Studienarbeit

Zielordnungen und Landmarken für SAS$^+$-Planer

Matthias Westphal
17. Juli 2007

Betreut durch Dr. Malte Helmert

Inhaltsverzeichnis

1 Einleitung **3**

2 Allgemeine Definitionen und Notation **4**
2.1 SAS$^+$ 4
2.2 Allgemeines 5

3 Definitionen von Landmarken und Ordnungen für SAS$^+$ **5**
3.1 Landmarken 5
3.2 Befolgen von Ordnungen 6
3.3 Lookahead-notwendige Ordnungen 7
3.4 Notwendige Ordnungen 8
3.5 Greedy-notwendige Ordnungen 9
3.6 Vernünftige Ordnungen 9
3.7 Folgende vernünftige Ordnungen 12
3.8 Landmarkengraph 12
3.9 Komplexität 14

4 Bisherige Approximierung von Landmarken und zwingenden Ordnungen **14**
4.1 Verfahren zur Approximierung von Landmarken und notwendigen Ordnungen 14
4.1.1 Pseudocode für die Approximierung von Landmarken und notwendigen Ordnungen 15
4.2 Verfahren zur Approximierung von Landmarken und greedy-notwendigen Ordnungen 15
4.2.1 Generierung von Kandidaten für Landmarken und greedy-notwendige Ordnungen 16
4.2.2 Pseudocode für die Approximierung von Kandidaten für Landmarken und greedy-notwendige Ordnungen . 17
4.2.3 Verifikation von Landmarken 18
4.3 Lookahead-notwendige Ordnungen 19

5 Verbesserte Approximierung von Landmarken und zwingenden Ordnungen für SAS$^+$ **19**
5.1 Approximierung von Landmarken und zwingenden Ordnungen 20
5.2 Landmarken, Ordnungen und der Domänentransitionsgraph . 21
5.2.1 Greedy-notwendige Ordnungen 21
5.2.2 Lookahead-notwendige Ordnungen 23
5.3 Erweiterung auf den propositionalen Teil von PDDL2.2 . . . 24
5.4 Pseudocode für die Approximierung von Landmarken und Ordnungen 25
5.4.1 Laufzeit 26

5.5 Vergleich mit dem alten Verfahren 26
5.5.1 Evaluierung der Approximierungsverfahren für Landmarken und zwingende Ordnungen 27

6 Approximierung von vernünftigen Ordnungen 28
6.1 Definition Inkonsistenz 29
6.1.1 Approximierung von Inkonsistenzen 29
6.2 Definition Interferenz 29
6.3 Approximierung der *Aftermath*-Relation 30
6.4 Approximierung von vernünftigen Ordnungen 31
6.4.1 Pseudocode für die Approximierung von Vernünftige Ordnungen 31
6.5 Approximierung von folgenden vernünftige Ordnungen 31
6.5.1 Zyklen im Landmarkengraphen 32

7 Nutzung von Landmarken bei der Planung 32
7.1 Das Downward-Planungssystem 32
7.2 Bisherige Verwendung 33
7.3 Landmarken-Heuristik 34
7.3.1 Preferred Operators 36
7.4 Bestensuche einschränken 36
7.5 Landmarken und Ordnungen explizit in Task einbringen . . . 37

8 Testergebnisse mit Landmarken 39
8.1 Übersicht 39
8.2 Plots zu ausgewählten Domänen 40
8.3 Fazit 42

1 Einleitung

In vielen Planungsproblemen gibt es eine sinnvolle zeitliche Ordnung für das Erreichen seiner Ziele. Beispielsweise ist es in einer Blockswelt meistens das Ziel einen Turm aus Blöcken zu konstruieren. Dabei gibt das Ziel genau für jeden Block des Turmes vor, auf welchem anderen Block er stehen soll. Nun ist es vernünftig diesen Turm gleich von Beginn an von unten her zu konstruieren, da eine andere Vorgehensweise nicht zum Erreichen aller Ziele führen kann, ohne den Teil-Turm wieder auseinander zunehmen und später wieder neu stapeln zu müssen. Es gibt also eine vernünftige Ordnung, die vorgibt, dass die Blöcke sukzessive von unten her gestapelt werden sollten.

Solche Ordnungen können nicht nur für die eigentlichen Ziele definiert werden, sondern es ist möglich, diese auf sogenannte Landmarken auszudehnen. Eine Landmarke ist dabei die Belegung einer Variable, welche nicht notwendigerweise ein Ziel sein muss, aber dennoch in jeder Lösung vorkommt. Landmarken sind damit sowohl die eigentlichen Ziele, als auch nicht explizit gekennzeichnete Ziele, die zwingend in jeder Lösung zu einem Zeitpunkt erreicht werden. Solche Landmarken finden sich zum Beispiel in Logistikproblemen, bei denen Pakete erst verladen werden müssen, um dann zum Ziel transportiert zu werden. Hierbei ist das Ziel meist, die Pakete an ihre Bestimmungsorte zu bringen, nicht aber das eigentliche Verladen. Bei vielen Problemen, lassen sich auch diese Landmarken zeitlich ordnen. In dem angesprochenen Logistikproblem, gibt es eine vernünftige zeitliche Ordnung für das Verladen, Transportieren und Entladen der Pakete (siehe Abb.1).

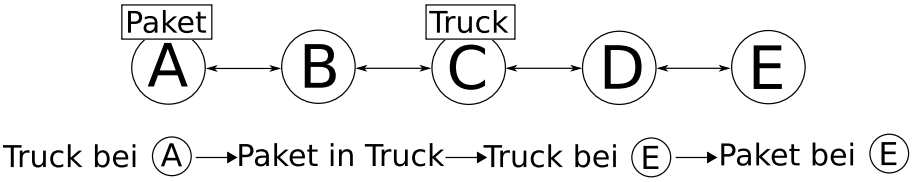


Abbildung 1: Ein Logistikproblem mit dazugehörigen Landmarken und Ordnungen. Ziel ist es, das Paket mit Hilfe des Trucks von A nach E zu befördern.

Sind solche Landmarken und Ordnungen gefunden worden, stellt sich die Frage, wie diese während der Planung ausgenutzt werden können. Von einer Verwendung erhofft man sich meist einen Geschwindigkeitsgewinn bei der Planung oder kürzere Pläne. Sowohl Möglichkeiten zum Auffinden von Landmarken, als auch ihre Verwendungen sind in der Literatur untersucht worden. Im Besonderen soll sich die hier vorliegende Studienarbeit auf die Arbeit von Hoffmann, Porteous und Sebastia, „Ordered Landmarks in Planning" [HPS04] stützen. Diese Arbeit wiederum ist eine Weiterentwicklung des älteren Artikels „On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm" von Hoffmann und Koehler [HK00].

Dabei werden in der neueren Arbeit [HPS04] die Verfahren des Vorgänger-Artikels in Bezug auf das Auffinden von Landmarken sowie die Verwendung der Landmarken und Ordnungen während der Planung verbessert.

Im wesentlichen gliedert sich diese Studienarbeit in zwei Teile. Der erste Teil widmet sich der Verbesserung und Portierung der bestehenden Verfahren aus dem Artikel „Ordered Landmarks in Planning" [HPS04] für die Approximierung von Landmarken und Ordnungen von STRIPS-Problemen auf SAS$^+$-Probleme. Im zweiten Teil werden mehrere Möglichkeiten der Verwendung von Landmarken und Ordnungen bei der Planung vorgestellt und auf Teilen der ICAPS-Planungsbenchmarks – die nach SAS$^+$ übersetzt werden – evaluiert. Implementiert und verwendet werden diese innerhalb des Downward-Planungssystems [Hel06], welches auch die Übersetzung der STRIPS-Probleme nach SAS$^+$ übernimmt.

2 Allgemeine Definitionen und Notation

2.1 SAS$^+$

Ein SAS$^+$-Planungsproblem ist ein Tupel $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$. Dabei bezeichnet $\mathcal{V}$ eine Menge von Zustandsvariablen mit je endlicher Domäne $\mathcal{D}_v$, $\mathcal{O}$ eine Menge von Operatoren, s_0 einen Initialzustand und $s_\star$ einen Teil eines Zielzustandes – eine partielle Belegung. Eine partielle Belegung s ist eine Funktion auf einer Teilmenge $\mathcal{W} \subseteq \mathcal{V}$ mit $s(v) \in \mathcal{D}_v$ für $v \in \mathcal{W}$. Gilt $\mathcal{W} = \mathcal{V}$, so ist s eine vollständige Belegung und wird auch als Zustand bezeichnet. Somit ist $s_\star$ eine partielle und s_0 eine vollständige Belegung. Ein Operator $o \in \mathcal{O}$ besitzt die Form $o = \langle pre_o, \mathit{eff}_o \rangle$, wobei pre_o, eff_o partielle Belegungen sind. Dabei sind pre_o die Vorbedingungen und eff_o die Effekte des Operators. Der Operator $o \in \mathcal{O}$ ist genau dann anwendbar, wenn s ein Zustand ist und $s \supseteq pre_o$ gilt. Ein Zustand s' ist genau dann der Nachfolgezustand von o angewendet auf s, wenn o anwendbar in s ist und $s' \supseteq \mathit{eff}_o$, sowie $s' \backslash \mathit{eff}_o \subseteq s$.

Der *Kausalgraph* eines SAS$^+$-Planungsproblems Π wird mit $CG(\Pi)$ bezeichnet. Dieser ist ein gerichteter Graph $(\mathcal{V}, E)$, bei dem für $u, v \in \mathcal{V}, u \neq v$ eine Kante (u, v), genau dann in E liegt, wenn ein Operator $o \in \mathcal{O}$ existiert, für den $\mathit{eff}_o(v)$ definiert und weiter $pre_o(u)$ oder $\mathit{eff}_o(u)$ definiert ist. Somit zeigt der Kausalgraph die Zusammenhänge zwischen den Variablen.

Der *Domänentransitionsgraph* einer Variable $v \in \mathcal{V}$, G_v ist ein gerichteter Graph $(\mathcal{D}_v, E)$. Die Kante (d, d') mit $d, d' \in \mathcal{D}_v$ ist genau dann in E, wenn ein Operator $o \in \mathcal{O}$ existiert mit $\mathit{eff}_o(v) = d'$ und weiter $pre_o(v) = d$ oder $pre_o(v)$ undefiniert. Weiter kann diese Kante noch mit den Vorbedingungen pre_o annotiert werden, wobei eine Belegung von v in pre_o aus den Annotation meist weggelassen wird, da diese bereits im Graph kenntlich gemacht ist. Aus dem Domänentransitionsgraph geht hervor, wie sich die Werte einer Variable verändern können.

2.2 Allgemeines

Im Kontext eines Planungsproblems $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ wird eine beliebige Sequenz von Aktionen durch $\langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*, n \in \mathbb{N}_0$ gekennzeichnet. Die Ausführung einer solchen Aktionssequenz von einem Zustand s aus wird wie folgt rekursiv definiert:
Sei $o \in \mathcal{O}$, s ein Zustand über $\mathcal{V}$.

$$
\begin{array}{lrcl}
n = 0: & \mathit{Result}(s, \langle\rangle) & := & s \\
n = 1: & \mathit{Result}(s, \langle o \rangle) & := & \begin{cases} \emptyset & o \text{ nicht anwendbar in } s \\ s' & s' = o \text{ angewendet auf } s \end{cases} \\
n > 1: & \mathit{Result}(s, \langle o_1, \ldots, o_n \rangle) & := & \mathit{Result}(\mathit{Result}(s, \langle o_1 \rangle), \langle o_2, \ldots, o_n \rangle)
\end{array}
$$

Sei im Folgenden stets $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ ein SAS$^+$-Planungsproblem und $\{v \mapsto d\}, \{v' \mapsto d'\}$ zwei Zuweisungen, wobei $v, v' \in \mathcal{V}, d \in \mathcal{D}_v, d' \in \mathcal{D}_{v'}$.

3 Definitionen von Landmarken und Ordnungen für SAS$^+$

Analog zu den Definitionen in „Ordered Landmarks in Planning" [HPS04] können Landmarken und Ordnungen für SAS$^+$ definiert werden.

3.1 Landmarken

Eine Zuweisung $\{v \mapsto d\}$ ist eine Landmarke genau dann wenn für alle Pläne $\langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*, n \in \mathbb{N}_0$, $s_\star \subseteq \mathit{Result}(s_0, \langle o_1, \ldots, o_n \rangle)$ gilt:

$$\{v \mapsto d\} \subseteq \mathit{Result}(s_0, \langle o_1, \ldots, o_i \rangle), \text{ für ein } i \in \{0, \ldots, n\}$$

Landmarken sind somit genau die Zuweisungen, die während der Ausführung einer Lösung immer in mindestens einem Zustand auftreten. Durch diese Definition folgt sofort, dass alle Zuweisungen die in s_0 sowie auch alle Zuweisungen die in $s_\star$ enthalten sind, Landmarken darstellen. Dies entspricht genau dem Gedanken, dass Landmarken eine Erweiterung zu den explizit gegebenen Zielen darstellen sollen. In dem Beispiel von Abb. 1 ist dies bei geeigneter Formalisierung[1] neben dem eigentlichen Ziel $\{\mathit{Paket} \mapsto E\}$ auch $\{\mathit{Truck} \mapsto E\}$, $\{\mathit{Truck} \mapsto A\}$ sowie $\{\mathit{Paket} \mapsto \mathit{Truck}\}$. Die Landmarken im Initialzustand $\{\mathit{Paket} \mapsto A\}$, $\{\mathit{Truck} \mapsto C\}$ können noch hinzugenommen werden, genauso wie $\{\mathit{Truck} \mapsto B\}$, $\{\mathit{Truck} \mapsto D\}$, da diese Knoten durchfahren werden müssen. Keine Landmarken stellen $\{\mathit{Paket} \mapsto B\}$, …, $\{\mathit{Paket} \mapsto D\}$ dar.

Hierbei zeigt sich auch gleich ein Problem der Landmarken-Definition: Gibt es mehr als einen Truck, der in der Lage ist, das Paket von A nach E zu

[1] Zwei Variablen *Truck*, *Paket* und $\mathcal{D}_{\mathit{Truck}} = \{A, \ldots, E\}, \mathcal{D}_{\mathit{Paket}} = \{A, \ldots, E, \mathit{Truck}\}$.

befördern, so gehen die meisten Landmarken verloren. In der Tat wäre neben den Landmarken im Initialzustand die einzig weitere nur noch $\{Paket \mapsto E\}$, da ein beliebiger Truck zur Beförderung verwendet werden könnte. Dieses Problem tritt generell immer dann auf, wenn es mehrere Möglichkeiten gibt, über die eine Landmarke erreicht werden kann. Eine Grundidee, ein solches Problemen zu lösen, wäre die Verwendung von disjunktiven Landmarken, die Mengen von Landmarken darstellen und somit solche Wahlmöglichkeiten abdecken könnten. Dies wurde in „Ordered Landmarks in Planning" [HPS04] nicht weiter betrachtet und soll auch hier nicht benutzt werden, da damit neue Probleme entstehen.

Nach Definition ist in jedem unlösbaren Problem jede gültige Zuweisung eine Landmarke. Dies soll nicht weiter stören, da in erster Linie lösbare Probleme betrachtet werden. Es werden später bei der Vorstellung konkreter Algorithmen nur kurz Kriterien genannt, mit denen die Unlösbarkeit gefolgert werden könnte. Diese sind allerdings einfacher Natur und sollen auch nicht das Ziel dieser Arbeit sein.

Nachdem nun Landmarken definiert wurden, kann man sich der Frage zuwenden, wie Landmarken untereinander geordnet werden können. Die Definitionen hierfür werden allgemein für Zuweisungen erfolgen, obwohl in erster Linie Ordnungen zwischen Landmarken interessant sind. $\{v \mapsto d\} \rightarrow \{v' \mapsto d'\}$ steht dabei für $\{v \mapsto d\}$ zeitlich vor $\{v' \mapsto d'\}$. Ohne auf eine genauere Bedeutung der Ordnung einzugehen, zuerst eine kurze Definition, was das Einhalten von Ordnungen bedeuten soll.

3.2 Befolgen von Ordnungen

Ein Aktionssequenz $\langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*, n \in \mathbb{N}_0$ *befolgt* eine Ordnung zwischen zwei Zuweisungen, $\{v \mapsto d\} \rightarrow \{v' \mapsto d'\}$ gdw.

$$\{v \mapsto d\} \subseteq s_0 \text{ oder } \exists 1 \leq i \leq n : \{v \mapsto d\} \subseteq \mathit{eff}_{o_i}, \{v' \mapsto d'\} \not\subseteq \mathit{eff}_{o_k} \forall 0 \leq k \leq i$$

Um die Ordnung zu befolgen, muss also $\{v \mapsto d\}$ im Initialzustand liegen[2] oder $\{v \mapsto d\}$ zeitlich vor $\{v' \mapsto d'\}$ erreicht werden. Wobei nicht gefordert wird, dass $\{v' \mapsto d'\}$ überhaupt vorkommt. Dies ist sinnvoll, da hier Aktionssequenzen und Zuweisungen, und nicht zwingend Pläne und Landmarken betrachtet werden.

Für die Ordnung $\{v \mapsto d\} \rightarrow \{v' \mapsto d'\}$ selbst können zwei Grundarten unterschieden werden:

- Zwingende Ordnungen
  In allen Aktionssequenzen, die von s_0 aus starten, wird $\{v \mapsto d\}$ immer vor $\{v' \mapsto d'\}$ erreicht. Diese Ordnungen sind zwingend, in dem Sinne,

[2] Womit nichts für $\{v' \mapsto d'\}$ gefordert wird. Demnach wird die Ordnung auch für $\{v \mapsto d, v' \mapsto d'\} \subseteq s_0$ befolgt. Dies wurde so aus der ursprünglichen Definition in „Ordered Landmarks in Planning" [HPS04] übernommen.

dass jede Aktionssequenz, die eine von beiden Zuweisungen in einem Zustand erreicht, diese Ordnung befolgt.

- Vernünftige Ordnungen
  Ordnungen, deren Einhaltung vernünftig erscheint, da dies womöglich zusätzliche Arbeit vermeidet. Hierbei gilt im Gegensatz zu den zwingenden Ordnungen nicht, dass $\{v \mapsto d\}$ immer vor $\{v' \mapsto d'\}$ erreicht wird. Es können also sehr wohl Aktionssequenzen, sogar Lösungen gefunden werden, die diese Ordnung nicht befolgen. Diese Art von Ordnungen wurde erstmals in „On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm" [HK00] vorgestellt.

Diese Grundarten sollen nun präzisiert werden.

3.3 Lookahead-notwendige Ordnungen

Zwei Zuweisungen $\{v \mapsto d\}, \{v' \mapsto d'\}$ sind genau dann lookahead-notwendig geordnet, wenn in jeder Aktionssequenz von s_0 aus $\{v \mapsto d\}$ immer zeitlich vor Erreichen von $\{v' \mapsto d'\}$ auftreten muss. Es handelt sich also um eine zwingende Ordnung. Formal:
Eine Lookahead-notwendige Ordnung $\{v \mapsto d\} \rightarrow_{ln} \{v' \mapsto d'\}$ gilt gdw.
$\{v' \mapsto d'\} \not\subseteq s_0$ und alle Aktionssequenzen $\langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*, n \in \mathbb{N}$ erfüllen:

$$\{v' \mapsto d'\} \subseteq \mathit{Result}(s_0, \langle o_1, \ldots, o_n \rangle) \Rightarrow \{v \mapsto d\} \subseteq \mathit{Result}(s_0, \langle o_1, \ldots, o_i \rangle) \quad \text{für ein } i \in \{0, \ldots, n-1\}$$

Dies ist die schwächste zwingende Ordnung, die definiert wird. Trivialerweise können alle Zuweisungen aus s_0 lookahead-notwendig vor beliebigen anderen Zuweisungen, die nicht in s_0 liegen, geordnet werden. Ein sinnvolleres Beispiel bietet das Logistikproblem aus Abb. 1. Hier sind nicht alle angegebenen Ordnungen lookahead-notwendig. Es gilt: $\{\mathit{Truck} \mapsto A\} \rightarrow_{ln} \{\mathit{Paket} \mapsto \mathit{Truck}\}$, $\{\mathit{Paket} \mapsto \mathit{Truck}\} \rightarrow_{ln} \{\mathit{Paket} \mapsto E\}$ und $\{\mathit{Truck} \mapsto E\} \rightarrow_{ln} \{\mathit{Paket} \mapsto E\}$. Falsch ist jedoch $\{\mathit{Paket} \mapsto \mathit{Truck}\} \rightarrow_{ln} \{\mathit{Truck} \mapsto E\}$, da der Truck nach E fahren kann, bevor das Paket geladen wurde. Diese Ordnung ist nicht zwingend, jedoch wäre sie vernünftig, wie bei einer späteren Definition deutlich werden wird. Ähnlich verhält es sich mit Ordnungen in der Blockswelt. Die Zielordnungen sind ebenfalls nicht zwingend, da die oberen Teile der Türme auch schon vorher gebaut werden können, obwohl dies nicht sinnvoll ist. Zwingende Ordnungen wären hier, dass Blöcke frei liegen müssen, bevor ein anderer Block darauf gestapelt werden kann.

In dem Logistikbeispiel traten die zwei Ordnungen: $\{\mathit{Truck} \mapsto E\} \rightarrow_{ln} \{\mathit{Paket} \mapsto E\}$ und $\{\mathit{Paket} \mapsto \mathit{Truck}\} \rightarrow_{ln} \{\mathit{Paket} \mapsto E\}$ auf. Schön wäre es nun wenn daraus gefolgert werden könnte, dass vor dem Erreichen des Zieles der Truck sowohl bei E stehen, als auch das Paket bereits geladen sein muss. Dies entspricht zwar den Operatorvorbedingungen des Abladens

an der Stelle E, jedoch kann dies nicht allgemein alleine aus den beiden Ordnungen gefolgert werden. Es ist schon nicht möglich, etwas über die zeitliche Ordnung von $\{Truck \mapsto E\}$ und $\{Paket \mapsto Truck\}$ zu folgern. Mit Hilfe einer strikteren Definition einer zwingenden Ordnung lässt sich dies verbessern.

3.4 Notwendige Ordnungen

Eine notwendige Ordnung $\{v \mapsto d\} \rightarrow_n \{v' \mapsto d'\}$ gilt gdw. $\{v' \mapsto d'\} \not\subseteq s_0$ und alle Folgen von Aktionen $\langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*, n \in \mathbb{N}$ erfüllen:

$$\{v' \mapsto d'\} \subseteq Result(s_0, \langle o_1, \ldots, o_n \rangle) \Rightarrow \{v \mapsto d\} \subseteq Result(s_0, \langle o_1, \ldots, o_{n-1} \rangle)$$

Dies bedeutet also, dass $\{v \mapsto d\}$ immer in einem Zustand direkt vor $\{v' \mapsto d'\}$ gelten muss. Tatsächlich gilt für die beiden lookahead-notwendigen Ordnungen aus dem Logistikproblem nun: $\{Truck \mapsto E\} \rightarrow_n \{Paket \mapsto E\}$, $\{Paket \mapsto Truck\} \rightarrow_n \{Paket \mapsto E\}$ impliziert $\{Truck \mapsto E, Paket \mapsto E\}$ gilt in einem Zustand in jeder Lösung.

Es ergibt sich allerdings ein Problem mit dieser Definition, denn für viele Zuweisungen gibt es keine weiteren Zuweisungen die man direkt notwendig davor ordnen könnte. $\{v \mapsto d\} \rightarrow_n \{v' \mapsto d'\}$ bedeuted eben nach Definition, dass immer vor dem Erreichen von $\{v' \mapsto d'\}$, $\{v \mapsto d\}$ gelten soll. Wenn es nun mehrere Möglichkeiten gibt, diese Zuweisung zu erreichen, sind diese meist so unterschiedlich, dass vorgehende Zustände keine Gemeinsamkeiten aufweisen müssen. An dieser Stelle ist es hilfreich den Domänentransitionsgraphen der Variable *Paket* zu betrachten (Abb. 2).

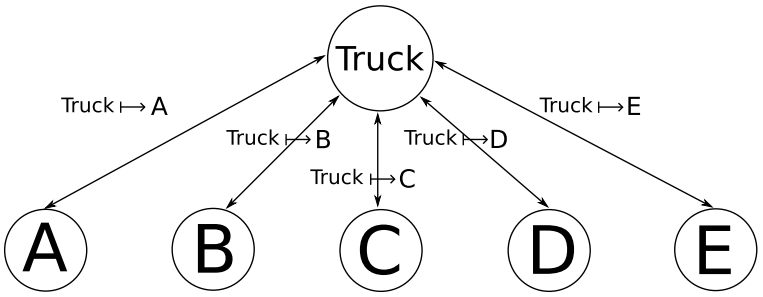


Abbildung 2: Domänentransitionsgraph G_{Paket}.

Zu sehen ist hier die Notwendigkeit von $\{Truck \mapsto E\}$ und $\{Paket \mapsto Truck\}$ direkt vor $\{Paket \mapsto E\}$, da es sich um den einzigen Übergang zu $\{Paket \mapsto E\}$ handelt. Für $\{Paket \mapsto Truck\}$ ist jedoch zu erkennen, dass viele mögliche Zuweisungen vor $\{Paket \mapsto Truck\}$ gelten können. Es ist möglich das Paket von einem beliebigen Punkt A-E aufzuheben, und jeder dieser Punkte stellt eine andere Bedingung an die *Truck*-Variable. Somit

existiert in diesem Beispiel keine Zuweisungen für die *Truck*- oder *Paket*-Variable, die notwendig vor $\{Paket \mapsto Truck\}$ geordnet werden könnte.

Um dieses Problem zu lösen, soll die Definition nun abgeschwächt werden. Interessieren soll nur noch das **erstmalige** Erreichen einer Zuweisung. Damit bleibt die Ordnung zwingend und mit dieser Einschränkung wird klar, dass *B-E* für das Einladen – also das Erreichen von $\{Paket \mapsto Truck\}$ – ignoriert werden können, da sich das Paket zu Beginn bei A befindet. Für Landmarken ist diese Einschränkung in den meisten Fällen auch nicht schlimm, weil nur davon ausgegangen werden kann, diese einmal zu erreichen.

3.5 Greedy-notwendige Ordnungen

Eine Greedy-notwendige Ordnung $\{v \mapsto d\} \rightarrow_{gn} \{v' \mapsto d'\}$ gilt gdw. alle beliebigen ausführbaren Folgen von Aktionen $\langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*, n \in \mathbb{N}$ erfüllen:

$$\begin{aligned}
&\{v' \mapsto d'\} \subseteq \mathit{Result}(s_0, \langle o_1, \ldots, o_n \rangle) \text{ und} \\
&\{v' \mapsto d'\} \not\subseteq \mathit{Result}(s_0, \langle o_1, \ldots, o_i \rangle) \quad \forall\ 0 \leq i < n \\
&\Rightarrow \{v \mapsto d\} \subseteq \mathit{Result}(s_0, \langle o_1, \ldots, o_{n-1} \rangle)
\end{aligned}$$

Nun wird nur noch das erste Erreichen von $\{v' \mapsto d'\}$ betrachtet, womit folgt: $\{Paket \mapsto A\} \rightarrow_{gn} \{Paket \mapsto Truck\}$, $\{Truck \mapsto A\} \rightarrow_{gn} \{Paket \mapsto Truck\}$. Trotz dieser Einschränkung gilt, dass aus drei Zuweisungen Z_1, Z_2, Z_3 mit $Z_1 \rightarrow_{gn} Z_3$, $Z_2 \rightarrow_{gn} Z_3$ folgt, Z_1, Z_2 gelten im Zustand direkt vor dem **erstmaligen** Erreichen von Z_3 gleichzeitig.

Dies waren alle zwingenden Ordnungen, die betrachtet werden. Nach ihren Definitionen gilt außerdem:

$$\{v \mapsto d\} \rightarrow_{n} \{v' \mapsto d'\} \Rightarrow \{v \mapsto d\} \rightarrow_{gn} \{v' \mapsto d'\}$$

$$\{v \mapsto d\} \rightarrow_{gn} \{v' \mapsto d'\} \Rightarrow \{v \mapsto d\} \rightarrow_{ln} \{v' \mapsto d'\}$$

$$\{v \mapsto d\} \rightarrow_{gn} \cdots \rightarrow_{gn} \{v' \mapsto d'\} \Rightarrow \{v \mapsto d\} \rightarrow_{ln} \{v' \mapsto d'\}$$

Diese zwingenden Ordnungen werden in jeder Lösung eingehalten. Um aus Ordnungen einen Vorteil für die Planlängen zu ziehen, sollten auch nicht zwingende Ordnungen eingeführt werden. Insbesondere ist man daran interessiert für das Logistikbeispiel $\{Paket \mapsto Truck\} \rightarrow \{Truck \mapsto E\}$ zu bekommen.

3.6 Vernünftige Ordnungen

Um eine vernünftige Ordnung zu erhalten ist die zugrunde liegende Idee, Zuweisungen Z_1, Z_2 zu identifizieren, bei denen das Erreichen von Z_2 vor Z_1

dazu führt, dass Z_2 bei dem Erreichen von Z_1 verloren geht und dann neu erreicht werden muss. Für das Logistikbeispiel wäre dies gerade, $\{\mathit{Truck} \mapsto E\}$ zu erreichen ohne das Paket je geladen zu haben. Um das Paket noch zu laden, ist es nötig nach A zu fahren, womit $\{\mathit{Truck} \mapsto E\}$ verloren geht. Ist das Paket dann geladen, muss $\{\mathit{Truck} \mapsto E\}$ wieder neu erreicht werden um das Ziel $\{\mathit{Paket} \mapsto E\}$ zu erhalten. Ein weiteres Beispiele hierfür sind Türme in der Blockswelt. Es ist sinnvoll, diese von unten her zu konstruieren, allerdings nicht zwingend. Türme, die nicht von unten her konstruiert werden, müssen dann eben wieder auseinander genommen und später, wenn der Grundbau des Turmes steht, wieder gestapelt werden. Dieser Mehraufwand soll durch vernünftige Ordnungen kenntlich gemacht werden.

Um dies für $\{v \mapsto d\}, \{v' \mapsto d'\}$ zu formalisieren, betrachtet man zuerst alle Zustände, die von s_0 aus erreichbar sind, bei denen gerade $\{v' \mapsto d'\}$ im Zustand erreicht wurde, aber $\{v \mapsto d\}$ bisher nicht auftrat. Die Menge dieser Zustände wird mit $S_{(\{v' \mapsto d'\}, \neg\{v \mapsto d\})}$ gekennzeichnet. Daraufhin definiert man, was es bedeutet, dass $\{v' \mapsto d'\}$ nach oder bei Erreichen von $\{v \mapsto d\}$ noch benötigt wird. Dies bildet die *Aftermath*-Relation. Gilt nun, dass $\{v' \mapsto d'\}$ in der *Aftermath*-Relation von $\{v \mapsto d\}$ liegt und es bei allen Zuständen $s \in S_{(\{v' \mapsto d'\}, \neg\{v \mapsto d\})}$ nötig ist, für das Erreichen von $\{v \mapsto d\}$, $\{v' \mapsto d'\}$ zu zerstören, so ist es vernünftig, $\{v \mapsto d\}$ vor $\{v' \mapsto d'\}$ zu erreichen.
Formal zusammengefasst ergibt dies:

1. $s \in S_{(\{v' \mapsto d'\}, \neg\{v \mapsto d\})}$ gdw. $\exists P = \langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*, n \in \mathbb{N}$:
   $\mathit{Result}(s_0, P) = s, \{v' \mapsto d'\} \subseteq \mathit{eff}_{o_n}$ und
   $\{v \mapsto d\} \not\subseteq \mathit{Result}(s_0, \langle o_1, \ldots, o_i \rangle) \quad \forall\ 0 \leq i \leq n$

2. $\{v' \mapsto d'\} \in \mathit{Aftermath}(\{v \mapsto d\})$ gdw. $\forall s \in S_{(\{v' \mapsto d'\}, \neg\{v \mapsto d\})}$ und für alle Pläne $P = \langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*, n \in \mathbb{N}$ von s aus gilt:
   $s_\star \subseteq \mathit{Result}(s, P)$ und $\exists i, j : 1 \leq i \leq j \leq n$:
   $\{v \mapsto d\} \subseteq \mathit{Result}(s, \langle o_1, \ldots, o_i \rangle), \{v' \mapsto d'\} \subseteq \mathit{Result}(s, \langle o_1, \ldots, o_j \rangle)$

3. $\{v \mapsto d\} \rightarrow_r \{v' \mapsto d'\}$ gdw. $\{v' \mapsto d'\} \in \mathit{Aftermath}(\{v \mapsto d\})$ und
   $\forall s \in S_{(\{v' \mapsto d'\}, \neg\{v \mapsto d\})} \forall P \in \mathcal{O}^*, n \in \mathbb{N}$:
   $\{v \mapsto d\} \subseteq \mathit{Result}(s, P) \Rightarrow \exists o \in P : \{v' \mapsto \widetilde{d}\} \subseteq \mathit{eff}_o, \widetilde{d} \in \mathcal{D}_{v'} \backslash \{d'\}$

Mit diesen Definitionen erhält man $\{\mathit{Paket} \mapsto \mathit{Truck}\} \rightarrow_r \{\mathit{Truck} \mapsto E\}$ für das Logistikbeispiel.

Obwohl die vernünftigen Ordnungen sinnvoll sind, entstehen doch einige Probleme. So drückt eine vernünftige Ordnung $\{v \mapsto d\} \rightarrow_r \{v' \mapsto d'\}$ **nicht** aus, dass es gut ist, $\{v \mapsto d\}$ vor $\{v' \mapsto d'\}$ zu erreichen. Es wird eher gesagt, dass $\{v' \mapsto d'\}$ vor $\{v \mapsto d\}$ mit Aufwand verbunden ist. Ob sich dieser Aufwand dann durch die zeitliche Abfolge $\{v \mapsto d\} \rightarrow_r \{v' \mapsto d'\}$ beseitigen lässt, ist **nicht** garantiert. Man betrachte ein Beispiel aus

„Ordered Landmarks in Planning" [HPS04]:

$$\mathcal{V} = \{S, L, L', \}, \mathcal{D}_S = \{0, \ldots, 4\}, \mathcal{D}_L = \mathcal{D}_{L'} = \{T, F\}$$

$$s_0 = \{S \mapsto 0, L \mapsto F, L' \mapsto F\}, s_\star = \{L \mapsto T, L' \mapsto T\}$$

Operator	*pre*	*eff*
o_{L_1}	$\{S \mapsto 0\}$	$\{L \mapsto T, S \mapsto 1\}$
$o_{L'_1}$	$\{S \mapsto 0\}$	$\{L' \mapsto T, S \mapsto 2\}$
$o_{L'_2}$	$\{S \mapsto 1\}$	$\{L' \mapsto T, L \mapsto F, S \mapsto 3\}$
o_{L_2}	$\{S \mapsto 2\}$	$\{L \mapsto T, L' \mapsto F, S \mapsto 4\}$
o_{L_3}	$\{S \mapsto 3\}$	$\{L \mapsto T\}$
$o_{L'_3}$	$\{S \mapsto 4\}$	$\{L' \mapsto T\}$

Hier gilt: $\{L \mapsto T\} \rightarrow_r \{L' \mapsto T\}$ und $\{L' \mapsto T\} \rightarrow_r \{L \mapsto T\}$. Die Variable S schränkt die Auswahl an möglichen Operatoren sehr genau ein. So ist es nur am Anfang möglich sich zwischen o_{L_1} und $o_{L'_1}$ zu entscheiden. Von dort an, ist es in beiden Fällen nötig die Zuweisung $\{L \mapsto T\}$, bzw. $\{L' \mapsto T\}$ wieder zu verlieren $(o_{L'_2}, o_{L_2})$, um die jeweils andere Zuweisung zu erreichen. Nach den Definition gelten damit beide vernünftigen Ordnungen. Würde nun versucht werden beide Ordnungen während der Planung zwangsweise einzuhalten, so wäre das Problem unlösbar. Diese Problematik wird später noch betrachtet werden.

In den Definitionen zu den vernünftigen Ordnungen werden beliebige Aktionssequenzen zum Erreichen von Zuständen in $S_{(\{v' \mapsto d'\}, \neg\{v \mapsto d\})}$ und bei der *Aftermath*-Relation betrachtet. Dies bedeutet, dass diese Aktionssequenzen lediglich zwingende Ordnungen einhalten. Nachdem nun vernünftige Ordnungen bekannt sind, kann man sich die Frage stellen, ob es nicht weitere vernünftige Ordnungen geben kann, wenn man davon ausgeht, dass die bekannten vernünftigen Ordnungen eingehalten werden. Solche Fälle treten tatsächlich auf. Durch das Befolgen von vernünftigen Ordnungen wird die Abfolge des Erreichens von Zuweisungen stärker festgelegt, womit Zuweisungen nun eventuell erstmals in *Aftermath*-Relationen von anderen Zuweisungen liegen. Die Arbeit „Ordered Landmarks in Planning" [HPS04] listet dafür folgendes Beispiel für Zuweisungen Z_1, Z_2, Z_3 mit $Z_1 \rightarrow_r Z_3$, $Z_2 \rightarrow_n Z_3$ und Z_2 geht verloren, falls Z_1 erreicht wird. Es könnte der Fall eintreten, dass Z_2 zuerst erreicht wird. Wenn nun Pläne generiert werden sollen, die vernünftige Ordnungen befolgen, würde in diesem Falle Z_2 wieder verloren gehen, um Z_1 zu erreichen und Z_2 müsste neu erreicht werden, um Z_3 zu erhalten. Die Ordnung $Z_1 \rightarrow_r Z_2$ kann jedoch **nicht** angenommen werden, weil Z_2 im Allgemeinen nicht in $\mathit{Aftermath}(Z_1)$ liegt. Ein Planer könnte die Ordnung $Z_1 \rightarrow_r Z_3$ nicht befolgen und damit Z_1 erst nach den beiden anderen erreichen. Die Problematik entsteht also erst dadurch, dass die vernünftige Ordnung $Z_1 \rightarrow_r Z_3$ eingehalten werden soll. Diese Tatsache soll durch eine Ordnung $Z_1 \rightarrow_r^{\{Z_1 \rightarrow_r Z_3\}} Z_2$ ausgedrückt werden, also einer weiteren Art von vernünftigen Ordnungen.

3.7 Folgende vernünftige Ordnungen

Sei O eine Menge von Ordnungen. Es werden nun folgende vernünftige Ordnungen definiert. Diese sind gerade vernünftig Ordnungen nach der bekannten Definition, falls zusätzlich alle Ordnungen in O befolgt werden:

1. $s \in S^O_{(\{v' \mapsto d'\}, \neg\{v \mapsto d\})}$ gdw. $\exists P = \langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*, n \in \mathbb{N}$, mit P befolgt alle Ordnungen in $O : \mathit{Result}(s_0, P) = s, \{v' \mapsto d'\} \subseteq \mathit{eff}_{o_n}$ und $\{v \mapsto d\} \not\subseteq \mathit{Result}(s_0, \langle o_1, \ldots, o_i \rangle), 0 \leq i \leq n$

2. $\{v' \mapsto d'\} \in \mathit{Aftermath}^O(\{v \mapsto d\})$ gdw. $\forall s \in S^O_{(\{v' \mapsto d'\}, \neg\{v \mapsto d\})}$ und für alle, diese Ordnungen befolgenden Pläne $P = \langle o_1, \ldots, o_n \rangle \in \mathcal{O}^*$, $n \in \mathbb{N}$ gilt: $s_\star \subseteq \mathit{Result}(s_0, P)$ und $\exists i, j, k : 1 \leq k \leq i \leq j \leq n :$ $s = \mathit{Result}(s_0, \langle o_1, \ldots, o_k \rangle)$, $\{v \mapsto d\} \subseteq \mathit{Result}(s, \langle o_1, \ldots, o_i \rangle)$, sowie $\{v' \mapsto d'\} \subseteq \mathit{Result}(s, \langle o_1, \ldots, o_j \rangle)$

3. $\{v \mapsto d\} \rightarrow^O_r \{v' \mapsto d'\}$ gdw. $\{v' \mapsto d'\} \in \mathit{Aftermath}^O(\{v \mapsto d\})$ und $\forall s \in S^O_{(\{v' \mapsto d'\}, \neg\{v \mapsto d\})} \forall P \in \mathcal{O}^* : \{v \mapsto d\} \subseteq \mathit{Result}(s, P) \Rightarrow \exists o \in P :$ $\{v' \mapsto \widetilde{d}\} \subseteq \mathit{eff}_o, \widetilde{d} \in \mathcal{D}_{v'} \backslash \{d'\}$

Die größte Änderung in den Definitionen ist $\mathit{Aftermath}^O$, da es hier nun nicht ausreichend ist, Pläne ab s zu betrachten, sondern diese schon seit s_0 die Ordnungen in O befolgen müssen. In Punkt 3. werden allgemeine Aktionssequenzen zugelassen, anstatt diese auf solche zu beschränken, die Ordnungen in O einhalten. Dies ist so aus „Ordered Landmarks in Planning" [HPS04] übernommen worden, was mit dem Hinweis auf die späteren Algorithmen begründet wird, welche diese Einschränkung im Falle 3. nicht nutzen. Wie auch bei den vernünftigen Ordnungen können hier widersprüchliche Ordnungen auftauchen (ein Beispiel hierzu findet sich in „Ordered Landmarks in Planning" [HPS04]).

Genau genommen könnte man nun „folgende folgende" vernünftige Ordnungen definieren, indem gefundene folgende vernünftige Ordnungen zu O hinzugefügt werden. Dies könnte weiter iteriert werden, bis für O ein Fixpunkt erreicht ist. In „Ordered Landmarks in Planning" [HPS04] wird hierauf verzichtet, da vermutet wird, dass in späteren Iterationen kaum nützliche Ordnungen auftauchen werden.

Da O in den späteren Algorithmen die Menge aller bekannten vernünftigen Ordnungen ist, wird in dieser Arbeit für $L \rightarrow^O_r L'$ meist nur $L \rightarrow_{or} L'$ geschrieben.

3.8 Landmarkengraph

Landmarken und ihre Ordnungen können in einem *Landmarkengraphen* repräsentiert werden.

Ein Landmarkengraph zu einem SAS$^+$-Planungsproblem Π, ist ein gerichteter Graph $LG(\Pi) = (LG_V, LG_E)$, mit $LG_V \subseteq \{\text{Landmarken in } \Pi\}$ als Knoten und $LG_E \subseteq \{\text{Ordnungen über } LG_V \text{ in } \Pi\}$ als annotierten Kanten. Sind $L, L' \in LG_V$, so ist $(L \rightarrow_o L') \in LG_E$ wenn eine Ordnung $L \rightarrow_o L'$ in Π existiert, wobei o die Art der Ordnung bezeichnet: $o \in \{n, gn, ln, r, or\}$. Nach dieser Definition ist der Landmarkengraph nicht eindeutig, da immer Ordnungen und Landmarken weggelassen werden können.

$LG(\Pi)$ ist azyklisch, falls Π lösbar ist und keine Art von vernünftigen Ordnungen in O existiert. Dies folgt aus der Tatsache, dass es sich um zeitliche Abfolgen handelt und Π lösbar ist, also die Reihenfolgen eingehalten werden können. Existieren vernünftige Ordnungen in LG_E, so könnte $LG(\Pi)$ zyklisch sein. Neben direkten Zyklen, die nur aus vernünftigen Ordnungen bestehen – wie in dem Beispiel bei den vernünftigen Ordnungen – können Zyklen auch zwischen vernünftigen und zwingenden Ordnungen auftreten. Im Logistikbeispiel gilt nach Definition auch $\{Truck \mapsto B\} \rightarrow_r \{Truck \mapsto C\}$ und sogar $\{Truck \mapsto A\} \rightarrow_r \{Truck \mapsto B\}$. Diese beiden Ordnungen können aufgrund der zwingenden Ordnungen nicht eingehalten werden. Weiter können Zyklen auch deutlich größer werden, so dass eine Entfernung aller Zyklen nicht trivial ist (In Abb. 3 würde die Ordnung $\{Truck \mapsto A\} \rightarrow_r \{Truck \mapsto C\}$ einen größeren Zykel bilden). Je nach Verwendung der Landmarken und Ordnungen während der Planung können Zyklen zur vermeintlichen Unlösbarkeit führen, so dass es bei manchen Verwendungen nötig ist, den Graph durch das Entfernen von Ordnungen azyklisch zu machen. Auf diese Details wird in einem späteren Kapitel eingegangen.

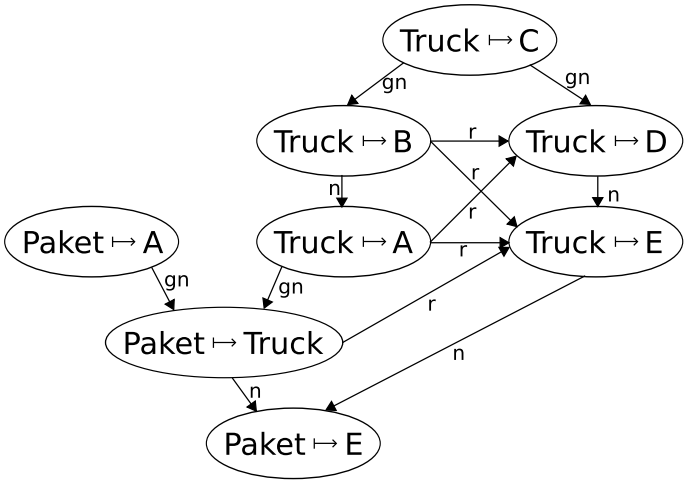


Abbildung 3: Ein Landmarkengraph für das Logistikbeispiel.

3.9 Komplexität

Die Entscheidungsprobleme, ob eine Zuweisung eine Landmarke ist oder ob für zwei gegebene Landmarken eine bestimmte Ordnung gilt, sind *PSPACE*-vollständig für STRIPS (Beweise siehe „Ordered Landmarks in Planning" [HPS04]). Für SAS$^+$ gelten die gleichen Aussagen, da die Beweise mit Reduktion der Plan-Existenz arbeiten – welche auch im SAS$^+$ Fall *PSPACE*-vollständig ist (siehe Bäckström und Nebel, „Complexity Results for SAS$^+$ Planning" [BN95]). Aufgrund dieser Komplexität sind in erster Linie Approximierungen für Landmarken und Ordnungen interessant.

4 Bisherige Approximierung von Landmarken und zwingenden Ordnungen

In diesem Kapitel sollen die Approximierungsverfahren für Landmarken und zwingende Ordnungen aus „Ordered Landmarks in Planning" [HPS04] direkt auf den SAS$^+$-Formalismus portiert werden. Wie später deutlich wird, lassen sich hier jedoch einige Verbesserungen bei den Verfahren erzielen. Zuerst wird nun die direkte Portierung mit ihren Problemen besprochen, und dann werden im nächsten Kapitel die erweiterten Algorithmen vorgestellt.

4.1 Verfahren zur Approximierung von Landmarken und notwendigen Ordnungen

Zuerst wird ein Approximierungsverfahren vorgestellt, das iterativ vom Ziel ausgehend Landmarken und notwendige Ordnungen findet. Die Zielbedingung $s_\star$ besteht nach Definition aus Landmarken. Nun ist es die Idee, für jede bekannte Landmarke $\{v' \mapsto d'\}$, die nicht unmittelbar in s_0 liegt, die Vorbedingungen aller Operatoren zu betrachten, die diese Landmarke erreichen.[3] Haben all diese Operatoren eine gemeinsame Vorbedingung, so ist diese wiederum eine Landmarke und kann notwendig vor $\{v' \mapsto d'\}$ geordnet werden. Dies folgt aus der Überlegung, dass jede Landmarke, die nicht in s_0 liegt, in jeder Lösung durch einen Operator erreicht werden muss – was also bedeutet, dass die Operatorvorbedingungen in dem Zustand direkt davor erfüllt sein müssen. Da im Voraus nicht klar ist, welcher Operator verwendet wird, werden die gemeinsamen Vorbedingungen aller dieser Operatoren betrachtet. Die gefunden Landmarken und Ordnungen werden in einem Landmarkengraphen gespeichert.

[3] Falls kein Operator dieser Art existiert, ist das Problem nicht lösbar, da die Landmarke nicht in s_0 liegt und somit nie erreicht werden kann. Es wird hier davon ausgegangen ein lösbares Problem zu betrachten. Die Algorithmen werden dieses Kriterium nicht untersuchen, da die meisten Planer bessere Methoden zur Erkennung von unlösbaren Problemen zur Verfügung stellen.

Für das Logistikbeispiel liefert dieses Verfahren lediglich die Ordnungen und Landmarken: $\{Truck \mapsto D\} \rightarrow_n \{Truck \mapsto E\}$, $\{Truck \mapsto E\} \rightarrow_n \{Paket \mapsto E\}$, $\{Paket \mapsto Truck\} \rightarrow_n \{Paket \mapsto E\}$. Dies sind nicht alle notwendigen Ordnungen und Landmarken des Beispiels. Insbesondere $\{Truck \mapsto B\} \rightarrow_n \{Truck \mapsto A\}$ wird nicht gefunden, da keine Landmarken existieren, die sich notwendig vor $\{Paket \mapsto Truck\}$ ordnen lassen und das Verfahren damit die Iteration an dieser Stelle nicht fortführt. Außer des Zieles werden also höchstens weitere Landmarken gefunden, die sich notwendig vor einer bereits bekannten Landmarke ordnen lassen.

4.1.1 Pseudocode für die Approximierung von Landmarken und notwendigen Ordnungen

Bezeichne $LG = (LG_V, LG_E) = LG(\Pi)$ den Landmarkengraphen.

$LG := (s_\star, \emptyset)$
$N := s_\star$
`while` $N \neq \emptyset$
    $N' := \emptyset$
    `for each` $\{v' \mapsto d'\} \in N, \{v' \mapsto d'\} \not\subseteq s_0$
        $O := \{o | o \in \mathcal{O}, \{v' \mapsto d'\} \subseteq eff_o\}$
        `for each` $\{v \mapsto d\} \subseteq \bigcap_{o' \in O} pre_{o'}$
            `if` $\{v \mapsto d\} \notin LG_V$
                $LG_V := LG_V \cup \{v \mapsto d\}$
                $N' := N' \cup \{v \mapsto d\}$
            $LG_E := LG_E \cup \{(\{v \mapsto d\} \rightarrow_n \{v' \mapsto d'\})\}$
    $N := N'$

4.2 Verfahren zur Approximierung von Landmarken und greedy-notwendigen Ordnungen

Wie bereits bei den Definitionen der Ordnungen gezeigt wurde, ist es eine starke Einschränkung, nur rein notwendige Ordnungen zu betrachten. Insofern ist es von Interesse, das Verfahren zu modifizieren, um auch solche Landmarken zu finden, die nur greedy-notwendig geordnet werden können. Um dies zu erreichen, wird in „Ordered Landmarks in Planning" [HPS04] ein zweistufiges Verfahren eingeführt:

1. Der bisherige Algorithmus wird so modifiziert, dass er eine Menge von Landmarken-Kandidaten mit Ordnungen generiert, wobei weder gesichert ist, dass es sich bei den Zuweisungen um Landmarken handelt, noch dass die gefundenen Ordnungen greedy-notwendig sind.

2. Aus der Kandidatenmenge werden alle diejenigen Zuweisungen entfernt, für die nicht hinreichend nachgewiesen werden kann, dass es

sich um Landmarken handelt. Die eventuell falschen Ordnungen bleiben jedoch erhalten.

Diesem zweistufigen Verfahren gelingt es somit nicht immer, korrekte greedy-notwendige Ordnungen zu finden, was aber nach Aussagen in „Ordered Landmarks in Planning" [HPS04] in der Praxis selten vorkommt.

4.2.1 Generierung von Kandidaten für Landmarken und greedy-notwendige Ordnungen

Die Generierung der Kandidaten wird mit Hilfe eines relaxierten Planungsgraphen durchgeführt. Für ein SAS$^+$-Planungsproblem Π lässt sich dies wie folgt formalisieren: Es werden keine Zustände mehr betrachtet, sondern Mengen von Zuweisungen, die erreicht wurden. Formal: Sei $i \in \mathbb{N}$,

$$S_i := \begin{cases} s_0 & i = 0 \\ S_{i-1} \cup \{\mathit{eff}_o | o \in \mathcal{O}_{i-1}\} & i > 0 \end{cases}$$

$$\mathcal{O}_i := \begin{cases} \{o \in \mathcal{O} | \mathit{pre}_o \subseteq S_0\} & i = 0 \\ \{o \in \mathcal{O} | \mathit{pre}_o \subseteq S_i\} \setminus \bigcup_{j=0}^{i-1} \mathcal{O}_j & i > 0 \end{cases}$$

S_i ist somit die Menge der erreichten Zuweisungen und $\mathcal{O}_i$ ist die Menge der anwendbaren Operatoren, die bisher noch nicht angewendet wurden. Die Menge S_i erreicht einen Fixpunkt nach polynomiell vielen Schritten, da es nach Konstruktion von S_0 aus nur polynomiell viele verschiedene S_i gibt. S_i stellt eine Überapproximierung der Erreichbarkeit von Zuweisungen dar. Ist das Planungsproblem lösbar, so ist $s_\star$ erreichbar.[4] Geht man von einem lösbaren Problem aus, ist der relaxierte Planungsgraph von folgender Gestalt:

$$S_0 \overset{\mathcal{O}_0}{\rightsquigarrow} S_1 \overset{\mathcal{O}_1}{\rightsquigarrow} \dots \overset{\mathcal{O}_{n-1}}{\rightsquigarrow} S_n \supseteq s_\star$$

Tatsächlich reicht die Bedingung $s_\star \subseteq S_n$ für die Verwendungszwecke in diesem Kapitel aus, so dass nicht notwendigerweise ein Fixpunkt erreicht sein muss. Nun wird der gegebene Algorithmus zur Approximierung von Landmarken und notwendigen Ordnungen in modifizierter Form verwendet. Anstatt einen Landmarkengraphen zu generieren, wird nun ein Graph konstruiert, der aus Kandidaten und ihren möglichen greedy-notwendigen Ordnungen besteht. Das Vorgehen ist dabei fast identisch zu dem alten Verfahren. Geändert wird lediglich ein kleines Detail: Anstatt alle Operatoren zu betrachten die einen Landmarken-Kandidaten erreichen können, beschränkt sich dieses Verfahren nun auf diejenigen Operatoren, die als erste im relaxierten Planungsgraphen diesen Kandidaten erreichen können. Dies bedeutet also, dass für jeden Kandidaten $\{v' \mapsto d'\}$, der nicht in $s_0 = S_o$

[4] Die Unerreichbarkeit von $s_\star$ impliziert die Unlösbarkeit von Π. Man könnte hier zwar unlösbare Probleme erkennen, jedoch soll dies nicht das Ziel dieser Arbeit sein.

und damit in $S_i, i > 0$ liegt, die Menge der Operatoren O gebildet wird, die genau aus den Operatoren besteht, die $\{v' \mapsto d'\}$ als Effekt haben und auch in O_{i-1} liegen. Im Endeffekt werden nun weniger Operatoren gleichzeitig betrachtet. Dies hat zu Folge, dass dieses Verfahren hier immer noch all die notwendigen Ordnungen und dazugehörigen Landmarken findet, die durch Verfahren für rein notwendige Landmarken gefunden worden wären. Hier werden die Ordnungen nun allerdings als greedy-notwendige Ordnungen gespeichert, was für die späteren Verwendungszwecke kaum einen Nachteil darstellt, da nach Definition nur davon ausgegangen werden kann, Landmarken einmal zu erreichen. Hier tritt nun allerdings das Problem der Kandidaten und Ordnungen auf. Da nur noch die ersten Operatoren, die einen Kandidaten erreichen, benutzt werden, ist nicht gesichert, dass eine gemeinsame Vorbedingung eine Landmarke ist. Deutlich wird dies an einem Beispiel in Abb. 4, das wieder aus „Ordered Landmarks in Planning" [HPS04] stammt.

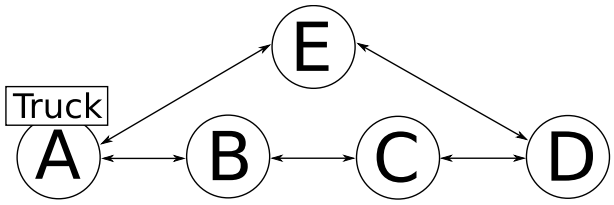


Abbildung 4: Ein Problem, bei dem ein Weg von A nach D gefunden werden soll.

Da der untere Weg von A nach D länger ist, betrachtet das Verfahren nur den Übergang von E nach D und ignoriert den Weg von C nach D. Der Operator für den Übergang von E nach D steht in $\mathcal{O}_1$, wohingegen der Operator C nach D erst in der nächsten Operatormenge $\mathcal{O}_2$ auftritt. Folglich wird E zu einem Landmarken-Kandidaten und die Ordnung $\{E \rightarrow_{gn} D\}$ wird eingefügt. Bei diesem Beispiel, würde der Kandidat E innerhalb des zweiten Schrittes als Nicht-Landmarke erkannt und mit der Ordnung $\{E \rightarrow_{gn} D\}$ zusammen entfernt werden. Ist jedoch E aus anderen Gründen eine Landmarke, da zum Beispiel ein Paket bei E abgeholt werden muss, so wird – falls das Verfahren E als Landmarke erkennt – die falsche Ordnung nicht entfernt.

4.2.2 Pseudocode für die Approximierung von Kandidaten für Landmarken und greedy-notwendige Ordnungen

Bezeichne $KG = (KG_V, KG_E)$ den Graphen für die Kandidaten und Ordnungen.

Generiere den relaxierten Planungsgraphen zu Π: $S_0, \mathcal{O}_0, \ldots, \mathcal{O}_{n-1}, S_n$
$KG := (s_\star, \emptyset)$
$K := s_\star$
`while` $K \neq \emptyset$
    $K' := \emptyset$
    `for each` $\{v' \mapsto d'\} \in K, \{v' \mapsto d'\} \subseteq S_i, i \neq 0$
        $O := \{o | o \in \mathcal{O}, \{v' \mapsto d'\} \subseteq \mathit{eff}_o, \{v' \mapsto d'\} \in \mathcal{O}_{i-1}\}$
        `for each` $\{v \mapsto d\} \subseteq \bigcap_{o' \in O} \mathit{pre}_{o'}$
            `if` $\{v \mapsto d\} \notin KG_V$
                $KG_V := KG_V \cup \{v \mapsto d\}$
                $K' := K' \cup \{v \mapsto d\}$
            $KG_E := KG_E \cup \{(\{v \mapsto d\} \rightarrow_{gn} \{v' \mapsto d'\})\}$
    $K := K'$

4.2.3 Verifikation von Landmarken

Nachdem der Graph mit den Kandidaten konstruiert wurde, werden nun alle Kandidaten inklusive ihrer Ordnungen entfernt, für die nicht hinreichend nachgewiesen werden kann, dass sie Landmarken sind.
Dazu wird folgendes hinreichendes Kriterium verwendet:

Sei Π das SAS$^+$-Planungsproblem und $\{v \mapsto d\}$ eine Zuweisung.
Ist $\overline{\Pi} := \langle \mathcal{V}, \{o \in O | \{v \mapsto d\} \not\subseteq \mathit{eff}_o\}, s_0, s_\star \rangle$ unlösbar, so ist $\{v \mapsto d\}$ eine Landmarke.

Die Korrektheit folgt unmittelbar: Ist Π unlösbar, so ist $\overline{\Pi}$ ebenfalls immer unlösbar. Dies ist korrekt, da nach Definition im unlösbaren Fall alle Zuweisungen Landmarken sind. Sei nun Π lösbar. Aus der Unlösbarkeit von $\overline{\Pi}$ folgt, dass ein Operator mit Effekt $\{v \mapsto d\}$ angewendet hätte werden müssen. Somit tritt $\{v \mapsto d\}$ in jeder Lösung von Π auf und ist damit eine Landmarke.

Da es zu aufwendig ist, für jeden Kandidaten ein eigenes Planungsproblem zu lösen, wird $\overline{\Pi}$ lediglich relaxiert betrachtet. Mit der oben gegebenen Definition des relaxierten Planungsgraphen, gilt $s_\star \subseteq S_n$ für jedes lösbare Problem. In der Implementierung wird nun der relaxierte Planungsgraph zu $\overline{\Pi}$ gebildet und getestet, ob $s_\star \subseteq S_n$ gilt. Denn ist $s_\star$ nicht in S_n enthalten, so ist $\overline{\Pi}$ nicht lösbar. Somit ist das Kriterium selbst relaxiert noch hinreichend.

Kandidaten, bei denen das relaxierte Planungsproblem lösbar ist, werden nun inklusive ihrer Ordnungen entfernt. Übrig bleibt nun ein Graph, der als Knoten echte Landmarken besitzt. Dies bedeutet allerdings noch nicht, dass dieser Graph ein Landmarkengraph nach Definition ist, da die enthaltenen Ordnungen möglicherweise nicht korrekt sind. In „Ordered Landmarks in Planning" [HPS04] werden diese Ordnungen nicht korrigiert. Diese falschen Ordnungen sind zwar keine Ordnungen im Sinne der greedy-notwendigen Definition, jedoch wurden einige Gründe aufgeführt, warum es nicht unbe-

dingt schlecht ist, diese nicht extra zu entfernen:

- Bei der Ordnung $L \rightarrow L'$ handelt es sich um Informationen darüber, welche Vorbedingungen im relaxierten Planungsgraphen für das erste Erreichen von L' nötig sind. Es ist also auch eine Hinweis darauf, wie man L' womöglich frühzeitig erreichen könnte.
- Die Ordnungen werden in den meisten Fällen mit einem Kandidaten zusammen entfernt. Wirklich falsche Ordnungen treten also selten auf.
- In „Ordered Landmarks in Planning" [HPS04] werden bei der Verwendung von Landmarken und Ordnungen während der Planung keine Ordnungen erzwungen.
- Ohne den relaxierten Planungsgraphen würde man wieder rein notwendige Ordnungen erzeugen, von denen meist nur sehr wenige gefunden werden.

4.3 Lookahead-notwendige Ordnungen

In „Ordered Landmarks in Planning" [HPS04] wird kurz auf ein Verfahren zur Approximierung von lookahead-notwendigen Ordnungen für STRIPS eingegangen. Man versucht dabei Fälle in den Griff zu bekommen, bei denen Operatoren, die eine Landmarke erreichen, selbst keine gemeinsamen Vorbedingungen haben, jedoch ein Fakt gelten muss, um die Vorbedingungen dieser Operatoren zu erhalten. Um solche Zuweisungen zu finden, wird wieder der relaxierte Planungsgraph verwendet. Das Verfahren beschränkt sich dabei auf eine maximale Entfernung von 2 Schritten im relaxierten Planungsgraphen und verfolgt dabei nur Fakten, die aus gleichen Prädikaten stammen.

Diese Beschreibung sollte hier nur der Vollständigkeit halber einen sehr groben Überblick über das Verfahren geben – für eine genauere Beschreibung siehe ursprünglichen Artikel [HPS04]. Auf eine Implementierung wird in dieser Arbeit zugunsten eines speziellen SAS$^+$-Verfahrens verzichtet. Dieses Verfahren findet sich im nächsten Kapitel.

5 Verbesserte Approximierung von Landmarken und zwingenden Ordnungen für SAS$^+$

Die bisher vorgestellten Approximierungsverfahren stellen eine direkte Portierung der Verfahren aus der ursprünglichen Arbeit [HPS04] dar, welche für STRIPS-Probleme entwickelt wurden. Mit einigen Änderung und unter Ausnutzung des SAS$^+$-Planungsformalismus lassen sich einige Verbesserungen erzielen. Unter anderem sollen nun korrekte greedy-notwendige Ordnungen gefunden werden. Zumindest bei einigen Domänen traten nach

„Ordered Landmarks in Planning" [HPS04] während der Verwendung der Landmarken und Ordnungen bei der Planung Probleme auf, die auf falsche greedy-notwendige Ordnungen zurückgeführt wurden. In solchen Fällen wurde zu Vergleichszwecken das Verfahren, das rein notwendigen Ordnungen findet, herbeigezogen. Dieses findet jedoch meist deutlich weniger Landmarken, weswegen der Vergleich etwas schwierig ist. Hier soll nun diese Problematik umgangen werden, indem Ordnungen korrekt approximiert werden.

5.1 Approximierung von Landmarken und zwingenden Ordnungen

Es gibt eine andere Möglichkeit, das Grundverfahren zur Approximierung von Landmarken und rein notwendigen Ordnungen auf greedy-notwendige Ordnungen auszudehnen. Dort wurden alle Operatoren, die eine Landmarke L erreichen, in die Menge O aufgenommen. Weitere Landmarken werden dann nur gefunden, falls alle Operatoren in O gemeinsame Vorbedingungen aufweisen. Das alte Verfahren für greedy-notwendige Ordnungen umging dieses Problem, indem nur noch Operatoren betrachtet wurden, die als erste innerhalb des relaxierten Planungsgraphen auftraten. Allerdings werden mit diesem Vorgehen nicht mehr zwingend Landmarken, sondern Kandidaten generiert.

In diesem Verfahren soll O nun so klein wie möglich gewählt werden, ohne dass relevante Operatoren fehlen. Dafür wäre es am besten, nur genau die Operatoren zu betrachten, die zum erstmaligen Erreichen der Landmarke von s_0 aus benutzbar sind. Der Test, ob ein Operator $o \in O$ von s_0 aus vor dem ersten Erreichen einer Landmarke $L \not\subseteq s_0$ angewendet werden kann, ist jedoch *PSPACE*-vollständig.[5] Aus diesem Grund soll eine Approximierung verwendet werden. Dazu sei *possibly_applicable_before*(s_0, o, L) ein Test ob o in einer Aktionssequenz die von s_0 aus startet, möglicherweise anwendbar sein kann, bevor die Landmarke L erreicht wird. Es wird gefordert, dass *possibly_applicable_before*(s_0, o, L) ein notwendiges Kriterium für die Anwendbarkeit darstellt. Wird nun O mit Hilfe des Tests auf die Operatoren eingeschränkt, die möglicherweise anwendbar sind, so ist wegen der Notwendigkeit gesichert, dass mindestens alle wirklich anwendbaren Operatoren in O enthalten sind. Auf diese Weise sind nun durch gemeinsame Vorbedingungen sowohl korrekte greedy-notwendige Ordnungen, als auch Landmarken gegeben. Zusätzlich dazu können noch lookahead-notwendige Ordnungen über einen Domänengraphen gefunden werden.

[5] In *PSPACE* mit guess & check, Vollständigkeit mit Reduktion der Plan-Existenz: Ersetze Zielbedingung $s_\star$ durch neue Variable N und füge Operator $o := \langle s_\star, N \rangle$ ein. Operator anwendbar gdw. ursprüngliches Planungsproblem lösbar.

5.2 Landmarken, Ordnungen und der Domänentransitionsgraph

Wie bereits bei den Definitionen der zwingenden Ordnungen angedeutet wurde, lassen sich aus einem Domänentransitionsgraphen Landmarken und Ordnungen herauslesen. So entspricht die Menge O aller Operatoren, die eine Zuweisung $\{v \mapsto d\}$ erreichen, nach der Definition des Domänentransitionsgraphen, gerade den eingehenden Kanten des Knotens d in G_v. Für den Algorithmus, der rein notwendige Ordnungen erzeugt, ist diese Beobachtung uninteressant, denn hier bildet die Menge O gerade alle eingehenden Kanten. Anders sieht es jedoch aus, wenn man an greedy-notwendigen Ordnungen interessiert ist.

5.2.1 Greedy-notwendige Ordnungen

In jedem Plan startet die Variable v mit dem Wert d_0, $\{v \mapsto d_0\} \subseteq s_0$. Operatoren, die diesen Wert verändern, bilden die Kanten im Domänentransitionsgraphen. In jeder Aktionssequenz entsprechen Werteänderungen von v Anwendungen von Operatoren mit Effekten für v, also Übergänge zwischen Knoten in G_v. Geht man davon aus, dass die Zuweisung $\{v \mapsto d\}$ bisher noch nicht erreicht wurde, so entspricht dies der Betrachtung, dass der Knoten d in G_v ebenfalls nicht erreicht wurde. Damit ist es ausgeschlossen, dass Knoten in G_v erreicht wurden, die von d_0 aus nur über Pfade erreichbar sind, die d enthalten. Auf diese Weise können die möglichen Werte die v vor dem Erreichen von d annehmen kann, reduziert werden. Betrachtet man die Menge O, sind für greedy-notwendige Ordnungen vor $\{v \mapsto d\}$ gerade solche Operatoren nicht relevant, die mit Kanten korrespondieren, die von nicht erreichten Knoten ausgehen. Es handelt sich also um ein Kriterium das den Anforderungen von *possibly_applicable_before*$(s_0, o, \{v \mapsto d\})$ genügt. Beispiele, bei denen dieses Vorgehen gut funktioniert finden sich in Abb. 2 und in Abb. 5.

Prinzipiell ist man für *possibly_applicable_before*(s_0, o, L) eigentlich daran interessiert zu wissen, ob die Vorbedingungen von $o \in \mathcal{O}$ erfüllt werden können, ohne $L = \{v \mapsto d\}$ zu erreichen. Ist $L \not\subseteq s_0$, so entspricht dies der Frage, ob $\Pi_o(L) := \langle \mathcal{V}, \{o' \in O | L \not\subseteq \mathit{eff}_{o'}\}, s_0, \mathit{pre}_o \rangle$ lösbar ist. Für die geforderte Eigenschaft von *possibly_applicable_before*(s_0, o, L) reicht es eine Relaxierung zu verwenden, deren Unlösbarkeit die Unlösbarkeit von $\Pi_o(L)$ zeigt. Das obige Vorgehen stellt eine solche Relaxierung dar, die sogar mit reinen Graphsuchverfahren auskommt. Allerdings ignoriert diese Relaxierung jegliche Interaktionen mit anderen Variablen, denn hier wird nur v und seine möglichen Werteänderungen betrachtet – es werden also alle Operatorvorbedingungen an andere Variablen immer als erfüllt angesehen.

Der Zeitbedarf für das Approximieren von Landmarken und Ordnungen ist um ein Vielfaches kleiner als der Zeitbedarf für die Planung. Aus diesem

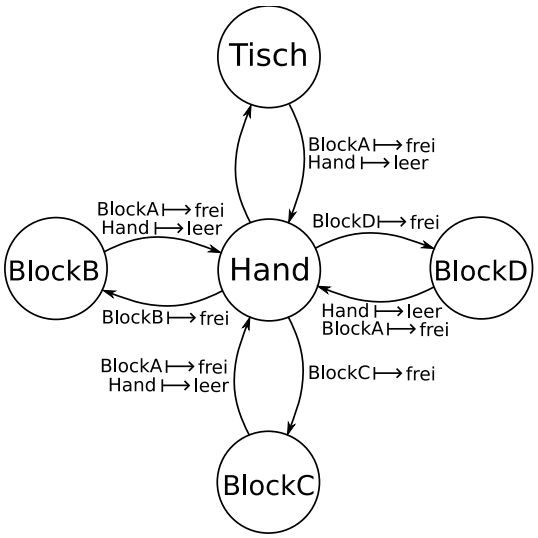


Abbildung 5: Blockswelt – Domänentransitionsgraph G_A gibt an, worauf der Block A gerade steht. Ist $\{BlockA \mapsto Hand\}$ nie erreicht worden, so hat die Variable noch ihren Startwert.

Grund soll hier trotz der Einfachheit der obigen Approximierung über Graphsuche wieder der relaxierte Planungsgraph als Approximierung verwendet werden. Mit dem relaxierten Planungsgraphen wird immer eine mindestens gleich gute Approximierung erzielt, da innerhalb des Planungsgraphen keine Zuweisung $\{v \mapsto d'\}$ auftreten kann, die nur über Pfade im Domänentransitionsgraphen, die d durchlaufen, erreichbar ist.

Nun könnte man einfach den relaxierten Planungsgraphen für $\Pi_o(L)$ aufbauen und entscheiden ob es relaxiert lösbar ist. Bei genauerer Betrachtung fällt allerdings auf, dass dieser Test für alle Operatoren in O durchgeführt werden muss, welche alle L im Effekt enthalten. Alle $\Pi_o(L)$ unterscheiden sich für verschiedene $o \in O$ nur in der Zielbedingung. Daraus kann ein Vorteil gezogen werden. Für die Approximierung von $\Pi_o(L)$ wird ein relaxierter Planungsgraph verwendet, der nun allerdings nicht für jedes $o \in O$ neu aufgebaut wird, sondern vielmehr für die je aktuelle Landmarke L, einmal mit beliebiger Zielbedingung konstruiert wird, bis der Fixpunkt S_n erreicht ist. S_n entspricht mindestens all denen Zuweisungen, die vor Erreichen von L auftreten können. Für $possibly\_applicable\_before(s_0, o, L)$ genügt dann der einfache Test, ob $pre_o \subseteq S_n$ gilt.

Ein weiterer Vorteil der Menge S_n besteht darin, dass durch das Komplement von S_n gerade solche Zuweisungen gegeben sind, die erst nach oder mit L zusammen erreicht werden können. Ist es ausgeschlossen, eine solche Zuweisung mit L gleichzeitig erreichen zu können, lässt sich diese lookaheadnotwendig hinter L ordnen. Allerdings geht aus S_n selbst nicht hervor, welche Zuweisungen Landmarken sind.

5.2.2 Lookahead-notwendige Ordnungen

Der Domänentransitionsgraph kann aber trotz des relaxierten Planungsgraphen benutzt werden um Landmarken mit lookahead-notwendige Ordnungen zu finden. Für den Nachweis einer lookahead-notwendige Ordnung vor eine Zuweisung $\{v \mapsto d\}$ versucht man innerhalb des Domänentransitionsgraphen G_v Knoten zu bestimmen, die für das Erreichen von d notwendig sind. Dies beschränkt lookahead-notwendige Ordnung an dieser Stelle auf Zuweisungen einer gleichen Variable. Betrachtet man hierzu das Beispiel in Abb. 6, wird deutlich, dass gerade die Knoten, die auf jedem Pfad von d_0 nach d auftreten lookahead-notwendig vor d geordnet werden können. In Abb. 6 wäre dies unter anderem $\{Paket \mapsto Truck1\} \rightarrow_{ln} \{Paket \mapsto Airport3\}$.

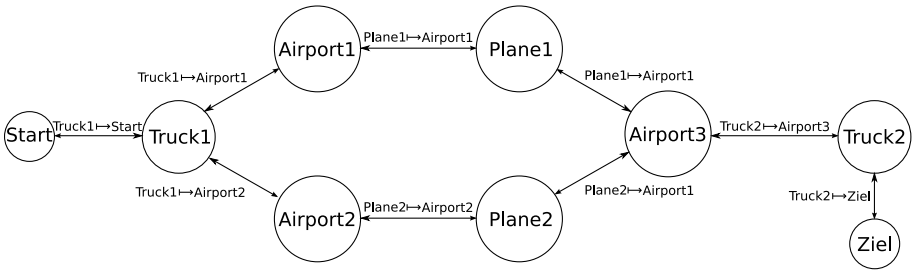


Abbildung 6: Domänentransitionsgraph G_{Paket} in einem größeren Logistikproblem.

Solche Knoten können mit einfachen Graphsuchverfahren gefunden werden. Hierzu entfernt man einfach immer je genau einen Knoten und testet, ob d_0 mit d im Graphen noch verbunden ist. Werden die greedy-notwendigen Ordnungen mit Hilfe des relaxierten Planungsgraphen approximiert, so kann die resultierende Menge S_n hier für die Graphsuche genutzt werden. Man entfernt aus G_v alle nicht erreichten Werte von v außer d. Diese Werte werden erst nach $\{v \mapsto d\}$ erreicht, und müssen deswegen nicht getestet werden.

Diese Graphsuche stellt – wie auch bei *possibly_applicable_before*(s_0, o, L) – eine vereinfachte, beschränkte Form eines relaxierten Planungsgraphen dar. Für einen relaxierten Planungsgraphen wäre dieses Vorgehen folgendermaßen zu beschreiben: Wähle einen Knoten d' in v, verbiete alle Operatoren, die $\{v \mapsto d'\}$ erreichen, und prüfe ob $\{v \mapsto d\}$ von s_0 aus nicht mehr erreichbar ist. Eine Beschränkung auf die Variable v und ihre Werteänderungen ist hier jedoch eher sinnvoll, da ein relaxierter Planungsgraph mindestens für jede Zuweisung $\{v \mapsto \widetilde{d}\}, \widetilde{d} \in D_v \backslash (\{d\} \cup \overline{S_n})$ nötig wäre, was sich im Hinblick auf das Ergebnis wahrscheinlich nicht rentiert. Es gibt allerdings eine Verallgemeinerung dieser greedy-notwendigen und lookahead-notwendigen Approximierungen die nur auf relaxierten Planungsgraphen basiert, welche am Ende dieses Kapitels kurz erläutert wird. In diesem Teil des Verfahren soll nun aber weiter die Graphsuche zum Einsatz kommen.

Den Nachteil der Graphsuche, sich auf lookahead-notwendige Ordnungen zwischen Zuweisungen einer Variable zu beschränkt, kann mit Hilfe von S_n behoben werden. Bereits bekannte Landmarken, die in S_n liegen, können – falls es unmöglich ist, diese zeitgleich mit $\{v \mapsto d\}$ zu erreicht – lookahead-notwendig **hinter** $\{v \mapsto d\}$ geordnet werden. Auf diese Weise entstehen lookahead-notwendige Ordnungen zwischen Zuweisungen verschiedener Variablen. Dabei werden allerdings keine neuen Landmarken gefunden. Während das Verfahren noch Landmarken approximiert, könnte man so nur die bereits bekannten Landmarken ordnen. Aus diesem Grund sollen alle Zuweisungen, die nicht in S_n liegen, zu der zugehörigen Landmarke L in $S(L)$ gespeichert werden. Die daraus resultierenden lookahead-notwendige Ordnungen werden dann nach Beendigung der Landmarken-Approximierung eingefügt.

5.3 Erweiterung auf den propositionalen Teil von PDDL2.2

Es ist das Ziel, dieses Verfahren innerhalb des Downward-Planungssystems zu implementieren. Dieses Planungssystem unterstützt den propositionalen Teil von PDDL2.2. Aus diesem Grund wird das Verfahren auf einfache Weise auf die notwendigen Features von PDDL2.2 ausgeweitet. Dies sind im Besonderen:

- Axiome und abgeleitete Variablen
  Um die Implementierung einfach zu halten, werden abgeleitete Variablen – genauso wie Axiome – ignoriert. Somit werden abgeleitete Variablen nie Landmarken. In seltenen Fällen ist es möglich, dass in einem PDDL2.2 Problem das Ziel nur aus abgeleiteten Variablen besteht – ein Beispiel hierfür ist die *miconic-fulladl* Domäne aus den ICAPS-Planungsbenchmarks. Der Algorithmus findet in solchen Fällen überhaupt keine Landmarke, da nicht iterativ vom Ziel ausgegangen werden kann.

- Bedingte Effekte
  Innerhalb des Verfahrens werden häufig Vorbedingungen eines Operators $o \in \mathcal{O}$, der eine Landmarke erreicht, verwendet. Von Interesse ist nicht die allgemeine Anwendung von o, sondern vielmehr die Anwendung des Operators, die auch zur Folge hat, dass die Landmarke im nachfolgenden Zustand erreicht wird. Ist die Landmarke ein bedingter Effekt in o, so ist die Verwendung von pre_o ohne die Bedingungen an den Effekt mit der Landmarke zu beachten nicht falsch, sondern nur etwas ungenauer. Man ignoriert damit eventuelle weitere nützliche Informationen. Insofern wäre es hilfreich, die Bedingung an den Effekt zu pre_o hinzuzufügen. Hierbei ergibt sich das Problem, dass diese Bedingung eine Disjunktion darstellen könnte, indem der Effekt mit der Landmarke mehrfach mit verschiedenen Bedingungen vorkommt. Aus

diesem Grund beschränkt sich die Implementierung darauf, die Bedingung an den Effekt genau dann zu pre_o hinzuzufügen, wenn die Landmarke nur genau einmal in eff_o auftaucht.

Weiter muss auch der relaxierte Planungsgraph bei $\Pi(L)$ mit bedingten Effekten umgehen können. Hier werden Operatoren, die L nur als bedingten Effekt haben zugelassen. Bei dem Aufbau des relaxierten Planungsgraphen werden die Bedingungen an Effekte immer als erfüllt angesehen, falls der Effekt nicht die Landmarke L ist. Auf diese Weise ist sichergestellt, dass die Landmarke L nicht erreicht wird. S_n bildet nun eine noch stärke Überapproximierung.

5.4 Pseudocode für die Approximierung von Landmarken und Ordnungen

Sei $LG := (LG_V, LG_E) = LG(\Pi)$.

$LG := (s_\star, \emptyset)$
$N := s_\star$
`while` $N \neq \emptyset$
    $N' := \emptyset$
    `for each` $\{v' \mapsto d'\} \in N, \{v' \mapsto d'\} \not\subseteq s_0, \{v' \mapsto d'\} \notin N'$
        Bestimme $S_0, \mathcal{O}_0, \ldots, S_n$ für $\Pi(\{v' \mapsto d'\})$ bis zum Fixpunkt
        $O := \{o | o \in \mathcal{O}, \{v' \mapsto d'\} \subseteq \mathit{eff}_o, pre_o \subseteq S_n\}$
        `for each` $\{v \mapsto d\} \subseteq \bigcap_{o \in O}(pre_o \cup$ einzige Vorbedingung an $\{v' \mapsto d'\})$
            `if` $\{v \mapsto d\} \notin LG_V$
                $LG_V := LG_V \cup \{v \mapsto d\}$
                $N' := N' \cup \{v \mapsto d\}$
            $LG_E := LG_E \cup \{(\{v \mapsto d\} \rightarrow_{gn} \{v' \mapsto d'\})\}$
        d_0 so gewählt, dass $\{v' \mapsto d_0\} \subseteq s_0$
        `for each` $d \in \mathcal{D}_{v'}, d \neq d'$,[a] es gibt keinen Weg von d_0 nach d' in $D_{v'}|_{\mathcal{V} \setminus \{d, d'' | \{v' \mapsto d''\} \notin S_n\}}$
            `if` $\{v' \mapsto d\} \notin LG_V$
                $LG_V := LG_V \cup \{v' \mapsto d\}$
                $N' := N' \cup \{v' \mapsto d\}$
            $LG_E := LG_E \cup \{(\{v' \mapsto d\} \rightarrow_{ln} \{v' \mapsto d'\})\}$
        $S(\{v' \mapsto d'\}) := \overline{S_n}$
    $N := N'$
`for each` $\{v \mapsto d\} \in LG_V, \{v \mapsto d\} \not\subseteq s_0$
    `for each` $\{v' \mapsto d'\} \in LG_V, \{v' \mapsto d'\} \subseteq S(\{v \mapsto d\})$
        `if` $\nexists o \in \mathcal{O} : \{v \mapsto d\}, \{v' \mapsto d'\} \in \mathit{eff}_o$
            $LG_E := LG_E \cup \{(\{v' \mapsto d\} \rightarrow_{ln} \{v' \mapsto d'\})\}$

[a] Für $d = d_0$ existiert kein Weg

5.4.1 Laufzeit

Die Laufzeit dieses neuen Verfahrens wäre vergleichbar mit dem alten Verfahren, falls auf die lookahead-notwendigen Ordnungen verzichtet wird, da in beiden für jede Landmarke, bzw. Kandidat ein gleich relaxiertes Planungsproblem betrachtet wird. Bei dem neuen Verfahren kommt nun allerdings zur Laufzeit noch die Graphsuche in den Domänengraphen sowie die Schleifen über die Menge der Landmarken hinzu. Insgesamt ist das Verfahren immer noch polynomiell in der Anzahl an Variablen und der Größe der zugehörigen Domänen.

5.5 Vergleich mit dem alten Verfahren

Verzichtet man auf die lookahead-notwendigen Ordnungen und Landmarken und betrachtet Planungsprobleme ohne Axiome und bedingte Effekte, so ergibt sich zumindest bei der Verwendung des relaxierten Planungsgraphen für *possibly_applicable_before*(s_0, o, L), der folgende Zusammenhang zwischen den beiden Verfahren:[6]

$$\{\text{Landmarken, altes Verfahren}\} \supseteq \{\text{Landmarken, neues Verfahren}\}$$

Zuerst einmal gilt: {Kandidaten} $\supseteq$ {Landmarken, neues Verfahren}. Kandidaten sind das Ergebnis von $\bigcap_{\{o | o \text{ erreicht Landmarke } L \text{ als erstes}\}} pre_o$. Die Vorbedingungen dieser Operatoren o sind damit auf jeden Fall relaxiert erfüllt, bevor L erreicht wurde. Folglich schneidet das neue Verfahren auch mindestens über diese pre_o und erreicht so höchstens die gleichen gemeinsamen Vorbedingungen.

Bei der Verifikation der Landmarken werden nur Kandidaten entfernt, die das neue Verfahren nicht gefunden hätte. Zeigen lässt sich dies über die Tatsache, dass alle gefundenen Landmarken aus dem neuen Verfahren dem hinreichenden Kriterium genügen.[7] Induktiv: Das neue Verfahren startet mit $s_\star$, welche von dem hinreichenden Kriterium korrekt erkannt werden. Wird nun L' von dem Kriterium erkannt, so sind die L die das neue Verfahren findet und davor ordnet, gerade alle die Vorbedingungen, die im relaxierten Planungsgraph notwendig sind um L' zu erreichen. Somit kann ohne Operatoren, die L erreichen, L' nicht im relaxierten Planungsgraphen erreicht werden. Ohne L' ist aber $s_\star$ nicht erreichbar, da L' durch das Kriterium korrekt erkannt wird. Induktiv folgt, dass alle solchen Landmarken durch das hinreichenden Kriterium korrekt erkannt werden.

Trotz dieser Eigenschaft ist das neue Verfahren meist genauso gut wie das alte Verfahren, was die Anzahl an gefunden Landmarken angeht. Im neuen

[6] Die Aussagen gelten ebenso für die Verwendung der Graphsuche, da die Graphsuche höchstens genauso viele Operatoren ausschließen kann wie der relaxierte Planungsgraph.

[7] Das hinreichende Kriterium sollte hierbei nur auf Kandidaten, die nicht in s_0 liegen, angewendet werden.

Verfahren werden eben auch zusätzlich lookahead-notwendigen Ordnungen und Landmarken gefunden. Nicht zu vergessen ist, dass die Ordnungen nun alle korrekt sind.

Durch ein Zusammenlegen der Verfahren könnte die Anzahl an approximierten Landmarken verbessert werden. Innerhalb des neuen Verfahrens ist es nur wichtig, dass Vorbedingungen von Operatoren betrachtet werden, die eine Landmarke erreichen. Man könnte nun zuerst eine Menge von Landmarken finden und für Elemente dieser dann Operatorvorbedingungen und den Domänentransitionsgraphen betrachten, um die Ordnungen zu approximieren. Bei den Operatorvorbedingungen und der Suche im Domänentransitionsgraphen könnten dabei auch neue Landmarken gefunden werden. Würde man zum Finden einer solchen Landmarkenmenge das alte Verfahren verwenden, so könnte sogar die Verifikation der Landmarken mit $\Pi_o(L)$ zusammengelegt werden. Insgesamt würde das neue Verfahren auf diese Weise mindestens alle Landmarken des alten approximieren. Man könnte auch weiter gehen und andere Strategien oder Heuristiken konstruieren, um eventuell mehr Landmarken zu finden.

Hier gibt es nun auch die Möglichkeit, das hinreichende Landmarkenkriterium auf alle Zuweisungen anzuwenden, um eine solche Landmarkenmenge zu generieren. Dieses Vorgehen verallgemeinert sowohl $\Pi_o(L)$ als auch die Suche nach lookahead-notwendigen Ordnungen im Domänengraphen, falls das Planungsproblem keine Operatoren mit bedingten Effekten enthält.[8] Kombiniert man nun die Menge aller Landmarken, die dem hinreichenden Kriterium genügen, mit dem neuen Verfahren, so erhält man eine sehr große Menge an Landmarken und Ordnungen, die auf jeden Fall alle Landmarken enthält, die das alte Verfahren gefunden hätte. Es ist allerdings nicht überraschend, dass diese Kombination gerade bei sehr großen Planungsproblemen ziemlich langsam ist.

5.5.1 Evaluierung der Approximierungsverfahren für Landmarken und zwingende Ordnungen

Um einen Überblick über die Verfahren zu erhalten, hier ein kurze Evaluierung auf ausgewählten Domänen aus den ICAPS-Planungsbenchmarks. Es wird dabei die jeweils größte Probleminstanz aus der zugehörigen Domäne betrachtet. Getestet werden dabei alle fünf Verfahren:

- *Grund*verfahren mit rein notwendige Ordnungen
- *Altes* Verfahren mit relaxiertem Planungsgraphen für greedy-notwendige Ordnungen

[8] Enthält das Planungsproblem solche Operatoren, kann aufgrund des sehr einfachen Umgangs mit bedingten Effekten innerhalb des relaxierten Planungsgraphen nicht davon ausgegangen werden, dass alle Landmarken, die durch die Vorbedingungen und die Suche im Domänengraphen gefunden werden auch dem hinreichenden Kriterium genügen.

- *SAS*$^+$-Verfahren mit reiner Graphsuche für die Einschränkung von O und die lookahead-Ordnungen
- *RPG/SAS*$^+$-Verfahren mit relaxiertem Planungsgraphen für die Einschränkung von O und Graphsuche für lookahead-Ordnungen
- *Kombination* aus RPG/SAS$^+$ und einer vorberechneten Landmarkenmenge, die mit dem hinreichenden Kriterium – angewendet auf alle Zuweisungen – generiert wurde.

Domäne	Grund	Altes	SAS$^+$	RPG/SAS$^+$	Kombination
Blocksworld					
Landmarken	50	67	78	79	86
Ordnungen	64	116	108	197	244
Depot					
Landmarken	79	110	102	127	258
Ordnungen	82	142	118	186	733
Grid					
Landmarken	28	71	52	56	130
Ordnungen	22	52	65	75	89
Logistics00					
Landmarken	21	33	68	68	70
Ordnungen	6	24	136	136	136

Für die nachfolgenden Kapitel soll das in diesem Kapitel vorgestellte RPG/SAS$^+$-Verfahren zum Einsatz kommen.

6 Approximierung von vernünftigen Ordnungen

Mit Hilfe des Landmarkengraphen – der bisher nur zwingende Ordnungen enthält – können nun vernünftige Ordnungen approximiert werden. Die Verfahren hierzu sind wieder aus „Ordered Landmarks in Planning" [HPS04] portiert worden. Es ergeben sich lediglich kleinere Änderungen für die Anpassung an den SAS$^+$-Planungsformalismus.

Nach den Definitionen ist es nötig für vernünftige Ordnungen zwischen Landmarken L, L' die *Aftermath*-Relation von L hinreichend zu approximieren sowie sicherzustellen, dass L' bei dem Erreichen von L, verloren geht. Die *Aftermath*-Relation kann durch zwingende Ordnungen hinreichend approximiert werden. Zu erkennen, ob L' bei dem Erreichen von L verloren geht, erfordert jedoch neue Kriterien. Ein naheliegendes Kriterium ist es auszunutzen, dass die beiden Zuweisungen nicht gleichzeitig in einem Zustand gelten können. Dies ist zwar einleuchtend aber leider nicht einfach zu bestimmen. Solche Fälle werden unter der Definition *Inkonsistenz* zusammengefasst. Eine Inkonsistenz zwischen L, L' ist zwar hinreichend, allerdings aber deutlich

strikter als es die Definition der vernünftigen Ordnungen verlangt. Daher werden Kriterien vorgestellt, die nicht nur direkte Inkonsistenzen der beiden Zuweisungen selbst betrachten, sondern vielmehr auch Inkonsistenzen in Operatorvorbedingungen und Effekten, ausnutzen – die sogenannten *Interferenzen*. Mittels der Interferenzen wird dann hinreichend approximiert, ob L bei dem Erreichen von L' verloren geht.

6.1 Definition Inkonsistenz

Zwei Zuweisungen sind inkonsistent, wenn es von s_0 aus keinen erreichbaren Zustand gibt, in dem sie gleichzeitig gelten.

6.1.1 Approximierung von Inkonsistenzen

In „Ordered Landmarks in Planning" [HPS04] wurde hierfür die TIM-API [FL98] verwendet. Da der SAS$^+$-Planungsformalismus gegenüber STRIPS gerade versucht solche Inkonsistenzen bereits in der Formalisierung kenntlich zu machen, soll hier nun die Inkonsistenz direkt über den Formalismus approximiert werden. Es wird benutzt:

Sind $\{v \mapsto d\}, \{v' \mapsto d'\}$ zwei Zuweisungen mit $v = v', d \neq d'$, so sind $\{v \mapsto d\}, \{v' \mapsto d'\}$ inkonsistent.

Dadurch ergibt sich leider eine starke Abhängigkeit von der konkreten Formalisierung eines Planungsproblemes. Die in dieser Arbeit betrachteten Planungsprobleme liegen alle in PDDL2.2 vor und werden vom Downward-Planungssystem nach SAS$^+$ übersetzt. Bei der Übersetzung wird darauf geachtet, die SAS$^+$-Formalisierung möglichst kompakt zu halten. Dies bedeutet, dass in der Formalisierung die ursprünglichen Zustandsvariablen je nur in eine SAS$^+$-Domäne einfließen und nicht mehrfach verwendet werden.

Besonders deutlich wird dies bei der Blockswelt-Domäne. Die Kodierung drückt hier nicht aus, dass ein Block, auf dem ein weiterer platziert ist, nicht in der Hand gehalten werden kann. Im Endeffekt werden somit leider die Zielordnungen für die Reihenfolge des Turmbaus nicht gefunden werden. Dies wird nach der Erweiterung dieser Definition auf die Interferenz deutlich.

6.2 Definition Interferenz

Die Zuweisung $\{v \mapsto d\}$ interferiert mit der Zuweisung $\{v' \mapsto d'\}$, falls:

1. $\{v \mapsto d\}, \{v' \mapsto d'\}$ inkonsistent sind.
2. $\exists \{v'' \mapsto d''\} \in \bigcap_{\{o \in \mathcal{O} | \{v \mapsto d\} \subseteq \mathit{eff}_o\}} \mathit{eff}_o : \{v' \mapsto d'\}, \{v'' \mapsto d''\}$ inkonsistent.

   Beim Erreichen von $\{v \mapsto d\}$ wird auch $\{v'' \mapsto d''\}$ erreicht, welches mit $\{v' \mapsto d'\}$ inkonsistent ist. Um Planungsprobleme mit bedingten

Effekten korrekt zu behandeln, kann man sich bei $\{v'' \mapsto d''\}$ auf unbedingte Effekte beschränken.

3. $\exists\{v'' \mapsto d''\} \rightarrow_{gn} \{v \mapsto d\} : \{v' \mapsto d'\}, \{v'' \mapsto d''\}$ inkonsistent.

   Bei der späteren Verwendung dieser Definition, handelt es sich bei $\{v \mapsto d\}$, $\{v' \mapsto d'\}$, $\{v'' \mapsto d''\}$ um Landmarken L, L', L'' und es wird von einem Zustand $s \in S_{(L', \neg L)}$ ausgegangen. In s wurde L' nach Definition gerade zum ersten mal erreicht. Wegen der Inkonsistenz zwischen L' und L'' kann L'' in diesem Zustand nicht gelten. Jede Lösung muss nun L' zerstören, um L'' zu erreichen, was nötig ist um L zu erhalten.

Mit der simplen Inkonsistenz gilt für die Fälle $1.-3.$: $v = v'$ oder $(v, v') \in CG(\Pi)$. In der Tat, bedingt 1. die Gleichheit und $2., 3.$ $(v, v') \in CG(\Pi)$. Für Fall 2. kommt noch $(v', v) \in CG(\Pi)$ hinzu, da die gemeinsame Vorbedingung inkonsistent zu v' sein soll. Dies lässt sich als Ausschlusskriterium verwenden und erspart häufig die Schnitte über die Operatoren und die Betrachtung von davor geordneten Landmarken.

Für die Blockswelt ergibt sich nun folgendes Bild: Angenommen, das Ziel ist es drei Blöcke, A, B, C aufeinander zu stapeln – A auf B, B auf C. Um die Ordnung $\{B_{\text{auf}} \mapsto C\} \rightarrow_r \{A_{\text{auf}} \mapsto B\}$ zu erhalten wäre es auf jeden Fall nötig zu erkennen, dass $\{A_{\text{auf}} \mapsto B\}$ zerstört werden muss um $\{B_{\text{auf}} \mapsto C\}$ zu erreichen. Dies wäre gerade die Interferenz von $\{B_{\text{auf}} \mapsto C\}$ mit $\{A_{\text{auf}} \mapsto B\}$. Genau diese gilt nun aber aufgrund der Kodierung mit diesen Definitionen nicht. Eigentlich sollte der Punkt 3. zutreffen, denn Voraussetzung für das Stapeln von B auf C ist es, dass B in der Hand liegt. Die Kodierung beschreibt diesen Sachverhalt allerdings nicht mit einer gemeinsamen Variable, womit die Interferenz nicht festgestellt wird.

6.3 Approximierung der *Aftermath*-Relation

Seien neben dem Planungsproblem Π zwei Landmarken L, L' gegeben.
$L' \in \mathit{Aftermath}(L)$ wenn eine der folgenden beiden Bedingungen gilt:

- $L' \subseteq s_\star$
- $\exists$ Landmarke $L'' \neq L' : L \rightarrow_{ln} L''$ und $L' \rightarrow_{gn} L''$

  Geht man von einem Zustand $s \in S_{(L', \neg L)}$ aus, so kann L'' noch nie erreicht worden sein, da L nie galt. Alle Lösungen von diesem Zustand aus müssen noch L'' erreichen, also auch vorher noch L. Ist L erreicht worden, so muss L' exakt vor dem ersten Erreichen von L'' gelten. Daraus folgt: L, L' gelten in einem Zustand gleichzeitig oder L' gilt nachdem L erreicht wurde.

  Diese Ordnungen können im Landmarkengraph auch nur implizit gegeben sein. So kann die greedy-notwendige Ordnung auch notwendig

sein und die lookahead-notwendige Ordnung eine Kette von zwingenden Ordnungen darstellen.

6.4 Approximierung von vernünftigen Ordnungen

Kombiniert man nun die Approximierung der *Aftermath*-Relation mit der Definition der Interferenz, so erhält man ein hinreichendes Kriterium für eine vernünftige Ordnung.

6.4.1 Pseudocode für die Approximierung von Vernünftige Ordnungen

Sei $LG := (LG_V, LG_E) = LG(\Pi)$.

```
for each L' ∈ LG_V
   if L' ⊆ s_⋆
      for each L ∈ LG_V, L ≠ L'
         if L interferiert mit L'
            LG_E := LG_E ∪ (L →_r L')
   else if L' ⊈ s_0
      for each C : (L' →_gn C) ∈ LG_E
         for each P : (P →_ln C) ∈ LG_E, P ≠ L'
            if P interferiert mit L'
               LG_E := LG_E ∪ (P →_r L')
```

Eine Implementierung sollte hierbei auch implizierte Ordnungen verwenden.

6.5 Approximierung von folgenden vernünftige Ordnungen

Die folgenden vernünftigen Ordnungen sind nach ihrer Definition gerade dann vernünftig, wenn eine Menge von bisherigen Ordnungen eingehalten wird. Hierzu wird anstatt der *Aftermath*-Relation die *Aftermath*^{O} Relation verwendet. Bei dieser ändert sich lediglich die Bedingung:

$$\exists \text{ Landmarke } L'' \neq L' : L \rightarrow_{ln} L'' \text{ und } L' \rightarrow_{gn} L''$$

Hier werden nun anstatt der $L \rightarrow_{ln} L''$-Ordnung, Ketten von Ordnungen

$$L = L_1 \rightarrow \cdots \rightarrow L_n \rightarrow L''$$

erlaubt, wobei $n \geq 1$ und $L_n \neq L'$ sein muss. Diese Ordnung sind entweder zwingende Ordnungen oder vernünftige Ordnungen in der Menge O.

Es kann fast der gleiche Pseudocode wiederverwendet werden. Nun kann allerdings der Fall $L' \subseteq s_\star$ weggelassen werden, da hier nur die Ordnungen gefunden werden, die bereits als vernünftig eingefügt wurden. Ansonsten muss die Ordnung $(P \rightarrow_{ln} C)$ auf die Ketten aus der *Aftermath*^{O}-Relation erweitert werden und in der letzten Zeile sollten folgende vernünftige Ordnungen eingefügt werden.

6.5.1 Zyklen im Landmarkengraphen

Da alle Kanten im Landmarkengraphen als zeitliche Ordnungen angesehen werden, wäre es gut, wenn diese keine Zykel enthalten würden. Einige der Verwendungsmöglichkeiten der Landmarken und Ordnungen während der Planung interessieren sich für die jeweils als nächstes zu erreichenden Landmarken. Sind nun Zykel enthalten, so könnte keine Landmarke existieren, die als nächste gesehen werden kann. Um zu garantieren, dass keine solche Landmarke existiert, genügt es den Landmarkengraphen $LG(\Pi)$ azyklisch zu machen. Die Bestimmung einer minimalen Teilmenge der Ordnungen um $LG(\Pi)$ azyklisch zu machen ist jedoch NP-vollständig (*Feedback Arc Set* – [GJ79]). Deswegen wird hier ähnlich wie in „Ordered Landmarks in Planning" [HPS04] mit einem greedy-Verfahren vorgegangen. Es werden Kanten identifiziert, die an einem Zykel beteiligt sind. Unter diesen findet sich mindestens eine vernünftige Ordnung, da zwingende Ordnungen keine Kanten bilden können. Nun werden solange Kanten entfernt, bis der Zykel gebrochen ist. Dabei werden zuerst folgende vernünftige Ordnungen aufgrund ihrer schwächeren Bedeutung entfernt. Nach „Ordered Landmarks in Planning" [HPS04] bringen bessere Verfahren zum Auffinden von Zyklen kaum bessere Ergebnisse.

Nachdem der Graph nun azyklisch ist, wird zwar garantiert, dass immer eine Landmarke als nächste identifiziert werden kann, jedoch ist bei allen vernünftigen Ordnungen nicht gesichert, ob die beteiligten Landmarken tatsächlich in dieser zeitlichen Reihenfolge erreicht werden können. Es ist außerdem möglich, wegen des Befolgens einer vernünftigen Ordnung das Planungsproblem nicht mehr lösen zu können. Aus diesem Grund sollten die vernünftigen Ordnungen generell nicht erzwungen, sondern eher vorgeschlagen werden.

7 Nutzung von Landmarken bei der Planung

Nachdem nun zu einem SAS$^+$ Planungsproblem Π der Landmarkengraph $LG(\Pi)$ gebildet wurde, stellt sich die Frage, wie die Landmarken und Ordnungen sinnvoll während der Planung, mit dem Downward-Planungssystem, verwendet werden können. Neben dem Vorgehen aus „Ordered Landmarks in Planning" [HPS04] sollen auch einige Alternativen untersucht werden.

7.1 Das Downward-Planungssystem

Wie bereits erwähnt wurde, übernimmt Downward die Übersetzung der Probleme nach SAS$^+$. Für die Lösung der Planungsprobleme selbst kommt eine Vorwärtssuche zum Einsatz. Interessant ist hierbei die Unterstützung von mehreren Heuristiken gleichzeitig. Dieser Multiheuristikbetrieb wird als Bestensuche folgendermaßen implementiert:

- Jede Heuristik besitzt eine eigene Liste von offenen Zuständen.
- Der beste Zustand aus einer alternierend gewählten Liste wird expandiert und von allen Heuristiken bewertet. Dabei werden die Nachfolger in alle Listen mit ihrem jeweils zugehörigen heuristischen Wert eingefügt.

Somit müssen die Werte der Heuristiken nicht miteinander vergleichbar sein. Als wichtigste Heuristiken sind die *Fast-Forward-* und die *Causalgraph-*Heuristik vorhanden.

Ein weiterer Punkt ist die Verwendung von *preferred operators*. Heuristiken können bei der Bewertung von Zustände eine Menge von anwendbaren Operatoren als bevorzugt markieren. Die nachfolge Zustände, die mit bevorzugten Operatoren entstehen, sollen dann eher betrachtet werden, als alle anderen möglichen Nachfolgezustände. Für genauere Details der hier angesprochenen Eigenschaften, siehe „The Fast Downward Planning System" [Hel06].

7.2 Bisherige Verwendung

In „Ordered Landmarks in Planning" [HPS04] wurde ein Verfahren vorgeschlagen und getestet, das eine etappenweise Planung vornimmt. Die Idee ist es, stückweise zwischen Landmarken zu planen, anstatt direkt das Planungsproblem mit dem eigentlichen Ziel zu lösen. Hierbei wird folgendermaßen iterativ vorgegangen:

Berechne $LG(\Pi)$
Entferne alle Landmarken mit ihren Ordnungen aus $LG(\Pi)$, die im Initialzustand sind.
$P := \langle\rangle$
`while` $LG(\Pi)$ nicht leer
    Ersetze das Ziel mit einer Disjunktion der Wurzeln von $LG(\Pi)$
    Generiere Plan P'
    `if` kein Plan P' existiert
        Gib auf
    Entferne alle während der Ausführung des Plans und im Endzustand erreichten Wurzeln aus $LG(\Pi)$
    $P := \langle P, P' \rangle$
    Ersetze den Initialzustand durch den Endzustand des Planes P'
Löse das eigentliche Planungsproblem, mit aktuellen Initialzustand.
Gibt es dafür einen Plan P', dann ist $P := \langle P, P' \rangle$ das Ergebnis.

Es wird also versucht, sukzessive einen Plan zu generieren, der alle Landmarken und am Ende das eigentliche Ziel erreicht. Geplant wird immer nur bis zu den „nächsten" Landmarken – die aktuellen Wurzeln von $LG(\Pi)$. Der

Plan wird dann soweit festgelegt und weiter von diesem Endzustand aus geplant. Sind alle Landmarken erreicht worden, so ist nicht garantiert, dass alle Ziele in einem Zustand gleichzeitig erreicht wurden. Hier wird dann einfach weiter auf das eigentliche Ziel geplant. Für eine genaue Analyse der Probleme und Eigenschaften dieses Verfahrens siehe original Arbeit [HPS04]. Hier sollen nur einige Punkte festgehalten werden:

- Während der Planung auf die Wurzeln wird das eigentliche Ziel des Planungsproblemes ausgeblendet. Es ist jedoch generell vernünftig anzunehmen, dass man sich durch die Landmarken dem eigentlichen Ziel nähert.
- Die Endzustände der Etappen werden als neuer Initialzustand beibehalten. Diese werden also als gute Zustände angesehen.
- Die Planung der Etappen ist weitestgehend unabhängig voneinander. Dies bedeutet sowohl, dass Zustände mehrfach erreicht werden können, als auch, dass in Etappenplänen bereits erreichte Landmarken verloren gehen können. In letzterem Falle kann sich dies negativ auswirken, falls diese Landmarken noch benötigt werden.
- Es wird aufgegeben, falls kein Plan für eine Etappe existiert. Hier wäre zwar Backtracking möglich, jedoch ist unklar wie weit man zurück gehen müsste. In „Ordered Landmarks in Planning" [HPS04] wurden einige Strategien hierzu besprochen, implementiert wurde nur das einfache obige Verfahren.

Als Alternative zu diesem Vorgehen wäre es nun interessant, von dieser Etappenplanung auf eine normale, globale Planung zum eigentlichen Planungsziel überzugehen, ohne dabei auf die Informationen aus Landmarken und Ordnungen zu verzichten.

7.3 Landmarken-Heuristik

Aufgrund der Multiheuristik-Unterstützung ergibt sich die einfache Möglichkeit, Landmarken mit Hilfe einer Heuristik für die Planung einzusetzen. Wenn auf die Wurzeln geplant wird, ergibt sich automatisch durch die Heuristiken eine Abschätzung für den Abstand zu den aktuellen Wurzeln, aber nicht zum eigentlichen Planungsziel. Die Abstände sind sowohl vom aktuellen Zustand als auch den aktuellen Wurzeln abhängig und somit nicht für die normale Planung zu gebrauchen. Sie stellen sozusagen nur lokale Informationen dar. Eine Alternative dazu ist, aus der Anzahl von abgearbeiteten Landmarken eine normale Heuristik zu konstruieren, die damit nicht lokal arbeitet, sondern global.

Für eine Abschätzung der Entfernung des aktuellen Zustandes zum Ziel, ist die Anzahl der erreichten Landmarken im Zustand selbst weniger nützlich, da deren Anzahl stark zwischen Zuständen schwankt. Deswegen soll

vielmehr der bisherige Teilplan beurteilt werden – also alle bisher auf diesem Teilplan erreichten Landmarken aus $LG(\Pi)$. Als heuristischer Wert könnte nun die Anzahl an nicht erreichten Landmarken benutzt werden. Mit diesem Vorgehen entsprechen allerdings Zustände, mit dem heuristischen Wert 0 nicht unbedingt den erwünschten Zielzuständen. Der Wert 0 impliziert lediglich alle Landmarken innerhalb des Planes und nicht die Zielzustände im aktuellen Zustand erreicht zu haben. Ein expliziter Zieltest könnte dies beheben, jedoch hätte diese Heuristik ein weiteres Problem, denn sie wäre monoton. Nachfolgezustände könnten nie schlechtere Werte bekommen. Demnach würde die Heuristik nicht erkennen ob man sich vom Ziel, bzw. neuen Landmarken wegbewegt.

Wegen der oben genannten Probleme soll eine nicht monotone Variante dieser Heuristik konstruiert werden. Es wird wieder der Teilplan bewertet, allerdings wird auch der aktuelle Zustand verwendet, um herauszufinden, welche bekannten und erreichten Landmarken wieder verloren wurden. Die Idee ist nun, als heuristischen Wert die Anzahl an bekannten Landmarken zurückzugeben, die noch erreicht werden müssen. Dies sind eben nicht nur solche Landmarken, die noch nie erreicht wurden, sondern vielmehr auch einige verlorene Landmarken. Nach Definition müssen Landmarken nur mindestens einmal in jeder Lösung vorkommen. Insofern benötigt man einige Kriterien, nach denen gesichert ist, dass eine Landmarke, die bereits einmal vorkam, nochmals auftreten muss:

Eine Landmarke, die erreicht wurde, aber im aktuellen Zustand nicht mehr gilt, tritt nochmals auf, falls:

- sie ein Teil der Zielbedingung ist.
- sie (mindestens) greedy-notwendig vor einer bisher noch nicht erreichten Landmarke geordnet ist.[9]

Der heuristische Wert entspricht nun also der Anzahl noch zu erreichender Landmarken aus dem Landmarkengraphen, womit die Heuristik nicht monoton und ein expliziter Zieltest nicht nötig ist. Problematisch ist allerdings, dass der heuristische Wert über gewisse Strecken gleich bleibt, was in solchen Fällen eher zu einer Breitensuche führt.

Ein weiteres Problem bleibt: vernünftige Ordnungen werden ignoriert. Um dies zu beheben könnte man den heuristischen Wert für jede missachtete vernünftige Ordnung erhöhen. Auf diese Weise würde man den Planer für das Nicht-Einhalten bestrafen. Allerdings ist der Unterschied in der Praxis nur

[9] Alle vorgestellten Approximierungsalgorithmen für Landmarken und zwingende Ordnungen, die greedy-notwendige Ordnungen finden, prüfen bei diesen nicht nach, ob es sich auch um echte notwendige Ordnungen handelt. Würden sinnvolle notwendige Ordnungen vorliegen, so könnte man hier auch ein Kriterium einfügen, welches in solchen Fällen notwendige Vorbedingungen einer solchen Landmarke rekursiv weiter verfolgt.

marginal, zumal es mit den *preferred operators* eine elegantere Möglichkeit gibt, dem Planer Ordnungen vorzuschlagen.

7.3.1 Preferred Operators

Ähnlich zur Planung in Etappen sollen *preferred operators* verwendet werden um möglichst schnell weitere Landmarken zu erreichen. Interessant ist dabei, dass auf diese Weise auch entschieden werden kann, welche Landmarke als nächstes erreicht werden sollte. Damit kann der Planer indirekt auf vernünftige Ordnungen hingewiesen werden.

Es soll wieder auf die Wurzeln von $LG(\Pi)$ geplant werden, und damit das Verhalten der Etappensuche möglichst nachgebildet werden. Da die Heuristik selbst nun allerdings nicht von den Abständen zu den Landmarken profitiert, müssen diesmal nur *preferred operators* gefunden werden, die zu einem Blatt führen sollen. Dazu wird nun eine relaxierte Planung mittels der Fast Forward Heuristik auf das disjunktive Ziel aus Landmarken durchgeführt, um daraus die *preferred operators* zu gewinnen. Sind bereits alle Landmarken erreicht worden, so sollen *preferred operators* auf das Ziel hin gefunden werden. Dabei kommt jedoch eine kleine Änderung mit zum Einsatz. Bei dem disjunktiven Ziel wird auf eines der „nahesten" Blätter geplant,[10] ohne Betrachtung der anderen anderen Möglichkeiten. Für die weiter entfernten Blätter ist dieses Vorgehen sicher akzeptabel, da es sich um eine relaxierte Betrachtung handelt. Sind Blätter jedoch in einem Nachfolgezustand erreichbar, so kann hier etwas genauer differenziert werden. In solchen Fällen soll das Blatt erreicht werden, das die längste Kette von Ordnungen $L = L_1 \rightarrow L_2 \rightarrow \cdots \rightarrow L_n \subseteq s_\star$ im Landmarkengraphen besitzt. Dieses Vorgehen macht nur dann einen Unterschied, wenn tatsächlich mehrere verschiedene Blätter in einem Nachfolgezustand erreichbar sind. Solche Fälle treten allerdings recht häufig in blocksworld-, freecell- und den beiden pipesworld-Domänen auf. Dort werden durch dieses Vorgehen mehr Probleme gelöst.

7.4 Bestensuche einschränken

Nachdem nun die Idee auf Wurzeln des Landmarkengraphen zu planen als Heuristik umgesetzt wurde, kann man sich überlegen, wie ein weiterer Aspekt der Etappenplanung umgesetzt werden kann. Die Etappenplanung geht davon aus, dass nach jeder Etappe ein guter Zustand erreicht wurde, von dem an dann weiter geplant wird. Dieses Konzept lässt sich auch auf die Bestensuche anwenden.

Die Landmarken-Heuristik wird dazu verwendet, den aktuellen Zustand, der

[10] Bei Nutzung der Fast Forward Heuristik entspricht dies dem Blatt mit dem geringsten h_{add} Wert.

gerade aus der Liste von offenen Zuständen ausgewählt wurde, zu bewerten. Ist der Wert besser als alle bisher gesehenen Werte, so wird die Liste von offenen Zuständen geleert und dann die möglichen Nachfolgezustände eingefügt. Der Planer wird somit gezwungen, von diesem Zustand aus weiter zu planen.

Ein wichtiger Punkt hierbei ist das Beibehalten der Liste von bereits betrachteten Zuständen. Damit soll eine Erhöhung der Planlängen möglichst vermieden werden. In der Praxis führt dieses Vorgehen jedoch häufig in Sackgassen. Aus diesem Grund wurde hier noch ein Stack hinzugefügt, der die alten Listen mit offenen Zuständen sowie die alte Anzahl an offenen Landmarken enthält und somit ein Backtracking ermöglicht.

7.5 Landmarken und Ordnungen explizit in Task einbringen

Neben der Idee, den alten Ansatz global anzuwenden, gibt es noch die Möglichkeit, die Landmarken und Ordnungen explizit in die Kodierung des SAS$^+$-Planungsproblems einzufügen. Auf diese Weise ist das Vorgehen unabhängig vom verwendeten Planungssystem und dem genauen Vorgehen bei der Planung. Die Landmarken und Ordnungen stellen hierbei eine Art Überlagerung dar, die dem Planer eine gewissen Reihenfolge vermitteln soll (siehe Abb. 7).

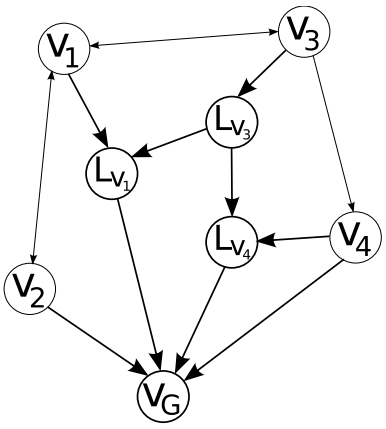


Abbildung 7: Kausalgraph mit Überlagerung aus Landmarken und Ordnungen.

Um dies zu erreichen, wird der Landmarkengraph $LG(\Pi)$ zu dem Kausalgraphen hinzugenommen und mit den ursprünglichen Variablen verbunden. Für jede Landmarke, die nicht im Initialzustand gilt, wird dazu eine neue Variable – eine sogenannte Landmarkenvariable – eingefügt, die angibt, ob die entsprechende Landmarke bereits erreicht wurde. Dazu gibt es dann passende Operatoren, die solche Landmarkenvariablen als erreicht markieren

können, falls die Landmarke selbst, und die davor geordnete Landmarkenvariablen bereits erreicht wurden. Die Landmarken im Ziel werden zusätzlich zu einer Variable zusammengefasst, die nur erreicht werden kann, wenn das eigentlich Planungsziel und deren Landmarkenvariablen erreicht wurden. Dies geschieht auf folgende Weise:

Bestimme $LG(\Pi)$
Entferne alle Landmarken im Initialzustand samt ihren Ordnungen
$\Pi^* = \langle \mathcal{V}^*, \mathcal{O}^*, s_0^*, s_\star^* \rangle$
$\Pi^* := \Pi$
`for each` $L' \in LG_V$
$\quad \mathcal{V}^* := \mathcal{V}^* \cup \{v_{L'}\}, \mathcal{D}_{v_{L'}} = \{T, F\}$
$\quad s_0^* := s_0^* \cup \{v_{L'} \mapsto F\}$
$\quad \mathcal{O}^* := \mathcal{O}^* \cup \{\langle \{v_L \mapsto T | \exists L \in LG(\Pi) : L \to L' \in LG(\Pi)\} \cup L',$
$\qquad \{v_{L'} \mapsto T\} \rangle\}$
$\mathcal{V}^* := \mathcal{V}^* \cup \{v_G\}, \mathcal{D}_{v_G} = \{T, F\}$
$\mathcal{O}^* := \mathcal{O}^* \cup \{\langle \{v_L | L \in s_\star\} \cup s_\star, \{v_G \mapsto T\} \rangle\}$
$s_\star^* := v_G$

Bei dem Setzen der Vorbedingungen der neuen Operatoren, taucht die Frage auf, welche Ordnungen benutzt werden sollten. Generell werden Ordnungen hier nicht unmittelbar erzwungen. Erzwungen wird lediglich, dass die Landmarkenvariablen in dieser Reihenfolge geschaltet werden müssen. Insofern gibt es hier die zwei sinnvollen Möglichkeiten, entweder nur zwingende Ordnungen oder alle Ordnungen zu verwenden. Bei der Verwendung von allen Ordnungen zeigt sich jedoch ein deutlich schlechteres Ergebnis. Da dabei außerdem in dem meisten Fällen die Planlänge nicht sinkt, wird hier auf eine weitere Betrachtung verzichtet und es werden nur zwingende Ordnungen für die Operatorvorbedingungen verwendet.

Eine Integrierung der Landmarkenvariablen in die bereits existierenden Operatoren brachte eine Verschlechterung, was eventuell darauf zurückzuführen ist, dass damit die Landmarkenvariablen im Kausalgraphen in beide Richtungen mit normalen Variablen verbunden sind.

Außerdem ist Vorsicht bei den Vorbedingungen der neuen Operatoren geboten. Die meisten Heuristiken treffen gewisse Unabhängigkeitsannahmen für Vorbedingungen, die hier nicht gegeben sind – die Landmarkenvariablen sind keine echten eigenständigen Variablen, die viel Aufwand zum Erreichen benötigen, sondern vielmehr Markierungen, die zu den entsprechenden Zeitpunkten gesetzt werden müssen. Dies führt zu höheren Abschätzungen. Insofern sollten die Vorbedingung nicht zu viele Landmarkenvariablen enthalten. Für die bekannten Approximierungsverfahren gilt, dass der Landmarkengraph nicht alle Ordnungen explizit enthält, somit sind die Vorbedingungen nach obigem Vorgehen noch nicht zu groß.

Am Ende der Planung werden die neuen Operatoren aus dem resultie-

renden Plan entfernt, um einen gültigen Plan für das eigentliche Planungsproblem zu erhalten.

8 Testergebnisse mit Landmarken

Zum Abschluss nun eine Evaluierung der verschiedenen Nutzungsmöglichkeiten. Da sich sehr viele Kombinationsmöglichkeiten ergeben, wird nur eine Auswahl auf einigen ICAPS-Planungsbenchmark-Domänen betrachtet. Getestet werden (jeweils mit *preferred operators*):

- Causalgraph- und Fast-Forward-Heuristik gleichzeitig – als Vergleich (STD)
- Etappenplanung – wegen einfacherer Implementierung lediglich mit der Fast Forward Heuristik (E)
- Landmarken-Heuristik (LH)
- Landmarken-Heuristik mit Causalgraph- und Fast-Forward-Heuristik zusammen (LH+STD)
- Causalgraph und Fast Forward Heuristik mit der eingeschränkten Bestensuche (eSTD)
- Modifiziertes Problem mit Landmarken und Ordnungen im Kausalgraph und der Causalgraph-Heuristik[11] (MP)

Die Tests wurden auf einem Xeon 3 GHz mit 3GB RAM und einem Zeitlimit von 300 Sekunden durchgeführt. Die Übersetzungszeiten von PDDL2.2 nach SAS^+ wurden nicht mit eingerechnet.

8.1 Übersicht

Anzahl an ungelösten Problemen: [12]

Domäne (# Probleme)	STD	E	LH	LH+STD	eSTD	MP
blocksworld (35)	0	0	0	0	0	0
depot (22)	4	1	4	4	4	15
freecell (80)	3	23	4	4	3	5
grid (5)	0	0	0	0	0	1
mystery (19)	0	4	5	0	1	2
pathways (30)	0	5	4	1	0	23
pipesworld-notankage (50)	6	11	9	6	7	31
pipesworld-tankage (50)	17	27	28	17	15	37
schedule (150)	1	67	11	0	0	3
Gesamt	**31**	**138**	**65**	**32**	**30**	**117**

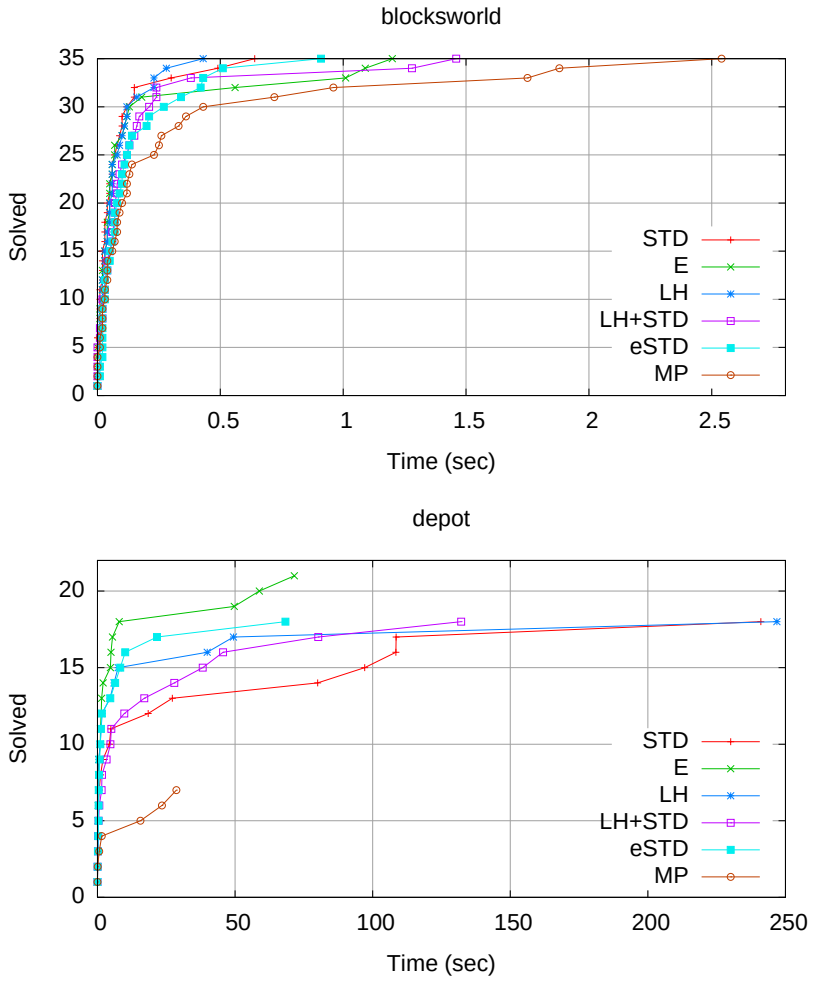
[11] Die Fast-Forward-Heuristik würde nicht von der neuen Kausalgraph-Struktur profitieren.

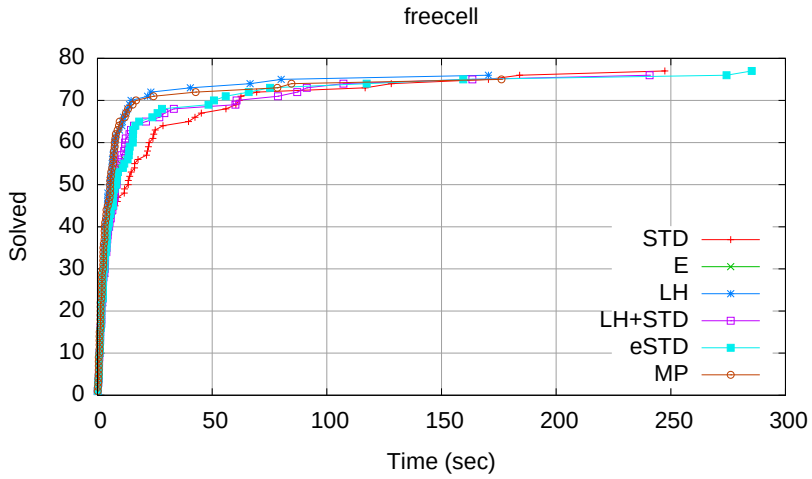
[12] Bei mystery sind nur die lösbaren Probleme angegeben.

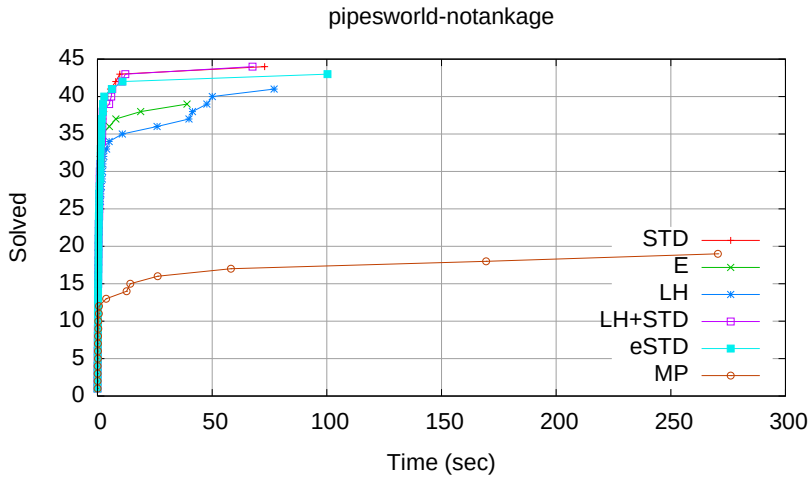
Die Planlängen variieren dabei nur minimal zwischen den verschiedenen Möglichkeiten. Generell sind die Planlängen bei der eingeschränkten Bestensuche sowie bei Verwendung des modifizierten Problemes leicht erhöht. Bei Verwendung der Landmarken-Heuristik hingegen verringern sich die Planlängen meistens.

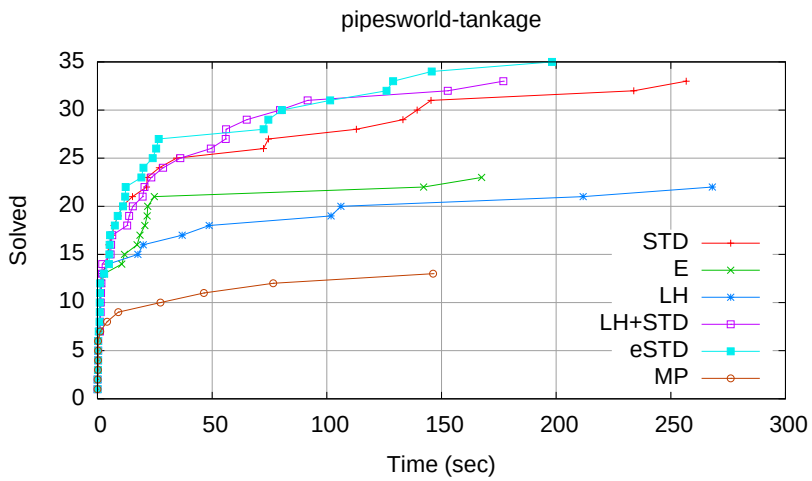
Weiter ist anzumerken, dass die Causalgraph-Heuristik auf den ursprünglichen, nicht modifizierten Problemen 128 nicht löst. Durch die Modifizierung der Probleme, fällt dies hier zwar auf 117, doch der Unterschied wird durch die Freecell Domäne stark zugunsten des modifizierten Problems verzerrt – auf den ursprünglichen, nicht modifizierten Freecell-Problemen bleiben 23 ungelöst, mit der Modifizierung nur 5.

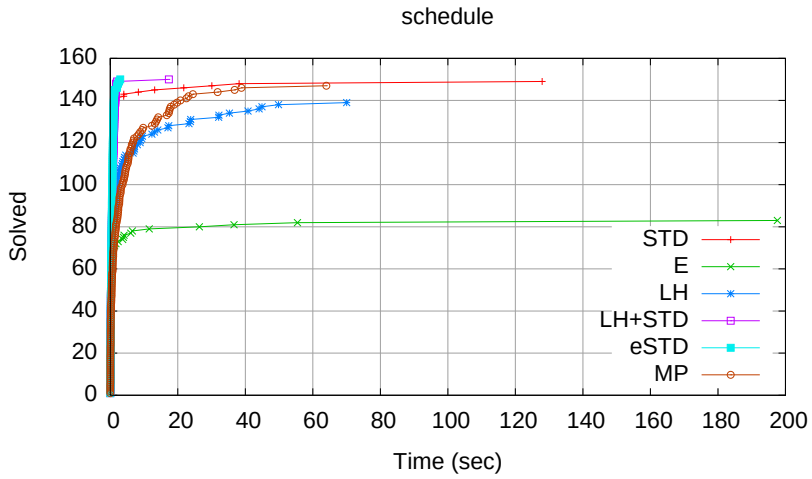
8.2 Plots zu ausgewählten Domänen




freecell
Solved
Time (sec)
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
STD
E
LH
LH+STD
eSTD
MP


pipesworld-notankage
Solved
Time (sec)
0
5
10
15
20
25
30
35
40
45
0
50
100
150
200
250
300
STD
E
LH
LH+STD
eSTD
MP


pipesworld-tankage
Solved
Time (sec)
0
5
10
15
20
25
30
35
0
50
100
150
200
250
300
STD
E
LH
LH+STD
eSTD
MP



8.3 Fazit

Bei den hier untersuchten Domänen zeigt sich, dass ein Vorgehen mit einer Landmarken-Heuristik und *preferred operators* im Vergleich zu einer reinen Etappenplanung durchaus gute Ergebnisse erzielen kann. Die Etappenplanung hat Schwierigkeiten mit der Frecell- und Schedule-Domäne, wohingegen die Landmarken-Heuristik bei diesen weniger Schwierigkeiten hat, dafür aber in meisten anderen Domänen schlechtere Planungszeiten erzielt.

Das Modifizieren des Kausalgraphen bringt in einigen Domänen Verbesserungen mit sich (Blockswelt), jedoch verschlechtern sich die Ergebnisse in anderen (Depot). Womöglich ist der Kausalgraph in solchen Domänen stärker verbunden und die zusätzlichen Knoten und Kanten verwirren die Heuristik.

Eine Einschränkung der Bestensuche bietet eine interessante Möglichkeit, die Planung in einigen Domänen zu beschleunigen. Hier würde man erwarten, dass die Planung auf einigen Probleminstanzen schneller abläuft, bei einigen anderen jedoch deutlich langsamer – je nachdem, ob die Einschränkungen der Liste von offenen Zuständen, an wirklich guten Zuständen erfolgt. Es zeigt sich hier erstaunlicherweise kein größerer Zeitverlust. Dies deutet darauf hin, dass nicht allzu viele Zustände auftreten, bei denen mehr Landmarken erreicht wurden, aber diese Zustände eher vom Ziel wegführen.

Literatur

[BN95] Bäckström, Christer und Bernhard Nebel: *Complexity Results for SAS$^+$ Planning.* Computational Intelligence, 11(4):625–655, 1995.

[FL98] Fox, Maria und Derek Long: *The Automatic Inference of State Invariants in TIM.* Journal of Artificial Intelligence Research, 9:367–421, 1998.

[GJ79] Garey, M. R. und D. S. Johnson: *Computers and Intractability – A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA, 1979.

[Hel06] Helmert, Malte: *The Fast Downward Planning System.* Journal of Artificial Intelligence Research, 26:191–246, 2006.

[HK00] Hoffmann, Jörg und Jana Koehler: *On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm.* Journal of Artificial Intelligence Research, 12:338–386, 2000.

[HPS04] Hoffmann, Jörg, Julie Porteous und Laura Sebastia: *Ordered Landmarks in Planning.* Journal of Artificial Intelligence Research, 22:215–278, 2004.