Albert-Ludwigs-Universität Freiburg
Fakultät für Angewandte Wissenschaften
Institut für Informatik
Lehrstuhl für Grundlagen der Künstlichen Intelligenz

Diplomarbeit

Zielauswahl für teilerfüllende Handlungsplanung

Benjamin Lempp

9. März 2007

Betreut von
Dipl.-Inf. Robert Mattmüller und Dr. Malte Helmert

Zusammenfassung

Die klassische Handlungsplanung mit einem fest vorgegebenen Ziel ist ein vielfältig untersuchter Bereich der Künstlichen Intelligenz. Es gibt jedoch zahlreiche Probleme, in denen die Zielmenge mit gegebenen Ressourcen nicht vollständig erfüllt werden kann. Teilerfüllende Handlungsplanung wählt aus der Menge der Ziele eine Teilmenge aus, die unter den gegebenen Umständen erreichbar ist. Für jede Teilzielmenge kann mit den Mitteln der klassischen Planung nach Plänen gesucht werden, die diese erfüllen. Ermittelt man die Plankosten für eine Auswahl von Teilzielen, lassen sich unter Umständen Schlüsse über die Kosten anderer Zielmengen ziehen. Das Konzept des Nettonutzens bietet die Möglichkeit flexibel zwischen Kosten und Nutzen von Plänen bzw. Zielen abzuwägen. Diese Arbeit untersucht Verfahren, wie mit Hilfe einer durch klassische Planungen erzeugten Basis Teilzielmengen mit hohem Nettonutzen identifiziert werden können.

Danksagung

Ich möchte mich an dieser Stelle bei allen bedanken, die zum Gelingen dieser Arbeit beigetragen haben:

Ganz besonders danke ich Robert Mattmüller und Malte Helmert für die hervorragende Betreuung dieser Arbeit und das Beisteuern vieler guter Ideen und Vorschläge. Herr Prof. Bernhard Nebel danke ich für die Möglichkeit, diese Diplomarbeit an seinem Lehrstuhl anzufertigen und die Bereitschaft, als Gutachter zur Verfügung zu stehen.
Für alles andere danke ich meiner Familie.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel erstellt habe. Alle Stellen, die wörtlich oder sinngemäß aus öffentlichen Schriften übernommen wurden, sind als solche kenntlich gemacht. Diese Diplomarbeit wurde nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt.

Freiburg, den 9. März 2007 Benjamin Lempp

Inhaltsverzeichnis

1 Einleitung

1.1 Handlungsplanung

Die automatisierte Handlungsplanung [NGT04] bildet einen wichtigen Zweig des Forschungsbereiches Künstliche Intelligenz [RN03]. Sie befasst sich mit der Fragestellung, welche Aktionen auszuführen sind, um ein Ziel zu erreichen [Rin05]. Der Agent hat die Aufgabe, anhand von Informationen über seine Umwelt und die eigenen Fähigkeiten zielgerichtet zu agieren. Hierzu werden Folgen von Aktionen entwickelt, welche die Umwelt in der gewünschten Form verändern.

Unser Interesse gilt vor allem den *domänenunabhängigen* Planungssystemen, welche auf jedes beliebige Problem anwendbar sind, das sich in vorgegebener Sprache formalisieren lässt. Da die reale Welt und insbesondere reale Aktionen kaum darstellbarbar sind, muss für Handlungsplanung auf vereinfachende Modelle zurückgegriffen werden, um die Umwelt sowie die Aktionen des Agenten darzustellen. In der Praxis werden häufig folgende Einschränkungen gemacht: Die Umwelt wird in endlich viele diskrete Zustände unterteilt, wobei der aktuelle Zustand für den Agenten zu jedem Zeitpunkt uneingeschränkt wahrnehmbar ist. Der Zustandsraum ist statisch, d.h. die Welt ist ausschließlich durch den Agenten veränderbar. Die hierzu verwendeten Aktionen sind deterministisch, ihre Auswirkungen sind eindeutig. Gelten alle diese Einschränkungen, spricht man von *klassischer* Planung.

Die Lösung von Problemen der klassischen Handlungsplanung ist ein von vielen Wissenschaftlern untersuchtes Gebiet und es existieren zahlreiche Planungssysteme mit unterschiedlichen Ansätzen. Erwähnt sei die *heuristische Suche* im Raum der Zustände [BG01] und das Erstellen eines *Planungsgraphen* [BF97]. Weitere Möglichkeiten bietet die Übersetzung von Planungsproblemen in das aussagenlogische Erfüllbarkeitsproblem [KS92] oder das Planen als Modellprüfung [EH01].

Einen Einblick in den Stand der Forschung bietet die alle zwei Jahre stattfindende *International Planning Competition* (IPC), bei der sich Planungssysteme in unterschiedlichen Kategorien an verschiedenen Planungsdomänen messen. Über Jahre wurden hierbei nur Planungsprobleme untersucht, in denen das zu erfüllende Ziel fest vorgegeben ist. Erst mit Einführung der Beschreibungssprache PDDL3 [GL05] im Jahr 2006 entstand die Möglichkeit, Zielvorgaben flexibel zu gestalten.

1.2 Teilerfüllendes Planen

Der Begriff *teilerfüllende Handlungsplanung* steht für Planungsprobleme, in denen das Planungssystem selbst entscheiden muss, welche Teilziele einer vorgegebenen

Zielmenge ein Plan erfüllen soll. Die in der Literatur als *partial satisfaction planning* [vSDK04] bzw. *oversubscription planning* [Smi04] bezeichnete Erweiterung der klassischen Planung ist trotz vielfältiger Anwendungsbereiche ein bisher relativ wenig erforschtes Gebiet.

Sinnvolle Anwendungen finden sich überall, wo für die Planausführung benötigte Ressourcen, wie Energie, Kosten oder Zeit begrenzt sind. Ist die Zielvorgabe mit den zur Verfügung stehenden Mitteln nicht vollständig zu erfüllen, gibt ein herkömmliches Planungssystem auf. Ein teilerfüllender Planer trifft hingegen selbst die Entscheidung, welche Teilziele unter den gegebenen Umständen erfüllt werden. Dabei kann für verschiedene Teile der Zielvorgabe unterschiedlicher Nutzen spezifiziert werden. Die Auswahl der erfüllten Ziele kann so erfolgen, dass für feste Ressourcenbegrenzungen ein hoher Nutzen erzielt wird. Der Planer kann aber auch zwischen Nutzen und Kosten abwägen. Hierbei wird entschieden, ob der Nutzen eines Ziels die aufgewandten Kosten rechtfertigt.

Im Vergleich zur herkömmlichen Planung werden damit weitere Aufgaben an den Agenten übertragen. Er ist dadurch weitaus flexibler, da er aus zahlreichen Zielkombinationen auswählen kann. Dies ermöglicht eine weitergehende Autonomisierung, die für immer mehr Systeme in unterschiedlichen Bereichen gefordert ist.

1.3 Verwandte Arbeiten

Mike Williams und Steve Hanks erweiterten bereits 1994 den Begriff der Planqualität auf teilerfüllende Planung [WH94]. Ihr Planungssystem *Pyrrhus* findet optimale teilerfüllende Pläne für sogenannte *utility models*.

Motiviert durch Problemstellungen für autonome Systeme im Bereich der Raumfahrt, befasste sich David Smith mit der Zielauswahl beim teilerfüllenden Planen und stellte eine Heuristik zur Auswahl der Teilzielmengen vor [Smi04].

Eine Forschergruppe der Arizona State University um Professor Subbarao Kambhampati beschäftigt sich seit einigen Jahren mit der Thematik der teilerfüllenden Planung und hat viele Veröffentlichungen zum Thema verfasst. Sie definierten das Konzept des Nettonutzens [vSDK04], mit dem ein Abwägen zwischen Kosten und Nutzen ermöglicht wird. Außerdem wurden zunächst die teilerfüllenden Planungssysteme *Optiplan*, $AltAlt^{ps}$ [vSK04] und $Sapa^{ps}$ [DK04] entwickelt. Wenig später kamen die Weiterentwicklungen *AltWlt* [SK05] und $Sapa^{ps}_{UD}$ [Ben06] sowie das System $Yochan^{PS}$ [BKD06] dazu.

Brafman und Chernyavsky wählten für die gleiche Klasse von Problemen einen anderen Ansatz, der die Definition von Abhängigkeiten und Präferenzen bezüglich des Ziels mittels sogenannter TCP-Netze ermöglicht [BC05].

Für die IPC5 im Jahr 2006 wurde der Sprachstandard PDDL3 eingeführt [GL05]. Er ermöglicht vielfältige Vorgaben bezüglich der Planausführung und bietet dadurch die Möglichkeit Ziele zu spezifizieren, die nicht vollständig erfüllt werden müssen.

1.4 Überblick

Die vorliegende Diplomarbeit ist folgendermaßen gegliedert: In Kapitel 2 werden die für das Verständnis der Arbeit grundlegenden Begriffe geklärt, wobei vor allem Definitionen aus der klassischen Handlungsplanung eingeführt werden. Anschließend werden einige Domänen vorgestellt, die im Rahmen dieser Arbeit Verwendung finden. Teil 3 erläutert die Begriffe, Konzepte und Herausforderungen der teilerfüllenden Handlungsplanung. Kapitel 4 beschreibt eine Klasse von Lösungsalgorithmen, die teilerfüllende Planung mit Hilfe eines klassischen Planungssystems durchführen. Hierbei kommen die Konzepte Hillclimbing sowie Verfahren, welche die Kosten der Teilzielmengen modellieren, zum Einsatz. Kapitel 5 zeigt die Ergebnisse der implementierten Verfahren. Die Arbeit endet mit einer Zusammenfassung und dem Ausblick auf weiterführende Möglichkeiten.

Diese Arbeit stellt Konzepte zur Lösung von Problemen aus dem Bereich der teilerfüllenden Handlungsplanung vor. Das Grundprinzip liegt hierbei in der Zerlegung des Problems in einzelne Teilprobleme bestehend aus Teilmengen des vorgegebenen Ziels. Es wird untersucht, wie Information über Kosten und Nutzen von Teilzielmengen auf andere übertragen werden kann. Man erhält dadurch Anhaltspunkte zur Auswahl von Teilzielmengen, die einen hohen Nutzen bei niedrigen Kosten erzielen.

2 Grundlagen und Definitionen

Das folgende Kapitel gibt eine Einführung in die wichtigsten Begriffe dieser Arbeit. Zunächst werden allgemeine Schreibweisen, dann Begriffe aus dem Bereich der Handlungsplanung vorgestellt.

2.1 Notation

In diesem Abschnitt werden kurz die in dieser Arbeit verwendeten mathematischen Schreibweisen eingeführt.

- Sei M eine beliebige Menge, so bezeichnen wir mit $Pot(M)$ die Menge ihrer Teilmengen und mit $|M|$ ihre Mächtigkeit.
- Für eine Abbildung f wird der Definitionsbereich mit $dom(f)$ bezeichnet. Wir schreiben $f \equiv k$ mit $k \in \mathbb{R}$, falls für alle Werte $z \in dom(f)$ gilt $f(z) = k$.
- Die Menge der Aussagenlogischen Formeln AL ergibt sich aus einer Menge aussagenlogischer Variablen X, den Konstanten $\bot$ (falsch) und $\top$ (wahr) und den Symbolen $\vee, \wedge, \rightarrow, \leftrightarrow$ und $\neg$ durch die Grammatik:

$$AL ::= X|\bot|\top|\neg AL|(AL \vee AL)|(AL \wedge AL)|(AL \rightarrow AL)|(AL \leftrightarrow AL)$$

- Ein Literal ist eine aussagenlogische Formel der Gestalt x oder $\neg x$ mit $x \in X$. Die Menge aller Literale über X wird mit $\bar{X}$ bezeichnet.
- Wir schreiben Konjunktionen von $l_i \in \bar{X}$ verkürzt $(l_1 \wedge \cdots \wedge l_m)$ statt $(l_1 \wedge (l_2 \wedge (\cdots \wedge (l_{m-1} \wedge l_m) \dots)$. Die Schreibweise gilt analog für Disjunktionen von Literalen $(l_1 \vee \cdots \vee l_m)$, die als Klauseln bezeichnet werden.
- Eine Variablenbelegung ist eine Abbildung $\beta : X \rightarrow \{0, 1\}$. Wir schreiben $\beta \models \Phi$ für $\Phi \in AL$, wenn die Belegung β die Formel Φ erfüllt. Hierbei gilt die übliche Semantik aussagenlogischer Formeln.
- Eine Formel Φ heißt erfüllbar genau dann, wenn es eine Belegung β mit $\beta \models \Phi$ gibt; falls keine solche Belegung existiert, nennen wir Φ unerfüllbar.
- Wir erlauben uns, für $\mathbb{R}$ definierte Rechenoperationen auf die Menge $\mathbb{R} \cup \{\infty\}$ zu übertragen.

2.2 Handlungsplanung

In diesem Abschnitt werden die Grundbegriffe der Handlungsplanung eingeführt. Nach der Definition von Zustandsmodell, Operatoren und Planungsinstanz lassen sich verschiedene Problemstellungen der klassischen Planung definieren. Es folgen Erläuterungen zum Zielzustand und der Unterscheidung optimaler und suboptimaler Planungssysteme. Zum Abschluss des Kapitels werden *Benchmark*-Domänen vorgestellt, die später als Probleme der teilerfüllenden Planung zum Einsatz kommen.

Definition 1 (Zustandsmodell). Ein **Zustandsmodell** ist ein 5-Tupel $M = (S, s_{init}, S_G, A, \theta)$. Hierbei bezeichnet

- S die endliche Menge der *Zustände*;
- s_{init} den *Anfangszustand*;
- $S_G \subseteq S$ die Menge der *Zielzustände*;
- A die endliche Menge der *Aktionen*;
- $\theta : S \times A \nrightarrow S$ die partielle *Übergangsfunktion*.

Die Menge aller Zustandsmodelle sei $\mathcal{M}$. Die Abbildung θ ist nur partiell definiert, da nicht jede Aktion in jedem Zustand anwendbar ist: Eine Aktion $a \in A$ ist im Zustand s **anwendbar**, falls $(s, a) \in dom(\theta)$. Der Folgezustand ist dann $s' = \theta(s, a)$.

Definition 2 (Plan). Sei $M = (S, s_{init}, S_G, A, \theta)$ ein Zustandsmodell . Eine Folge von Aktionen $P = (a_0, a_1, \ldots, a_n)$ heißt **Plan** genau dann, wenn es eine Folge von Zuständen $(s_0, s_1, \ldots, s_{n+1})$ gibt mit (i) $s_0 = s_{init}$, (ii) $(s_i, a_i) \in dom(\theta)$ für alle $i = 1, \ldots, n$ und (iii) $\theta(s_i, a_i) = s_{i+1}$ für alle $i = 1, \ldots, n$.

Die Definition weicht von der üblichen Formulierung ab, da auf die Forderung $\theta(a_n, s_n) \in S_G$ verzichtet wird. Ein Plan in unserem Sinne muss also keinen Zielzustand erreichen. Wir bezeichnen den Zustand, der durch einen Plan P erreicht wird, als s_P. Ein Plan P erreicht also das *Ziel*, falls $s_P \in S_G$.

Die Definition beschreibt *sequentielle Pläne*. Auf das Konzept des *parallelen Planers*, der bestimmte Aktionen gleichzeitig ausführen kann, soll nicht weiter eingegangen werden.

Die Menge aller Pläne sei $\mathcal{P}$. Für einen Plan $P = (a_0, a_1, \ldots, a_n)$ heißt die Anzahl der Aktionen in P **Planlänge** $L(P)$. Der leere Plan wird mit () bezeichnet.

Definition 3 (Planungsdomäne). Eine **Planungsdomäne** ist eine Abbildung $\mathcal{D} : \mathcal{L} \rightarrow \mathcal{M}$, wobei $\mathcal{L}$ eine Sprache über einem Alphabet Σ ist, die *Kodierungssprache*.

Die hier betrachteten Planungsprobleme sind in PDDL (*Planning Domain Definition Language*) spezifiziert. Die Beschreibungssprache befindet sich in einem permanenten Weiterentwicklungsprozess, um immer wieder neue Herausforderungen der

Handlungsplanung darstellen zu können. Daher existieren zahlreiche Standards verschiedener Aussagekraft. Ausgehen wollen wir in dieser Arbeit von PDDL2.1 [FL03]; es werden aber auch weiterführende Konzepte vorgestellt.

Das Zustandsmodell wird mittels logischer Ausdrücke beschrieben. Für eine Menge X von Aussagenvariablen ist ein Zustand damit immer eine Belegung $s : X \to \{0, 1\}$. Eine aussagenlogische Formel Φ über X repräsentiert damit die Menge aller Zustände, für die $s \models \Phi$ gilt. Die Variablen in X werden daher auch als Zustandsvariablen bezeichnet. Aktionen entsprechen logische Operatoren:

Definition 4 (Operator). Sei X eine Menge von Aussagenvariablen. Ein *Operator* ist ein Tupel $o = (prec, \mathit{eff})$ mit

- *prec*, einer aussagenlogischen Formel über X, der Vorbedingung.
- *eff*, dem Effekt von o.

Die Menge der Effektformeln E wird gebildet durch die Regelmenge:

$$E ::= a | \neg a | (E \wedge E)$$

Für die Anwendbarkeit gilt dann: Ein *Operator* $o = (prec, \mathit{eff})$ ist im Zustand s *anwendbar* gdw. $s \models prec$.

Die Definition für Vorbedingungen und Effekte ist denkbar einfach gehalten. Grundsätzlich hängt die Mächtigkeit und Flexibilität der Operatoren vom verwendeten Sprachstandard ab. Ein *STRIPS-Operator* etwa lässt für *prec* lediglich Konjunktionen von Literalen zu, während bei einem *ADL-Operator* auch quantifizierte und konditionale Ausdrücke möglich sind.

Der Operatorbegriff erlaubt die folgende Definition:

Definition 5 (Klassische Planungsinstanz). Eine klassische **Planungsinstanz** Π ist ein Vier-Tupel $\Pi = (X, I, G, O)$:

- Die endliche Menge der *Zustandsvariablen* X.
- Der vollständig definierte *Anfangszustand* $I : X \to \{0, 1\}$ beschreibt die Welt zu Beginn des Planungsvorgangs.
- Das *Ziel* ist gegeben durch die Zielformel G, eine Konjunktion von Literalen $G = (g_1 \wedge g_2 \wedge \cdots \wedge g_r),\ g_i \in \bar{X}$. s ist genau dann ein Zielzustand, wenn $s \models G$.
- Die endliche Menge der *Operatoren* O.

Man beachte, dass für die Zielbeschreibung lediglich eine Konjunktion von Literalen, also falls erfüllbar eine partielle Belegung $X \nrightarrow \{0, 1\}$, vorgesehen ist. Die Definition ist bewusst gewählt, da diese Arbeit sich mit der Zerlegung der Zielmenge

beschäftigt. Je nach Sprachstandard sind aber auch hier komplexere logische Ausdrücke möglich.

PDDL-Planungsinstanzen bestehen aus zwei Teilen: In der *Domänenbeschreibung* werden die verwendeten Prädikate und die Operatoren der Domäne spezifiziert.
In der *Problembeschreibung* wird ein konkretes Problem beschrieben. Hierzu werden sämtliche Objekte der *Planungswelt* aufgezählt. Logische Ausdrücke über diesen Objekten definieren den Start- sowie die Menge der Zielzustände.

2.2.1 Problemstellungen

In der Planungsinstanz ist zwar das Ziel der Planung spezifiziert, es ergeben sich jedoch unterschiedliche *Problemstellungen*, die von einem Planungssystem bearbeitet werden können.

Für Forschung und Anwendung sind vor allem Such- und Optimierungsprobleme interessant. Daher wird im Folgenden auf die Spezifikation von Entscheidungsproblemen verzichtet. Naheliegende und sinnvolle Problemstellungen für eine gegebene Planungsinstanz Π sind dann z.B. die folgenden:

- PlanGen-Suchproblem: Ausgabe ist ein Plan P mit $s_P \models G$, oder #, falls kein solcher Plan existiert.

- PlanLen-Optimierungsproblem: Ausgabe ist ein Plan P mit minimaler Länge $L(P)$ und $s_P \models G$ oder #, falls kein solcher Plan existiert.

Klassische Planungsprogramme wie *FF* [HN01] oder *STAN* [LF99] lösen PlanGen, versuchen dabei aber kurze Pläne zu bevorzugen.

2.2.2 Plankosten

Zum Vergleich der Güte von Plänen steht uns bis jetzt für einen Plan $P = (a_1, \ldots, a_m)$ lediglich die Planlänge $L(P) = m$ zur Verfügung. Dies ist in einigen Domänen ausreichend. Häufig macht es jedoch Sinn, verschiedenen Aktionen unterschiedliche Kosten zuzuweisen. Dadurch lässt sich z.B. der Verbrauch von Ressourcen wie Zeit, Treibstoff oder Speicherplatz modellieren. Es ist offensichtlich, dass in Transportdomänen das Aufladen einer Ware weniger Ressourcen benötigt als das Zurücklegen langer Wegstrecken. Die Kosten sind in der Regel nicht nur von der ausgeführten Aktion, sondern auch vom aktuellen Zustand abhängig.

Definition 6 (Aktionsausführungskosten). Die Kosten einer Aktionsausführung sind gegeben durch eine partielle Abbildung $\hat{c} : S \times A \nrightarrow \mathbb{R}^+$ mit $dom(\hat{c}) = dom(\theta)$.

Die Forderung bezüglich des Definitionsbereichs bewirkt, dass jeder ausführbaren Aktion auch Kosten zuweisbar sind.

Wir erweitern nun die Aktionsausführungskosten $\hat{c}$ auf Pläne P beliebiger Länge:

Definition 7 (Plankosten). Die **Plankosten** für einen Plan $P = (a_0, \ldots, a_n)$, der angewandt in s_0 die Zustände $s_1, \ldots, s_{n+1}$ durchläuft, ergeben sich durch

$$\hat{c}(P) = \sum_{i=0}^{n} \hat{c}(a_i, s_i)$$

Die Planlänge $L(P)$ entspricht also dem uniformen Kostenmaß $\hat{c}(a) = 1 \; \forall a \in A$.

Durch eine Erweiterung des Zustandsmodells um Aktionsausführungskosten ergibt sich für die kostenbehaftete Planungsinstanzen Π folgendes:

Definition 8 (Optimaler Plan). Ein Plan heißt **(kosten)optimaler Plan** P^* für Π, falls $s_P \models G$ und für jeden Plan P' für Π mit $s_P \models G$ $\hat{c}(P^*) \leq \hat{c}(P')$ gilt.

Unmittelbar folgt die neue Problemstellung:

- PlanCost-Optimierungsproblem: Ausgabe ist ein Plan P mit minimalen Kosten $\hat{c}$ und $s_P \models G$, oder #, falls kein solcher Plan existiert.

Wie eben erwähnt ist das Problem eine Verallgemeinerung von PlanLen.

2.2.2.1 Optimale und suboptimale Planer

Abhängig von der Problemstellung ändert sich auch die Vorschrift zur Bewertung von Planern. Für unsere Betrachtungen sind vor allem die Plankosten interessant. Ein optimaler Planer sucht stets nach einem Plan mit optimalen Kosten. Sofern er terminiert und einen Plan ausgibt, hat dieser minimale Plankosten. Es existieren zahlreiche Systeme für Handlungsplanung mit Kosten wie *Metric-FF* [Hof02], *SGPLAN* [CWH06] oder *LPG-TD* [GSS04]. Diese liefern in der Regel keine kostenoptimalen Pläne, versuchen jedoch die Plankosten klein zu halten.

2.2.2.2 Kodierung von Plankosten

Durch den Sprachstandard PDDL2.1 wurden verschiedene Möglichkeiten zur Darstellung von Zeit oder anderen Ressourcen bereitgestellt. Mit dem Konzept der *durative actions* kann die zur Ausführung von Aktionen benötigte Zeit explizit dargestellt werden. Die in dieser Arbeit vorgestellten Domänen verwenden zur Modellierung der Ressourcen *numerische Variablen*. Diese können mittels *Funktionen* gesetzt werden. Die Depots-Domäne z.B. verwendet die nullstellige Funktion `fuel-cost`, die in der Problembeschreibung initialisiert `(= (fuel-cost) 0)` und in den Effekten der Aktionen verändert wird `(increase (fuel-cost) 10)`. Die benötigte Zeit `total-time` entspricht in allen vorgestellten Problemen der Planlänge L.

Am Ende jeder Problembeschreibung kann dem Planer mitgeteilt werden, ob er eine spezielle Metrik verwenden soll, indem er z.B. Pläne sucht, die den Treibstoffverbrauch minimieren `(:metric minimize (fuel-cost))`.

In PDDL2.1-Spezifikationen lassen sich parameterbehaftete Funktionen beliebig schachteln und mit Rechenoperationen und Zahlen verknüpfen. Somit bieten sie ein mächtiges Instrument zur Darstellung beliebiger numerisch variabler Phänomene.

2.2.3 Die Zielbeschreibung

Die Menge der Zielzustände S_G ist definiert durch den logischen Ausdruck G, der in jedem $s_G \in S_G$ erfüllt sein muss. In der Regel ist die Zielbeschreibung G eine Konjunktion mehrerer Teilziele $G = (g_1 \wedge g_2 \wedge \cdots \wedge g_r)$. Bei allen in dieser Arbeit vorgestellten Verfahren wird angenommen, dass die Konjunktionsglieder $g_1, \ldots, g_r$ Literale sind. Diese Einschränkung gilt nicht für alle PDDL-Versionen, aber in den in dieser Arbeit betrachteten Probleminstanzen. Üblich ist auch die Mengenschreibweise $G = \{g_1, g_2, \ldots, g_r\}$, die für G den Begriff *Zielmenge* nahelegt. Die einzelnen Zielmengenelemente g_i werden dann als Zielelemente, Teilmengen $G' \subseteq G$ als Teilzielmengen bezeichnet. Einelementige Zielmengen $\{g_i\}$ werden häufig vereinfachend mit g_i bezeichnet.

Für das (im Gegensatz zu den in dieser Arbeit betrachteten *teilerfüllenden* Planungsproblemen) traditionelle Planen gilt die Forderung an einen gültigen Plan, dass die gesamte Zielmenge G erfüllt sein muss.

Kann ein traditioneller Planer nur $r - 1$ Teilziele erreichen, ist die Aufgabe nicht lösbar, und er gibt # aus. Das ist auch dann der Fall, wenn nicht ein einziges Teilziel g_i erreicht wird. Ein herkömmlicher Planer unterscheidet also nicht, ob fast alle oder kein Teilziel erreicht wurde. Insbesondere kann natürlich auch dann kein Plan gefunden werden, wenn die Zielformel einen logischen Widerspruch enthält.

2.3 Planungsdomänen

In diesem Abschnitt werden kurz vier Planungsdomänen vorgestellt. Es handelt sich um Probleme der IPC (*International Planning Competition*) des Jahres 2002. Verwendet wird jeweils die numerische Variante, spezifiziert gemäß dem Standard PDDL2.1, da hier Plankosten sinnvoll erzeugt werden können und die Herausforderung an das Planungssystem in der Suche nach kostenoptimalen Plänen besteht. Die bekannten *Benchmark*-Domänen dienen zur Erläuterung der in dieser Arbeit vorgestellten Konzepte sowie zum Test der entwickelten Verfahren. Man beachte, dass für den Zeitverbrauch `total-time` ein sehr reduziertes Modell verwendet wird, welches jeder Aktion die Zeitdauer 1 zuordnet. Bei Verwendung von sequenziellen Planern entspricht die Zeit also der Anzahl der verwendeten Aktionen. Parallele Planer geben wesentlich kleinere Kosten zurück, weil sie die gleichzeitige Ausführung mehrerer Aktionen erlauben.

2.3.1 Satellites

Die Domäne SATELLITES beschreibt eine Aufgabenstellung für unterschiedlich ausgestattete Satelliten im Weltraum. Mittels diverser Instrumente und unterschiedlicher Modi sollen Aufnahmen von Himmelskörpern gemacht werden. Die individuell ausgestatteten Geräte unterstützen verschiedene Modi, wie Wärme- oder Infrarotbild. Um ein Gerät aufnahmebereit zu machen, muss dieses zunächst eingeschaltet und dann an festgelegten Objekten kalibriert werden, wobei immer nur ein Gerät mit Strom

versorgt werden kann. Für Kalibrierung und Aufnahmen muss der ganze Satellit so gedreht werden, dass er auf das entsprechende Objekt ausgerichtet ist. Die Zielbeschreibung fordert für eine Menge von Himmelskörpern Bilder in festgelegten Modi und gegebenfalls eine Ausrichtung, die der Satellit im Zielzustand einnehmen muss. In der hier verwendeten Version wird der Treibstoff- und Speicherplatzverbrauch der verschiedenen Aktionen modelliert; beide Ressourcen können begrenzt sein. Die Herausforderung besteht also in einer effizienten Abdeckung der geforderten Aufnahmen mit den Mitteln und Ressourcen der Satelliten. Die Kosten eines Planes entsprechen der Menge des benötigten Treibstoffs. Es sollen also solche Pläne bevorzugt werden, bei denen sich die Satelliten möglichst wenig drehen müssen.

2.3.2 Depots

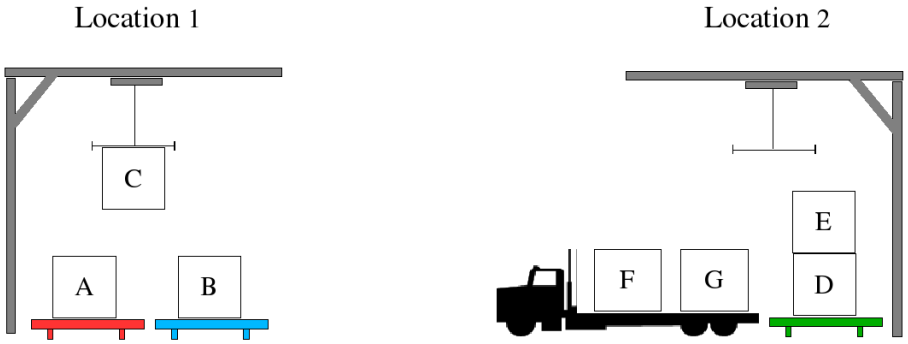


Abbildung 2.1: Ein DEPOTS-Szenario

Die DEPOTS-Domäne kann als Kombination aus BLOCKSWORLD [ST01] und einer Transport-Domäne betrachtet werden. Hierbei sollen Blöcke mittels Kränen an festgelegten Orten auf Paletten gestapelt und dabei gegebenenfalls auch umsortiert werden. Außerdem können sie an andere Orte transportiert werden, indem sie mit dem Kran auf einen LKW geladen werden. Die Traglast des Krans entspricht einem Block. Auf dem Lastwagen liegen die Blöcke ungestapelt, sind also alle zugänglich. Zur Lösung einer Instanz müssen Blöcke in vorgegebener Reihenfolge auf bestimmte Paletten gestapelt werden. Es wird eine Version verwendet, welche die Zuladung der LKWs begrenzt und den Treibstoff- sowie den Zeitverbrauch berechnet. Die Plankosten ergeben sich je nach Instanz aus der benötigten Zeit oder den anfallenden Treibstoffkosten.

Interessant ist zum einen die *BlocksWorld*-Problematik, dass nur der oberste Block eines Stapels vom Kran aufgenommen werden kann, was Umsortierungen notwendig macht. Hinzu kommt das Logistik-Problem, die Blöcke so zu anderen Orten zu transportieren, dass möglichst wenig bzw. kurze LKW-Fahrten nötig sind. Abbildung 2.1 zeigt ein Szenario in einem kleinen Problem mit zwei Orten, einem LKW, drei Paletten und sieben Blöcken.

2.3.3 Rovers

Die Probleme der ROVERS-Domäne beschreiben Aufgabenstellungen für ein autonomes Kleinfahrzeug, das die Oberfläche eines Planeten erforscht. Hierbei werden Erdreich- und Gesteinsproben genommen sowie Bilder gemacht. Die Daten müssen per Funk an die Basisstation übermittelt werden. In der Sonne kann der Rover verbrauchte Energie wieder aufladen. Die Plankosten ergeben sich aus der Anzahl der Wiederaufladungen.

Die Umwelt ist durch ein Netz von Wegpunkten realisiert, wobei von einem gegebenen Wegpunkt festgelegte Mengen von Wegpunkten erreich- bzw. sichtbar sind. Verschiedene Fahrzeuge sind mit unterschiedlichen Geräten ausgestattet. Um Daten über eine Probe zu übermitteln, muss der Rover sie aufnehmen und analysieren. Da das Fahrzeug für nur eine Probe Platz bietet, muss diese gegebenenfalls wieder abgelegt werden. Damit Bilder gemacht werden können ist es notwendig, dass zunächst die Kamera kalibriert wird. Alle Aktionen benötigen eine operatorabhängige Menge an Energie, die nur mit dem Operator `recharge` wieder aufgeladen werden kann.

Die Herausforderungen bestehen zum einen darin, einen kurzen Weg zu finden, der die Wegpunkte der Proben erreicht. Zweitens gilt es die verfügbaren Instrumente effektiv einzusetzen. Beide Bereiche werden durch Einschränkungen bezüglich der Erreichbarkeit und Sichtbarkeit erschwert.

2.3.4 ZenoTravel

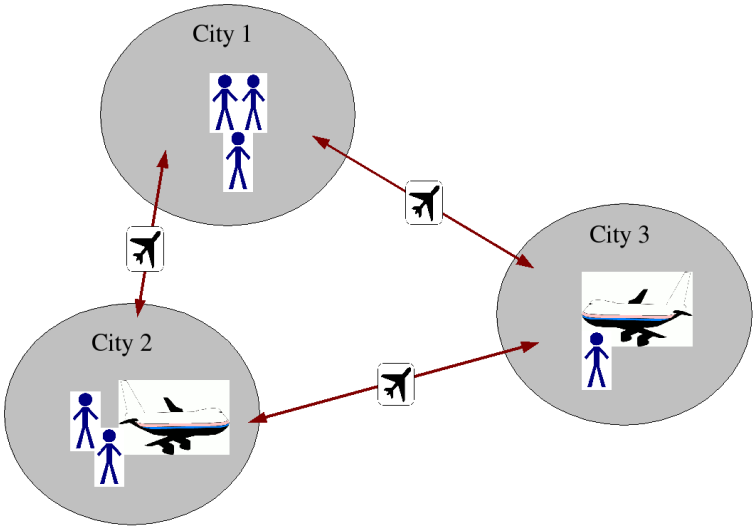


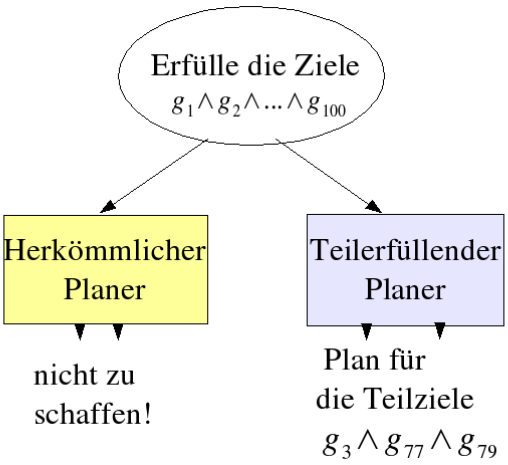
Abbildung 2.2: Ein ZENOTRAVEL-Szenario

In der Domäne ZENOTRAVEL geht es um den Transport von Passagieren zwischen Städten mittels Flugzeugen. Hierbei kann zwischen zwei verschiedenen Flug-Modi mit unterschiedlichem Treibstoffverbrauch gewählt werden. In der verwendeten Version

ist der sparsame Modus Flügen mit wenigen Passagieren vorbehalten. Die Plankosten berechnen sich aus der mit verschiedenen Konstanten gewichteten Summe aus Zeit- und Treibstoffverbrauch.

Die zunächst relativ simple Logistik-Problematik bezieht ihren Reiz aus den unterschiedlichen Modi, in denen die Flugzeuge fliegen können. Bei der Planung gilt es abzuwägen, ob die treibstoffintensive Variante Vorteile bringt, weil unter Umständen weniger Flüge notwendig sind. Da `(total-time)` die Planlänge beschreibt, steigt dieser Wert im sparsamen Modus, der nur wenige Passagiere zulässt, an. Durch die für jede Probleminstanz neu gewichtete Kombination der Werte `(total-time)` und `(total-fuel-used)` als Plankosten-Maß bildet jede Entscheidung für einen Flug-Modus eine Herausforderung an das Planungssystem.

3 Teilerfüllendes Planen



Teilerfüllende Handlungsplanung behandelt Probleme mit einer Menge von möglichen Zielen. Ein Planer muss aus diesen Zielen eine Teilmenge auswählen, die er mit gegebenen Ressourcen erreichen kann [Smi04]. Die Bezeichnung *teilerfüllend* wurde gewählt, da im Allgemeinen nicht alle Konjunkte der Zielformel erfüllt werden. In dieser Arbeit wird die gängige Abkürzung PSP (*partial satisfaction planning*) [vSDK04] verwendet. PSP wird in der Literatur teilweise auch als Over-Subscription-Planning (OSP) [Smi04] bezeichnet.

Der Einsatz von teilerfüllenden Planern bietet sich überall da an, wo ein traditionelles Planungssystem mit den gegebenen Ressourcen das komplette Ziel nicht erreichen kann. Es ist offensichtlich, dass es bei vielen Problemen sinnvoll sein kann, hier nicht aufzugeben, sondern möglichst viele und wichtige Teilziele zu erfüllen. PS-Planer können außerdem auch dann angewendet werden, wenn eine Zielformel im Ganzen nicht erfüllbar ist, da sie z.B. widersprüchliche Konjunkte enthält.

Die Ausgabe eines teilerfüllenden Planers ist ein Plan $P_{G'}$, der im Allgemeinen nur eine Teilmenge G' der gesamten Zielmenge G erfüllt. PS-Planen ist also weitaus flexibler und breiter anwendbar als klassische Planung.

Das Kapitel gibt einen Überblick über den Begriff der teilerfüllenden Handlungsplanung. Hierzu werden verschiedene Problemstellungen vorgestellt, wobei ein Schwerpunkt auf das PSPNetBenefit-Problem gelegt wird. Im Abschnitt 3.3 wird die Problematik der Abhängigkeiten zwischen Teilzielen behandelt, gefolgt von Erläuterungen zur Problemkodierung. Die Einführung eindeutiger Teilzielkosten ermöglicht Untersuchungen zur Verteilung des Nettonutzens auf die Teilzielmengen. Abschließend werden die Lösungsansätze einiger bestehender PS-Planungssysteme vorgestellt.

3.1 Grundlagen und Beispiel

In dieser Arbeit wird die teilerfüllende Handlungsplanung in zwei Fragestellungen eingeteilt.

1. Welche Teilziele sollen erfüllt werden?

2. Welche Aktionen müssen dazu ausgeführt werden?

Letztere ist exakt die Fragestellung der klassischen Planung – ein vielfältig untersuchtes Forschungsfeld. Die vorliegende Arbeit befasst sich daher mit der ersten Frage.

Definition 9 (G'-erfüllender Plan). Sei $G' \subseteq G$. Ein Plan $P = (a_1, a_2, \ldots, a_n)$ heißt **G'-erfüllender Plan**, wenn im erreichten Zustand s_P gilt: $s_P \models G'$.

Ein G'-erfüllender Plan wird mit $P_{G'}$, die Menge aller G'-erfüllenden Pläne mit $\mathcal{P}_{G'}$ bezeichnet.

Definition 10 (Nicht erfüllbare Zielmenge). Eine Zielmenge G' heißt **nicht erfüllbar**, wenn kein G'-erfüllender Plan existiert, also genau dann, wenn $\mathcal{P}_{G'} = \emptyset$.

Da grundsätzlich $\emptyset \subseteq G$ gilt, ist jeder beliebige in s_{init} anwendbare Plan P inklusive der leeren Folge ein gültiger *teilerfüllender*, genauer $\emptyset$-*erfüllender* Plan. Plankosten reichen also nicht aus, um Pläne zu bewerten, da in der Regel unterschiedliche Zielmengen erfüllt werden. Es ist damit notwendig, das Erreichen verschiedener Zielmengen zu bewerten. Hierzu benötigt man eine Nutzenfunktion, die Mengen von Teilzielen einen Wert zuweist. Die folgende Definition führt diese ein.

Definition 11 (Teilerfüllende Planungsinstanz). Eine **teilerfüllende Planungsinstanz** ist ein 6-Tupel $\Psi = (X, I, G, O, \hat{c}, u)$. Hierbei ist wiederum

- X die endliche Menge der *Aussagenvariablen*.
- I der *Anfangszustand*.
- $G = (g_1 \wedge g_2 \wedge \cdots \wedge g_r)$ das *Gesamtziel*.
- O die endliche Menge der *Operatoren*.
- $\hat{c} : S \times A \nrightarrow \mathbb{R}^+$ die partielle *Kostenfunktion*.
- $u : Pot(G) \rightarrow \mathbb{R}^+$ die *Nutzenfunktion*.

3.1.1 Die Nutzenfunktion

Die Nutzenfunktion dient dazu festzulegen, welchen Wert die Erfüllung welcher Teilzielmenge hat. Sie bildet für das PS-Planungssystem den einzigen Anreiz, Ziele zu erfüllen.

Als Definitionsbereich von u wurde die Potenzmenge der Ziele gewählt, um u zunächst möglichst allgemein zu halten. In der Praxis ist die Nutzenfunktion oft genau auf den einelementigen $g \in G$ definiert: $\hat{u} : G \rightarrow \mathbb{R}^+$. Für beliebige zusammengesetzten Teilzielmengen G' ergibt sich der Nutzen dann durch Addition: $u(G') = \sum_{g \in G'} \hat{u}(g)$. Denkbar sind jedoch auch Probleme, bei denen spezielle Kombinationen von Zielen sinnlos oder eben besonders sinnvoll sind. Der Nutzen eines linken Schuhs alleine ist z.B. sehr gering. Zwei linke Schuhe sind kaum nützlicher; ein rechter und ein linker hingegen bringen mehr als die Nutzensumme der einzelnen Schuhe. Treten solche Nutzen-Abhängigkeiten [Ben06] auf, ist die additive Nutzenfunktion natürlich ungeeignet und $\hat{u}$ sollte einen größeren Definitionsbereich abdecken.

Da Nutzen-Abhängigkeiten in der Praxis häufig sind, sollten sie auch in der teilerfüllenden Handlungsplanung Beachtung finden.

Das Problem stellt jedoch keine große Herausforderung dar, da der Nutzen jeder beliebigen Teilzielmenge entweder aus der Problembeschreibung ausgelesen oder einfach berechnet werden kann. Dies gilt unabhängig davon, für welche Mengen ein Nutzen angegeben ist: Sei $\hat{u}$ eine gegebene Nutzenfunktion mit einem Definitionsbereich $dom(\hat{u}) \subseteq Pot(G)$. Der Nutzen u einer beliebigen Menge $G' \subseteq G$ ergibt sich dann durch

$$u(G') = \sum_{\hat{G} \in dom(\hat{u}), \hat{G} \subseteq G'} \hat{u}(\hat{G})$$

Zudem lässt sich jede Zielmenge leicht in ein einzelnes Zielelement mit entsprechendem Nutzen übersetzen:

Für ein Teilziel $G1 = (g_1 \wedge \cdots \wedge g_l)$ mit $u(G1) = k$ genügt hierzu die Einführung eines Operators `opG1` und des neuen Zielelementes `reachedG1`. Man wählt nun *prec*(`opG1`) $= G1$ und *eff*(`opG1`) $=$ `reachedG1`, sowie $\hat{c}$(`opG1`) $= 0$. Die Zielmenge erweitert sich um $g_{neu} =$ `reachedG1` mit $u(g_{neu}) = k$.

In dieser Arbeit wird daher stets eine Nutzenfunktion verwendet, die nur für die elementaren Ziele definiert ist und sich ansonsten durch Addition ergibt.

3.1.2 Beispiel

Im Folgenden wird eine einfache für teilerfüllendes Planen typische Aufgabenstellung vorgestellt. Geeignet hierzu ist das Reiseproblem [vSDK04], bei dem es um die Auswahl von Reisezielen geht. Es wird in dieser Arbeit immer wieder zur Erläuterung der theoretischen Konzepte verwendet.

Abbildung 3.1 zeigt den Graphen eines Reiseproblems. Ein Student aus Las Vegas (LV) soll eine Arbeit bei der AAAI in San Jose (SJ) präsentieren. Um dieses Ziel $g_1 =$*Attend_AAAI(SJ)* mit Nutzen $u(g_1) = 300$ zu erreichen muss er lediglich

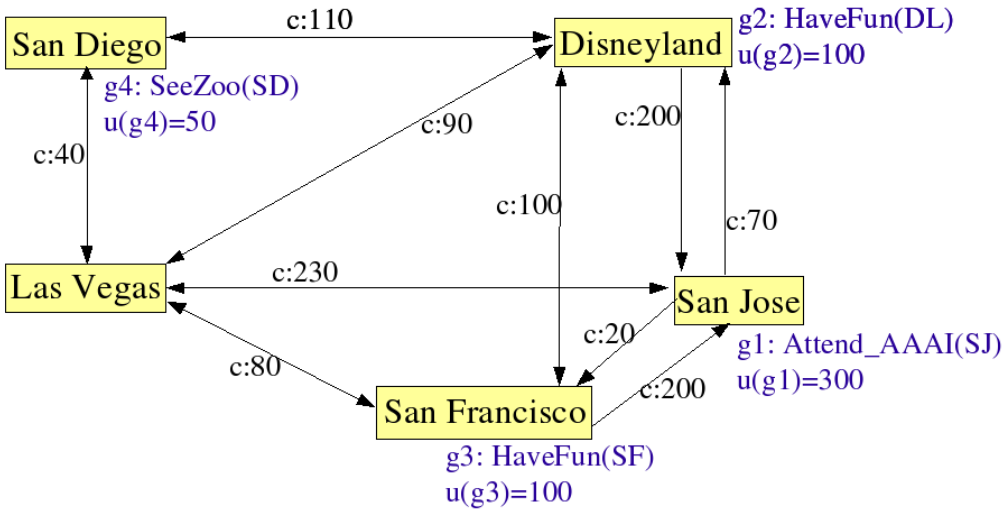


Abbildung 3.1: Reiseproblem mit Kosten und Nutzen.

dorthin reisen. Die Reisekosten hierfür betragen $\hat{c}(travel(LV,SJ)) = 230$. Außerdem sucht der Student Kurzweil in Disneyland (DL) und San Francisco (SF) und er würde gerne den Zoo von San Diego (SD) besuchen. Auch hierfür reicht es aus, zu den entsprechenden Orten zu gelangen. Da es sich um ein teilerfüllendes Planungsproblem handelt, muss der Student nicht alle Teilziele erfüllen. Sinnvoll wäre z.B. eine Reiseroute mit niedrigen Kosten und hohem Nutzen.
Das Reiseproblem als formal dargestellte teilerfüllende Planungsinstanz (verkürzt): $\Psi = (X, I, G, O, \hat{c}, u)$ mit

- $X = \{at(LV), at(SD), at(DL), at(SF), at(SJ), SeeZoo(SD), HaveFun(DL), HaveFun(SF), Attend\_AAAI(SJ)\}$.
- $I = \{at(LV)\}$.
- $G = (SeeZoo(SD) \wedge HaveFun(DL) \wedge HaveFun(SF) \wedge Attend\_AAAI(SJ))$.
- $O = \{travel(LV, SJ), travel(LV, SD), travel(DL, SJ), travel(SJ, DL), \dots\}$.
- $\hat{c}(travel(LV, SJ)) = 230, \hat{c}(travel(LV, SD)) = 40, \hat{c}(travel(DL, SJ)) = 200, \hat{c}(travel(SJ, DL)) = 70, \dots$
- $u(at(LV)) = u(at(SD)) = u(at(DL)) = u(at(SJ)) = u(at(SF)) = 0, u(SeeZoo(SD)) = 50, u(HaveFun(DL)) = 100, u(HaveFun(SF)) = 100, u(Attend\_AAAI(SJ)) = 300$.

Hierbei ist u nur auf den einelementigen Teilzielen definiert und ansonsten additiv zu verstehen. Die übrigen Kosten können aus der Graphik 3.1 entnommen werden.

Auch die *Benchmark*-Domänen der IPC bieten sich zu einem großen Teil für PSP an, da sich die Zielmenge sinnvoll in Teilziele zerlegen lässt. Eine Ausnahme bilden lediglich so genannte *manipulation domains* [Hel01] wie z.B. FREECELL, bei denen die Zerlegung des Ziels keinen Sinn macht. Bei den in 2.3 vorgestellten Domänen, sowie in vielen logistischen Planungsproblemen kann es durchaus geboten sein, gegebenenfalls nur einen Teil des Gesamtziels zu erfüllen.

3.2 Teilerfüllende Problemstellungen

Da selbst der leere Plan in PSP-Problemen *erfüllend* ist, wären sämtliche in 2.2.1 vorgestellten Problemstellungen wie PLANEX und PLANGEN trivial lösbar. Problemstellungen für teilerfüllende Planung erfordern also bestimmte Eigenschaften. Sie sollten nach Möglichkeit einbeziehen, dass die Güte eines PS-Planes sowohl von der Qualität der erreichten Zielmenge, als auch von den Kosten des Plans abhängt. Die hier vorgestellten Problemstellungen wurden von van den Briel, Sanches Nigenda, Do und Kambhampati ausgewählt [vSDK04] und werden im folgenden wieder als Optimierungsprobleme vorgestellt. Insbesondere das PSPNETBENEFIT-Problem soll dann genauer betrachtet werden. Abbildung 3.2 zeigt eine hierarchische Darstellung für einige Problemstellungen der herkömmlichen und teilerfüllenden Handlungsplanung. Verbindungen zeigen direkte Verallgemeinerungen an.

3.2.1 Einfache PSP-Problemstellungen

Um triviale Lösungen ausschließen zu können, benötigen die zwei einfachsten PSP-Problemstellungen eine Schranke N für die minimale Anzahl an erfüllten Teilzielen. Für gegebenes PSP-Problem Ψ ergeben sich z.B.:

- PSPGOAL-Suchproblem: Ausgabe ist ein beliebiger Plan P, der eine Zielmenge $G' \subseteq G$ erfüllt, so dass für gegebenes N gilt: $|G'| \geq N$; oder #, falls kein solcher existiert.

- PSPGOALLEN-Optimierungsproblem: Ausgabe ist ein Plan P, der eine Zielmenge $G' \subseteq G$ erfüllt, so dass für gegebenes N gilt: $|G'| \geq N$; oder #, falls kein solcher existiert. Außerdem gelte $\forall P' \in \mathcal{P}_{G'} : \ L(P') \geq L(P)$.

- PSPUTIL-Optimierungsproblem: Ausgabe ist ein Plan P, der ein $G' \subseteq G$ erfüllt, wobei für alle erfüllbaren $\hat{G} \subseteq G$ gilt: $u(G') \geq u(\hat{G})$

Bei PSPGOAL handelt es sich um die zu PLANGEN analoge PSP-Problemstellung. Man beachte, das PLANGEN damit eine Spezialisierung von PSPGOAL für $N = |G|$ darstellt. Für unser Reiseproblem existieren für jedes $N \leq 4$ mehrere Pläne, die jeweils bis zu N Ziele erfüllen können.

PLANGOALLEN verallgemeinert PLANLEN für teilerfüllendes Planen. Falls $N = |G|$, sind die beiden Problemstellungen identisch. Da im Beispiel jeder Ort immer

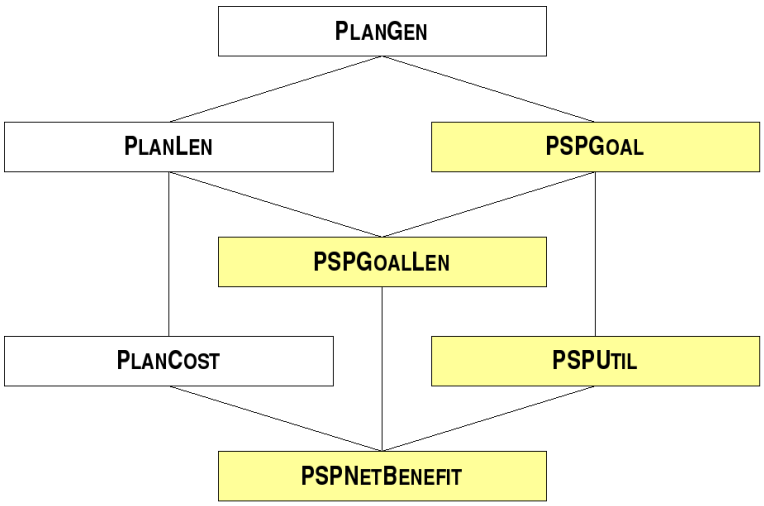


Abbildung 3.2: Taxonomie einiger Problemstellungen der klassischen und teilerfüllenden Handlungsplanung.

mit einer Aktion erreicht werden kann, existiert z.B. genau dann ein G'-erfüllender Plan P mit $L(P) \leq L$, wenn $L \geq |G'|$.

Eine Lösung des PSPUTIL-Suchproblems bildet jeder Plan, der einen Zielzustand mit maximalem Nutzen erreicht. Da hier die Bewertung des Plans über die erreichten Ziele definiert ist, kann wie auch bei den folgenden Problemen auf die Schranke N verzichtet werden. Der Student würde nach maximalem Nutzen streben, ohne sich für die Kosten zu interessieren. Eine Lösung wäre also jeder Plan beliebiger Länge und Kosten, der alle Teilziele erreicht, was im gegebenen Beispiel keine große Herausforderung darstellt.

3.2.2 Nettonutzen

In diesem Abschnitt wird die Problemstellung PSPNETBENEFIT [vSDK04] vorgestellt. Hierzu wird zunächst der Begriff des Nettonutzens (*net-benefit*) definiert, der uns ein Maß für die Güte von Plänen zur Verfügung stellt. Dabei findet ein Abwägen zwischen den Kosten und dem Nutzen eines Planes statt. PSPNETBENEFIT bildet damit eine sehr interessante Problemstellung im Bereich PSP. Die Kosten-Nutzen-Abwägung bietet sich für viele Problembereiche an. Das Sparen von Ressourcen wird flexibel honoriert und ist nicht allein durch eine starre Begrenzung gefordert. Die zentrale Definition ist hierbei folgende:

Definition 12 (Nettonutzen). Gegeben sei ein PS-Problem Ψ und ein Plan P, der zum Zustand s_P führt. Der **Nettonutzen** ist gegeben durch die Abbildung $\hat{b} : \mathcal{P} \rightarrow \mathbb{R}$ mit

$$\hat{b}(P) = \sum_{g:\ s_P \models g} [u(g)] - \hat{c}(P)$$

.

Die Kosten des Plans werden vom Nutzen des erreichten Zustandes subtrahiert. Hiermit lassen sich für gegebenes Ψ die folgende Problemstellungen formulieren:

- PSPNetBenefit-Optimierungsproblem: Ausgabe ist ein Plan P mit maximalem Nettonutzen, d.h. $\forall P' \in \mathcal{P}$ gilt $\hat{b}(P') \leq \hat{b}(P)$.
- PSPNetBenEx-Entscheidungsproblem: Existiert ein Plan P, so dass für gegebenes B gilt: $\hat{b}(P) \geq B$?

Das PSPNetBenEx-Entscheidungsproblem ist wie bereits bewiesen PSPACE-*vollständig* [vSK04].

Für unsere Anwendungen interessiert uns wieder vor allem das Optimierungsproblem. Damit ist auch die Reiseinstanz zu einer kleinen Herausforderung geworden, dessen Lösung nicht mehr sofort erkennbar ist. Der Plan mit dem höchsten Nettonutzen von 190 ist (*travel(LV,DL)*,*travel(DL,SJ)*,*travel(SJ,SF)*). Diese Reiseroute ignoriert das Ziel *SeeZoo(SD)*, erreicht also die Zielmenge $\{g_1, g_2, g_3\}$ und einen Nettonutzen von $500 - 310 = 190$.

Die PSPNetBenefit-Problemstellung wurde zunächst von den Planern *Optiplan* und *AltAlt*^{ps} behandelt [vSK04]. Wenig später kamen weitere PS-Planer wie *Sapa*^{ps} [DK04] oder *AltWlt* [SK05] hinzu. *Optiplan* ist ein *optimaler* teilerfüllender Planer, die anderen genannten suchen Pläne mit hohem aber nicht zwingend maximalem Nettonutzen. Auch diese Arbeit befasst sich im Folgenden mit dem NetBenefit-Problem in der Art, dass der Nettonutzen als Maß zur Bewertung von gegebenenfalls suboptimalen Plänen dient. Alternativ wird hierfür der Begriff PS-Planen verwendet.

3.3 Kosten und Nettonutzen von Zielmengen

Die vorliegende Arbeit befasst sich mit Teilzielmengen beim teilerfüllenden Planen. Im Speziellen wird versucht, Wissen über einzelne Teilziele auf andere zu übertragen und Zielmengen untereinander zu vergleichen. Gesucht werden Zielkombinationen, deren Erfüllung einen hohen Nettonutzen bringt. Um dies vernünftig untersuchen zu können, macht es Sinn, die auf der Menge der Pläne definierte Kostenfunktion auf die Menge aller Teilziele zu übertragen. Jede Zielmenge besitzt dann nicht nur einen eindeutigen Nutzen, sondern auch eindeutige Kosten. Damit lassen sich verschiedene Teilzielmengen auch bezüglich ihres Nettonutzens vergleichen und bewerten. Es können also Aussagen über *gute* und *schlechte* Teilziele mit hohem bzw. niedrigem Nettonutzen getroffen werden.

3.3.1 Die Optimalitätsannahme

Tatsächlich existiert zu jeder erfüllbaren Teilzielmenge G', mindestens ein *optimaler G'-erfüllender Plan* $P^*_{G'}$, dessen eindeutige Kosten $\hat{c}(P^*_{G'})$ sich auf G' übertragen lassen.
Gegeben sei jeweils eine teilerfüllende Planungsinstanz Ψ.

Definition 13 (Teilzielkosten). Die **Teilzielkosten** sind gegeben durch eine Abbildung $c : Pot(G) \rightarrow \mathbb{R}^+ \cup \{\infty\}$. Sei $P^*_{G'}$ ein kostenoptimaler Plan, der $G' \subseteq G$ erfüllt, so sind die Kosten von G'

$$c(G') = \hat{c}(P^*_{G'}).$$

Für nicht erfüllbare G' gilt $c(G') = \infty$.

Hiermit lässt sich nun leicht der Nettonutzen einer Teilzielmenge definieren:

Definition 14 (Teilziel-Nettonutzen). Der **Nettonutzen** für Zielmengen ist gegeben durch eine Abbildung $b : Pot(G) \rightarrow \mathbb{R} \cup \{-\infty\}$. Sei $P^*_{G'}$ ein kostenoptimaler Plan der G' erfüllt, dann ist der Nettonutzen

$$b(G') = \hat{b}(P^*_{G'}).$$

Für nicht erfüllbare G' gilt $b(G') = -\infty$.

Definition 15 (Optimale Zielmenge). Eine **(Nettonutzen-)optimale Zielmenge** G^* ist eine Zielmenge mit maximalem Nettonutzen:

$$G^* = \text{argmax}_{G' \subseteq G}\, b(G')$$

Wegen $b(\emptyset) = \hat{b}(()) = \sum_{g:\ s_{init} \models g} u(g) \geq 0$, existiert G^* immer und es gilt $b(G^*) \geq 0$.

Der optimale Plan zur Erfüllung einer Zielmenge und seine Kosten sind theoretisch mit Hilfe eines optimalen Planers jederzeit zugänglich. In der Praxis muss aus Gründen der Performanz häufig auf suboptimale Planer zurückgegriffen werden. Für die Vergleichbarkeit von Teilzielkosten und Teilziel-Nettonutzen genügt im Prinzip die Forderung der Eindeutigkeit. Zusätzlich sollten die Kosten konsistent sein, das heißt $c(G'') \leq c(G')$, falls $G'' \subseteq G'$. Weist ein klassisches Planungssystem jeder Teilzielmenge eindeutige Kosten zu, können diese als optimal angenommen, und damit als Teilzielkosten verwendet werden.

Durch die Einschränkung, sich nur für optimale Plankosten zu interessieren, wird die PS-Planungsproblematik entscheidend vereinfacht. Der Einfluss von verschiedenen Plänen mit unterschiedlichen Kosten wird gänzlich ausgeklammert. Die Annahme nicht eindeutiger Teilzielkosten bringt einen komplett neuen Aspekt ins Spiel, der in dieser Arbeit jedoch vernachlässigt werden soll.

Die Eindeutigkeit der Kosten bzw. des Nettonutzen einer Zielmenge wird als *Optimalitätsannahme* bezeichnet.

3.3.2 Bewertung von PS-Planern

Die Rückgabe eines PS-Planungssystems ist ein Plan und eine zugehörige erfüllte Teilzielmenge G'. Falls eine optimale Teilzielmenge G^* bekannt ist, kann der relative Nettonutzen ϕ berechnet werden:

Definition 16 (Relativer Nettonutzen). Sei PS ein teilerfüllendes Planungssystem. Bei Eingabe eines PS-Planungsproblems Ψ sei P sein zurückgegebener Plan, s_P der erreichte Zustand und G' die maximal große Zielmenge mit $s_P \models G'$. G^* sei eine optimale Zielmenge. Der erreichte **relative Nettonutzen** ϕ des Planers PS ist dann für $b(G^*) \neq 0$ gegeben durch

$$\phi = \frac{b(G')}{b(G^*)}.$$

Der relative Nettonutzen beschreibt also, wie gut der erreichte Nettonutzen im Vergleich zum maximal erreichbaren Wert ist, und ist für *optimale* teilerfüllende Planer immer 1. Damit ist er ein aussagekräftiges Maß für die Güte eines PS-Planungssystems.

3.4 Korrelation von Teilzielen

Eine zentrale Schwierigkeit der Handlungsplanung ergibt sich durch Kosten-Abhängigkeiten: Zielelemente beeinflussen sich gegenseitig, wenn das Erreichen eines Teilzieles durch das Erreichen anderer erschwert oder vereinfacht wird. Auch beim teilerfüllenden Planen sind die auftretenden Herausforderungen durch die Existenz solcher Abhängigkeiten gegeben. Eine Menge von Zielen ist unabhängig, wenn die Kosten der einzelnen Teilziele sich zu den Kosten der Zielmenge aufaddieren.

Definition 17 (Unabhängigkeit). Die Teilziele einer Menge G' sind **unabhängig** genau dann, wenn

$$\sum_{g \in G'} c(g) = c(G')$$

Definition 18 (Strenge Unabhängigkeit). Die Teilziele einer Menge G' sind **streng unabhängig** genau dann, wenn

$$\forall G'' \subseteq G', \sum_{g \in G''} c(g) = c(G'').$$

Wäre dies im Allgemeinen gegeben, ließen sich die Kosten für jede beliebige Teilzielmenge durch Aufaddieren der Elementkosten berechnen. Nur wenn unter allen Zielen der Teilzielmenge G' keinerlei Abhängigkeiten auftreten, verhalten sich die Kosten auch für alle Teilmengen additiv. Strenge Unabhängigkeit schließt aus, dass sich die Auswirkungen einzelner Abhängigkeiten gegenseitig ausgleichen.

Definition 19 (Abhängigkeit). Eine **Abhängigkeit** in einer Zielmenge G' ist durch eine Teilmenge $G'' \subseteq G'$ mit $\sum_{g \in G''} c(g) \neq c(G'')$ gegeben.

Definition 20 (Positive/Negative Abhängigkeit). Eine Abhängigkeit G' heißt **positiv** genau dann, wenn

$$\sum_{g \in G'} c(g) > c(G'),$$

sie heißt **negativ** genau dann, wenn

$$\sum_{g \in G'} c(g) < c(G').$$

Der Begriff *Abhängigkeit* bezieht sich im Folgenden immer auf die eben definierten Kostenabhängigkeiten und ist nicht mit den in 3.1.1 beschriebenen Nutzenabhängigkeiten zu verwechseln.

Ein typisches Beispiel für positive Beeinflussung ist z.B. in logistischen Problemen wie unserem Reiseproblem gegeben, wenn auf der Route zu einem Ziel andere Ziele erreicht werden.
Sei etwa $G' = \{Attend\_AAAI(SJ), HaveFun(SF)\}$. Vom Startpunkt Las Vegas aus kostet die Reise nach San Jose mindestens 230, die nach San Francisco 80 Einheiten. Ist SJ bereits erreicht, kostet die Reise nach SF hingegen nur weitere 20 Einheiten. Also

$$c(Attend\_AAAI(SJ)) + c(HaveFun(SF)) = 230 + 80 = 310$$

$$c(\{Attend\_AAAI(SJ), HaveFun(SF)\}) = 230 + 20 = 250$$

Es findet sich aber auch ein Beispiel für negative Abhängigkeit. Das Erreichen von Disneyland und San Diego ergibt folgendes Ergebnis:

$$c(SeeZoo(SD)) + c(HaveFun(DL)) = 40 + 90 = 130$$

$$c(SeeZoo(SD), HaveFun(DL)) = 40 + 110 = 150$$

Negative Abhängigkeiten finden sich auch häufig bei SATELLITES-Problemen, da verwendete Instrumente zwischen verschiedenen Aufnahmen an Fixpunkten kalibriert werden müssen. Auch notwendige Leerfahrten in Logistik-Problemen führen zu solchen Korrelationen.

3.5 Kodierung von PSP-Problemen

Es sei an dieser Stelle nochmals auf die praktische Anwendbarkeit des PS-Planens im Allgemeinen und des PSPNETBENEFIT-Problems im Speziellen hingewiesen. Mit den bereits erwähnten Möglichkeiten, die durch die Kosten- und Nutzenfunktion gegeben sind, lassen sich unterschiedliche Aufgabenstellungen spezifizieren. Die Kostenfunktion kann zur Modellierung verschiedenster Ressourcen, wie Energie, Zeit, Geld, Arbeit oder Ähnlichem verwendet werden. Durch die Nutzenfunktion lässt sich die Wichtigkeit von Teilzielen unterscheiden. Arbeitsaufträge lassen sich somit z.B. in Haupt- und Nebenaufgaben unterteilen. Des weiteren können Nutzenabhängigkeiten zwischen den Zielen vorgegeben werden, da manche Errungenschaften eben nur gemeinsam oder getrennt sinnvoll sind.

Es gibt in der Praxis kaum Probleme, für deren Lösung unbegrenzt Zeit oder Geld zur Verfügung steht. Tatsächlich ist meist gefordert, dass an Ressourcen gespart wird. Die Entscheidung, welche Ziele genau erfüllt werden, wird also auf das

PS-Planungssystem übertragen und im Falle von PSPNetBenefit durch eine einfache Kosten-Nutzen-Abwägung gelöst. Hierbei können nicht nur starre Grenzen berücksichtigt werden. Durch Maximierung des Nettonutzens werden Pläne gesucht, in denen der erreichte Nutzen die aufgewandten Kosten rechtfertigt.

Mit den Werten für Nutzen und Kosten ist es möglich, vielfältige Vorgaben, Präferenzen und Grenzen darzustellen, über die Einfluss auf den Planungsvorgang genommen werden kann. Bei der Verwendung eines PS-Planers muss nicht mehr im Voraus beachtet werden, ob die Menge der Ziele grundsätzlich oder mit gegebenen Ressourcen überhaupt erreicht werden kann. Es können damit durchaus widersprüchliche oder scheinbar unlösbare Ziele gesetzt werden. Das Planungssystem versucht selbstständig aus den gegebenen Umständen *das Beste herauszuholen*. Dies ist im Zuge der zunehmenden Autonomisierung, z.B. im Bereich der Robotik, folgerichtig und notwendig.

3.5.1 Kodierung durch Kosten und Nutzen

Man beachte, das eine Nutzenfunktion in der vorgestellten Form kein Bestandteil der Beschreibungssprache PDDL ist. Wir werden jedoch sehen, dass PDDL3 und auch PDDL2.1 grundsätzlich die Möglichkeit bieten, ein solches Konzept auszudrücken.

Die Teilzielauswahl eines teilerfüllenden Planers lässt sich mit Hilfe von Kosten und Nutzen vielfältig beeinflussen. Das System kann etwa angeleitet werden, bestimmte Ziele oder Zielkombinationen zu bevorzugen oder zu vermeiden. Mit einer auf $Pot(G)$ definierten Nutzenfunktion kann theoretisch jeder Kombination von Zielen ein spezieller Nutzen zugeordnet werden. Präferenzen und Abhängigkeiten können also mit Hilfe der Nutzenfunktion ausgedrückt werden. Dieses Konzept ist zwar sehr ausdrucksstark, aus Sicht eines Anwenders jedoch auch anspruchsvoll, da alles in Zahlenwerten beschrieben werden muss. Eine Abhängigkeitsvorschrift, wie „das Ziel g_1 soll nur erfüllt werden, wenn auch g_2 erfüllt ist“, müsste z.B. folgendermaßen kodiert werden:

$u(g_1) = 0; u(g_2) = k; u(\{g_1, g_2\}) = K$, wobei $K > k$. Zusätzlich muss die Regel in allen Obermengen von g_1 und g_2 beachtet werden. Wie wir in 3.6 sehen werden, ergibt sich noch eine weitere Schwierigkeit, da die Werte für Kosten und Nutzen aufeinander abgestimmt werden müssen. Es kann dabei gewichtet werden, welcher Nutzen welche Kosten rechtfertigt.

3.5.2 Planen mit Ziel-Präferenzen und Abhängigkeiten.

Eine wesentlich natürlichere Möglichkeit zur Problemspezifikation bietet das *Planen mit Abhängigkeiten und Präferenzen*. Die Arbeit von Brafman und Chernyavsky [BC05] ermöglicht es, bei der Problembeschreibung Präferenzen und Abhängigkeiten direkt anzugeben. Hierzu werden sogenannte TCP-Netze verwendet. *Planen mit Präferenzen und Abhängigkeiten* ist eng mit teilerfüllendem Planen verwandt und stellt einen Lösungsansatz für dieselbe Klasse von Problemen dar. Wie beim PS-Planen wird vom Planungssystem selbst zur Laufzeit ausgewählt, welche Ziele erfüllt

werden. Die Unterschiede liegen in erster Linie in der Art der Spezifikation.

3.5.3 PDDL3: Plan-Abhängigkeiten und Präferenzen

Die für die fünfte *International Planning Competition* entwickelte Beschreibungssprache PDDL3 bietet die Möglichkeit, verschiedene Vorlieben und Abhängigkeiten direkt zu spezifizieren. Die Gewichtung erfolgt auch hier über die Angabe numerischer Werte. Die Sprache ermöglicht die Definition von Zielen, die erfüllt werden *müssen* neben solchen, die erfüllt werden *sollten*. Außerdem lassen sich verschiedene Abhängigkeiten beschreiben, die sich auch auf Zwischenzustände während der Planausführung beziehen können. Die Planqualität kann schließlich an der Anzahl von verletzten Präferenzen und Abhängigkeiten bemessen werden. Die Metrik zur Bewertung der Pläne lässt sich dann so gestalten, dass für nicht erfüllte Präferenzen *Kosten* entstehen. Kambhampati und Do nutzen dies bei der Entwicklung des PS-Planers $Sapa^{ps}$. Das System wandelt die Elemente der Zielmenge intern in PDDL3-*soft-constraints* um [DK04].

3.5.3.1 Übersetzung in PDDL3

Teilerfüllende Planungsinstanzen lassen sich in PDDL3 übertragen. Hierzu muss aber die konkrete Problemstellung, z.B. PSPNetBenefit miteincodiert werden. Bei der Übersetzung werden alle Elemente der Zielbeschreibung in Präferenzen verwandelt. Die Metrik der PDDL3-Planungsinstanz kann dann so gewählt werden, dass sie dem Nettonutzen entspricht. Ein Planungssystem, welches den besten Plan bezüglich der gegebenen Metrik sucht, bearbeitet dann *automatisch* das PSPNetBenefit-Optimierungsproblem. Das folgende Beispiel zeigt die Übersetzung der Zielbeschreibung eines kleinen Satellites-Problems in PDDL3. Die hier für das PS-Problem verwendete Schreibweise, in der auf jedes Teilziel der entsprechende Nutzen folgt, ist nicht PDDL-konform.

```
(:goal (and
        ((have_image Planet3 infrared0) 79.5087 )
        ((have_image Planet4 infrared0) 37.3212 )
        ((have_image Phenomenon5 image2) 74.1062 )
        ((have_image Phenomenon6 infrared0) 75.5580 )
        ((have_image Star7 infrared0) 86.2710 )
       )
)
(:metric minimize (fuel-used))
```

Die Metrik verlangt die durch `fuel-used` gegebenen Kosten zu minimieren. Der Nettonutzen eines Planes ergibt sich dann aus dem Nutzen der erreichten Ziele abzüglich des verbrauchten Treibstoffs. Dies lässt sich mit PDDL3 wie folgt in die Metrik eincodieren:

```
(:goal (and
        (preference g0 (have_image Planet3 infrared0))
        (preference g1 (have_image Planet4 infrared0))
        (preference g2 (have_image Phenomenon5 image2))
        (preference g3 (have_image Phenomenon6 infrared0))
        (preference g4 (have_image Star7 infrared0))
        )
)
(:metric maximize    (+    (* (is-violated g0) -79.5087)
                           (* (is-violated g1) -37.3212)
                           (* (is-violated g2) -74.1062)
                           (* (is-violated g3) -75.5580)
                           (* (is-violated g4) -86.2710)
                           (*  (fuel-used) -1 )
                           352.7651              ;;Summe aller Nutzen
                      ) )
```

Der zu maximierende Ausdruck beschreibt nichts anderes als den Nettonutzen in einer etwas umständlichen Formulierung, die der PDDL3-Syntax geschuldet ist. Es gilt im Übrigen auch die umgekehrte Übersetzbarkeit von PDDL3-Instanzen der Kategorie *simple preferences* in PSP-Probleme. Das von Benton, Khambhampati und Do entwickelte System *Yochan*^{PS} [BKD06] führt diese Umcodierung automatisch durch.

3.5.4 Übersetzung von Nutzen in Kosten

Betrachtet man den Ausdruck, der den Nettonutzen als Metrik beschreibt, kann man erkennen, dass die Konzepte *Nutzen* und *Kosten* nichts grundlegend unterschiedliches beschreiben. Die folgende Überlegung zeigt, dass theoretisch auch in PDDL2.1 eine Spezifikation von PS-Problemen mit Nettonutzen möglich ist. Tatsächlich lassen sich die Nutzen für Zielmengen in Aktionsausführungskosten übertragen. Durch Zulassung negativer Kosten, erhält man die Möglichkeit, für jedes Zielelement einen Operator einzuführen, der dessen Nutzen repräsentiert. Soll etwa für die Zielmenge $g1$ gelten $u(g1) = k$, wird der Operator `util2costg1` wie folgt definiert:
prec(`util2costg1`) = $g1$, `(not(haveUtilg1))` ; *eff*(`util2costg1`) = `haveUtilg1` ;
c(`util2costg1`) = $-k$. Die Zielmenge erweitert sich um g_{neu} = `haveUtilg1`.
Die Kosten eines Planes in einer solchen Instanz entsprechen dem Nettonutzen des zugrundeliegenden PS-Problems. Ein Planer, der hier versucht, die Kosten zu minimieren, bearbeitet das PSPNetBenefit-Optimierungsproblem.

3.6 Verteilung des Nettonutzens

In diesem Abschnitt wird betrachtet, wie sich der Nettonutzen auf die verschiedenen Teilmengen der Gesamtzielmenge verteilt und wie sich die Nutzenfunktion auf diese Verteilung auswirkt. Wieder wird von additivem Nutzen ausgegangen, so dass dieser

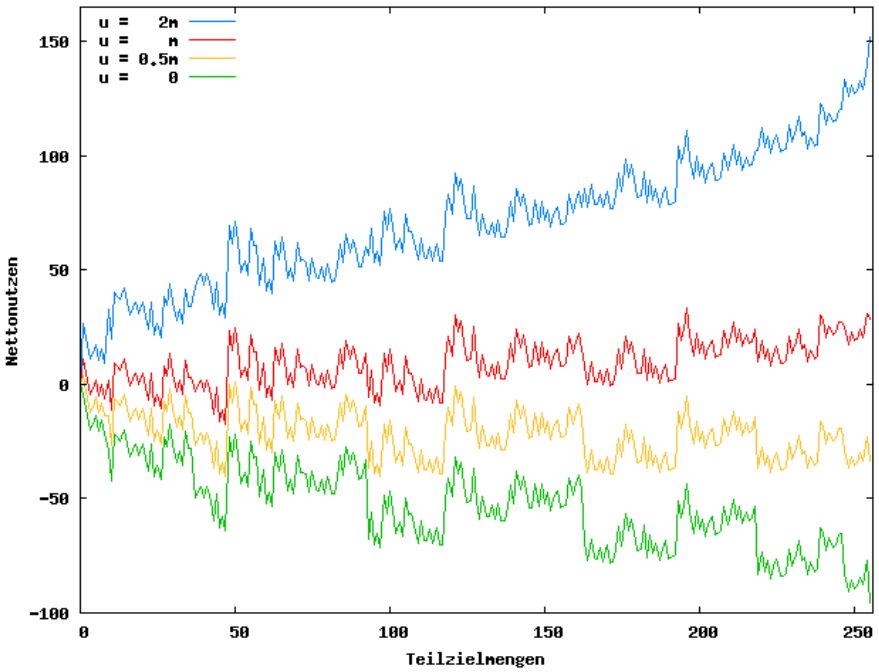


Abbildung 3.3: Nettonutzen der 256 nach Kardinalität sortierten Teilzielmengen eines SATELLITES-Problems für verschiedene u.

nur für die Elementarziele definiert werden muss. Die Aktionskosten seien unveränderlich. Zur Vereinfachung erhalten hier alle Elemente der Zielmenge den gleichen Nutzenwert, u ist also eine konstante Funktion. Werden nun etwa alle Zielelemente als gänzlich nutzlos definiert ($u \equiv 0$), ergibt sich ein nicht-positiver Nettonutzen für alle Teilziele. Eine Teilzielmenge mit maximalen Nettonutzen ist dann die leere Menge mit $b(\emptyset) = 0$. Wählt man hingegen den Nutzen der Elemente relativ zu deren Kosten übermäßig groß, steigt der Nettonutzen mit wachsender Teilzielgröße an. Das beste Ergebnis liefert dann im Grenzwert das vollständige Ziel G, da die Plankosten nicht mehr ins Gewicht fallen.

Abbildung 3.3 zeigt für ein SATELLITES-Problem mit Zielgröße 8 die Verteilung des Nettonutzens, wobei dieser für jede der aufsteigend nach Mächtigkeit angeordneten 256 Teilzielmengen aufgetragen ist. Der Nutzen u der Zielelemente wurde relativ zur Konstanten $m = \frac{1}{|G|} \sum_{g \in G} c(g)$, dem Mittelwert der Kosten für einzelne Zielelemente, gewählt. Man sieht, dass auch innerhalb der Teilzielmengen gleicher Größe durch unterschiedliche Kosten hohe Schwankungen des Nettonutzens auftreten.

Es zeigt sich, dass die nettonutzenoptimale Zielmenge G^* leicht die Extreme $\emptyset$ oder G annimmt, wenn die Kosten und Nutzenfunktion nicht eng aufeinander abgestimmt werden.

Die folgende Tabelle zeigt die Mächtigkeit von G^* in einer SATELLITES-Instanz für verschiedene Elementnutzenwerte u.

u:	$\leq 0.2m$	$0.5m$	$0.7m$	m	$1.1m$	$\geq 1.2m$
$\lvert G^* \rvert$	0	1	4	5	7	8 ($= \lvert G \rvert$)

3.6.1 Größenordnungen von Nutzen und Kosten

Die Größenordnung der Kosten, die zur Erfüllung von Zielen benötigt werden, unterscheidet sich domänenabhängig sehr stark. Das PSPNetBenefit-Optimierungsproblem stellt vor allem dann eine Herausforderung dar, wenn der optimale Wert nicht offensichtlich für eine der zwei extremen Teilzielmengen $\emptyset$ oder G erreicht wird.

Der einfach zu berechnende Wert m bietet einen Anhaltspunkt zur Wahl des Nutzens für Zielelemente.

In Abbildung 3.3 sieht man, dass für $u \equiv m$ (rote Kurve) der mittlere Nettonutzen mit wachsender Teilzielgröße nur sehr leicht ansteigt. Tatsächlich ist hier auch die optimale Zielmenge mit Mächtigkeit 5 mittelgroß. In anderen Instanzen kann die Wahl $u \equiv m$ aber auch zu einer stark ansteigenden oder abfallenden Kurve führen. Relevant hierfür ist, ob der Einfluss positiver oder negativer Kosten-Abhängigkeiten überwiegt. Die zwei Kurven in Abbildung 3.4 zeigen die Verteilung des Nettonutzens für zwei Probleme der Domäne ZenoTravel, jeweils mit $u \equiv m$. Die Kurve für die eine Instanz steigt mit wachsender Kardinalität der Teilziele immer mehr an. Offensichtlich überwiegen positive Abhängigkeiten. In der zweiten Graphik sieht man, dass die negativen Abhängigkeiten überwiegen. Sie treten allerdings vermehrt in größeren Teilzielen ab vier Elementen auf. Offensichtlich handelt es sich also vor allem um Abhängigkeiten, welche die Korrelation zwischen mindestens vier Zielelementen beschreiben.

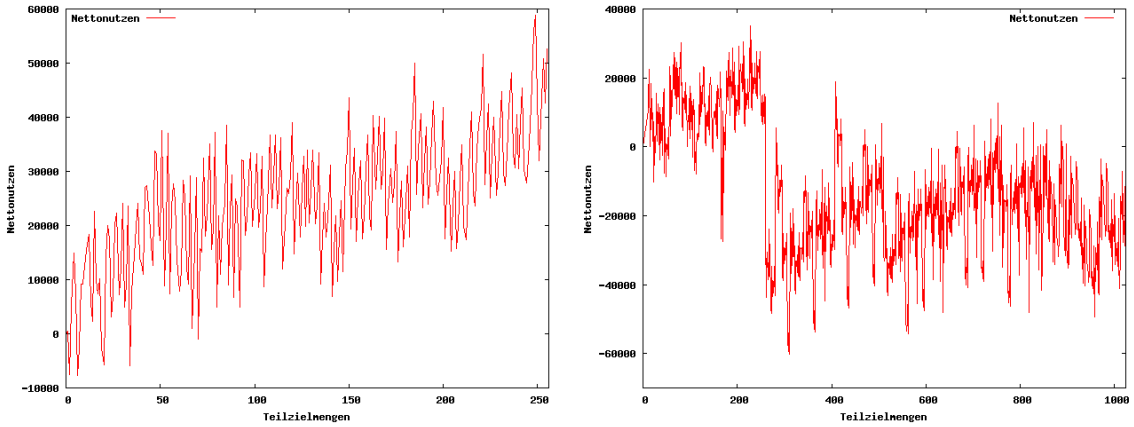


Abbildung 3.4: Nettonutzen für die nach Kardinalität sortierten Teilzielmengen zweier ZenoTravel-Instanzen mit $u = m$.

3.7 Lösen von PS-Problemen

Die derzeit verwendeten Planungssysteme wurden nicht für teilerfüllendes Planen entwickelt. Eine Erweiterung zur Anwendbarkeit auf PSP-Probleme geht mit den folgenden Herausforderungen einher [DK04]:

- Das Kriterium zur Terminierung des Planungsprozesses muss geändert werden, da die Menge der konjunktiven Ziele nicht mehr festgelegt ist.
- Die Heuristiken des traditionellen Planens wurden in Hinblick auf eine feste Zielmenge entwickelt und sind daher zunächst ungeeignet.
- Zur Bewertung von Plänen müssen sowohl die Kosten der ausgeführten Aktionen als auch der Nutzen der erreichten Zielmenge verwendet werden.

Die Schwierigkeit besteht nun zum einen darin, dass die Kosten einer Zielmenge G' nur über die Kosten eines G'-erfüllenden Plans, der erst erstellt werden muss, zugänglich sind. Zum Zweiten ist die Anzahl der Teilzielmengen in der Regel sehr groß.

3.7.1 PS-Planung als Suche im Raum der Teilzielmengen

Vergleicht man teilerfüllende mit klassischer Handlungsplanung, kommt vor allem ein Element hinzu: Die Auswahl der zu erfüllenden Zielmenge ist die neue Herausforderung des PS-Planens. Diese Auswahl entspricht einer Suche im Raum, der aus allen Teilzielen $G' \subseteq G$ besteht. Nach der Auswahl von G' bieten sich die verschiedenen Möglichkeiten zur Lösung klassischer Planungsinstanzen an.

Ein naiver Ansatz wäre, jedes mögliche Teilziel G' als Ziel eines herkömmlichen Planungsproblems Π zu definieren. Ein Planer könnte nun einen Plan erstellen, der G' mit minimalen Kosten k löst. Der Nettonutzen wäre unmittelbar zugänglich. Besteht das Ziel G aus N Teilzielen, muss hierzu allerdings das PlanCost-Optimierungsproblem 2^N-mal gelöst werden, was schon für moderat große N extrem aufwendig wird. Es müssen also Möglichkeiten gefunden werden, die Teilziele *vor* dem Planen zu bewerten. Hierzu benötigt man z.B. Heuristiken oder Modelle, die vielversprechende Teilziele identifizieren, für welche dann die Plankosten berechnet werden. Ein Hauptbestandteil dieser Arbeit ist die Untersuchung von Modellen, die mit Hilfe von Basiswissen, bestehend aus den Plankosten ausgewählter Zielmengen, berechnet werden.

3.7.2 Bestehende Lösungsansätze und Planer

In diesem Abschnitt sollen beispielhaft PS-Planer und ihre Funktionsweise dargestellt werden. Wie in der klassischen Handlungsplanung bietet sich auch beim teilerfüllenden Planen das Prinzip der heuristischen Suche [BG01] an. Die Pfadsuche im Zustandsraum wird hierbei durch Heuristiken geleitet, welche die Qualität von Pfaden *schätzen* können. Bei der Berechnung der für PS-Planung verwendeten Heuristiken sollten sowohl die Kosten als auch die Nutzen der Zielteilmengen einbezogen werden.

Das Problem lässt sich auch umgekehrt angehen – durch eine Suche im Raum der Aktionsfolgen ausgehend vom Startzustand, wobei geprüft wird, welche Teilzielmenge erfüllt ist.

3.7.2.1 Planen mit Hilfe des Orienteering Problems

Die Arbeit von David E. Smith [Smi04] ist motiviert durch Problemstellungen im Bereich der Raumfahrt mit begrenzten Ressourcen. Entwickelt wird eine Heuristik zur Schätzung der Kosten für Teilziele. Hierzu wird zunächst der Planungsgraph erstellt. Auf dessen Basis wird in einem zweiten Schritt das *Orienteering Problem*, eine Variante des Handlungsreisenden-Problems mit Nutzen und Kosten, entwickelt. Die durch den Planungsgraphen berechneten Kosten für einzelne Zustandsübergänge bilden hierbei die Kantengewichte des *Orienteering-Graphen*. Das *Orienteering Problem* bildet eine Abstraktion der PS-Planungsinstanz. Seine Lösungen, Pfade mit hohem Nutzen bei begrenzten Kosten, sind nicht notwendig Lösungen des entsprechenden PSP-Problems, eignen sich jedoch als aussagekräftige Heuristik für ein Planungssystem.

3.7.2.2 OptiPlan

Der Planer *OptiPlan* [vSK04, vSDK04] ist für gegebene maximale Planlänge L ein *optimaler* PS-Planer. Das System formuliert die teilerfüllende Planungsinstanz als Aufgabenstellung der *ganzzahligen linearen Programmierung* und löst diese. Variablen für Zustandsübergänge drücken die möglichen Veränderungen von Zustandsvariablen x durch Operatoren aus. Mit weiteren Variablen werden die Abhängigkeiten des Zustandsübergangs-Modells beschrieben, so dass zulässige Pläne garantiert sind. Außerdem kann die Problemstellung spezifiziert werden, so dass z.B. PSPNetBenefit-Optimierung durchgeführt wird. Die von *OptiPlan* ausgegebenen Pläne sind dann stets nettonutzenoptimal.

3.7.2.3 AltAlt^{ps}

Der von van den Briel, Sanches, Do und Khambhampati [vSK04, vSDK04] vorgestellte Planer *AltAlt*^{ps} basiert auf dem klassischen Planer *AltAlt* [NKS02]. Das Konzept der *cost-propagation* ermöglicht die Verwendung kostensensibler Heuristiken die zu einer Variante des *relaxierten* Plans [HN01] führen.

Das System verwendet einen *Hillclimbing*-Algorithmus zur Auswahl der Teilzielmenge: Ausgehend vom Zielelement mit dem höchsten Nettonutzen, wird die Zielmenge iterativ vergrößert. In jedem Schritt wird für alle möglichen Erweiterungen ein *relaxierter* Plan R erstellt, dessen Kosten zur Berechnung des geschätzten Nettonutzens b_R dienen. Die Zielmenge G' mit dem höchsten b_R bildet die Grundlage für Erweiterungen in der nächsten Iteration. Das Besondere ist, dass die neue Zielmenge stets als Erweiterung von G' betrachtet wird: Zur Schätzung der *verbleibenden* Kosten des neuen Ziels g auf Basis der bekannten Zielmenge G' wird der relaxierte Plan für $G' \cup g$ berechnet. Hierbei werden die Aktionen bevorzugt behandelt, die

bereits im relaxierten Plan für G' vorkommen. Der Algorithmus terminiert, wenn durch Hinzunahme weiterer Zielelemente keine Verbesserung von b_R eintritt oder keine Erweiterungen mehr möglich sind.

Für die im ersten Schritt identifizierte Teilzielmenge wird nun ein Plan erstellt. Hierbei wird wiederum die kostensensible Heuristik zur Steuerung der Suche im Zustandsraum verwendet. Das Zielauswahlverfahren garantiert keine Optimalität, was gegebenenfalls die Ausgabe suboptimaler Pläne zur Folge hat.

Mit *AltWlt* [SK05] existiert eine weiterentwickelte Variante des Systems. Das Zielauswahlverfahren wurde dahingehend erweitert, dass die Menge der berücksichtigten Teilziele etwas breiter angelegt ist. Dadurch wird vermieden, dass das System bei der Auswahl der Ziele zu gierig vorgeht.

3.7.2.4 Sapa^{ps}

Eine Schwäche von *AltAlt*^{ps} liegt in der frühen Festlegung auf eine Teilzielmenge, die nicht mehr geändert werden kann. Das System *Sapa*^{ps} [DK04, vSDK04] wählt daher die umgekehrte Vorgehensweise: Jede in s_{init} ausführbare Aktionsfolge wird als mögliche Lösung angenommen; als Qualitätsmaß dient der erreichte Nettonutzen. Statt vorher eine Zielmenge auszuwählen, werden die einzelnen Zielelemente bei der Planung als PDDL3-*soft-constraints* behandelt.

Das System führt eine progressive A^*-Suche im Raum dieser Aktionsfolgen aus. Der g-Wert beschreibt hierbei den Nettonutzen der im erreichten Zustand s erfüllten Ziele. Der h-Wert schätzt, welcher Nettonutzen zusätzlich durch Erweiterungen des Planes P_s möglich ist. Die Heuristik wird aus dem relaxierten Plan für die gesamte Zielmenge G extrahiert, indem Aktionen identifiziert werden, die allein der Erfüllung einzelner Ziele dienen. Nach der Entfernung von Zielelementen mit negativem Nettonutzen kann ein effektiver, wenn auch nicht zulässiger (*inadmissible*) h-Wert abgeleitet werden. Der ausgegebene Plan kann also durchaus suboptimal sein.

Zur speziellen Behandlung von Problemen mit ausgezeichneten Nutzenabhängigkeiten existiert eine Erweiterung des Planungssystem namens *Sapa*$^{ps}_{UD}$ [Ben06].

4 Teilerfüllendes Planen mit klassischem Planer und Basiswissen

In diesem Kapitel werden neue Konzepte zur Lösung teilerfüllender Planungsprobleme vorgestellt.

Alle Verfahren nutzen ein klassisches Planungssystem zur Berechnung der Teilzielkosten für vorher ausgewählte Zielmengen. Gesucht ist jeweils ein Plan, der die optimale Teilzielmenge mit dem höchsten erreichbaren b-Wert erfüllt, bzw. einen hohen relativen Nettonutzen aufweist. Es wird jeweils von einer additiven Nutzenfunktion ausgegangen. Es gilt also $u(G') = \sum_{g \in G'} u(g)$. Außerdem wird ein klassischer Planer verwendet, der gemäß der Optimalitätsannahme für gegebene Teilzielmengen das PlanCost-Optimierungsproblem löst.

Die zentrale Idee der vorliegenden Arbeit ist, diese Suche im Teilzielraum in zwei Schritte aufzuteilen.

1. Schaffen einer Wissensbasis $\mathcal{K}$ durch klassische Planung einiger Elemente des Teilzielraums $Pot(G)$.

2. Ausnutzen der Information aus $\mathcal{K}$ zur Identifikation einer oder mehrerer Teilzielmengen mit hohem Nettonutzen.

Der erste Schritt wird als *Basisphase* bezeichnet. Die Bezeichnung für den zweiten Schritt richtet sich nach dem verwendeten Algorithmus.

Das Kapitel erläutert zunächst die Verwendung klassischer Planungssysteme innerhalb der teilerfüllenden Planung. Es folgen Abhandlungen zum Aufbau und zur Erzeugung verschieden strukturierter Wissensbasen. In Abschnitt 4.3 wird gezeigt, wie das Basiswissen zur Steuerung verschiedener Hillclimbing-Varianten verwendet wird. Anschließend werden Möglichkeiten untersucht, wie *Nettonutzenmodelle* aus der Wissensbasis erzeugt werden können. Hierbei wird zwischen *statischer* und *dynamischer* Modellierung unterschieden. Zum Abschluss des Kapitels wird aufgezeigt, welche Grenzen der Modellierung aus Basiswissen gesetzt sind.

4.1 Nutzung klassischer Planungssysteme

Im Vergleich zur teilerfüllenden Handlungsplanung ist die klassische Variante ein sehr gut untersuchtes Forschungsfeld mit zahlreichen und guten Lösungskonzepten.

Der Einsatz eines aktuellen klassischen Planers ermöglicht also die Verwendung von weit entwickelten Konzepten, die häufig schnell gute Ergebnisse liefern können. Dennoch ist und bleibt die klassische Handlungsplanung ein komplexes Problem, das oft zu hohen Laufzeiten führt [Byl94, Hel03, Hel06]. In den vorgestellten Verfahren zur teilerfüllenden Planung entfällt tatsächlich ein Großteil der Laufzeit auf die Plangenerierung. Es muss also darauf geachtet werden, dass der klassische Planer nicht zu oft, vor allem nicht mit schwierigen Problemen aufgerufen wird.

4.1.1 Das Blackbox-Prinzip

Die hier vorgestellten Ideen zur teilerfüllenden Planung binden den verwendeten klassischen Handlungsplaner stets als *Blackbox* ein. Die Eingabeprobleme werden als PDDL-Dateien erzeugt und übergeben. Die Ausgabe des Planungsprogramms besteht aus einem Plan und seinen Kosten. Wie der Planer intern funktioniert und zu seinen Ergebnissen gelangt, spielt keine Rolle. Das System ist damit beliebig austauschbar, solange es sich an die gängigen Ein- und Ausgabekonventionen für PDDL-Probleme hält. Es wird also kein konkreter Planer für die Anforderungen des PS-Planens erweitert oder weiterentwickelt, sondern es wird ein System um einen beliebigen klassischen Handlungsplaner herum entwickelt.

Dies bietet den Vorteil, dass unterschiedliche Planer sehr einfach eingebunden werden können und sich somit für verschiedenartige Probleme die Stärken und Vorteile bestimmter Systeme ausnutzen lassen. Auch Weiterentwicklungen in diesem Bereich können quasi unmittelbar verwendet werden. Nachteile zeigen sich jedoch in der Anwendung, wo durch das *Blackbox*-Prinzip die Leistungsfähigkeit des Planungssystems beeinträchtigt werden kann. Jede vorgegebene Teilzielmenge wird separat geplant, ohne dass der klassische Planer Informationen aus dem Planungsvorgang vorhergehender Teilziele nutzen kann. Ein Planer wie SGPLAN [CWH06] etwa, der intern ohnehin für Teilziele plant, kann dieses Wissen nicht mehr ausnutzen, wenn er neu gestartet wird.

Zudem nutzen die hier vorgestellten Verfahren lediglich die errechneten Kosten des ausgegebenen Plans. Auf eine Analyse der Plansequenz wird verzichtet.

4.1.2 Ein naiver optimaler PS-Planer

Der hier kurz vorgestellte Algorithmus zur exakten Lösung des PSPNetBenefit-Optimierungsproblems benötigt für ein PS-Problem mit Zielmengengröße $|G| = N$ genau 2^N Planungen. Dadurch ist er extrem ineffizient. Das Verfahren wird dennoch erwähnt, da mit seiner Hilfe der optimale Nettonutzen und die zugehörigen Teilzielmengen zugänglich sind. Damit ist eine Bewertung schnellerer suboptimaler Algorithmen durch Vergleich mit den optimalen Werten möglich.

Das Verfahren führt für jede Teilzielmenge $G' \subseteq G$ eine klassische Planung durch und identifiziert so garantiert die Teilmenge mit optimalem Nettonutzen.

4.2 Basiswissen

Die Begriffe *Basiswissen* oder *Wissensbasis* stehen in dieser Arbeit für gesicherte Informationen über Kosten und Nutzen bestimmter Teilzielmengen. Techniken wie *cost-propagation* [vSDK04] zur heuristischen Kostenberechnung sind hierzu nicht geeignet. Die einzige Möglichkeit zur exakten Berechnung der Kosten einer Zielmenge G' ist die Erstellung eines Plans, der diese erfüllt. Hierfür gibt es keine Alternative, als eine klassische Planung durchzuführen. Bei der Wahl der Basiswissensgröße muss also die Laufzeit der Basisphase im Auge behalten werden.

Definition 21 (Wissensbasis). Die Wissensbasis ist formal gegeben durch eine partielle Abbildung $\mathcal{K} : Pot(G) \nrightarrow (\mathbb{R}^+ \cup \{\infty\}) \times \mathbb{R}^+$.
Für eine Teilzielmenge $G' \subseteq G$ ergibt sich der Wert des Tupels durch die Kosten- und Nutzenwerte: $\mathcal{K}(G') = (c(G'), u(G'))$.

Basiswissen unterscheidet sich durch die enthaltenen Teilziele und kann sehr verschiedene Größen annehmen. Die hier verwendeten Wissensbasen speichern die Teilziele nur mit den korrespondierenden Kosten und Nutzen. Denkbar wäre auch eine Speicherung der Pläne, durch deren Analyse weitere Erkenntnisse möglich sind.

Erkenntnisse aus dem Basiswissen sollen schließlich helfen, Voraussagen über möglichst viele Teilzielmengen treffen zu können. Dazu wird davon ausgegangen, dass sich die Kosten einzelner Teilzielmengen in irgendeiner Weise auf andere übertragen lassen. Die Schwierigkeiten dieses Schließens ergeben sich wiederum durch Abhängigkeiten.

4.2.1 Größe des Basiswissens

Bei der Wahl der Größe des Basiswissens gilt es zwischen Laufzeit und enthaltener Information abzuwägen. Da die Wissensbasis ausschließlich gesicherte Informationen enthält, ist jedes ihrer Elemente aufgrund des laufzeitintensiven Planungsvorgangs sehr *teuer*. In der Regel steigt außerdem der Aufwand zur Planung für eine Zielmenge mit deren Kardinalität an. Eine *vollinformierte* Wissensbasis, bestehend aus allen $2^{|G|}$ Zielmengen, die sich z.B. durch Anwendung des oben erwähnten naiven optimalen Algorithmus erstellen lässt, stößt also in der Praxis schnell an ihre Grenzen.

Zusätzlich sollte bei der Auswahl der Basiszielmengen darauf geachtet werden, dass diese vor allem dann sinnvoll verwendbar sind, wenn für möglichst viele Teilzielmengen überhaupt Information abgeleitet werden kann. Hierzu sollten möglichst alle Zielelemente in irgendeiner Form in der Basis vertreten sein. Als einfache Wissensbasis drängt sich also die Menge der einelementigen Teilziele $\{g1, \ldots, g_r\}$ auf.

4.2.2 Basis aus einzelnen Teilzielen

Wählt man $dom(\mathcal{K}) = \{G' \subseteq G : \ |G'| = 1\}$ erhält man eine Basis aus Zielelementen, die sehr einfach anwendbares und meistens schnell erzeugbares Wissen enthält. Sie wird mit $\mathcal{K}^1$ bezeichnet und enthält zu jedem Elementarziel einen Informationssatz,

bestehend aus Kosten und Nutzen. Damit lässt sich die Basis zur Modellierung jeder Teilzielmenge verwenden.

Bei $|G| = N$ müssen in der Basisphase also genau N klassische Planungen durchgeführt werden. Pläne zur Erfüllung einzelner Ziele sind meist einfach und kurz, so dass die Erzeugung des Basiswissens in der Regel schnell möglich ist. Der Algorithmus INITK1 erzeugt $\mathcal{K}^1$ als Vektor von Nettonutzen:

```
1: procedure INITK1(Zielmenge G)
2:     vector<float> K^1
3:     for each g ∈ G do
4:         P(g) = generatePlan(g)
5:         K^1.insert(b̂(P(g)))
6:     return K^1
```

Die Schwäche einer solchen Wissensbasis liegt auf der Hand: Sie enthält keinerlei Information über positive oder negative Abhängigkeiten.

4.2.3 Basis aus Paaren von Teilzielen

Um Aussagen über Abhängigkeiten treffen zu können, muss notwendig Kosteninformation über mehrelementige Teilzielmengen zur Verfügung stehen. Eine Wissensbasis bestehend aus allen ein- und zweielementigen Zielen bietet etwa die Möglichkeit zu bewerten, ob sich zwei Ziele positiv oder negativ beeinflussen.

Wir werden sehen, dass die Informationen der Wissensbasis $\mathcal{K}^{1,2}$, bestehend aus den Teilzielen der Mächtigkeiten 1 und 2, in der Praxis häufig ausreichen, um ein Modell zu erstellen, das eine Zielmenge mit hohem Nettonutzen identifizieren kann.

Problematisch ist hingegen der große Aufwand, der zur Basiserstellung notwendig ist. Eine Zielmenge der Größe N besitzt $N(N-1)$ zweielementige Teilmengen. Zur Initialisierung der aus ein- und zweielementigen Zielen bestehenden Wissensbasis $\mathcal{K}^{1,2}$ müssen also $N(N-1) + N = N^2$ klassische Planungen durchgeführt werden. Auch wenn diese im einzelnen leicht zu lösen sind, führt die große Anzahl schnell zu hohen Laufzeiten.

4.2.4 Basis mit größeren Elementen

Theoretisch kann das Basiswissen Teilzielmengen beliebiger Größe enthalten. Große Ziele haben jedoch erstens den Nachteil, dass die zugehörigen Pläne oft sehr lang sind und ihre Erstellung entsprechend aufwendig ist. Zweitens stellt sich die Frage, welche Schlüsse aus den Ergebnissen ableitbar sind. Ähnlich wie beim *credit-assignment*-Problem [RN03] ist aus dem Nettonutzen großer Zielmengen nicht erkennbar, warum er hoch oder niedrig ist. Erkenntnisse über einzelne Elemente oder das Zusammenspiel diverser Abhängigkeiten sind aus dem Ergebnis nicht ableitbar.

Die hier eingesetzten Algorithmen verwenden daher in der Wissensbasis ausschließlich Teilziele der Größen 1 und 2.

4.2.5 Relaxiertes Basiswissen

Es kann vorkommen, dass die Erzeugung der Pläne für einzelne Elemente der Wissensbasis zu viel Laufzeit in Anspruch nimmt. Eine Möglichkeit diese zu reduzieren ist, auf eine vollständige Planung zu verzichten, stattdessen lediglich einen *relaxierten* Plan [HN01] zu erstellen und dessen Kosten zu verwenden. Ein solcher Plan lässt sich mit sehr reduziertem Aufwand entwickeln. Die Eigenschaft des Basiswissens, nur gesicherte, also *wahre* Informationen zu enthalten, geht damit allerdings verloren. In der Praxis können die Ergebnisse relaxierter Pläne aber durchaus ausreichen, um brauchbare Modelle zu erstellen. Für kleine Teilziele sind die relaxierten und die tatsächlichen Pläne in den untersuchten Domänen häufig sehr ähnlich. Der Laufzeitgewinn ist stark von der verwendeten Domäne und Instanz abhängig. Größerer Zeitgewinn ist natürlich bei den hier für die Basis nicht verwendeten mächtigeren Zielmengen zu erwarten. Im Rahmen dieser Arbeit wurde eine Variante untersucht, die $\mathcal{K}^{1,2}$ so erstellt, das für die zweielementigen Teilziele ein relaxierter Plan erstellt wird.

4.2.6 Verwendung des Basiswissens

Wir haben nun einige Möglichkeiten betrachtet, wie das Basiswissen aufgebaut sein kann. Mit dem folgenden Abschnitt wenden wir uns dem zweiten Schritt der Teilzielsuche zu, in dem die Erkenntnisse der Basisphase genutzt werden sollen. Hierzu werden zunächst einige durch Basiswissen gesteuerte Hillclimbing-Algorithmen vorgestellt. Im zweiten Teil wird darauf eingegangen, wie sich die Wissensbasis zu Modellen erweitern lässt, mit denen für alle $G' \subseteq G$ ein modellierter Nettonutzen berechnet werden kann.

4.3 Hillclimbing

Eine sehr einfache Realisierung der Suche im Raum der Teilzielmengen bietet sich durch die bekannte Technik des *Hillclimbing*.

Es werden einige Varianten vorgestellt, die $\mathcal{K}^1$ als Wissensbasis verwenden. In jedem Hillclimbing-Schritt wird eine Teilzielmenge ausgewählt und klassisch für sie geplant. Die Auswahl erfolgt mit Hilfe von $\mathcal{K}^1$ und unter Einbeziehung des Ergebnisses aus dem letzten Schritt.

In einem Präprozess werden die Elemente der Wissensbasis gemäß ihrem Nettonutzen sortiert. Ihre Rolle besteht darin, während des Hillclimbings eine Reihenfolge für die benachbarten Teilzielmengen zu bestimmen.

4.3.1 Aufsteigendes Hillclimbing

Beim aufsteigenden Hillclimbing wird beginnend mit dem besten Zielelement, in jedem Schritt ein Element g hinzugefügt und für das resultierende Ziel geplant. Die Funktion der Wissensbasis besteht dabei ausschließlich darin, dass Ziele mit hohem

```
1:  procedure HILLCLIMB(sorted list<singleGoal> K^1)
2:      GoalSet G' = K^1.pop_front
3:      bestBenefit = b(G')                          ▷ bekannt aus Wissensbasis
4:      while Erweiterungen möglich do
5:          g = K^1.pop_front
6:          G' = G' ∪ g
7:          b(G') = b̂(generatePlan(G'))                  ▷ klassische Planung!
8:          if b(G') < bestBenefit then
9:              G' = G' \ g
10:             K^1.push_back(g)
11:         else
12:             bestBenefit = b(G')
13:         if alle Erweiterungen getestet then break
```

b-Wert bevorzugt hinzugefügt werden. Tritt keine Verschlechterung des Nettonutzens ein, wird sofort das neue größere Ziel als Erweiterungsgrundlage G' verwendet. Weitere Teilzielmengen derselben Größe werden nicht mehr geprüft. Diese schnellere und *gierigere* Variante des Hillclimbings kann gewählt werden, da die Information aus der Wissensbasis nahelegt, dass kein besseres Teilziel mehr zu erwarten ist. Durch die Sortierung der Wissensbasiselemente werden die vielversprechenden Einzelziele mit dem höchsten Nettonutzen zuerst hinzugefügt.

Führt eine Erweiterung zu einer Verschlechterung, wird das entsprechende Element wieder hinten in die Wissensbasis eingefügt. Da dieses in Kombination mit den Elementen aus der bereits gewählten Menge G' offenbar keine Verbesserung bringt, wird es bis auf Weiteres nicht mehr bevorzugt integriert. Es bleibt jedoch in der Kandidatenmenge enthalten und kann auf einer späteren Stufe durchaus nochmals einbezogen werden.

Der Algorithmus startet aus praktischen Gründen nicht mit der leeren Menge: Falls alle Zielelemente negativen Nettonutzen haben, würde er sonst auf Stufe 0 terminieren. Dieses Verhalten ist nicht sinnvoll, da für größere Zielmengen durchaus positive b-Werte erreicht werden können. In allen anderen Fällen ist das Verhalten identisch zur vorgestellten Variante.

Im Reiseproblem macht der Algorithmus folgende Schritte:

1. Sortiere $\mathcal{K}^1$ nach Nettonutzen: $(g_1\ (70), g_3\ (20), g_2\ (10), g_4\ (10))$.
2. Plane klassisch für $\{g_1, g_3\}$, erhalte Nettonutzen $150 = (300+100)-(230+20)$.
3. Da $150 \not< 70$, plane klassisch für $\{g_1, g_3, g_2\}$, erhalte Nettonutzen 190.
4. Da $190 \not< 150$, plane klassisch für $\{g_1, g_3, g_2, g_4\}$, erhalte Nettonutzen 170.
5. Keine weiteren Erweiterung der Menge $\{g_1, g_3, g_2\}$ mehr möglich $\Rightarrow$ Ende.

Die tatsächlich beste Zielmenge $\{g_1, g_3, g_2\}$ mit Nettonutzen 190 wurde gefunden, wobei in der Hillclimb-Phase nur drei klassische Planungen notwendig waren.

Ein Vorteil des Verfahrens ist seine Schnelligkeit: Bei Anwendung des Algorithmus auf die Probleme der Domäne SATELLITES werden für N Teilziele während des Hillclimbings im Durchschnitt etwa $N \cdot 1{,}15$ klassische Planungen durchgeführt. Die Basisphase benötigt für $\mathcal{K}^1$ ebenfalls nur N Planungen.

Eine untere Schranke für die Anzahl der Planungen in der Hillclimbing-Phase mit der verwendeten Variante ist $N-1$. Sie kann auf verschiedene Arten erreicht werden. Der Wert ergibt sich z.B., wenn alle zweielementigen Obermengen ein schlechteres Ergebnis liefern als das Startelement; der Algorithmus bricht dann nach der ersten Stufe ab. Er wird aber auch erreicht wenn auf jeder Stufe genau einmal geplant wird.

Eine obere Schranke ist gegeben durch $\frac{N(N-1)}{2}$. Sie wird erreicht, wenn auf jeder Stufe jede Kombination geprüft werden muss und jeweils die letzte das Ergebnis verbessert.

Das einfach aufsteigende Hillclimbing-Verfahren hat eine grundlegende Schwäche: Es wird nur ein sehr kleiner Bereich des Teilzielraumes berücksichtigt. Das Zielelement mit dem besten Nettonutzen ist z.B. in jeder Kandidatenmenge vorhanden. Außerdem kann der Algorithmus zu früh terminieren, wenn keine Erweiterung zu einer Verbesserung des Nettonutzens führt. Der gesamte Bereich großer Zielmengen wird dann nicht erreicht.

4.3.2 Hillclimbing mit randomisiertem Neustart

Um den berücksichtigten Teil des Zielraumes etwas breiter zu gestalten, bietet sich das Verfahren des *randomisierten Neustarts* an. Hierbei wird nach dem ersten Durchlauf das Hillclimbing einige Male neu gestartet, wobei das Startelement sowie das Element für die erste Erweiterung zufällig gewählt werden. Übergeht man hierbei bereits geplante Teilzielmengen, erhöht sich die Anzahl der Planungen für N Teilziele und drei Neustarts z.B. bei SATELLITES um den Faktor 4,08. Obwohl immer wieder bekannte Zielkombinationen übersprungen werden, terminiert das Verfahren für zufällige Startpunkte offensichtlich etwas langsamer. Dies zeigt, dass durch die vorgegebene Erweiterungsreihenfolge tatsächlich schneller gute Obermengen gefunden werden. Man beachte, dass bei den randomisierten Neustarts die Information der Wissensbasis nur im ersten Durchgang wirklich verwendet wird, da danach die Startpunkte zufällig gewählt sind. Durch die breitere Abdeckung des Suchraums können jedoch auch dann gute Ergebnisse erzielt werden, wenn die Informationen der Wissensbasis zu Fehleinschätzungen führen. Tatsächlich zeigen Tests, dass das Verfahren für Instanzen, in denen alle anderen basisgestützten Algorithmen schlecht abschneiden, oft die besten Ergebnisse liefert. Da der komplette Hillclimbing-Algorithmus viermal durchgeführt wird, neigt die Prozedur zu langen Laufzeiten. Auf eine Vertiefung oder Kombination mit anderen Varianten wird daher verzichtet.

4.3.3 Absteigendes Hillclimbing

Eine weitere Variante bietet das *absteigende Hillclimbing*. Der Algorithmus ist identisch zum aufsteigenden Verfahren, es werden jedoch, beginnend mit der gesamten

Zielmenge G, sukzessive Elemente entfernt. Die Reihenfolge ist wiederum durch die nach Nettonutzen sortierten Zielelenmente gegeben, wobei hier schlechtere Teilziele bevorzugt entfernt werden. Die Schwächen und Laufzeitschranken des Verfahrens sind die gleichen wie beim aufsteigenden Hillclimbing. Der bei früher Terminierung vernachlässigte Bereich enthält hier jedoch Teilziele kleiner Mächtigkeit. Die Auswertung der SATELLITES-Testdaten ergibt hier für die Hillclimbing-Phase $N \cdot 1{,}37$ Planungen. Der Wert, der natürlich von den getesteten Instanzen abhängt, ist gegenüber der aufsteigenden Version leicht angestiegen.

4.3.4 Bidirektionales Hillclimbing

Eine elegante Verwendung der Hillclimbing-Technik ergibt sich sehr einfach durch Kombination von auf- und absteigender Suche. Da optimale Teilzielmengen häufig sehr groß oder sehr klein sind, ist das Verfahren naheliegend und sinnvoll. Hierzu wird das aufsteigende und das absteigende Hillclimbing hintereinander ausgeführt, wobei bekannte Teilzielmengen übersprungen werden. Die Abdeckung des Suchraumes ist besonders in den Randbereichen sehr gut, insbesondere werden sicher alle Ziele der Größen 1 und $|G|$ geprüft. Angewendet auf die SATELLITES-Instanzen führt das Verfahren in der Hillclimbing-Phase für $|G| = N$ zu $N \cdot 2{,}59$ Planungen.

Dieser Faktor übersteigt geringfügig die Summe der einzelnen einfach gerichteten Varianten. Offensichtlich werden also immer wieder Zielkombinationen von beiden Varianten besucht. Das Überspringen derselben führt zu einer leichten Erhöhung der Anzahl besuchter Knoten. Der Nettonutzen kann dadurch das Maximum des durch auf- und absteigendes Hillclimbing erreichten Wertes sogar übertreffen.

Das *bidirektionale Hillclimbing* ist ein schnelles Verfahren, welches, wie wir sehen werden, sehr gute Ergebnisse liefert.

4.4 Nettonutzenmodelle

Dieser Abschnitt beschreibt das Konzept des *Nettonutzenmodells*. Mit Hilfe der Wissensbasis werden Modelle erstellt, die für jede mögliche Zielmenge einen modellierten Nettonutzen-Wert berechnen. Ziel der Modellierung ist die Identifikation von Teilzielen mit gutem bzw. optimalem b-Wert. Das Modell wird nicht als Heuristik für ein Suchverfahren verwendet, sondern liefert direkt Teilzielmengen, für die ein hoher Nettonutzen zu erwarten ist.

Definition 22 (Nettonutzenmodell). Ein **Nettonutzenmodell** ist eine **totale** Abbildung $m : \mathit{Pot}(G) \rightarrow \mathbb{R} \cup \{-\infty\}$, die einer Teilzielmenge einen geschätzten Nettonutzen zuweist.

Eine Teilzielmenge, die vom Modell m einen maximalen Nettonutzen zugewiesen bekommt, wird mit G^*_m bezeichnet.

Ein gutes Modell sollte die tatsächliche Verteilung des Nettonutzens möglichst treffend darstellen. Wichtig ist hierbei jedoch nicht die Differenz zum tatsächlichen

Nettonutzen einer Zielmenge $|m(G') - b(G')|$. Es reicht völlig aus, wenn Teilzielmengen mit relativ hohem modellierten Nettonutzen auch tatsächlich einen relativ hohen Wert haben. Zur Lösung des PSPNetBenefit-Optimierungsproblems reicht es sogar aus, wenn gilt

$$\text{argmax}_{G' \subseteq G}\, m(G') = \text{argmax}_{G' \subseteq G}\, b(G')).$$

Definition 23 (Modellabstand einer Zielmenge). Sei $i(G')$ der Index in der nach tatsächlichem Nettonutzen b sortierten Liste aller Teilzielmengen und $i_m(G')$ der Index in der nach modelliertem Nettonutzen m sortierten Liste aller Teilzielmengen. Elemente einer Liste mit gleichem Nettonutzen erhalten jeweils den Mittelwert der Indizes als gemeinsamen Index.

Der **Modellabstand einer Zielmenge** G' im Modell m ist dann

$$\delta_m(G') = |i(G') - i_m(G')|$$

.

Definition 24 (Modellabstand). Der relative **Modellabstand** Δ eines Modells m ist die normalisierte Summe aller Modellabstände von Zielmengen.

$$\Delta(m) = \frac{2}{M^2} \sum_{G' \subseteq G} \delta_m(G')$$

.

Durch den von $M = |Pot(G)|$ abhängigen Normalisierungsfaktor vor der Summe gilt $\Delta \in [0, 1]$, mit $\Delta = 0$, falls $m \equiv b$. Der maximale nicht normalisierte Wert für die Summe der Modellabstände aller Zielmengen bei geradem M ist $\frac{M^2}{2}$ [Ron98].

Modelle haben also einen kleinen Modellabstand, wenn sie bei einer Sortierung gemäß den b-Werten die Teilzielmengen in ähnlicher Reihenfolge anordnen. Ein Nettonutzenmodell m heißt **exakt**, wenn $\Delta(m) = 0$ gilt.

Der Modellabstand bietet ein gutes Maß zur Bewertung von Modellen, ist aber für die in der Praxis benötigte Lösung des PSPNetBenefit-Optimierungsproblems nicht notwendig. Hierzu genügt es, wenn die Zielmenge mit dem besten modellierten Nettonutzen tatsächlich ein gutes Ergebnis erzielt.

Definition 25 (Modellfehler). Seien G^*_m und G^* die Teilzielmengen mit dem besten modellierten bzw. tatsächlichen Nettonutzen. Dann ist für $b(G^*) \neq 0$ der relative **Modellfehler** gegeben durch

$$d(m) = 1 - \frac{b(G^*_m)}{b(G^*)}$$

.

Der Modellfehler d ist also genau dann 0, wenn das Modell eine Zielmenge mit maximalem Nettonutzen identifiziert. Offensichtlich ist der Modellfehler jedes exakten Modells 0. Stellt man an m die naheliegende Forderung, dass $m(\emptyset) \geq 0$ gilt, folgt $b(G^*_m) \geq 0$ und damit $d \in [0, 1]$, da dann $0 \leq b(G^*_m) \leq b(G^*)$ gewährleistet ist.

4.4.1 Berechnung von Nettonutzenmodellen

Grundlage zur Berechnung von Nettonutzenmodellen sind die in der Wissensbasis $\mathcal{K}$ enthaltenen Zielmengen. Zweck der Modellierung ist, diese gesicherten Informationen möglichst gut auf andere Teilziele zu übertragen. Die Totalität eines Modells garantiert einen Wert für den modellierten Nettonutzens aller $G' \subseteq G$. Die Modellierungsvorschrift muss also in der Praxis gegebenenfalls $2^{|G|}$-mal angewendet werden. Es ist damit notwendig, dass m sehr schnell berechnet werden kann.

4.4.2 Nicht erfüllbare Zielmengen

Wie bereit erwähnt, ist es im Rahmen der teilerfüllenden Planung durchaus möglich, dass bestimmte Teilzielmengen nicht erfüllbar sind. Diese Teilmengen sollen mit unendlichen Kosten, also negativ unendlichem Nettonutzen, modelliert werden.
Eine Sinnvolle Forderung an ein Modell ist, diese Teilziele G' mit $\mathcal{P}_{G'} = \emptyset$ zu identifizieren, also $m(G') = -\infty \Leftrightarrow b(G') = -\infty$. Dies ist allerdings im Allgemeinen nicht effizient zu erfüllen, da das PLANEX-Entscheidungsproblem PSPACE-vollständig ist [Byl94]. Es kann lediglich aus der Nichterfüllbarkeit einer Teilmenge auf Nichterfüllbarkeit der Obermengen geschlossen werden, also $\mathcal{P}_{G''} = \emptyset \Rightarrow \mathcal{P}_{G'} = \emptyset \;\; \forall G' \supseteq G''$.

4.4.3 Die Top-h des Modells

In der Praxis tritt der Fall auf, dass die Zielmenge mit dem höchsten modellierten Nettonutzen in Wirklichkeit ein schlechtes Ergebnis liefert. Dies kann in der Regel jedoch nicht erkannt werden, da der Vergleichswert fehlt. Eine Anpassung des Modells ist also relativ aufwendig. Sehr einfach ist es jedoch, den Konzepten des verwendeten Modells weiterhin zu vertrauen und lediglich die Existenz einzelner *Ausreißer* anzunehmen. Die naheliegende Reaktion hierauf ist, mehrere Teilziele zu testen. Plant man für die h besten Elemente des Modells, steigt die Wahrscheinlichkeit, ein gutes oder sogar das optimale Element zu finden, deutlich an. Die Anzahl der klassischen Planungen erhöht sich dabei nur um eine Konstante h.

4.5 Statische Modellierung

Als *statische* Modelle werden solche Nettonutzenmodelle bezeichnet, die einmalig mit Hilfe des Basiswissens erstellt werden. Sie bieten den Vorteil, dass nach dem Initialisieren der Basis theoretisch keine klassischen Planungen mehr durchgeführt werden müssen. Das Modell liefert direkt die Teilzielmenge mit dem höchsten erwarteten Nutzen. Um gute Ergebnisse erzielen zu können, muss jedoch eine Wissensbasis verwendet werden, die ausreichend Informationen z.B. über Abhängigkeiten zur Verfügung stellt.

Eine Basis aus einelementigen Teilzielen, wie sie etwa beim Hillclimbing verwendet wird, reicht in der Praxis meistens nicht aus, um mit statischer Modellierung einen geringen Modellfehler zu erhalten.

4.5.1 Ein statisches Modell

In diesem Abschnitt wird ein Nettonutzenmodell vorgestellt, welches sich die Informationen der ein- und zweielementigen Wissensbasis $\mathcal{K}^{1,2}$ zunutze macht. Durch die Zielpaare erhält das Modell Kenntnisse über Abhängigkeiten. Dadurch kann ein Modell mit häufig sehr kleinem Modellfehler entwickelt werden. Zur Verfügung stehen Nutzen und Kosten der Zielelemente $u(g_1), \ldots, u(g_r)$; $c(g_1), \ldots, c(g_r)$, sowie die der Paare $u(\{g_1, g_2\}), \ldots, u(\{g_{r-1}, g_r\})$ und $c(\{g_1, g_2\}), \ldots, c(\{g_{r-1}, g_r\})$. Hierbei gilt zwar für alle (i, j) mit $i \neq j$: $u(\{g_i, g_j\}) = u(g_i) + u(g_j)$ aber nicht zwingend $c(\{g_i, g_j\}) = c(g_i) + c(g_j)$. Beschränkt man sich für m auf lineare Modellierungsvorschriften, hat jedes auf $\mathcal{K}^{1,2}$ basierende sinnvolle Modell die folgende Form:

$$m(G') = \frac{1}{\eta(|G'|)} \sum_{g \in G'} [\alpha u(g) - \beta c(g)] + \sum_{\{g,g'\} \subseteq G'} [\gamma u(\{g, g'\}) - \zeta c(\{g, g'\})]$$

Hierbei sind $\alpha, \beta, \gamma, \zeta$ reelle Zahlen und η ein von der Teilzielgröße abhängiger *Normalisierungsfaktor* und es gelte $g \neq g'$. Wählt man $\alpha = \beta = \gamma = \zeta = 1$ und $\eta \equiv 1$ ergibt sich

$$m_2(G') = \sum_{g \in G'} b(g) + \sum_{\{g,g'\} \subseteq G'} b(\{g, g'\}) = \sum_{G'' \subseteq G', |G''| \leq 2} b(G'')$$

Der Bezeichner m_2 deutet an, dass alle Zielelemente mit maximal 2 Elementen eingerechnet werden.

Für die Zielmenge $G' = \{g_1, g_2, g_3\}$ aus dem Reiseproblem ergibt sich z.B.:
$m_2(G') = b(g_1) + b(g_2) + b(g_3) + b(\{g_1, g_2\}) + b(\{g_1, g_3\}) + b(\{g_2, g_3\}) = 70 + 10 + 20 + 110 + 150 + 20 = 390$.

Da $b(\{g_3, g_4\}) = -10$ den einzigen negativen Wert in der Wissensbasis darstellt, gilt $G^*_{m_2} = \{g_1, g_2, g_3, g_4\}$, obwohl $G^* = \{g_1, g_2, g_3\}$ die bessere Wahl wäre. Der Modellfehler bleibt jedoch mit $d(m_2) = 1 - \frac{b(\{g_1,g_2,g_3,g_4\})}{b(\{g_1,g_2,g_3\})} = 1 - \frac{180}{190} \approx 0{,}053$ sehr gering.

Abbildung 4.1 zeigt für die 128 nach Kardinalität sortierten Teilzielmengen einer SATELLITES-Instanz die Verteilung des tatsächlichen und des mit m_2 modellierten Nettonutzens. Das Modell produziert betragsmäßig deutlich höhere Ergebnisse, modelliert die Verteilung mit einem relativen Modellabstand von $\Delta = 0{,}31$ aber relativ gut. Der in der Praxis relevante relative Modellfehler zeigt, dass eine Modellierung dieser Gestalt sinnvoll sein kann: Die sechselementige Zielmenge G_{120} erreicht den tatsächlich höchsten b-Wert von 48,1. Das Modell ist maximal für die Zielmenge G_{105}, die den tatsächlichen Nettonutzen 41,2 erreicht. Der relative Modellfehler ist damit nur $d(m_2) \approx 0{,}143$.

Sortiert man die Zielmengen nun absteigend nach modelliertem Nettonutzen, erkennt man, das die *beste* Zielmenge G_{120} Platz 3 belegt. Ergebnisse der IPC-Domänen zeigen, dass die Top-5 des vorgestellten Modells häufig die optimale Zielmenge enthält.

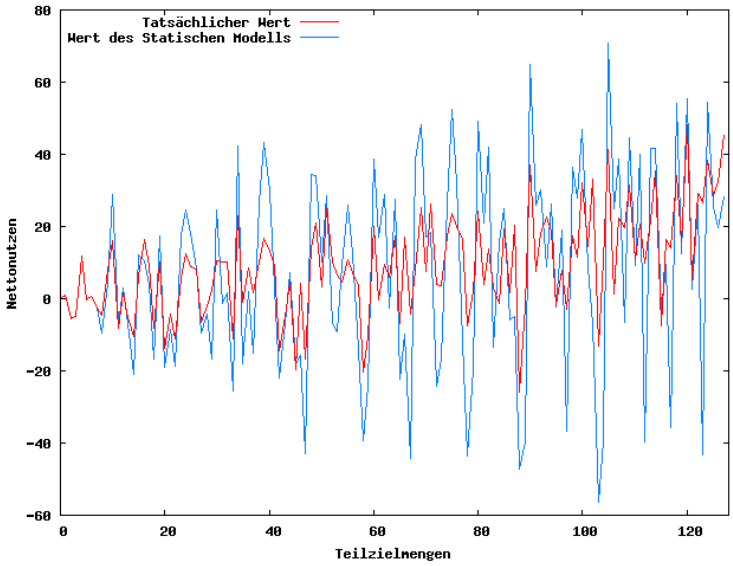


Abbildung 4.1: Tatsächliche und modellierte Verteilung des Nettonutzens für die Teilzielmengen eines SATELLITES-Problems mit 6 Teilzielen.

4.5.2 Ein statisches Modell mit größerer Basis

Das eben vorgestellte Modell verwendet als Basis $\mathcal{K}^{1,2}$, modelliert also höchstens zweistellige Abhängigkeiten. Verwendet man eine größere Wissensbasis $\mathcal{K}^{1,2,\ldots,M}$ mit allen Teilzielen bis Größe M, lässt sich die Modellierungsvorschrift analog anwenden:

$$m_M(G') = \sum_{G'' \subseteq G', |G''| \leq M} b(G'')$$

Das resultierende Modell berücksichtigt Abhängigkeiten bis zur Teilzielgröße M. Die Basisphase kann jedoch sehr laufzeitintensiv werden. Die hier vorgestellten Algorithmen verwenden daher nur die Modelle m_1 und m_2.

4.6 Dynamische Modellierung

Bei der *dynamischen* Modellierung wird das aus der Wissensbasis erzeugte Modell nachträglich mit Hilfe weiterer klassischer Planungen angepasst. Das Planungssystem *lernt* aus den Ergebnissen und passt sein Modell gemäß den neuen Erkenntnissen an. Die Problematik ist wiederum, dass jeder Anpassungsschritt sehr teuer ist, die meisten Konzepte aus dem Bereich des Maschinellen Lernens aber mit sehr vielen Modellupdates arbeiten. Der nachträglichen Anpassung dynamischen Basiswissens sind also im Rahmen vertretbarer Laufzeiten Grenzen gesetzt.

4.6.1 Ein dynamisches Modell

Die Idee des hier vorgestellten dynamischen Modells m_D ist die Folgende: Ausgehend von einem Initialmodell wird in jedem Anpassungsschritt eine Zielmenge G' klassisch geplant. Das Modell wird dann so modifiziert, dass der modellierte Nettonutzen der Menge G' in Richtung des tatsächlichen Wertes verschoben wird. Das Modell wird dabei so verändert, dass sich auch der Wert anderer Teilzielmengen verändert. Das Verfahren wird in Abschnitt 4.6.1.4 an einem Beispiel erläutert und und hier zunächst formal vorgestellt.

Das Modell m_D verwendet als Ausgangspunkt die Wissensbasis $\mathcal{K}^1$. Das Initialmodell wird mittels m_1 erstellt, einem einfachen Modell, das von Additivität der Elementkosten, also von der Unabhängigkeit aller Zielmengen, ausgeht. Im Gegensatz zu m_2 haben die von m_1 gebildeten Nettonutzen-Werte die gleiche Größenordnung, wie die tatsächlichen Werte. Die Regel zur Initial-Modellierung einer Zielmenge G' ist die Folgende:

$$m_D^0(G') = m_1(G') = \sum_{g \in G'} b(g)$$

Die anschließenden Anpassungen des Modells haben also vor allem die Aufgabe, Abhängigkeiten miteinfließen zu lassen.

Hierzu wird in jedem Schritt eine klassische Planung mit anschließender Anpassung des Modells durchgeführt. Die im n-ten Updateschritt geplante Zielmenge sei G_n. Ein n-mal angepasstes Modell wird mit m_D^n bezeichnet

Wiederum tritt allerdings die *credit-assignment*-Problematik auf: Die Kenntnis der Kosten für eine Zielmenge G_n sagt nichts über die Abhängigkeiten aus, die zu diesem Ergebnis führen. Es müssen also auch hier Annahmen getroffen werden. Eine sinnvolle Vereinfachung ist die Beschränkung auf Abhängigkeiten von zwei Elementen. Zweitens nehmen wir an, dass alle Zielpaare $\{g, g'\}, g \neq g'$, einheitlich negativ oder positiv abhängig bzw. unabhängig sind. Ist etwa die Menge G_n vom aktuellen Modell zu niedrig bewertet, also $b(G_n) > m_D^{n-1}(G_n)$, wird beim Update des Modells angenommen, dass alle Paare $\{g, g'\} \subseteq G_n$ positiv abhängig sind. Die Idee ist nun, das Modell für alle Obermengen dieser Zielpaare anzupassen. Im Fall von positiver Abhängigkeit werden die Werte für den modellierten Nettonutzen erhöht.

Für die Modellanpassung im n-ten Schritt unter Verwendung der Teilzielmenge G_n werden folgende Werte benötigt:
Der tatsächliche Nettonutzen $b(G_n)$, der modellierte Nettonutzen des aktuellen Modells $m_D^{n-1}(G_n)$ und $\lambda = b(G_n) - m_D^{n-1}(G_n)$. Die Zahl $\lambda \in \mathbb{R}$ beschreibt also die Differenz zwischen dem tatsächlichen und dem modellierten Nettonutzen für die Zielmenge G_n und ist genau dann positiv, wenn das Modell diesen Wert zu niedrig einschätzt.

4.6.1.1 Ausprägung der Modifikation

Der modellierte Nettonutzen für G_n des aktualisierten Modells m_D^n soll in Richtung des tatsächlichen Wertes $b(G_n)$ verschoben werden. Es soll also gelten:

$$m_D^n(G_n) = m_D^{n-1}(G_n) + \epsilon\lambda,$$

wobei mit der *Lernrate* $\epsilon \in [0,1]$ gesteuert werden kann, wie stark die Modellanpassung erfolgt. Bei $\epsilon = 1$ liefert das Modell im nächsten Schritt für G_n den tatsächlichen Nettonutzen, da sich für die eben eingeführte Gleichung $m_D^n(G_n) = m_D^{n-1}(G_n) + b(G_n) - m_D^{n-1}(G_n) = b(G_n)$ ergibt. Für kleinere ϵ-Werte wird ein Wert zwischen dem des alten Modells und dem Ergebnis der Planung angenommen.

4.6.1.2 Ablauf der Modifikation

Das Modell wird nicht nur punktuell an der Stelle G_n angepasst. Um Abhängigkeiten darzustellen, werden alle Obermengen der zweielementigen Teilmengen von G angepasst.

Wir bilden beim ersten Update einmalig eine Erweiterungsbasis $\mathcal{K}_+^2$, bestehend aus allen zweielementigen $\{g, g'\} \subseteq G$. Für jedes Element setzen wir zunächst die *Erweiterungskosten* $c_+(\{g, g'\}) = 0$.

Bei jeder Modellanpassung wird zuerst c_+ in $\mathcal{K}_+^2$ aktualisiert. Man beachte, dass die Erweiterungskosten c_+ dabei durchaus negative Werte annehmen können.

$$\forall \{g, g'\} \in \mathcal{K}_+^2, \{g, g'\} \subseteq G_n : \; c_+(\{g, g'\}) = \frac{1}{\binom{|G_n|}{2}} \cdot \epsilon\lambda$$

$$\forall \{g, g'\} \in \mathcal{K}_+^2, \{g, g'\} \nsubseteq G_n : \; c_+(\{g, g'\}) = 0$$

Die Vorschrift zur Modellaktualisierung für beliebige G' lautet dann:

$$m_D^n(G') = m_D^{n-1}(G') + \sum_{\{g,g'\} \in G'} c_+(\{g, g'\})$$

Durch den Binomialkoeffizienten $\binom{|G_n|}{2}$ wird erreicht, dass für aktuell geplante Zielmenge G_n gilt $\sum_{\{g,g'\} \subseteq G_n} c_+(\{g, g'\}) = \epsilon\lambda$. Das Modell wird an dieser Stelle um den Wert $\epsilon\lambda$ in Richtung des tatsächlichen Wertes verschoben.

Für $\lambda \neq 0$ verändert sich das Modell für genau die G', die gemeinsame zweielementige Teilmengen mit G_n besitzen. Nur hier gilt $m_D^n(G') \neq m_D^{n-1}(G')$.

Keine Veränderung ergibt sich für alle Zielmengen, deren zweielementigen Teilmengen $\{g, g'\}$ alle Erweiterungskosten 0 haben. Dies sind genau die $\{g, g'\} \nsubseteq G_n$.

4.6.1.3 Auswahl der Zielmenge zum Modellupdate

Die Auswahl der Zielmenge, für die klassisch geplant wird, kann z.B. so erfolgen, dass der Raum der Teilzielmengen möglichst breit abgedeckt wird. Hier wurde eine andere Lösung gewählt, welche vorhandenes Wissen verwendet: Das Modell nimmt diese Auswahl selbst vor, indem immer die Zielmenge mit dem aktuell besten modellierten Nettonutzen $G^*_{m_D^n}$ geplant wird. Ist deren tatsächlicher Nettonutzen bereits bekannt, wird die nächstbeste Menge gewählt. Bleibt das Modell in den Anpassungsschritten stabil, wurde also bei h Schritten am Ende die Top-h des Modells geplant. Dies ist zum einen vorteilhaft, wenn die Anzahl der Updates nicht ausreicht, um das Modell sinnvoll anzupassen, da dann wenigstens die Information des Initialmodells so gut

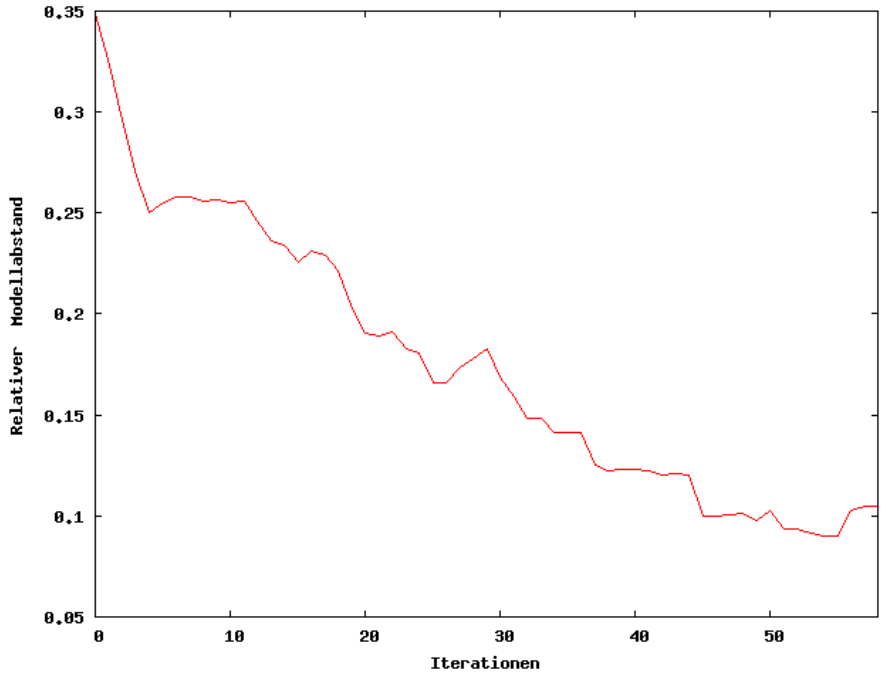


Abbildung 4.2: Entwicklung des Modellabstandes Δ über 58 Updateschritte ($\epsilon = \frac{1}{2}$) für eine DEPOTS-Instanz mit 64 Teilzielmengen.

wie möglich ausgenutzt wurde. Zum anderen sind natürlich genau solche Modelle stabil, die den tatsächlichen Nettonutzen ihrer Elemente treffend voraussagen. Hier befindet sich dann die optimale Zielmenge häufig in den Top-h.

Abbildung 4.2 zeigt die Entwicklung des Modellabstandes Δ für eine DEPOTS-Instanz über 58 Modellanpassungen mit $\epsilon = \frac{1}{2}$. Da die Kosten der 6 elementaren Ziele bereits in der Basisphase berechnet wurden, ist insgesamt für alle 64 Teilzielmengen der Instanz einmal klassisch geplant worden. Man kann erkennen, dass die Verbesserung des Modellabstandes zunächst sehr schnell, später deutlich langsamer stattfindet. Zusätzlich treten immer wieder Anpassungen auf, die den Δ-Wert wieder verschlechtern. Am Ende wird ein sehr guter Wert im Bereich 0,1 erreicht. Die Kurve dient zur Veranschaulichung des Algorithmus. Eine Iterationstiefe in dieser Größenordnung ist in der Praxis natürlich nicht sinnvoll und extrem ineffizient.

4.6.1.4 Beispiel

Das folgende Beispiel zeigt verkürzt die Initialisierung und den ersten Updateschritt für das Reiseproblem mit $\epsilon = \frac{1}{2}$:

- Berechne $m_D^0 = m_1$ für alle $G' \subseteq G$, hier exemplarisch $m_D^0(\{g_1, g_2, g_3\}) = 70 + 10 + 20 = 100$.
- Da alle Elementarziele $g_1, \ldots, g_4$ positiven Nettonutzen haben, ergibt sich der maximale Wert für $G_0 = \{g_1, g_2, g_3, g_4\}$ mit $m_D^0(G_0) = 110$.

- Klassische Planung ergibt $c(G_0) = 370$ für den optimalen Reiseweg ($LV \rightarrow SD \rightarrow DL \rightarrow SJ \rightarrow SF$) und $u(G_0) = 550$, also $b(G_0) = 180$ und $\lambda = 180 - 110 = 70$.
- Da $G_0 = G$ gilt, erhalten alle möglichen Zielpaare $\{g, g'\}$ Erweiterungskosten $c_+(\{g, g'\}) = \frac{1}{\binom{|G_n|}{2}} \cdot \epsilon\lambda = \frac{35}{6}$.

Hiermit werden nun alle Zielmengen neu modelliert:

- Für G_0 ergibt sich wegen $\epsilon = \frac{1}{2}$ der Wert $m_D^1(G_0) = 110 + 35 = 145$ genau zwischen $m_D^0(G_0)$ und $b(G_0)$.
- Exemplarisch berechnen wir noch $m_D^1(\{g_1, g_2, g_3\}) = 100 + 3 \cdot \frac{35}{6} = 117{,}5$.

Das einmalige Update hat also nicht ausgereicht, um die optimale Zielmenge $G^* = \{g_1, g_2, g_3\}$ mit $b(G^*) = 190$ zu erkennen. Das aktualisierte Modell berechnet den höchsten Wert immer noch für $G^*_{m_D^0} = \{g_1, g_2, g_3, g_4\}$ mit $b(G^*_{m_D^0}) = 180$. Der relative Modellfehler liegt wie beim statischen Modell bei 0,053.

4.7 Anwendbarkeit von Basiswissen

Obwohl Basiswissen moderater Größe meistens nicht alle positiven und negativen Abhängigkeiten repräsentieren kann, bietet es doch Informationen an, die geeignet sind, bei der Modellbildung oder Suche zu helfen. Dies gilt allerdings nur, wenn die Kosteninformationen sich überhaupt übertragen lassen. Die IPC-Domäne ROVERS erweist sich z.B. als gänzlich ungeeignet, um mit einer ein- und zweielementigen Wissensbasis zu operieren. Dies liegt an der verwendeten Kostenfunktion. Die Kosten der ROVERS-Probleme ergeben sich aus der Anzahl der *recharges*. Das Fahrzeug muss regelmäßig Sonnenenergie aufnehmen um sich fortbewegen zu können. Ein- und zweielementige Zielmengen können jedoch in der Regel ohne *recharges* erreicht werden, haben also Kosten 0. Es lassen sich damit keinerlei Rückschlüsse ableiten, welche Zielkombinationen kostenintensiv sind.

Geeignet für die Verwendung einer Wissensbasis aus Zielen kleiner Mächtigkeit sind also nur Domänen, deren Kostenfunktion so fein abgestuft ist, dass sich Ergebnisse von kleinen auf große Teilzielmengen übertragen lassen.

Die in anderen Domänen verwendeten Kostenmaße wie Treibstoffverbrauch, Planlänge oder Kombinationen aus diesen, erweisen sich hingegen als geeignet für Verfahren mit Wissensbasis. Abbildung 4.3 zeigt die Verteilung der Kosten für die nach Mächtigkeit sortierten Teilzielmengen einer SATELLITES- sowie einer ROVERS-Instanz mit jeweils 7 Zielelementen. Deutlich ist zu erkennen, dass bei SATELLITES auch für kleine Teilzielmengen differenzierte Kosteninformation vorliegt, die zur Modellierung verwendbar ist. Bei ROVERS bleiben die Kosten jedoch, mit Ausnahme des Gesamtziels G und einiger großer Teilmengen der Mächtigkeit 6, konstant bei 0. Die Kosten-Werte der kleineren Teilziele können keinerlei Hinweise auf die Erhöhung der Kosten für einige große Zielmengen enthalten.

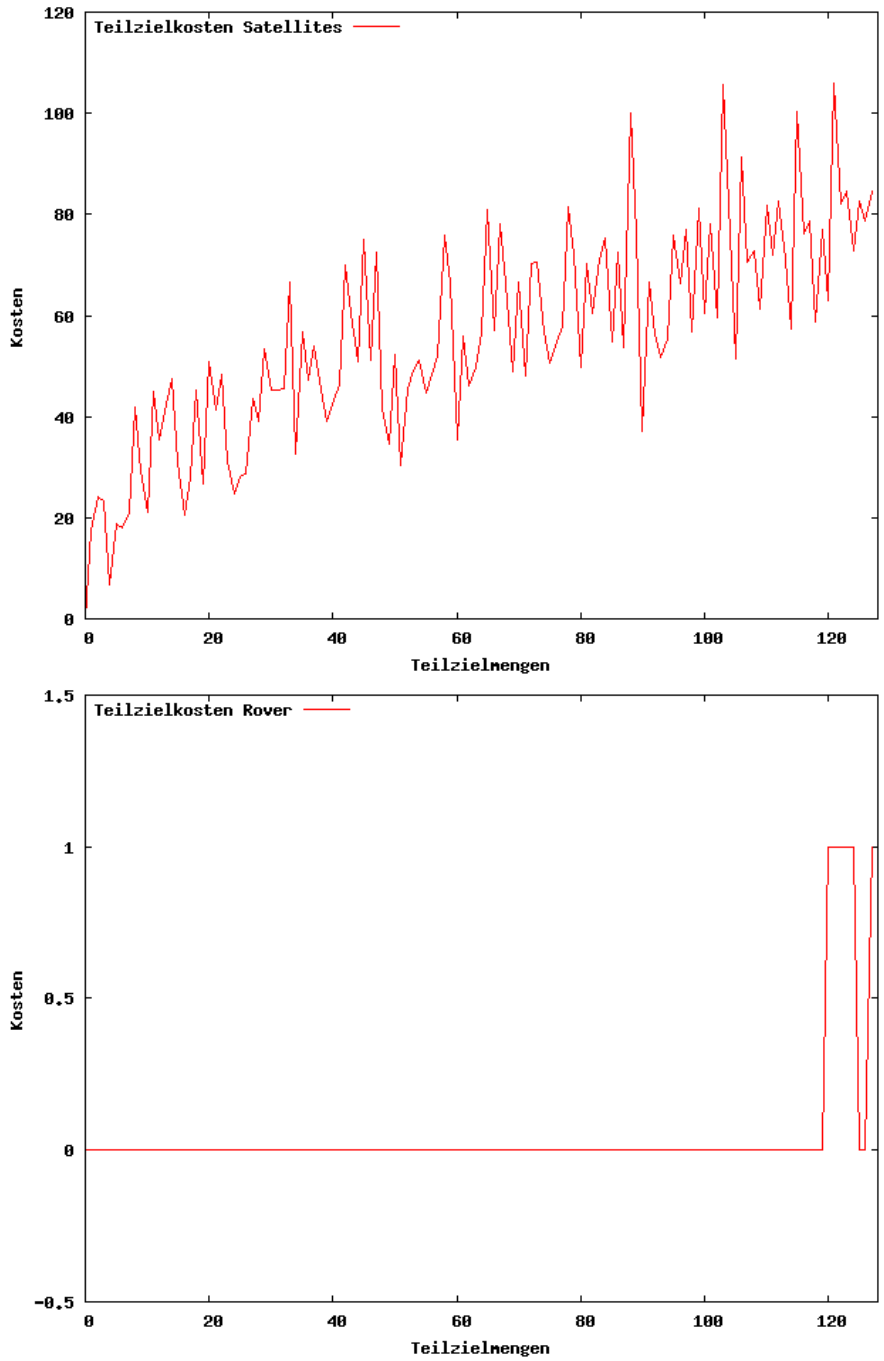


Abbildung 4.3: Kosten der nach Kardinalität sortierten Teilzielmengen einer SATELLITES- und einer ROVERS-Instanz mit jeweils 128 Teilzielmengen.

5 Implementierungen und Evaluation

Im Rahmen dieser Arbeit wurden diverse PS-Planungssysteme implementiert. Verwendet wurden die in Kapitel 4 vorgestellten Konzepte und Algorithmen. Zur Evaluation werden Planungsprobleme der Domänen SATELLITES, ZENOTRAVEL und DEPOTS verwendet. Als teilerfüllende Planungsprobleme dienen um eine Nutzenfunktion erweiterte Instanzen der IPC3.

Zur Einordnung der Leistungsfähigkeit wurde der vorgestellte naive optimale PS-Planer implementiert, mit dem soweit möglich der maximal erreichbare Nettonutzen berechnet wurde.

Das Kapitel erläutert zunächst Grundsätzliches zur Implementierung und untersucht anschließend die Ergebnisse der Hillclimbing-Varianten. Es folgt eine Evaluation der statischen und dynamischen Modellierung. In Abschnitt 5.5 werden die verschiedenen Konzepte zur Teilerfüllenden Planung untereinander verglichen. Anschließend wird versucht, die gewonnenen Ergebnisse mit denen anderer Planungssysteme zu vergleichen.

5.1 Grundsätzliches

In diesem Abschnitt wird erläutert, auf welche Weise der Nutzen für die Planungsprobleme festgesetzt wurde und wie die Einbindung des klassischen Planungsprogramms implementiert ist.

5.1.1 Spezifikation des Teilzielnutzens

Aus Mangel geeigneter *Benchmark*-Domänen mit numerischem Teilzielnutzen wurden die in Kapitel 2 vorgestellten Domänen der IPC3 für Tests und Auswertung verwendet. Dazu wurden sie um eine Nutzenfunktion erweitert, welche jedem Zielelement einen Wert zuordnet. Für größere Zielmengen ergibt sich der Nutzen dann additiv. Um interessante Probleminstanzen für PS-Planung zu erhalten, muss darauf geachtet werden, dass die Zielmenge mit dem höchsten Nettonutzen scheinbar zufällig aus der Potenzmenge der Teilziele gezogen ist. Das heißt es sollen Teilmengen großer, kleiner und mittlerer Kardinalität vertreten sein. Wie in 3.6 gezeigt, muss also darauf geachtet werden, dass der Nutzen der Zielelemente nicht zu klein oder zu groß gewählt wird. Hierzu wird Information über die Größenordnung der Kosten in der entsprechenden Instanz benötigt. Verwendet wurde der Mittelwert der Kosten für

Zielelemente $m = \frac{1}{|G|} \sum_{g \in G} c(g)$. Er ergibt sich unmittelbar aus der einelementigen Wissensbasis, die meist ohne Aufwand erstellt werden kann.

Den Zielelementen wurde dann bei uniformer Verteilung ein zufälliger Nutzen aus dem Intervall $[0, 2m]$ zugeordnet, so dass $\sum_{g \in G} c(g) \approx \sum_{g \in G} u(g)$ gilt. Die Tests mit den *Benchmark*-Domänen haben gezeigt, dass der maximale Nettonutzen damit in der Regel für Zielmengen $G' \notin \{G, \emptyset\}$ erreicht wird.

5.1.2 Verwendung des klassischen Planers

Die verschiedenen PS-Planungsvarianten sind alle so implementiert, dass wegen des *Blackbox*-Prinzips leicht verschiedene klassische Planer verwendet werden können. Das klassische Planungsprogramm erhält als Eingabedateien eine Domänen- und eine Problembeschreibung in PDDL. Letztere wird für jeden Aufruf so modifiziert, dass die Zielbeschreibung genau die gewünschten Zielelemente enthält.

Zur Erhebung der Evaluationsdaten wurde das System stets mit dem klassischen Planer SGPLAN5.2 [CWH06] gestartet. Wegen seiner Schnelligkeit erwies er sich als besonders geeignet, die oft zahlreichen Planungen durchzuführen. SGPLAN ist zwar ein suboptimaler Planer, liefert aber für das gleiche Problem stets dasselbe Ergebnis. Die Kosten für Teilziele sind also eindeutig und können somit als optimal angenommen werden. Bei Verwendung eines anderen Planungsprogramms können sich jedoch völlig andere Teilzielkosten und damit deutlich abweichende Ergebnisse einstellen Man beachte insbesondere bei Problemen, bei denen die Planlänge in die Kostenberechnung einfließt, dass SGPLAN ein sequentieller Planer ist. Die Ergebnisse der Domäne ZenoTravel und einiger Depots-Probleme sollten also nicht mit Daten verglichen werden, die mit einem parallelen Planer entstanden sind.

5.2 Hillclimbing-Algorithmen

In diesem Abschnitt werden die Ergebnisse der unterschiedlichen Hillclimbing-Varianten vorgestellt. Die Implementierung besteht jeweils aus einer Basisphase, welche als Basiswissen alle Elementarziele plant und deren Kosten und Nutzen speichert ($\mathcal{K}^1$). Für die Hillclimbing-Phase wurden die folgenden Systeme implementiert:

- Einfaches aufsteigendes Hillclimbing (*hillclimb up*, HCU)
- Einfaches absteigendes Hillclimbing (*hillclimb down*, HCD)
- Aufsteigendes Hillclimbing mit dreimaligem zufälligen Neustart (*hillclimb plus*, HCP)
- Kombiniertes auf- und absteigendes Hillclimbing (*hillclimb bidirectional*, HCB)

Abbildung 5.1 zeigt den relativen erreichten Nettonutzen ϕ der verschiedenen Varianten für die Depots-Instanzen. Man sieht, dass für die Instanzen 5 und 8 nur die Version mit randomisierten Neustarts (HCP) ein gutes Ergebnis erzielt. Ansonsten

findet eine der einfachgerichteten Varianten (HCU,HCD) ein guten Wert, welchen auch der bidirektionale Algorithmus (HCB) erreicht oder übertrifft.

Abbildung 5.2 zeigt die Anzahl der klassischen Planungen inklusive Basisphase, die für die verschiedenen Verfahren durchgeführt wurden. Die braune Kurve beschreibt die Anzahl der Zielelemente in der entsprechenden Instanz.

5.3 Statisches Modell

Das in Abschnitt 4.5.1 vorgestellte statische Modell, basierend auf den ein- und zweielementigen Teilzielmengen, wurde wie folgt implementiert: Die Basisphase stellt zwei Varianten bereit, wobei eine für die Zielpaare nur einen relaxierten Plan erstellt und dessen Kosten speichert. Für die verwendeten Instanzen genügt es hierzu, die Domänenbeschreibung zu modifizieren. Dabei werden negative Effekte und Ressourcenbeschränkungen, die keinen Einfluss auf die Metrik haben, entfernt. In beiden Varianten wurde nach der Modellierungsphase die Top-5 des Modells klassisch geplant. Das Maximum der hierbei erzielten Werte bildet den erreichten Nettonutzen des Systems.

Abbildung 5.3 zeigt den erreichten relativen Nettonutzen ϕ für die DEPOTS-Instanzen. Es ist zu erkennen, dass die Ergebnisse bei Verwendung relaxierter Pläne häufig gleich gut, und manchmal sogar besser sind als bei einer Basis aus korrekten Plänen. Außerdem kann man beobachten, dass schlechtere Ergebnisse für die gleichen Instanzen auftreten wie beim Hillclimbing. Abbildung 5.4 zeigt, dass für einige Instanzen die Laufzeit durch Verwendung relaxierter Pläne verringert werden konnte. Die Laufzeiten beziehen sich auf den gesamten teilerfüllenden Planungsvorgang inklusive Basisphase und Planen der Top-5, erstellt mit einem *AMD Athlon 64 3200+*.

5.4 Dynamisches Modell

Die Implementierung des in Abschnitt 4.6.1 beschriebenen statischen Modells enthält zwei Freiheitsgrade. Beim Start des Programms kann die Anzahl der Update-Schritte sowie der Wert für ϵ gewählt werden. Die durchgeführten Experimente haben gezeigt, dass für viele Probleminstanzen bei Verwendung unterschiedlicher ϵ-Werte gleich gute Ergebnisse erzielt wurden. Die Anzahl der Modellupdates spielte ebenso eine wesentlich kleinere Rolle, als man erwarten könnte. Das erzielte Ergebnis wurde bei einer Steigerung von 5 auf 10 oder 20 Aktualisierungsschritte nur in Einzelfällen deutlich verbessert.

Abbildung 5.5 zeigt die Entwicklung des relativen Modellabstands Δ über 20 Anpassungsschritte des Modells für $\epsilon = 1, \frac{1}{2}$ und $\frac{1}{4}$. Man erkennt, dass die Verbesserung bei $\epsilon = \frac{1}{4}$ langsamer erfolgt als für größere Werte. Die Kurve für $\epsilon = 1$ sinkt zunächst stärker ab, unterliegt jedoch starken Schwankungen.

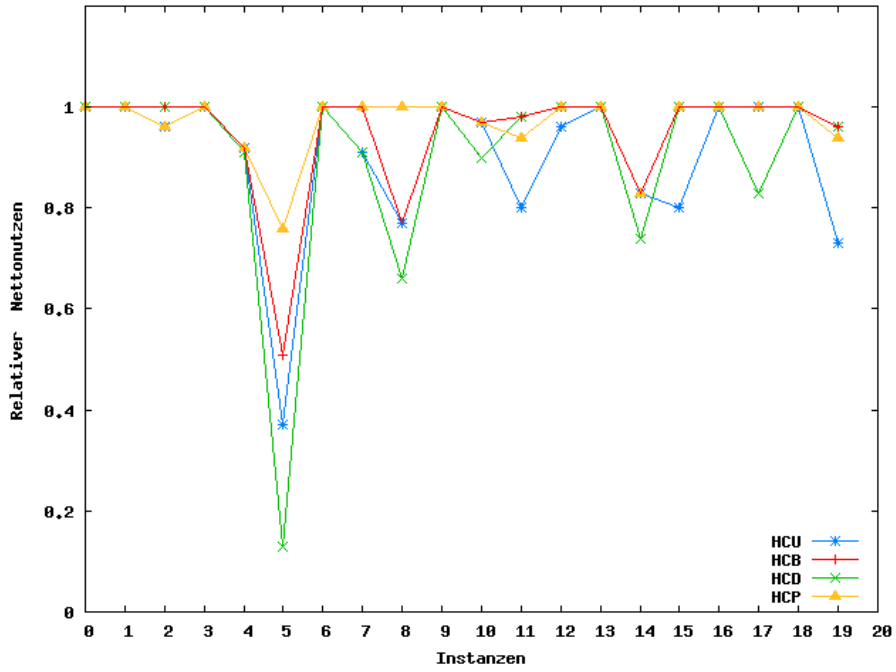


Abbildung 5.1: Erreichter relativer Nettonutzen ϕ der verschiedenen Hillclimbing-Varianten für 20 DEPOTS-Instanzen.

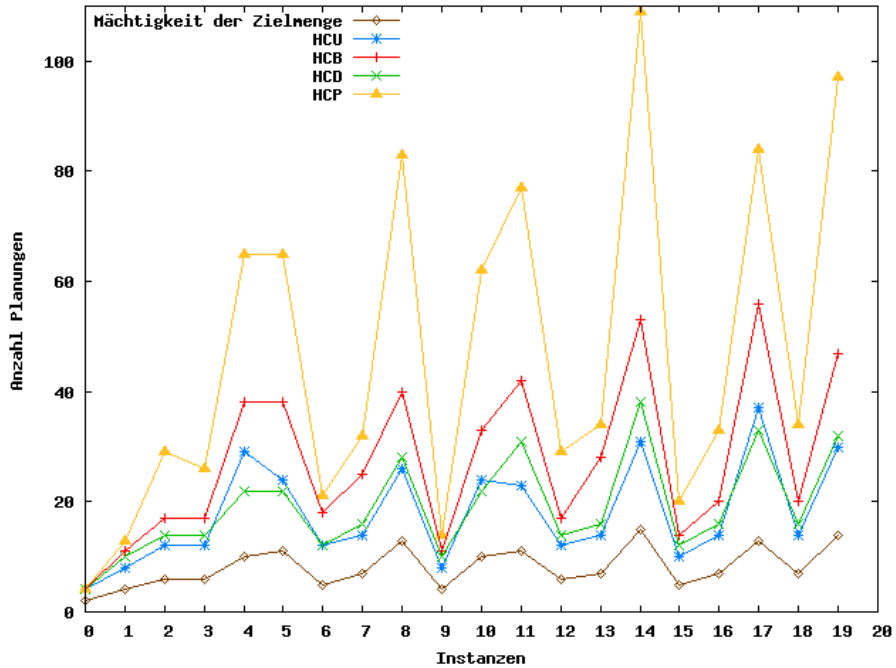


Abbildung 5.2: Anzahl klassischen Planungen in den verschiedenen Hillclimbing-Varianten für 20 DEPOTS-Instanzen.

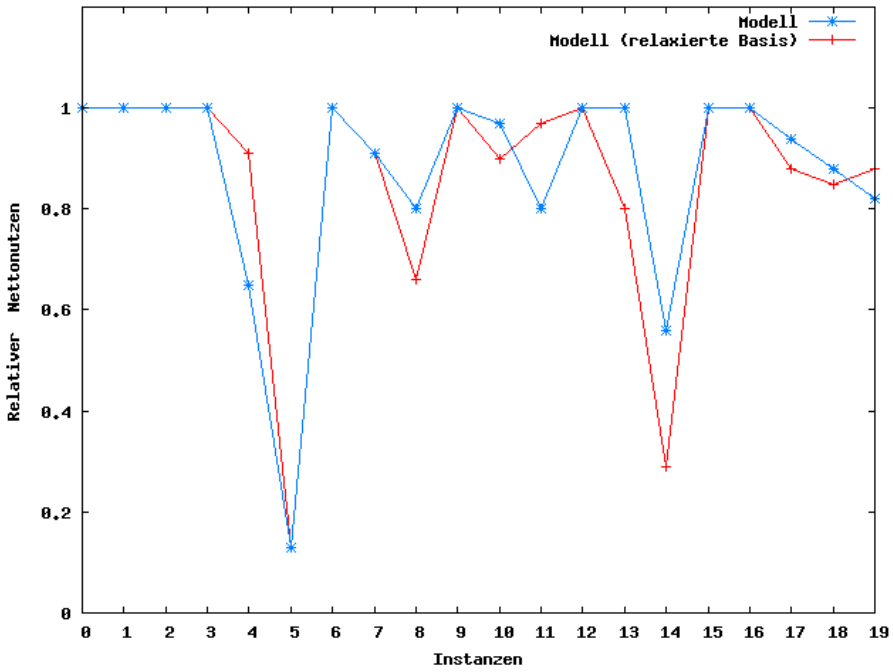


Abbildung 5.3: Maximaler relativer Nettonutzen ϕ in der Top-5 des statischen Modells mit bzw. ohne *relaxierte* Pläne in der Wissensbasis für 20 DEPOTS-Instanzen.

5.5 Vergleich der Konzepte

In diesem Abschnitt werden die drei untersuchten Konzepte Hillclimbing, statische, sowie dynamische Modellierung verglichen. Hierzu wurde jeweils eine Variante als Repräsentant ausgewählt. Die Gruppe der Hillclimbing-Algorithmen wird durch die bidirektionale Version (HCB) repräsentiert. Das Basiswissen des statischen Modells wird mit korrekt geplanten zweielementigen Zielmengen erstellt. Beim Dynamischen Modell ist ϵ auf $\frac{1}{2}$ und die Anzahl der Modellanpassungen auf 10 gesetzt. Alle Verfahren wurden mit den Domänen DEPOTS, ZENOTRAVEL und SATELLITES getestet. Zur Bewertung der Qualität dient der erreichte Nettonutzen. Die Performanz wird zunächst mit der Hardware-unabhängigen Größe *Anzahl der klassischen Planungen*, dann mit tatsächlichen Laufzeiten bewertet.

5.5.1 Nettonutzen

Die Abbildungen 5.6–5.9 zeigen den erreichten Nettonutzen der vorgestellten Verfahren in den verschiedenen Domänen. Zur Berechnung des relativen Nettonutzens ϕ wird der maximale erreichbare b-Wert benötigt. Die hierzu verwendete Implementierung des naiven optimalen Algorithmus konnte für große Zielmengen auch mit mehrtägigen Laufzeiten keine Ergebnisse mehr produzieren. Auch die modellierenden Verfahren scheiterten an einigen großen Instanzen. Zur Veranschaulichung wurde

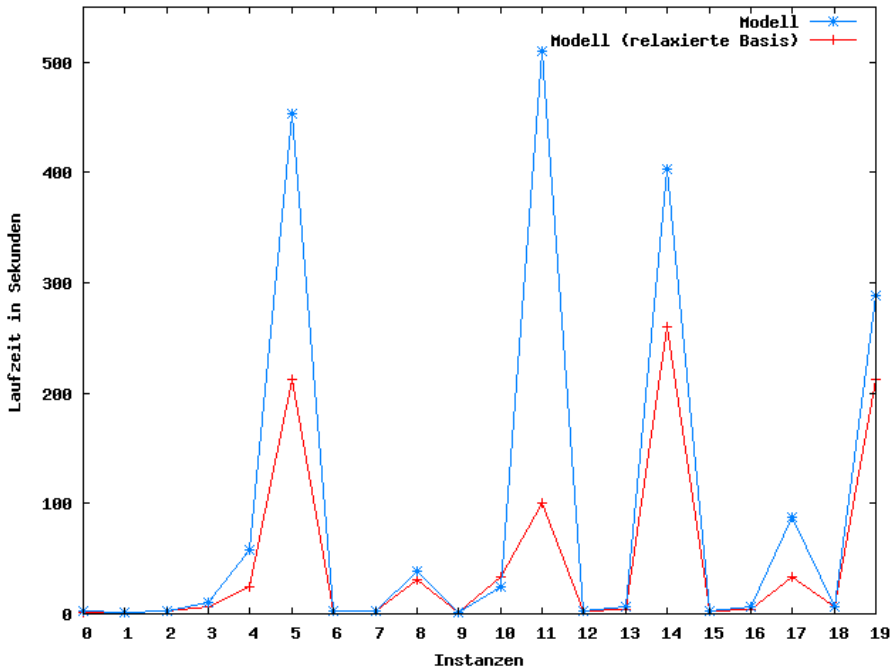


Abbildung 5.4: Laufzeit in Sekunden für PS-Planung durch das statische Modell mit und ohne *relaxierte* Pläne in der Wissensbasis für 20 DEPOTS-Instanzen.

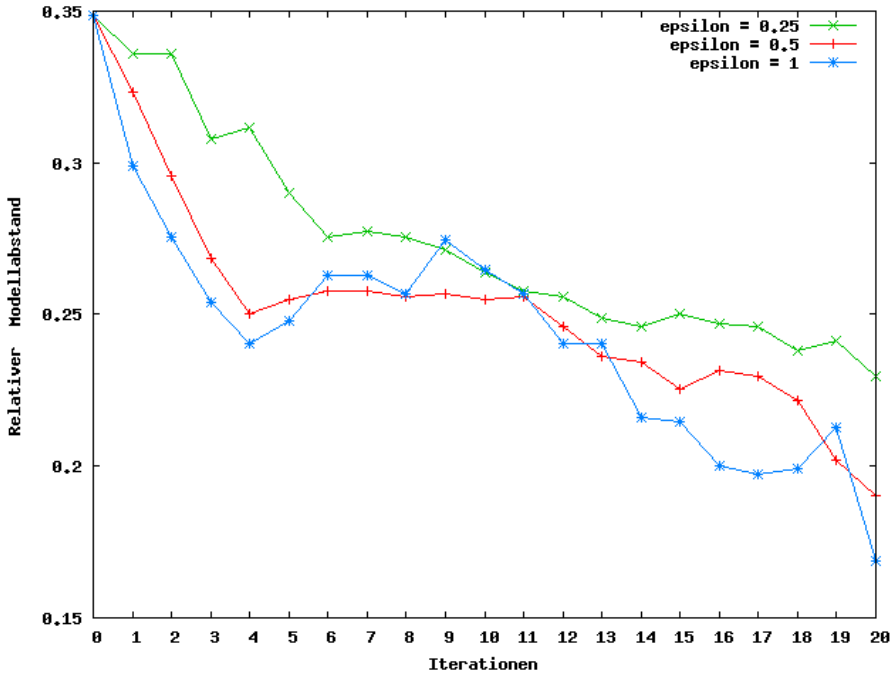


Abbildung 5.5: Entwicklung des relativen Modellabstands Δ über 20 Iterationen bei unterschiedlichen ϵ-Werten für eine DEPOTS-Instanz.

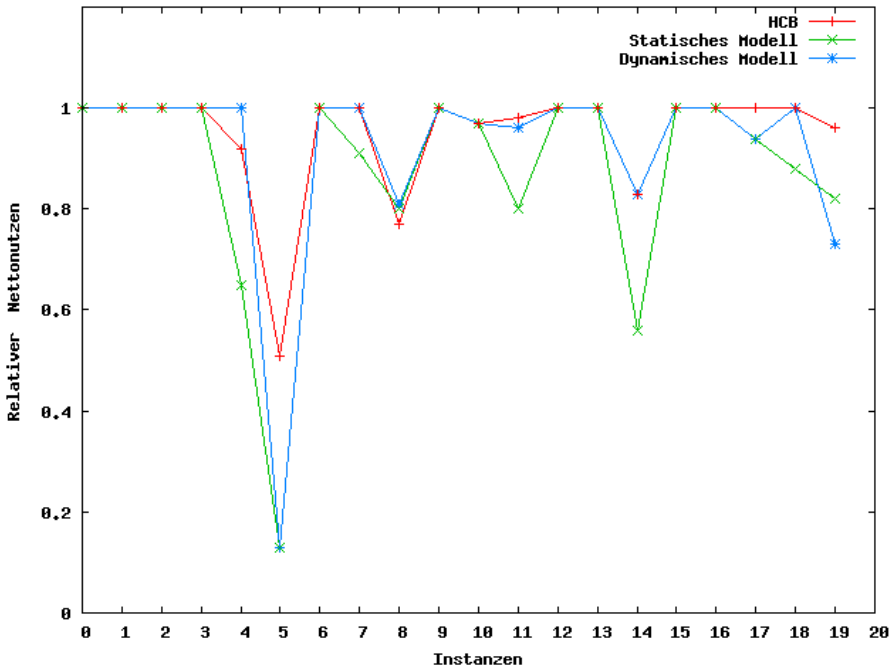


Abbildung 5.6: Erreichter relativer Nettonutzen ϕ verschiedener PS-Planungssysteme für 20 DEPOTS-Instanzen.

daher in zwei Abbildungen der absolute Nettonutzen verwendet. Abbildung 5.7 und 5.9 zeigen, dass nur mit dem Hillclimbing-Verfahren alle Instanzen gelöst werden konnten. Eine Einordnung des erzielten Wertes ist jedoch wegen des unbekannten maximalen Nettonutzens nicht möglich. Ansonsten zeigt sich, dass mit allen Verfahren, vor allem aber mit bidirektionalem Hillclimbing, häufig der optimale Wert erzielt werden konnte. Mit wenigen Ausnahmen werden im Übrigen auch sonst gute Ergebnisse gefunden.

5.5.2 Anzahl klassischer Planungen

Die Abbildungen 5.10, 5.11 und 5.12 zeigen die Anzahl durchgeführter klassischer Planungen der unterschiedlichen Verfahren. Die Werte des bidirektionalen Hillclimbing (HCB) sind empirisch ermittelt. Die Werte für die beiden Modelle lassen sich aus der Gesamtzielgröße $|G|$ berechnen. So ergibt sich die Anzahl der klassischen Planungen für das statische Modell aus den notwendigen Planungen zur Erzeugung der ein- und zweielementigen Teilzielmengen zuzüglich 5 Planungen in der Top-5 des Modells zu $|G| + \binom{|G|}{2} + 5$. Der Wert für das dynamische Modell besteht aus $|G|$ Planungen für die Zielelemente plus 10 Modifikationsschritte. Die Maße sind unabhängig von der verwendeten Implementierung und Hardware. Für alle Domänen gilt, dass das dynamische Modell in der Regel die wenigsten und das statische Modell die meisten Planungen durchführt. Die sehr großen ZENOTRAVEL- und SATELLITES-Instanzen

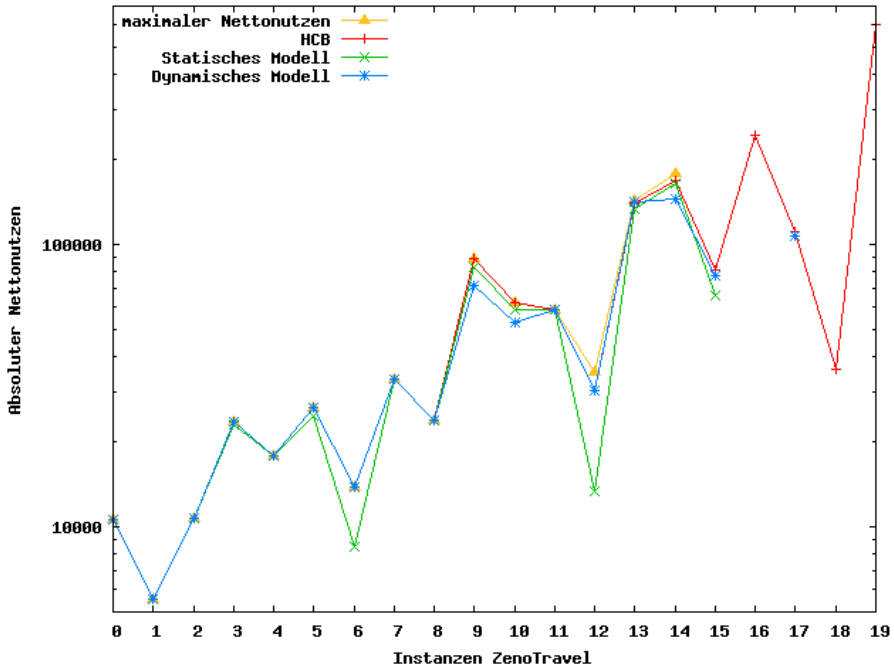


Abbildung 5.7: Erreichter absoluter Nettonutzen verschiedener PS-Planungssysteme für 20 ZenoTravel-Instanzen. Die y-Achse ist logarithmisch skaliert.

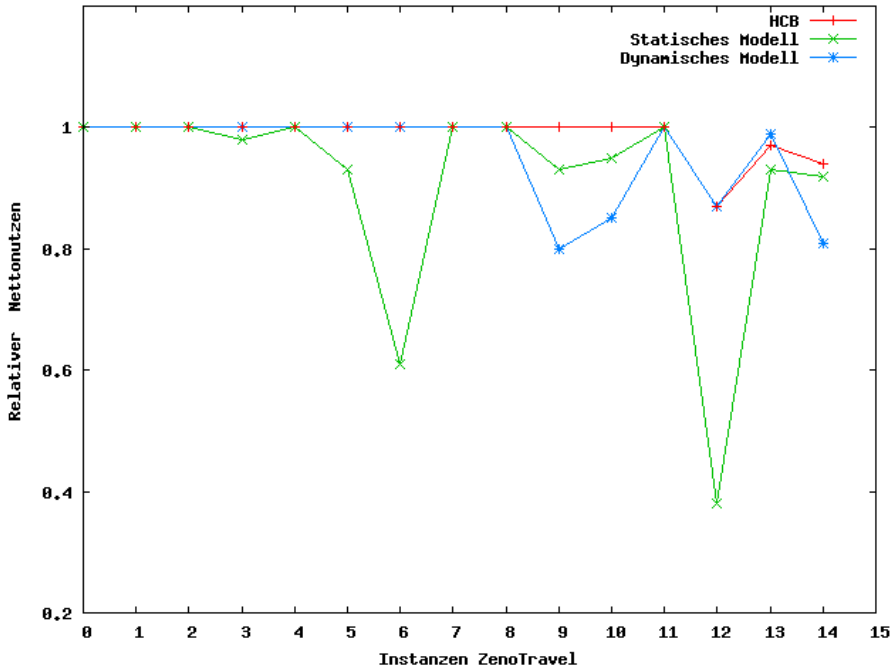


Abbildung 5.8: Erreichter relativer Nettonutzen ϕ verschiedener PS-Planungssysteme für 15 ZenoTravel-Instanzen.

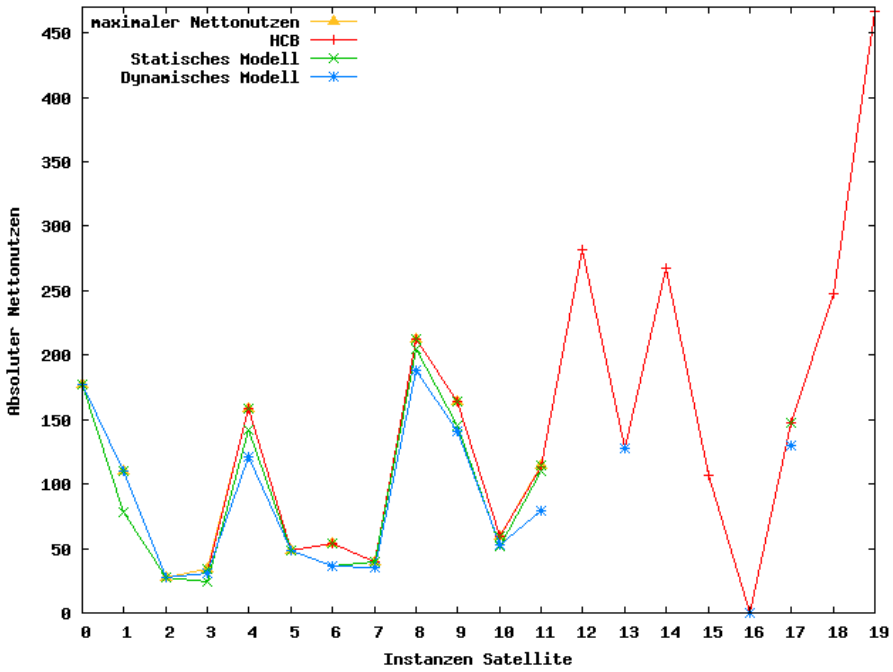


Abbildung 5.9: Erreichter absoluter Nettonutzen verschiedener PS-Planungssysteme für 20 SATELLITES-Instanzen.

mit mehr als 20 Teilzielen konnten in der Praxis mit den modellierenden Verfahren nicht gelöst werden.

5.5.3 Laufzeiten

Abbildung 5.13 zeigt die tatsächlichen Laufzeiten auf einem *AMD Athlon 64 3200+*. Es ist erkennbar, dass die Verfahren mit den meisten klassischen Planungen nicht zwingend die längsten Laufzeiten haben. Ein Grund hierfür besteht darin, dass die Planung für die *kleinen* Teilziele der Wissensbasis in der Regel relativ schnell möglich ist, während die mächtigeren Teilzielmengen aufwendigere und damit längere Planungen erfordern. Zusätzlich schwanken die Laufzeiten je nach Domäne und Instanz, so dass im Vergleich zur Anzahl der Planungen sehr unterschiedliche Ergebnisse erzielt werden. Für die sehr großen ZENOTRAVEL- und SATELLITES-Instanzen mit mehr als 20 Teilzielen zeigt sich das Hillclimbing-Verfahren überlegen. Hier konnten viele Instanzen in der Praxis mit den modellierenden Verfahren nicht gelöst werden oder die Laufzeit liegt im zweistelligen Minuten- oder sogar im Stundenbereich.

5.6 Vergleich mit anderen Systemen

Der Vergleich unserer Ergebnisse mit denen anderer Planungssysteme geht mit einigen Schwierigkeiten einher: Zur Evaluation steht nur ein PS-Planungssystem zur

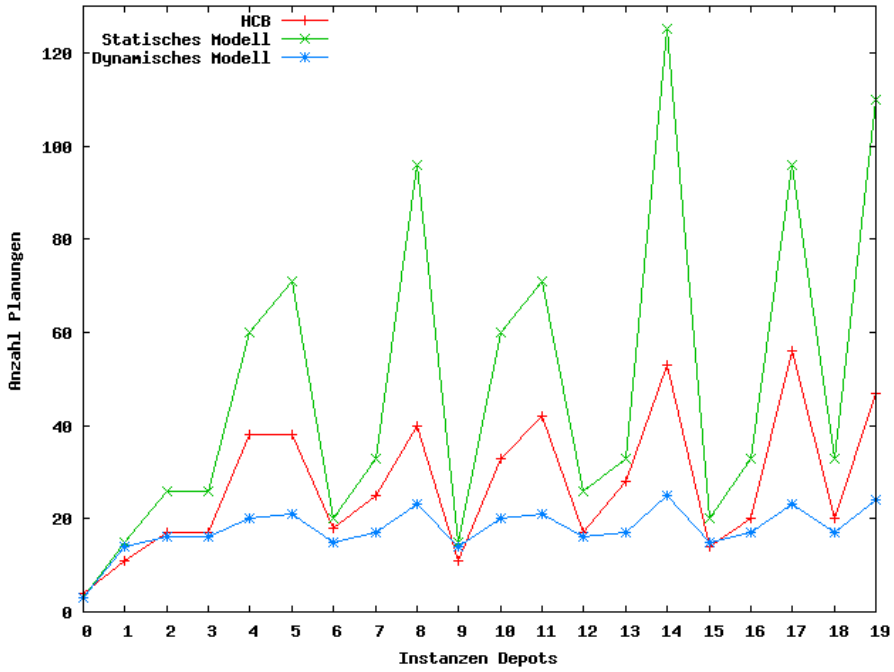


Abbildung 5.10: Anzahl klassischer Planungen verschiedener PS-Planungssysteme für 20 DEPOTS-Instanzen.

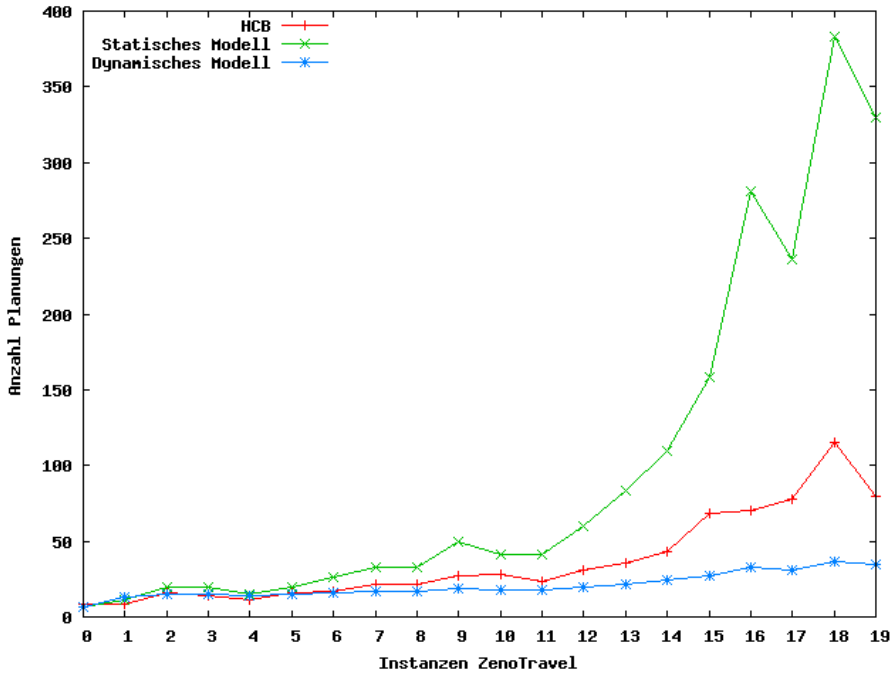


Abbildung 5.11: Anzahl klassischer Planungen verschiedener PS-Planungssysteme für 20 ZENOTRAVEL-Instanzen.

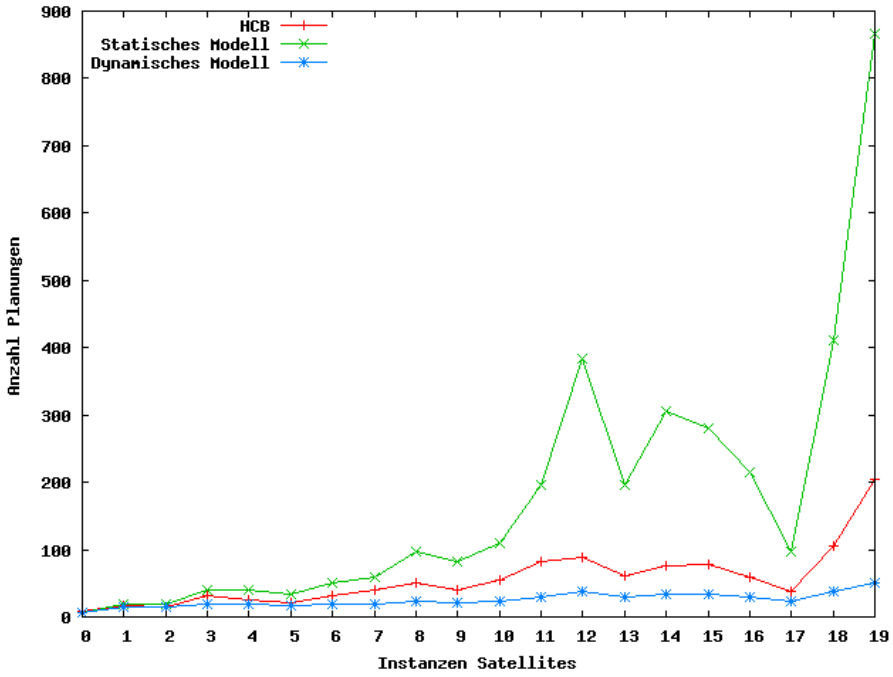


Abbildung 5.12: Anzahl klassischer Planungen verschiedener PS-Planungssysteme für 20 SATELLITES-Instanzen.

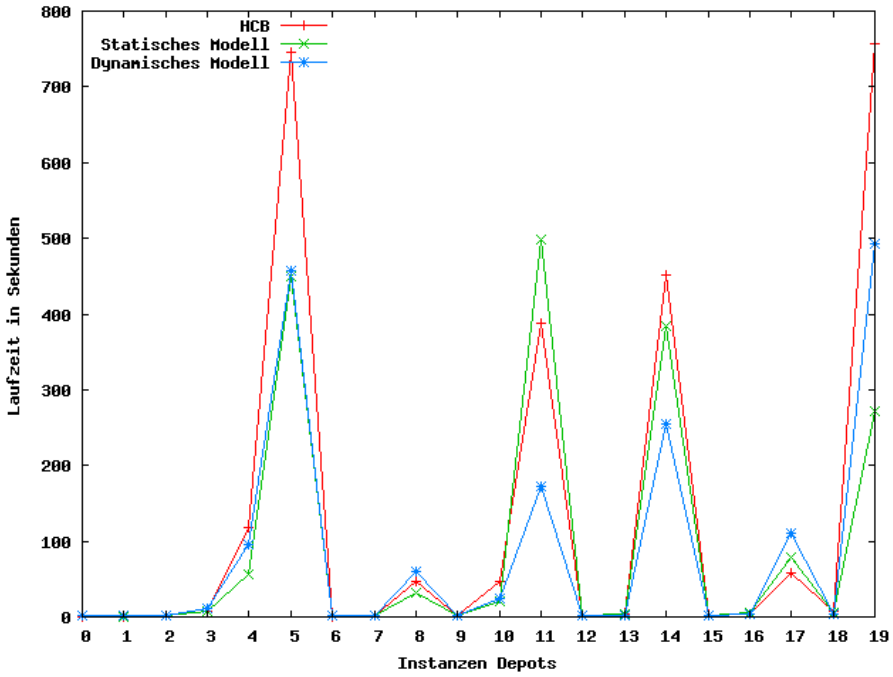


Abbildung 5.13: Laufzeit verschiedener PS-Planungssysteme für 20 DEPOTS-Instanzen.

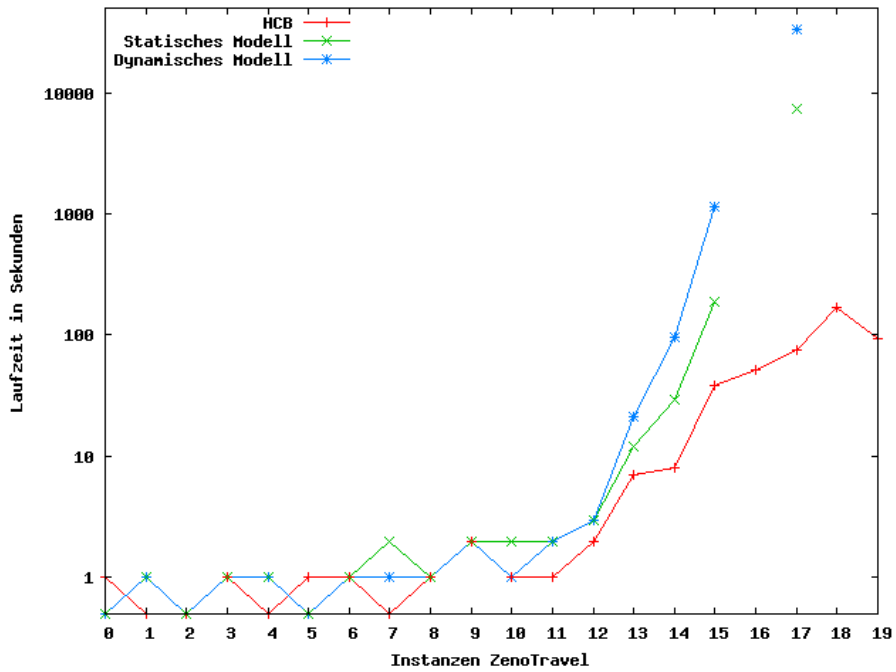


Abbildung 5.14: Laufzeit verschiedener PS-Planungssysteme für 20 ZenoTravel-Instanzen. Die y-Achse ist logarithmisch skaliert.

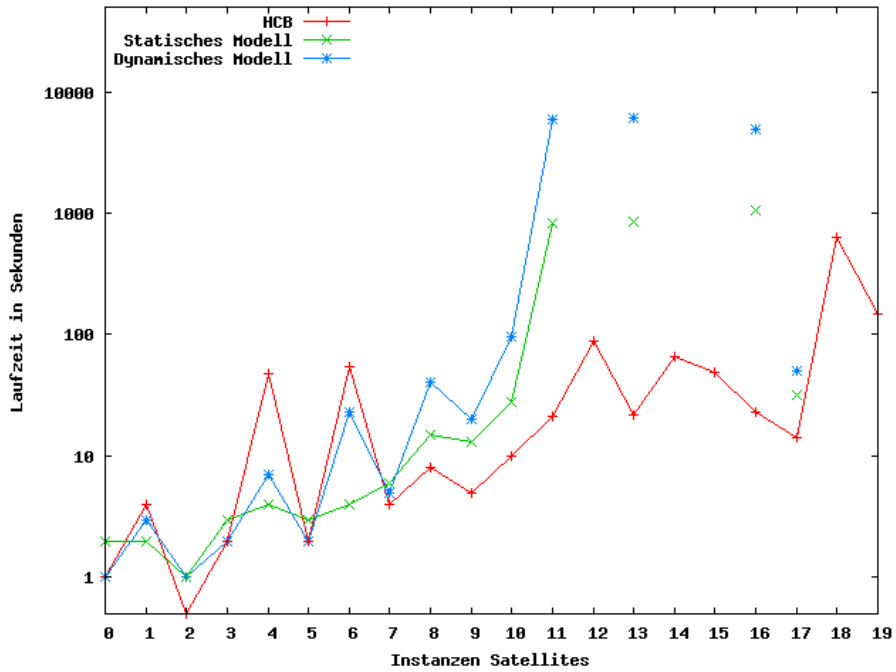


Abbildung 5.15: Laufzeit verschiedener PS-Planungssysteme für 20 Satellites-Instanzen. Die y-Achse ist logarithmisch skaliert.

Verfügung, welches eine Anpassung der PS-Planungsinstanzen verlangt. Systeme zur Planung von PDDL3-Problemen mit *simple preferences* liefern keine verwertbaren Ergebnisse.

5.6.1 $Sapa^{PS}$

Als weiteres teilerfüllendes Planungssystem steht lediglich der PS-Planer $Sapa^{ps}$ zur Verfügung. Dieser bezieht jedoch die in den hier verwendeten IPC3-Domänen gegebenen Metriken wie `(fuel-used)` nicht in die Planung ein, sondern verwendet offenbar grundsätzlich die Planlänge. Um überhaupt Ergebnisse erzielen zu können, wurden die 20 Probleme der Domäne DEPOTS modifiziert: Die Ladungsbeschränkungen der LKW wurden aus der Domänenbeschreibung entfernt, da $Sapa^{ps}$ die verwendete Syntax nicht beherrscht. Des weiteren wurde die Metrik in allen Problemen auf `(total-time)`, also die Planlänge gesetzt. Für diese veränderte DEPOTS-Version wurden neue Nutzenwerte berechnet, und die Instanzen wurden erneut mit bidirektionalem Hillclimbing geplant. Außerdem wurden die modifizierten Eingabedateien unter Verwendung der Nutzenfunktion in das spezielle Eingabeformat des Systems $Sapa^{ps}$ übersetzt und geplant.

Abbildung 5.16 zeigt den erreichten Nettonutzen des Systems im Vergleich zu den Ergebnissen des bidirektionalen Hillclimbings. Man beachte, das durch die Verwendung verschiedener Planer die Optimalitätsannahme nicht erhalten werden kann. Das heißt, dass die schlechteren Ergebnisse zwei verschiedene Ursachen haben können. Erstens kann eine Teilzielmenge mit niedrigerem Nettonutzen ausgewählt worden sein. Zweitens ist es auch möglich, dass die Pläne zur Erfüllung der Teilzielmengen höhere Kosten benötigen. Die Ergebnisse sind also nicht unmittelbar zu vergleichen. Es können keine sicheren Aussagen über die Qualität der Teilzielauswahl abgeleitet werden. $Sapa^{PS}$ gibt für viele Instanzen aus unbekannten Gründen keinen Plan zurück und erreicht deshalb nur Nettonutzen 0.

5.6.2 PDDL3-Planer

Eine sehr interessante Möglichkeit zum Vergleich von Ergebnissen bietet theoretisch die in Abschnitt 3.5.3.1 gezeigte Umcodierung in PDDL3-Instanzen mit *simple preferences*. Hierzu wurden die IPC3 Instanzen mit Nutzen versehen und in PDDL3 übersetzt. Leider konnte jedoch kein Planungssystem ausfindig gemacht werden, das für diese Instanzen sinnvolle Ergebnisse erzeugte. MIPS-XXL [EJN06] gibt grundsätzlich den leeren Plan zurück, während die verschiedenen SGPLAN-Versionen [CWH06] je nach Domäne häufig Fehlermeldungen oder negative Nettonutzen zurückliefern. Ein Vergleich der Ergebnisse war daher nicht möglich.

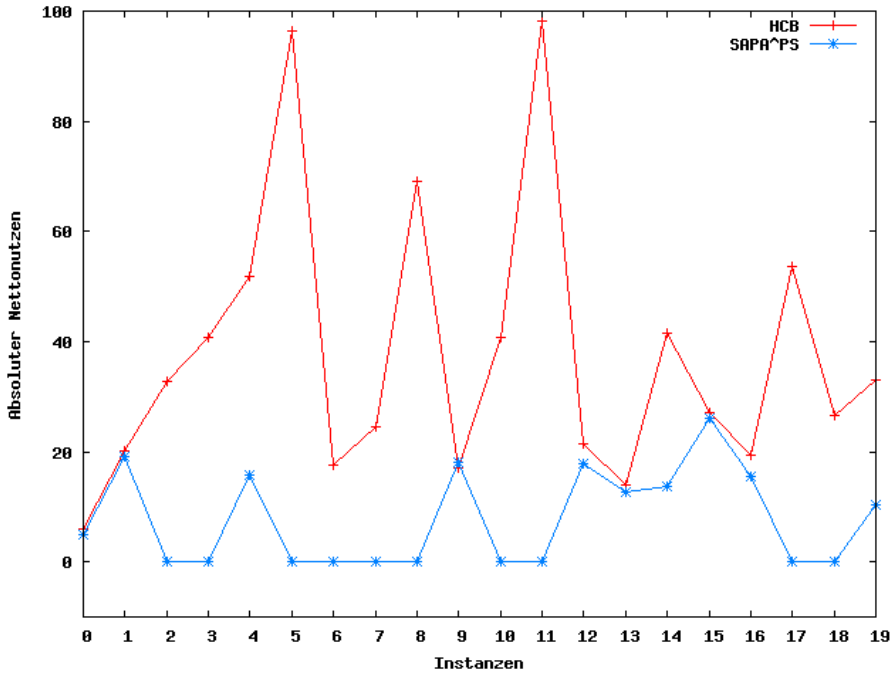


Abbildung 5.16: Absoluter erreichter Nettonutzen des bidirektionalen Hillclimbings (HCB) und des PS-Planers $Sapa^{ps}$ für 20 modifizierte DEPOTS-Instanzen.

5.7 Zusammenfassung

Anhand der vorgestellten Ergebnisse lässt sich keines der verwendeten Verfahren als grundsätzlich überlegen einstufen. Vielmehr schwanken die Ergebnisse sowohl bezüglich der Laufzeit als auch bezüglich des erreichten Nettonutzens je nach Planungsinstanz. Dies gilt sowohl im Vergleich der drei Konzepte Hillclimbing, statische und dynamische Modellierung, als auch für die verschiedenen Varianten. Auffällig ist weiterhin, dass sämtliche auf eine Wissensbasis gestützten Verfahren bei den gleichen Instanzen schlechte Ergebnisse liefern. Das Grundprinzip Basiswissen scheint also immer wieder an Grenzen zu stoßen. In der verwendeten Implementierung konnten nur mit den Hillclimbing-Verfahren für alle Instanzen der getesteten Domänen ein Ergebnis erzielt werden. Außerdem bleiben hier die Laufzeiten auch bei sehr großen Zielmengen moderat. Besonders die bidirektionale Variante sei an dieser Stelle nochmal hervorgehoben. Durch das sehr einfache zugrundeliegende Prinzip ergibt sich ein robuster und schneller PS-Planer, der häufig optimale und in der Regel gute Teilzielmengen identifiziert.

Betrachtet man alle Instanzen der Domänen DEPOTS und ZENOTRAVEL, in denen die optimale Zielmenge bekannt ist, ergeben sich folgende Ergebnisse. Von 35 Instanzen konnte mit bidirektionalem Hillclimbing bei 25 die optimale Zielmenge und bei 31 eine Zielmenge mit einem relativen Nettonutzen $\phi > 0{,}9$ erreicht wer-

den. Das statische Modell identifizierte 17 optimale und 26 sehr gute Teilzielmengen. Beim dynamischen Modell ergeben sich die Anzahlen 23 und 27. Man beachte jedoch, dass die 5 größten ZenoTravel-Probleme nicht eingerechnet sind, da der Wert des optimalen Nettonutzens hier unbekannt ist.

6 Zusammenfassung und Ausblick

6.1 Ergebnisse

Ziel dieser Arbeit war es, die teilerfüllende Handlungsplanung aus dem Blickwinkel der Zielmengen zu untersuchen. Der Schwerpunkt lag in der Auswahl der Teilzielmengen; die Planung selbst wurde mit einem bereits vorhandenen Planungssystem durchgeführt. Das Grundprinzip, für einzelne Teilzielmengen zu planen und Kosteninformation für andere Zielkombinationen abzuleiten, wurde auf verschiedene Arten eingesetzt. Als Maß für die Planqualität diente uns das Konzept des Nettonutzens. Es wurde eine Klasse von Hillclimbing-Algorithmen sowie ein statisches und ein dynamisches Nettonutzenmodell entwickelt. In den zur Evaluation verwendeten Planungsinstanzen gelang es allen Algorithmen, Pläne zu identifizieren, die einen hohen Nettonutzen erzielen.

Konkret ergaben sich für die 35 Instanzen der Domänen DEPOTS und ZENOTRAVEL mit bekanntem maximalem Nettonutzen die in der folgende Tabelle dargestellten Werte. Für drei repräsentative Verfahren ist der Anteil der Instanzen dargestellt, die den optimalen Nettonutzen ($\phi = 1$) bzw. einen sehr guten relativen Nettonutzen ($\phi > 0.9$) erreicht haben.

Verfahren	**Anteil identifizierte Teilziele mit**	
	$\phi = 1$	$\phi > 0{,}9$
HCB	0,71	0,89
Statisches Modell	0,49	0,74
Dynamisches Modell	0,66	0,77

Die Arbeit kann dazu beitragen, effiziente Planungssysteme für teilerfüllende Handlungsplanung zu entwickeln. Dies gilt insbesondere für den Bereich der Teilzielmengenauswahl.

6.2 Ausblick

Die vorliegende Arbeit hat gezeigt, dass die Zerlegung der Zielmenge in Teilziele mit assoziierten Planungsproblemen sinnvolle Konzepte ermöglicht. Es bieten sich jedoch vielfältige Möglichkeiten zur Verfeinerung bzw. anderweitiger Verwendung der Grundprinzipien.

In der Klasse der Hillclimbing-Algorithmen bieten sich etwa weitere Varianten an,

die eine breitere Abdeckung des Suchraumes mit sich bringen: Nicht untersucht wurde etwa der übliche Hillclimbing-Algorithmus, der jede mögliche Erweiterung der aktuellen Zielmenge testet. Auch auf eine Vertiefung von Verfahren mit randomisierten Elementen wurde verzichtet.

Im Bereich der Nettonutzenmodelle bleiben zahlreiche Möglichkeiten, den Aufbau der Wissensbasis oder die Modellierungsvorschrift zu gestalten. Des weiteren wurden die Modelle in den hier vorgestellten Verfahren direkt zur Teilzielauswahl verwendet. Nettonuzenmodelle eignen sich auch als Heuristiken für unterschiedliche heuristisch gesteuerte Suchverfahren.

Für dynamische Modelle bieten sich verschiedene Algorithmen aus dem Bereich des Maschinellen Lernens an. Möglich wäre z.B. ein neuronales Netz, das so trainiert wird, dass es für eingegebene Teilzielmengen einen Nettonutzenwert berechnet.

Grundsätzlich ist festzustellen, dass bisher nur die Kosten der Teilzielmengen in Betracht gezogen wurden. Durch eine genauere Untersuchung der Pläne, die zur Erfüllung notwendig sind, lassen sich jedoch weitere Informationen erzeugen. So kann z.B. erkannt werden, welche Ziele zusätzlich erfüllt werden oder welche Aktionen für viele Ziele notwendig sind. Die Möglichkeiten der vorgestellten Verfahren sind durch das Blackbox-Prinzip stark eingeschränkt. Ein komplettes PS-Planungssystem kann so entwickelt werden, dass wesentlich weniger Planungen durchgeführt werden müssen. Pläne zur Erfüllung von Teilzielmengen können z.B. für Obermengen erweitert werden. Pläne lassen sich im Hinblick darauf untersuchen, welche Aktionen zur Erfüllung welcher Teilziele dienen. Durch die erreichte Effizienzsteigerung können unter Umständen komplexere Algorithmen verwendet werden.

Abbildungsverzeichnis

Literaturverzeichnis

[BC05] Brafman, Ronen I. ; Chernyavsky, Yuri: Planning with Goal Preferences and Constraints. In: [BMR05], S. 182–191

[Ben06] Benton, J.: *Solving Goal Utility Dependencies and Simple Preferences in Partial Satisfaction Planning.* Doctoral Consortium of the International Conference on Automated Planning and Scheduling (ICAPS 2005), 2006

[BF97] Blum, Avrim L. ; Furst, Merrick L.: Fast Planning through Planning Graph Analysis. In: *Artificial Intelligence* 90 (1997), Nr. 1-2, S. 281–300

[BG01] Bonet, Blai ; Geffner, Hector: Planning as Heuristic Search. In: *Artificial Intelligence* 129 (2001), Nr. 1-2, S. 5–33

[BKD06] Benton, J. ; Kambhampati, Subbarao ; Do, Minh B.: *Yochan^{PS} PDDL3 Simple Preferences as Partial Satisfaction Planning.* Booklet of the Fifth International Planning Competition, 2006

[BMR05] Biundo, Susanne (Hrsg.) ; Myers, Karen L. (Hrsg.) ; Rajan, Kanna (Hrsg.): *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), June 5-10 2005, Monterey, California, USA.* AAAI, 2005

[Byl94] Bylander, Tom: The computational Complexity of Propositional STRIPS Planning. In: *Artificial Intelligence* 69 (1994), Nr. 1-2, S. 165–204

[CWH06] Chen, Y. ; Wah, B. W. ; Hsu, C.: Temporal Planning using Subgoal Partitioning and Resolution in SGPlan. In: *J. Artif. Intell. Res. (JAIR)* 26 (2006), S. 323–369

[DK04] Do, Minh B. ; Kambhampati, Subbarao: Partial Satisfaction (Over-Subscription) Planning as Heuristic Search. In: *Knowledge Based Computer Systems (KBCS)*, 2004

[EH01] Edelkamp, Stefan ; Helmert, Malte: The Model Checking Integrated Planning System (MIPS). In: *AI Magazine* 22 (2001), Nr. 3, S. 67–71

[EJN06] Edelkamp, Stefan ; Jabbar, Shahid ; Nazih, Mohammed: *Large-Scale Optimal PDDL3 Planning with MIPS-XXL.* Booklet of the Fifth International Planning Competition, 2006

[FL03] Fox, Maria ; Long, Derek: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. In: *J. Artif. Intell. Res. (JAIR)* 20 (2003), S. 61–124

[GL05] Gerevini, Alfonso ; Long, Derek: Plan Constraints and Preferences in PDDL3 / Department of Electronics for Automation, University of Brescia, Italy. 2005. – Forschungsbericht

[GSS04] Gerevini, Alfonso ; Saetti, Alessandro ; Serina, Ivan: *LPG-TD: a Fully Automated Planner for PDDL2.2 Domains.* International Planning Competition, 14th Int. Conference on Automated Planning and Scheduling (ICAPS-04), Booklet of the System Demo Section, Whistler, Canada, 2004

[Hel01] Helmert, Malte: *On the Complexity of Planning in Transportation and Manipulation Domains*, Albert-Ludwigs-Universität Freiburg, Diplomarbeit, 2001

[Hel03] Helmert, Malte: Complexity results for standard benchmark domains in planning. In: *Artificial Intelligence* 143 (2003), Nr. 2, S. 219–262

[Hel06] Helmert, Malte: New Complexity Results for Classical Planning Benchmarks. In: *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, 2006, S. 52–61

[HN01] Hoffmann, Jörg ; Nebel, Bernhard: The FF Planning System: Fast Plan Generation Through Heuristic Search. In: *J. Artif. Intell. Res. (JAIR)* (2001), S. 253–302

[Hof02] Hoffmann, Jörg: Extending FF to Numerical State Variables. In: *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-02).* Lyon, France, Juli 2002, S. 571–575

[KS92] Kautz, Henry A. ; Selman, Bart: Planning as Satisfiability. In: *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, 1992, S. 359–363

[LF99] Long, D. ; Fox, M.: Efficient Implementation of the Plan Graph in STAN. In: *J. Artif. Intell. Res. (JAIR)* 10 (1999), S. 87–115

[NGT04] Nau, Dana ; Ghallab, Malik ; Traverso, Paolo: *Automated Planning: Theory & Practice.* San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2004

[NKS02] Nguyen, XuanLong ; Kambhampati, Subbarao ; Sanches Nigenda, Romeo: Planning Graph as the Basis for Deriving Heuristics for Plan Synthesis by State Space and CSP Search. In: *J. Artif. Intell. Res. (JAIR)* 135 (2002), Nr. 1-2, S. 73–124

[Rin05] RINTANEN, Jussi: *Introduction to Automated Planning.* Vorlesungsskript, Principles of AI-Planning, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, 2005

[RN03] RUSSELL, Stuart ; NORVIG, Peter: *Artificial Intelligence: A Modern Approach.* 2nd edition. Prentice-Hall, Englewood Cliffs, NJ, 2003

[Ron98] RONALD, S.: More Distance Functions for Order-based Encodings. In: *Proceedings of the IEEE conference on evolutionary computation*, 1998

[SK05] SANCHEZ NIGENDA, Romeo ; KAMBHAMPATI, Subbarao: Planning Graph Heuristics for Selecting Objectives in Over-subscription Planning Problems. In: [BMR05], S. 192–201

[Smi04] SMITH, David E.: Choosing Objectives in Over-Subscription Planning. In: *ICAPS*, 2004, S. 393–401

[ST01] SLANEY, John ; THIEBAUX, Sylvie: Blocks World revisited. In: *Artificial Intelligence* 125 (2001), Nr. 1-2, S. 119–153

[vSDK04] VAN DEN BRIEL, Menkes ; SANCHEZ NIGENDA, Romeo ; DO, Minh B. ; KAMBHAMPATI, Subbarao: Effective Approaches for Partial Satisfaction (Over-Subscription) Planning. In: *AAAI*, 2004, S. 562–569

[vSK04] VAN DEN BRIEL, Menkes ; SANCHEZ, Romeo ; KAMBHAMPATI, Subbarao: *Over-Subscription in Planning: a Partial Satisfaction Problem.* ICAPS Workshop on Integrating Planning into Scheduling, 2004

[WH94] WILLIAMSON, Mike ; HANKS, Steve: Optimal Planning With a Goal-Directed Utility Model. In: *Proceedings of the Second International Conference on AI Planning Systems*, 1994, S. 176–181