Douglas Cunningham, Petra Hofstedt, Klaus Meer, Ingo Schmitt (Hrsg.): INFORMATIK 2015
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015

Ein Modell zur Beschreibung des Interaktionsablaufs zwischen Mensch und Computer bei der Lösung von Planungsproblemen

Anna Prenzel[1]

Abstract: Planungsprobleme, z.B. aus den Gebieten der Produktions-, Flotten- oder Personalplanung können mit Hilfe von Software in vielen Fällen nahezu vollständig automatisch gelöst werden. Trotzdem wird in der Regel auch ein menschlicher Disponent benötigt, der sicherstellt, dass bei der Planung alle betriebs- und situationsspezifischen Anforderungen erfüllt werden. Die Lösung eines Planungsproblems wird damit zu einem integrierten Prozess, an dem Mensch und Computer beteiligt sind. In diesem Papier wird ein Interaktionsmodell entwickelt, welches unabhängig vom konkreten Anwendungsfall das dabei stattfindende Zusammenspiel zwischen den beiden Parteien beschreibt. Die Aktivität des Disponenten wird dabei als schrittweiser Vorgang der Variablenbelegung für das zugrunde liegende Constraint-Erfüllungsproblem (CSP) aufgefasst. Das Modell erlaubt es, Faktoren zu identifzieren, die den Arbeitsaufwand für den Disponenten stark erhöhen können. Dazu gehört u.a. die Behandlung von Sackgassen bei der Variablenbelegung des CSPs und die Berücksichtigung von Optimierungszielen. Es wird geschlussfolgert, dass der für den Disponenten entstehende Arbeitsaufwand durch eine geeignete Computerunterstützung reduziert werden sollte.

Keywords: Planungsprobleme, Lösungsverfahren, Expertenwissen, Mensch-Computer-Interaktion, Backtracking, CSP

1 Einführung

1.1 Anwendungsgebiet

In diesem Papier werden Feinplanungsprobleme betrachtet, mit denen die Zuordnung von Aufgaben zu Startzeitpunkten und Ressourcen geregelt wird. Beispiele sind die Zuordnung von Transportgütern zu Fahrern, Schichten zu Personal oder Fertigungsaufträgen zu Maschinen. Es wird angenommen, dass die Planung für einen begrenzten Planungszeitraum P mit $h \in \mathcal{N}^+$ diskreten Zeitpunkten $P = \{1,..,h\}$ vorgenommen wird, für den eine feststehende Nachfrage an Gütern oder Dienstleistungen vorliegt. Die Nachfrage bestimmt eine Menge von n Aufgaben $AS = \{a_1,..,a_n\}$, die im Rahmen der Planung bestimmten Ressourcen und Ausführungszeitpunkten zugeordnet werden müssen. Es seien m verschiedene Ressourcen $RS = \{r_1,..,r_m\}$ vorhanden. An der Ausführung einer Aufgabe ist jeweils eine Ressource oder eine Kombination von Ressourcen beteiligt. Es sei RCS die Menge möglicher Ressourcen und Ressourcenkombinationen $RCS \subseteq \mathcal{P}(RS)$.

[1] Hochschule Zittau/Görlitz, Fakultät Elektrotechnik und Informatik, Brückenstraße 1, 02826 Görlitz, aprenzel@hszg.de

Das Planungsproblem kann als Constraint-Erfüllungsproblem (CSP) modelliert werden [HW07, Dec03]. Ein CSP wird durch ein Tripel (V,D,C) beschrieben. Dabei ist V eine Menge von Entscheidungsvariablen, D die Menge der Domänen der Variablen und C die Menge aller Randbedingungen (Constraints) über diesen Variablen. Sei VS mit $|VS| = n$ die Menge aller Variablen $start_i$, die den Startzeitpunkt der Aufgabe a_i repräsentieren und VR mit $|VR| = n$ die Menge aller Variablen res_i, die die Ressourcenzuweisung von $a_i, i \in \{1..n\}$ festlegen. Dann gilt im Allgemeinen $(VS \cup VR) \subseteq V$. Es können weitere problemspezifische Variablen vorliegen.
Die Zuordnung von Variablen zu Domänen wird durch die Abbildung $dom : V \to D$ geregelt. Für alle $v \in VS$ gilt: $dom(v) \subseteq P$ und für alle $v \in VR$ gilt: $dom(v) \subseteq RCS$. Jeder Variable $v \in V$ dürfen nur Werte zugewiesen werden, die in der zugehörigen Domäne $dom(v)$ enthalten sind. Für die Wertzuweisung werden die Abbildungen $\alpha_i : v_i \to dom(v_i), i \in \{1..|V|\}$ definiert.
Jede Randbedingung $c \in C$ wird über einer Sequenz $Sub_c = \langle v_1,..,v_s \rangle, v_j \in V, j \in \{1..s\}, s \in \mathcal{N}^+$ definiert. Dabei entspricht c einer Relation, d.h. einer Teilmenge des Kreuzproduktes der beteiligten Variablen: $c \subseteq dom(v_1) \times ... \times dom(v_s)$. Typische Randbedingungen bei Feinplanungsproblemen sind z.B. Reihenfolgeabhängigkeiten zwischen Aufgaben, Zeitfenster für die Startzeitpunkte von Aufgaben oder Beschränkungen bei der Ressourcenzuweisung von Aufgaben. Eine Lösung für ein Planungsproblem liegt dann vor, wenn für alle $c \in C$ gilt: $Sub_c = \langle v_1,..,v_s \rangle \Rightarrow (\alpha(v_1),..,\alpha(v_n)) \in c$. In der Praxis wird meist keine beliebige Lösung gesucht, sondern eine Lösung mit einem optimalen Zielfunktionswert, der sich aus den Werten der Variablen berechnet.
Problemklassen, die dieser Modellstruktur gehorchen, sind z.B. die Ressourcenbeschränkte Projektplanung (RCPSP, [BK12]) , die Werkstattplanung (JSSP, [BK12]) oder die Flottenplanung (VRPTW, [KLMS05]).

1.2 Ein allgemeines Verfahren zur Lösung von Planungsproblemen

Ein weit verbreitetes Basisverfahren zur automatischen Lösung von Planungsproblemen als CSP ist eine Tiefensuche mit chronologischem Backtracking [HW07]. In Algorithmus 1 wird das zugrunde liegende Lösungsprinzip dargestellt.
In jedem Rekursionsdurchgang wird mit einer problemspezifischen Variablenauswahlheuristik $select\_var : \mathcal{P}(V) \to V$ eine noch nicht zugewiesene Variable ausgewählt. Anschließend wird mit Hilfe einer Wertauswahlheuristik $select\_val_s : v_s \to dom(v_s), s \in \{1..|V|\}$ ein Wert ausgewählt, sodass *assigned* eine partielle Lösung [Dec03] bildet. Wenn es nicht gelingt, in den nächsten Rekursionsschritten die restlichen Variablen in *unassigned* mit Werten zu belegen, wird für v_s eine neue Wertauswahl vorgenommen. Dies wird solange wiederholt, bis alle nachfolgenden Rekursionsschritte erfolgreich sind (und somit eine Lösung B existiert) oder bis die Domäne d_s leer ist. In diesem Fall wird v_s wieder als nicht zugewiesen betrachtet (Zeile 20) und der Wert der zuvor zugewiesenen Variable wird geändert (Backtracking). Wenn die Domäne d der im ersten Aufruf von $backtrackSolve$ betrachteten Variable leer ist, existiert keine Lösung für das CSP.

Algorithmus 1 : Lösung eines CSPs mit Tiefensuche und Backtracking

input : Ein CSP (V,D,C)
output : Eine Lösung mit Wertzuweisungen für jede Variable, wenn eine Lösung existiert,
false, wenn keine Lösung existiert.

```
1  begin
2      unassigned ← V;
3      assigned ← ∅;
4      return backtrackSolve(assigned,unassigned);
5  end
6  backtrackSolve(assigned, unassigned) ≡
7  if unassigned ≠ ∅ then
8      v_s ← select_var(unassigned);
9      d_s ← dom(v_s);
10     while d_s ≠ ∅ do
11         value ← select_val_s(v_s);
12         d_s ← d_s \ {value};
13         v_select ← value;
14         assigned ← assigned ∪ {v_s};
15         unassigned ← unassigned \ {v_s};
16         B ← backtrackSolve(assigned, unassigned);
17         if B ≠ false then
18             return B;
19         else
20             assigned ← assigned \ {v_s};
21             unassigned ← unassigned ∪ {v_s};
22         end
23     end
24     returnfalse;
25 else
26     return assigned;
27 end
```

1.3 Die Beteiligung des Disponenten am Lösungsprozess

1.3.1 Gründe für die Beteiligung des Disponenten

Ein Planungsproblem ist strukturiert, wenn anhand des vorliegenden Modells (z.B. in Form eines CSPs) ohne Beteiligung des Disponenten automatisch ein Plan so erzeugt werden kann, dass alle betrieblichen Anforderungen erfüllt werden. In der Praxis sind die Probleme jedoch häufig semi-strukturiert, da einige Anforderungen nicht im Modell kodiert sind

[CvW12]. Dafür gibt es verschiedene Gründe:

1. Die Planungssoftware ist nicht individuell auf das Unternehmen oder die Organisation zugeschnitten, sondern wurde für einen breiten Anwenderkreis entwickelt (Standardanwendungen). Einige unternehmensspezifische Parameter, Variablen und Randbedingungen werden daher nicht bei der Planerstellung berücksichtigt.

Beispiel 1 (unternehmensspezifische Regeln) [dSvWJ11]:

- Fest angestellte Arbeiter sollten möglichst nicht in der gleichen Schicht wie befristet angestellte Mitarbeiter arbeiten.
- Fahrzeiten zwischen zwei Orten dürfen nicht mehr als 30 min. betragen.

2. Auch unternehmensspezifische Anwendungen bilden die Anforderungen häufig nicht vollständig ab, da es zum einen nicht immer möglich ist, während der Software-Herstellung alle Anforderungen zu ermitteln [Ceg08, CMWV99] und zum anderen nach der Einführung der Software neue Anforderungen entstehen können. Unspezifizierte Anforderungen umfassen häufig auch inoffizielle Planungsregeln: Der Disponent kennt aus Erfahrung die besten Ausführungszeitpunkte und Ressourcen für bestimmte Aufgaben.

Beispiel 2 (inoffizielle Regeln):

- Eine Aufgabe, deren Dauer mehr als 2 Stunden beträgt, sollte nicht zu Beginn des Arbeitstages eingeplant werden.
- Alle Aufgaben für Kunde X sollten von Arbeiter Y ausgeführt werden.

3. Das Planungsmodell wird bewusst gegenüber der Realität vereinfacht, um die Komplexität des Problems zu reduzieren und somit eine Lösungserstellung in akzeptabler Zeit zu ermöglichen. Es wird davon ausgegangen, dass der Disponent die zusätzlichen Anforderungen in einer automatisch generierten Ausgangslösung leicht manuell umsetzen kann [Ceg08].

4. Bestimmte Anforderungen können nur schwer kodiert werden. Nur der menschliche Disponent kann einschätzen, ob ein gegebener Plan zu einem bestimmten Zeitpunkt diese Anforderungen erfüllt oder nicht [dSvWJ11, CvW12].

Beispiel 3 (schwer kodierbare Anforderungen):

- Planänderungen sollten nachvollziehbar sein.
- Pläne sollten für die Mitarbeiter akzeptabel sein.
- Die Aufgaben sollten möglichst abwechslungsreich verteilt werden. [dSvWJ11].

6. Unabhängig von den restlichen Punkten wird das menschliche Urteilsvermögen des Disponenten benötigt, um Unsicherheiten in den Daten zu identifizieren und entsprechende Maßnahmen einleiten zu können [CvW12].

Beispiel 4 (vorausschauende Planung): Der Disponent einer Produktionsfirma erwartet eine Materiallieferung, die für nächste Woche angekündigt ist. Bei einer pünktlichen Lieferung könnte die Verarbeitung in der darauffolgenden Woche erfolgen. Da sich der Lieferant erfahrungsgemäß verspätet, plant sie der Disponent jedoch erst in der übernächsten Woche ein.

1.3.2 Die Umsetzung manueller Planungsentscheidungen

Es lassen sich zwei Arten von Planungsentscheidungen identifizieren, die vom Disponenten zur Umsetzung nicht kodierter Anforderungen getroffen werden können:

- Der Disponent beschränkt die Startzeitpunkte bestimmter Aufgaben jeweils auf bestimmte Bereiche im Planungshorizont.
- Der Disponent beschränkt die Ressourcenzuweisung bestimmter Aufgaben jeweils auf bestimmte Gruppen von Ressourcen.

Diese Planungsentscheidungen können auf zwei Wegen in den Planungsprozess eingebracht werden:

1. *Überführung in kodierte Anforderungen:* Dabei wird das Planungsmodell um unäre Randbedingungen erweitert, die den Wertebereich ausgewählter Variablen auf eine Teilmenge des ursprünglichen Wertebereiches (evtl. auf einen einzigen Wert) einschränken. Das erweiterte Planungsproblem kann anschließend ggf. automatisch gelöst werden.
2. *Manuelle Modifikation einer automatisch erzeugten Lösung:* Dabei werden die Werte von einzelnen (Startzeit- und Ressourcen-)Variablen in einer Lösung nachträglich geändert.

In vielen Fällen kann Expertenwissen nicht sofort in passende Planungsentscheidungen übersetzt werden [BPYS07]. Häufig ist nicht von vornherein bekannt, mit welchen Entscheidungen eine Anforderung am besten umgesetzt werden kann. In diesem Fall muss der

Disponent alternative Randbedingungen bzw. Wertzuweisungen aufstellen und bewerten. Dieser Vorgang lässt sich als 3-stufiger Entscheidungsprozess mit den Stufen *Intelligenz*, *Entwurf* und *Auswahl* modellieren [TSD11]:

Intelligenz: Die Intelligenzphase ist der Einstieg in den Entscheidungsprozess. In einer bestimmten Situation stellt der Disponent die Notwendigkeit fest, die Eigenschaften des Plans zu beeinflussen. Er sammelt Wissen, welches zunächst in Form von nicht kodierten Anforderungen vorliegt.

Entwurf: In dieser Phase werden Planungsentscheidungen gesucht, mit denen sich die Anforderungen beschreiben lassen. Dabei werden Alternativen entworfen, bewertet und miteinander verglichen. Folgende Kriterien fließen in die Bewertung ein:

- Wie gut werden die Anforderungen durch diese Entscheidungen repräsentiert?
- Existiert eine Lösung mit einer ausreichend guten Qualität, wenn die Entscheidungen im Modell bzw. in der Lösung umgesetzt werden?
- Wie gut ist der Kompromiss, der mit diesen Entscheidungen zwischen den einzelnen (ggf. widersprüchlichen) Anforderungen erreicht werden kann?

Auswahl: In der letzten Phase wird eine Alternative ausgewählt, d.h. es werden die Entscheidungen bestimmt, mit denen die Anforderungen am besten abgebildet werden können.

Der Entwurfsprozess kann übersprungen werden, wenn sich die Anforderungen direkt in Planungsentscheidungen überführen lassen.

2 Beschreibung des interaktiven Lösungsprozesses für Planungsprobleme

Der Disponent muss mit dem Planungssystem interagieren, um einzelne Planungsentscheidungen in den Planungsprozess einbringen zu können. Dazu gehört z.B. das Eintragen von Randbedingungen an der Benutzerschnittstelle oder das Ändern der Position einer Aufgabe im Plan (häufig können Aufgaben in einer interaktiven grafischen Visualisierung des Plans mittels Drag-and-Drop verschoben werden). Das System verarbeitet diese Aktionen und aktualisert das Planungsergebnis entsprechend, woraus der Disponent ggf. wiederum Schlussfolgerungen für neue oder geänderte Planungentscheidungen zieht. Es wird deutlich, dass im Rahmen des 3-stufigen Entscheidungsprozesses eine intensive Interaktion zwischen Mensch und Computer stattfinden kann. In diesem Abschnitt wird ein Modell zur Beschreibung der möglichen Interaktionsabläufe entwickelt.

2.1 Annahmen an das Planungssystem

Im nachfolgenden Szenario wird angenommen, dass das verwendete Planungssystem folgende Funktionalität bereitstellt:

- eine automatische Planerstellung, bei der (falls möglich) eine Lösung unter Einbeziehung aller Aufgaben oder wahlweise eine partielle Lösung unter Einbeziehung einer Teilmenge von Aufgaben erzeugt wird,
- die Möglichkeit zur Festlegung unärer Randbedingungen durch den Disponenten,
- die Möglichkeit der manuellen Wertzuweisung für beliebige Variablen durch den Disponenten (z.B. durch Verschieben einer Aufgabe in der grafischen Planansicht),
- die Möglichkeit, Wertzuweisungen für bestimmte Variablen offen zu lassen oder zu löschen, sodass eine partielle Instanziierung der Variablen vorliegt,
- die Benachrichtigung des Disponenten, wenn Randbedingungen verletzt werden,
- die Berechnung der Qualität des Plans in Form von Leistungskennzahlen.

2.2 Der übergeordnete Entscheidungsprozess

Algorithmus 2 modelliert den Ablauf des interaktiven Lösungsprozesses aus einer abstrakten Sichtweise. Die Zeilen 2 bis 14 umfassen den übergeordneten Entscheidungsprozess, in dem mehrere Alternativlösungen entwickelt werden, bis der Disponent den Vorgang abbricht. Jede Alternativlösung wird in den Zeilen 3 bis 13 erzeugt, was als eingebetteter Entscheidungsprozess aufgefasst werden kann:

Zeilen 3-7: Zunächst entscheidet der Disponent, ob vom Computer eine vollständige Ausgangslösung erzeugt werden soll ($userInitialize = true$) oder nicht ($userInitialize = false$). Durch die Prozedur $computerSolve$ wird ein beliebiger Lösungs- oder Optimierungsalgorithmus repräsentiert.

Zeilen 8-13: Wenn gilt $userInitialize = true$ und eine Lösung existiert, dann sind anschließend alle Variablen zugewiesen (und liegen in Liste *assigned* vor). Ansonsten sind noch keine Variablen zugewiesen (*assigned* ist leer). Der Disponent hat nun die Möglichkeit, neue Zuweisungen vorzunehmen oder bestehende Zuweisungen zu ändern oder zu lösen. Die Prozedur *userSolve* repräsentiert dabei den Entwurf einer alternativen (ggf. partiellen) Lösung, in der bestimmte Anforderungen umgesetzt werden sollen. Wenn die erzeugte Lösung noch unvollständig ist, kann sie der Disponent automatisch vervollständigen lassen (Zeile 13). Die in *assigned* bereits vorliegenden automatischen oder manuellen Wertzuweisungen werden dabei als Randbedingungen berücksichtigt.

Jede Alternative kann iterativ angepasst werden, bis der Disponent den Vorgang abbricht ($userCancelled = true$) und ggf. ab Zeile 2 die Umsetzung einer neuen Alternative beginnt. Eine weitere Iteration ist z.B. erforderlich, wenn durch $computerSolve$ festgestellt wird, dass für die in *assigned* vorgegebenen Zuweisungen keine Lösung existiert, sodass diese nochmals angepasst werden müssen. Auch wenn der Disponent aus der vervollständigten Lösung neue Präferenzen für die Lösungseigenschaften ableitet, ist eine weitere Iteration erforderlich.
Die automatische Planung kann vom Nutzer konfiguriert werden (Liste $settings$ als Para-

meter für *computerSolve*). Dies ermöglicht z.B. die Wahl eines Optimierungsverfahrens oder die Gewichtung von Zielfunktionen.

Algorithmus 2 : Interaktive Lösung eines CSPs für die Planung (Teil 1)

input : Ein CSP (V,D,C)
output : Eine Lösung mit Wertzuweisungen für jede Variable, wenn eine Lösung existiert,
false, wenn keine Lösung existiert.

```
1  begin
2      while !userCancelled do
3          unassigned ← V;
4          assigned ← ∅;
5          /*vgl. Algorithmus 3:*/
6          if userInitialize then
7              computerSolve(assigned, unassigned, settings);
8          end
9          while !userCancelled do
10             /*Zuweisungen manuell vervollständigen, ändern oder lösen:*/
11             userSolve(assigned, unassigned);
12             /*übrige Variablen in Liste unassigned zuweisen lassen:*/
13             computerSolve(assigned, unassigned, settings);
14         end
15     end
16 end
```

2.3 Der Prozess zum Entwurf einer Alternativlösung

In diesem Abschnitt steht der Entwurf einer einzelnen Alternativlösung im Vordergrund. Es wird davon ausgegangen, dass der Disponent eine Kombination von Planungsentscheidungen sucht, mit der eine oder mehrere Anforderungen beschrieben werden können. Der Prozess zur sukzessiven Eingabe der Entscheidungen an der Schnittstelle des Planungssystems besitzt Gemeinsamkeiten mit der Tiefensuche. Im Gegensatz zu Algorithmus 1 wird dabei sowohl die Variablen- als auch die Wertauswahl durch den Disponenten vorgenommen. Der vollständige Ablauf der Variablenbelegung wird in Algorithmus 3 nach dem Vorbild der Tiefensuche als rekursive Funktion modelliert[1]: In jedem Rekursionsdurchgang wird zunächst eine beliebige bereits zugewiesene oder noch nicht zugewiesene Variable ausgewählt (Zeile 2). Eine zugewiesene Variable kann vom Disponenten sofort in die Liste noch nicht zugewiesener Variablen verschoben werden (Zeilen 3-5). Dies ist erforderlich, wenn die Wertzuweisung für diese Variable nach der Entwurfsphase dem

[1] **Konventionen für die folgenden Interaktions-Algorithmen:** Funktionen mit dem Präfix *user* repräsentieren Entscheidungen oder (z.T. rein kognitive) Aktivitäten des Disponenten. Der Präfix *computer* kennzeichnet eine unterstützende Funktion, die vom Disponenten an der Benutzerschnittstelle aufgerufen werden kann. Im Kopf einer Bedingung (if..then) werden z.T. Variablen mit dem Präfix *user* eingesetzt, deren Wert für eine Ja/Nein-Entscheidung des Disponenten steht.

Computer überlassen werden soll (Algorithmus 2, Zeile 13) oder wenn der Disponent diese Variable zunächst ignorieren möchte, damit er andere Variablenzuweisungen leichter (d.h. ohne Verletzung von Randbedingungen) vornehmen kann. Wenn die Variable nicht ignoriert wird, wird für sie in den Zeilen 6 bis 43 ein (neuer) Wert gesucht. Dieser Vorgang gestaltet sich wie folgt:

Zeilen 10-12: Der Disponent wählt eine Reihe von alternativen Werten oder Teilbereichen des Wertebereiches aus, die aus seiner Sicht für eine Zuweisung in Frage kommen.

Zeilen 14-20: Jeder Wert wird daraufhin überprüft, ob er gültig und zufriedenstellend ist. Ein Wert ist gültig, wenn seine Zuweisung keine Randbedingungen mit bereits zugewiesenen Variablen verletzt. Er ist zufriedenstellend, wenn die um diese Wertzuweisung ergänzte (ggf. partielle) Lösung die Anforderungen des Disponenten ausreichend erfüllt.

Zeilen 21-32: Wenn kein passender Wert existiert, kann der Disponent die Zuweisung dieser Variable abbrechen (Zeilen 22-29) und den Lösungsprozess mit einer anderen Variable fortzusetzen (Zeile 41) bzw. den Entwurf dieser Alternative komplett abbrechen (Zeile 39). Eine Variable, die bereits zugewiesen war, erhält dabei ihren alten Wert zurück (Zeile 24). Alternativ kann der Disponent einen Backtracking-Prozess einleiten, indem er einige der vorhandenen Wertzuweisungen ändert oder löst (dazu wird in Zeile 30 die Funktion *userSolve* rekursiv aufgerufen, wobei nur die bereits zugewiesenen Variablen als Parameter übergeben werden). Anschließend kann ab Zeile 13 erneut eine Überprüfung der ausgewählten Werte stattfinden.
Im Gegensatz zu Algorithmus 1 findet kein systematischer Backtracking-Vorgang statt, bei dem jeweils die letzte Variablenzuweisung geändert wird. Die menschliche Kapazität des menschlichen Kurzzeitgedächtnisses ist gegenüber dem Computer stark begrenzt [Mac08], sodass nicht davon ausgegangen werden kann, dass der Disponent die Reihenfolge der betrachteten Variablen beibehält.

Zeilen 32-36: Wenn mindestens ein Kandidat existiert, kann der Disponent einen Wert aus der Liste kandidierender Werte auswählen und der betrachteten Variable zuweisen.

Die Zuweisung zu jeder ausgewählten Variable kann als eingebetteter Entscheidungsprozess mit Entwurfs- und Auswahlphasen betrachtet werden, da sich der Disponent in der Regel zwischen alternativen Wertzuweisungen entscheiden muss.

Algorithmus 3 : Interaktive Lösung eines CSPs für die Planung (Teil 2)

```
1  userSolve(assigned, unassigned) ≡
2  /*Variablenauswahl aus assigned oder unassigned:*/
   V_select ← user_select_var(assigned, unassigned);
3  if userIgnores && V_select ∈ assigned then
4      assigned ← assigned \ {V_select};
5      unassigned ← unassigned ∪ {V_select};
6  else
7      if V_select ∈ assigned then
8          old_value ← α(V_select);
9      end
10     unassigned ← unassigned \ {V_select};
11     sub_domain ← user_select_values(D_select);
12     candidates ← ∅;
13     while candidates = ∅ do
14         for each value ∈ sub_domain do
15             V_select ← value;
16             if computer_is_feasible?(V_select) && user_is_satisfying?(V_select) then
17                 candidates ← candidates ∪ value;
18             end
19             V_select ← null;
20         end
21         if candidates = ∅ then
22             if userCancelledWhile then
23                 if V_select ∈ assigned then
24                     V_select ← old_value;
25                 else
26                     unassigned ← unassigned ∪ {V_select};
27                 end
28                 /*Sprung zu Zeile 38:*/ break;
29             else
30                 userSolve(assigned);
31             end
32         else
33             value ← user_select_val(candidates);
34             V_select ← value;
35             assigned ← assigned ∪ {V_select};
36         end
37     end
38     if userCancelled then
39         return;
40     else
41         userSolve(assigned, unassigned);
42     end
43 end
```

3 Bewertung verschiedener Interaktionsszenarien hinsichtlich des zu erwartenden Arbeitsaufwandes

Aus dem in Algorithmus 3 beschriebenen Interaktionsmodell können verschiedene konkrete Interaktionsszenarien abgeleitet werden. Typische Szenarien sind z.B.:

Manuelle Planung: Der Disponent nimmt alle Variablenzuweisungen selbst vor (in Algorithmus 2 gilt $userInitialize = false$).

Automatische Planung mit nachträglicher manueller Modifikation: Es wird eine Ausgangslösung erzeugt ($userInitialize = true$). Anschließend werden einige Zuweisungen manuell geändert.

Automatische Planung unter Berücksichtigung zusätzlicher Randbedingungen: Es wird keine Ausgangslösung erzeugt ($userInitialize = false$). Der Disponent nimmt einige manuelle Wertzuweisungen vor (dies entspricht der Festlegung von Randbedingungen) und lässt die Lösung anschließend automatisch vervollständigen.

Die Szenarien können miteinander kombiniert werden. Im Folgenden wird jedes Szenario auf Faktoren untersucht, die den Zeitaufwand und die Schwierigkeit der Arbeit des Disponenten erhöhen oder seine Motivation zur Beteiligung am Lösungsprozess beeinträchtigen können. Es wird vorausgesetzt, dass die in Abschnitt 2.1 beschriebenen Annahmen an das Planungssystem gelten. In Abschnitt 3.4 werden daraus Schlussfolgerungen für die Gestaltung der Mensch-Computer-Schnittstelle gezogen.

3.1 Manuelle Planung

Eine manuelle Planung ist für den Disponenten in der Regel mit einem hohen Arbeitsaufwand verbunden. Dies gilt besonders dann, wenn während der Zuweisung häufig Sackgassen auftreten. Eine Sackgasse liegt vor, wenn der Disponent die Spezifikation einer Alternativlösung nicht mit weiteren Planungsentscheidungen fortsetzen kann, ohne Randbedingungen zu verletzen ($is\_feasible$ ist stets $false$ in Algorithmus 3, Zeile 16). Der Disponent muss daher die bisher erstellte partielle Lösung abwandeln (Zeile 30).
Der Disponent hat in der Regel keine Möglichkeit, das Auftreten von Sackgassen durch eine vorausschauende Planung zu verhindern. Wenn er eine Planungsentscheidung trifft, kann er die Auswirkungen auf die restlichen Variablen nicht sofort feststellen, denn bei einer großen Menge an Variablen und Randbedingungen ist es ihm nahezu unmöglich, im Kopf eine vorausschauende Propagierung durchzuführen. Der Grund ist zum einen die begrenzte Kurzzeitgedächtniskapazität des Menschen [Mac08], die von der Größe praktischer Planungsprobleme in der Regel weit überschritten wird. Zum anderen sind die zugrunde liegenden Randbedingungen in der Regel unbekannt, sodass sie nicht in die Propagierung einbezogen werden können.
Ob der Wertebereich einer Variablen nach der Propagierung seiner Entscheidung keine Werte mehr enthält, kann der Disponent also erst dann feststellen, wenn er die betroffene

Variable betrachtet. Diese steht u.U. in der vom Disponenten gewählten Zuweisungsreihenfolge sehr weit hinten. Es finden also u.U. zahlreiche Belegungen nicht betroffener Variablen statt, bevor ein Konflikt bemerkt wird. An dieser Stelle ist die Konfliktursache jedoch meist nicht mehr nachvollziehbar, d.h. der Disponent kann die Variable, für die ein Konflikt vorliegt, nicht mehr der Variable zuordnen, die den Konflikt verursacht hat. Letztere muss, wenn der Wertebereich leer ist oder nicht mehr den gewünschten Wert enthält, in einem zeitaufwändigen Backtracking-Prozess herausgefunden werden.
In Abhängigkeit vom Modell und von der Anzahl der Variablen kann es für den Disponenten zudem schwierig bzw. unmöglich sein, neben der Erfüllung der Randbedingungen auch eine Optimierung der Zielfunktionen anzustreben.

3.2 Automatische Planung mit nachträglicher manueller Modifikation

Verschlechterung der Qualität
Nachträgliche lokale Modifikationen können die Qualität des Plans bezüglich ein oder mehrerer Kennzahlen verschlechtern. Der Disponent arbeitet nach dem Prinzip einer Verbesserungsheuristik: Jede Planänderung, bei der die Randbedingungen berücksichtigt werden, führt zu einer Nachbarschaftslösung mit einer besseren oder schlechteren Qualität. Auch wenn der Disponent neben der Umsetzung von Anforderungen auch auf die Optimierung der Zielfunktionen achtet, besteht die Gefahr, ein lokales Optimum zu erreichen. Häufig befindet sich im Lösungsraum des Planungsproblems ein Plan, der den Vorstellungen des Disponenten ebenfalls entspricht und gleichzeitig eine bessere Qualität besitzt. Es ist jedoch schwierig, diesen Plan manuell zu finden. In der Regel ist dazu eine größere Umstellung notwendig, die mit einem hohen Arbeitsaufwand verbunden ist.

Selbstgefälligkeit
Dieser Effekt bezeichnet das übermäßige Vertrauen eines Nutzers in ein automatisiertes System und die damit verbundene Verschlechterung des Expertenwissens. Er kann dazu führen, dass nach einer automatischen Planerstellung keine manuellen Modifikationen am Plan vorgenommen werden, obwohl noch nicht erfüllte Präferenzen vorhanden sind. Zum einen weiß der Disponent, dass es ihm (im Gegensatz zum Computer) schwerfällt, gute Qualitätskennzahlen des Gesamtplans zu erreichen. Zum anderen sind ihm u.U. nicht alle Randbedingungen und Regeln bekannt, die in das Planungssystem eingepflegt wurden. Es entsteht eine Unsicherheit darüber, ob wichtige Faktoren bei der geplanten Modifikation nicht berücksichtigt wurden. Die Befürchtungen, die Qualität zu verschlechtern oder Fehlentscheidungen zu treffen, lassen es letztendlich am sichersten erscheinen, die Planung vollständig dem Computer zu überlassen.

Aufwand der manuellen Planung → Abschnitt 3.1

3.3 Automatische Planung unter Berücksichtigung zusätzlicher Randbedingungen

Planung durch Versuch und Irrtum
Dem Disponenten wird keine Unterstützung dabei geboten, die Randbedingungen nach

dem Gesichtspunkt der Lösbarkeit auszuwählen. Die automatische Planung muss u.U. mehrfach mit angepassten Randbedingungen aufgerufen werden, bevor das Problem lösbar ist (mehrere Iterationen in Algorithmus 2, Zeilen 9-14). Der Arbeitsaufwand der Planung wird dadurch erhöht.

Mangelnde Kontrolle über das Ausmaß der Umplanung
Die nachträgliche Modifikation eines Plans kann die Anpassung von bestehenden Zuweisungen erfordern (Backtracking). Wenn der Disponent die Anpassung nicht manuell vornehmen möchte, kann er einige der bestehenden Zuweisungen lösen (Algorithmus 3, Zeilen 4-5) und den Plan nach seiner Modifikation automatisch vervollständigen lassen (Algorithmus 2, Zeile 13). Häufig ist es bei einer nachträglichen Änderung jedoch erwünscht, dass die bestehenden Planungsentscheidungen, abgesehen von den Modifikationen des Disponenten, weitestgehend unverändert beibehalten werden. Bei einem Backtracking-Vorgang strebt der Disponent daher die geringstmögliche Anpassung des Plans an, die zur Auflösung des Konfliktes erforderlich ist. Eine automatische Neuplanung passt den Plan jedoch u.U. stärker an als erwünscht, sodass sich weitreichende und schwer nachvollziehbare Planänderungen ergeben.

3.4 Rückschlüsse für die Gestaltung der Mensch-Computer-Schnittstelle

Aus der Untersuchung der Algorithmen 2 und 3 können Anforderungen für die Gestaltung der Mensch-Computer-Schnittstelle von Planungssystemen abgeleitet werden:

1. Das Planungssystem sollte den Disponenten während der Planung bei der Auswahl von Werten aus dem Wertebereich einer Entscheidungsvariable unterstützen. Werte, die zur Verletzung einer Randbedingung oder in zukünftigen Zuweisungsschritten in eine Sackgasse führen, sollten entsprechend gekennzeichnet werden.
2. Der Disponent sollte einen gewünschten Wert auch dann auswählen können, wenn er einen Konflikt verursacht, d.h. wenn er im Zusammenhang mit den bisherigen Zuweisungsschritten zur Verletzung einer Randbedingung oder in zukünftigen Zuweisungsschritten in eine Sackgasse führt. Der Backtracking-Prozess zur Beseitigung des Konflikts sollte vom Planungssystem übernommen bzw. unterstützt werden. Dabei sollte eine möglichst geringe Abweichung von den Entscheidungen des Disponenten angestrebt werden.
3. Das Planungssystem sollte die Möglichkeit bereitstellen, Planungsentscheidungen automatisch innerhalb von vom Disponenten vorgegebenen Grenzen so anzupassen, dass Qualität des Plans optimiert wird.

Die Umsetzung dieser Anforderungen hilft dabei, den Arbeitsaufwand der manuellen Planung zu reduzieren und die Qualität der interaktiv erstellten Pläne zu verbessern. Sie trägt damit zu einer erfolgreicheren und zufriedenstellenderen Beteiligung des Disponentem am Lösungsprozess bei, wodurch auch der Effekt der Selbstgefälligkeit reduziert werden kann.

4 Zusammenfassung

In diesem Papier wurde die Beteiligung des Disponenten am Prozess zur Lösung von Planungsproblemen untersucht. Es wurde ein mehrstufiger Entscheidungsvorgang identifiziert, in dem der Disponent Zuweisungen für Startzeiten und Ressourcen einzelner Aufgaben sucht, um bestimmte Anforderungen zu erfüllen, die nicht im Planungsmodell abgebildet sind. Der Vorgang wurde nach dem Vorbild einer Tiefensuche mit Backtracking zur Lösung eines CSPs modelliert. Aus dem Modell wurden verschiedene Interaktionsszenarien abgeleitet und nach dem vom Disponenten benötigten Arbeitsaufwand bewertet. Es wurden Aufwandsfaktoren und Interaktionsdefizite wie z.B. die Notwendigkeit für Backtracking-Vorgänge, die potenzielle Verschlechterung der Qualität und der Effekt der Selbstgefälligkeit identifiziert. Bei der Gestaltung von Planungssystemen müssen Maßnahmen getroffen werden, um diese Defizite so weit wie möglich zu kompensieren.

4.1 Verwandte Arbeiten

Viele Studien stimmen darin überein, dass die Beteiligung des Disponenten an der Planerstellung in der Praxis unerlässlich ist. Es wurden verschiedene Konzepte der Funktionsaufteilung zwischen Mensch und Computer entwickelt: Nach dem Konzept der *gemischten Initiative* müssen beide Parteien als gleichberechtigte Agenten betrachtet werden, die gemeinsam ein bestimmtes Planungsziel erreichen wollen und dazu in jeder Stufe des Lösungsprozesses ihr jeweiliges Wissen auf eigene Initiative einbringen [BBIM96]. Unter dem Begriff der *interaktiven Optimierung* wurden Varianten der Beteiligung des Disponenten konkretisiert. Dazu gehört:

- Die Einbeziehung des Disponenten in die automatische Lösungserstellung, z.B. um in einem lokalen Optimierungsverfahren die Suche nach der optimalen Lösung in eine bestimmte Richtung zu lenken (vgl. [KLMM10]).
- Die Beteiligung des Disponenten an der Modell- und Wissensspezifikation. Verschiedene Möglichkeiten zur Modelländerung, darunter auch die in Abschnitt 2.1 beschriebenen Interaktionsmöglichkeiten, werden von [dNE05] zusammengefasst. Ein Beispiel ist das in [SHC05] vorgestellte System COMIREM (Interaktive Ressourceneinsatzplanung).
- Die teilweise Übernahme der Lösungserstellung durch den Anwender.

Eine Analyse der Nutzeraktionen im Kontext eines vollständigen Interaktionsvorgangs von der Feststellung des Handlungsbedarfs bis zur Fertigstellung des gewünschten Plans findet dabei nicht statt. Mögliche Interaktionsprobleme, die sich aus der Größe von Such- und Lösungsraum praktischer kombinatorischer Planungsprobleme ergeben, werden dadurch nicht identifiziert. In diesem Papier wird eine mögliche Vorgehensweise zur Durchführung einer solchen Analyse gezeigt.

Literatur

[BBIM96] Mark H. Burstein, Bolt Beranek, Newman Inc und Drew V. Mcdermott. Issues in the development of human-computer mixed-initiative planning. In *Cognitive Technology*, Seiten 285–303. Elsevier, 1996.

[BK12] Peter Brucker und Sigrid Knust. Scheduling Models. In *Complex Scheduling*, GOR-Publications, Seiten 1–28. Springer Berlin Heidelberg, 2012.

[BPYS07] Pauline Berry, Bart Peintner und Neil Yorke-Smith. Bringing the User Back into Scheduling: Two Case Studies of Interaction with Intelligent Scheduling Assistants. In *Interaction Challenges for Intelligent Assistants*, Seiten 10–11, 2007.

[Ceg08] Julien Cegarra. A cognitive typology of scheduling situations: A contribution to laboratory and field studies. *Theoretical Issues in Ergonomics Science*, 9(3):201–222, 2008.

[CMWV99] S. Crawford, B. L. MacCarthy, J. R. Wilson und C. Vernon. Investigating the Work of Industrial Schedulers through Field Study. *Cognition, Technology and Work*, 1:63–77, 1999.

[CvW12] Julien Cegarra und Wout van Wezel. Revisiting Decision Support Systems for Cognitive Readiness: A Contribution to Unstructured and Complex Scheduling Situations. *Journal of Cognitive Engineering and Decision Making*, 6(3):299–324, 2012.

[Dec03] R. Dechter. *Constraint Processing*. The Morgan Kaufmann Series in Artificial Intelligence. Elsevier Science, 2003.

[dNE05] Hugo A.D. do Nascimento und Peter Eades. User hints: a framework for interactive optimization. *Future Generation Computer Systems*, 21(7):1177–1191, 2005.

[dSvWJ11] Cees de Snoo, Wout van Wezel und René J. Jorna. An empirical investigation of scheduling performance criteria. *Journal of Operations Management*, 29(3):181–193, 2011.

[HW07] P. Hofstedt und A. Wolf. *Einführung in die Constraint-Programmierung: Grundlagen, Methoden, Sprachen, Anwendungen*. Springer, 2007.

[KLMM10] Gunnar W. Klau, Neal Lesh, Joe Marks und Michael Mitzenmacher. Human-guided search. *Journal of Heuristics*, 16(3):289–310, 6 2010.

[KLMS05] Brian Kallehauge, Jesper Larsen, OliB.G. Madsen und MariusM. Solomon. Vehicle Routing Problem with Time Windows. In Guy Desaulniers, Jacques Desrosiers und MariusM. Solomon, Hrsg., *Column Generation*, Seiten 67–98. Springer US, 2005.

[Mac08] M. MacDonald. *Your Brain: The Missing Manual*. Missing manual. O'Reilly Media, 2008.

[SHC05] Stephen F. Smith, David W. Hildum und David R. Crimm. Comirem: An Intelligent Form for Resource Management. *IEEE Intelligent Systems*, 20, 2005.

[TSD11] Efraim Turban, Ramesh Sharda und Dursun Delen. *Decision Support and Business Intelligence Systems*. Pearson, 2011.