuronaler Netze. In: J. Biethahn, et al., editors, *Betriebswirtschaftliche Anwendungen des Softcomputing*. Vieweg, Wiesbaden, 1998.

[Sau93] J. Sauer. *Wissensbasiertes Lösen von Ablaufplanungsproblemen durch explizite Heuristiken. DISKI*. Infix Verlag, 1993.

[Sau97] J. Sauer. Ablaufplanung. *KI Künstliche Intelligenz*, 2/97, 1997.

[SB97] J. Sauer, R. Bruns. Knowledge-Based Scheduling Systems in Industry and Medicine. In *IEEE-Expert*, February 1997.

[Smi92] S.F. Smith. Knowledge-based production management: approaches, results and prospects. In *Production Planning & Control*, Vol. 3, No. 4, 1992.

[ZF94] M. Zweben, M.S. Fox (Hrsg.). *Intelligent Scheduling*. Morgan Kaufmann Publishers, 1994.