MAXPLAN

Ein neuer Ansatz für probabilistisches Planen
(Stephen M. Majercik & Michael L. Littman 1998)
Ausarbeitung

Eric Auer

11. August 2000

Abstract

Klassische KI-Planungstechniken können in grossen deterministischen Welten planen, **Operations research Techniken** können mit Unsicherheiten arbeiten, versagen aber für grössere Welten. **MAXPLAN** wandelt eine probabilistische Planungsaufgabe in ein **E-MAJSAT** Problem um, das effizient bearbeitet werden kann.

Die Aufgabenstellung bei einem **E-MAJSAT Problem** lautet: Finde die Belegung der Wahlvariablen, die bei einer KNF-Formel mit Wahl- und Zufallsvariablen die Wahrscheinlichkeit der Erfüllung maximiert. **Bei MAXPLAN** entspricht die Belegung der Wahlvariablen dem gewählten Plan, die Zufallsvariablen repräsentieren die Unsicherheit in der Aufgabe.

1 Problemrepräsentation bei MAXPLAN

Probabilistische Planprobleme sind Quadrupel aus einer Zustandsmenge, einem Startzustand daraus, Aktionen und Zielzuständen (die eine Untermenge der Zustandsmenge darstellen): $< S, s_0, A, G >$ (Die von den Aktionen bewirkten Zustandswechsel können probabilistisch sein)

Bei MAXPLAN wird die Aufgabe als sequentieller Effektbaum (ST, sequential effects tree, sind äquivalent zu two timeslice Bayes networks) eingelesen und mit einem LISP-Programm in ein E-MAJSAT Problem umgewandelt.

Das E-MAJSAT Problem wird als KNF Formel dargestellt, für die die Belegung von Wahlvariablen gesucht wird, die unter Berücksichtigung der Zufallsvariablen die Erfüllungwahrscheinlichkeit der KNF Formel maximiert.

SAT Probleme sind NP-vollständig, die bei MAXPLAN vorkommenden E-MAJSAT Probleme sind NP^{PP}-vollständig, wenn Planhorizont und Plangrösse durch ein Polynom der Problemgrösse beschränkt sind (SAT: Erfüllbarkeitsproblem / E-MAJSAT: Erweiterung von SAT).

1.1 Beispiel „Sandburg-67"

Eine Sandburg soll gebaut werden, dabei stehen die Aktionen baueBurg und grabeGraben zur Verfügung.

Der Bau eines Burggrabens erhöht die Erfolgsaussichten des Burgenbaus.

Die Zustände sind Kombinationen der Prädikate Burg und Graben.

$$
\begin{aligned}
S = \{ \quad & (\neg\mathsf{Burg}, \neg\mathsf{Graben}),\ (\neg\mathsf{Burg}, \mathsf{Graben}), \\
& (\mathsf{Burg}, \neg\mathsf{Graben}),\ (\mathsf{Burg}, \mathsf{Graben})\ \} \\
s_0 = (\quad & \neg\mathsf{Burg}, \neg\mathsf{Graben}\ ) \\
G = \{ \quad & (\mathsf{Burg}, \neg\mathsf{Graben}),\ (\mathsf{Burg}, \mathsf{Graben})\ \} \\
A = \{ \quad & \text{baueBurg},\ \text{grabeGraben}\ \}
\end{aligned}
$$

ST für die Aktion „grabeGraben**":**

- Wirkung auf das Prädikat Graben:

  Die Erfolgswahrscheinlichkeit beträgt 0.5 (d_1), ein schon vorhandener Graben bleibt immer bestehen.

- Wirkung auf das Prädikat Burg:

  Die Aktion hat keine Wirkung auf dieses Prädikat.

ST für die Aktion „baueBurg**":**

- Wirkung auf das Prädikat Burg:

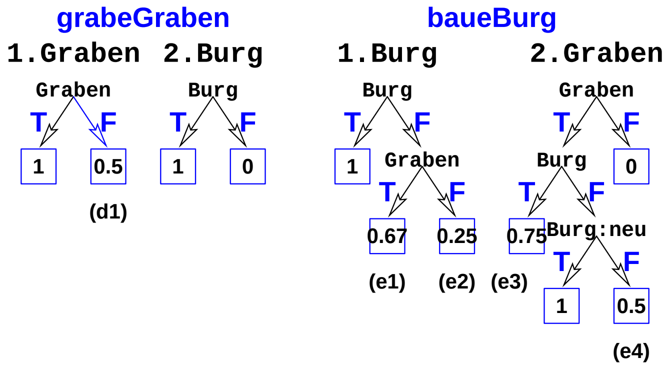
  Eine schon vorhandene Burg bleibt bestehen, sonst hängt die Erfolgswahrscheinlichkeit von der Existenz eines Grabens ab.

Burg muss hier vor Graben betrachtet werden, da der neue Wert von Burg auf den neuen Wert von Graben wirkt! Die Ordnung der ST ist also wichtig.

- Wirkung auf das Prädikat Graben:

  Der Versuch, eine Burg zu bauen, kann den Graben zerstören, wenn schon eine Burg stand oder aber der Versuch, eine Burg zu bauen, fehlgeschlagen ist (hier wird der Wert von Burg : neu verwendet). Beim Burgenbau entsteht niemals ein Graben.

Darstellung des ST als Baum:



1.2 Darstellung des Beispielproblemes als KNF

Zunächst wird der ST modifiziert, um die Umwandlung in eine KNF zu vereinfachen: Jedes Blatt mit einem Wert x **zwischen** 0.0 und 1.0 wird durch einen Knoten ersetzt, der über eine Zufallsvariable (die mit Wahrscheinlichkeit x wahr ist) zu 2 Blättern mit den Werten 1.0 und 0.0 verzweigt. Danach haben alle Blätter 1.0 oder 0.0 als Wert.

Für jede Aktion a_i und jeden Zeitschritt $j = t_1 \dots t_N$ (N ist der Planungshorizont) wird eine Variable $a_{i,j}$ eingeführt.

Für jede Zustandsvariable p_i und jeden Zeitschritt $j = t_0 \dots t_N$ wird eine Variable $p_{i,j}$ eingeführt.

Ausser für den Zeitpunkt t_0 werden diese mit Variablen 0.5 vorbelegt, denn sie sind nicht vollständig von den Aktionen determiniert. Die Gesamtwahrscheinlichkeit jedes Planes sinkt dadurch mit dem Faktor 0.5^{P*N}, eine Anpassung der Bewertung ist nötig.

Für jede Zufallsvariable r_i und jeden Zeitschritt $j = t_1 \dots t_N$ wird eine Variable $r_{i,j}$ eingeführt. Damit wird später die Unsicherheit modelliert.

Insgesamt verwendet die E-MAJSAT Repräsentation also $V = AN + P(N+1) + RN$ Variablen.

$r_{i,x} = r_{i,y}$ für alle i und alle $x, y \in 1 \dots N$, Von den $a_{i,j}$ ist pro i immer genau eine wahr, es gibt also unnötig viele Variablen – siehe „Zukünftige Verbesserungen" in Abschnitt 4.

1.3 Klauseln in der KNF Darstellung

Beispiel: KNF für Sandburg-67 mit 1 Zeitschritt:

Der Zweck der einzelnen Klauseln wird direkt im Anschluss beschrieben.

$(\overline{\text{Graben}_0}) \wedge (\overline{\text{Burg}_0}) \wedge (\text{Burg}_1) \wedge$

$(\text{grabeGraben}_1 \vee \text{baueBurg}_1) \wedge$
$(\overline{\text{grabeGraben}_1} \vee \overline{\text{baueBurg}_1}) \wedge$
$(\overline{\text{grabeGraben}_1} \vee \overline{\text{Graben}_0} \vee \text{Graben}_1) \wedge$

$(\overline{\text{grabeGraben}_1} \vee \text{Graben}_0 \vee \overline{d_{1,1}} \vee \text{Graben}_1) \wedge$
$(\overline{\text{grabeGraben}_1} \vee \text{Graben}_0 \vee d_{1,1} \vee \overline{\text{Graben}_1}) \wedge$

$(\overline{\text{baueBurg}_1} \vee \overline{\text{Graben}_0} \vee \overline{\text{Burg}_0} \vee \overline{e_{3,1}} \vee \text{Graben}_1) \wedge$
$(\overline{\text{baueBurg}_1} \vee \overline{\text{Graben}_0} \vee \overline{\text{Burg}_0} \vee e_{3,1} \vee \overline{\text{Graben}_1}) \wedge$
$(\overline{\text{baueBurg}_1} \vee \overline{\text{Graben}_0} \vee \text{Burg}_0 \vee \overline{\text{Burg}_1} \vee \text{Graben}_1) \wedge$

$(\overline{\text{b.B.}_1} \vee \overline{\text{Graben}_0} \vee \text{Burg}_0 \vee \text{Burg}_1 \vee \overline{e_{4,1}} \vee \text{Graben}_1) \wedge$
$(\overline{\text{b.B.}_1} \vee \overline{\text{Graben}_0} \vee \text{Burg}_0 \vee \text{Burg}_1 \vee e_{4,1} \vee \overline{\text{Graben}_1}) \wedge$
$(\overline{\text{baueBurg}_1} \vee \text{Graben}_0 \vee \overline{\text{Graben}_1}) \wedge$

$(\overline{\text{grabeGraben}_1} \vee \overline{\text{Burg}_0} \vee \text{Burg}_1) \wedge$
$(\overline{\text{grabeGraben}_1} \vee \text{Burg}_0 \vee \overline{\text{Burg}_1}) \wedge$

$(\overline{\text{baueBurg}_1} \vee \overline{\text{Burg}_0} \vee \text{Burg}_1) \wedge$
$(\overline{\text{baueBurg}_1} \vee \text{Burg}_0 \vee \overline{\text{Graben}_0} \vee \overline{e_{1,1}} \vee \text{Burg}_1) \wedge$

$(\overline{\text{baueBurg}_1} \vee \text{Burg}_0 \vee \overline{\text{Graben}_0} \vee e_{1,1} \vee \overline{\text{Burg}_1}) \wedge$
$(\overline{\text{baueBurg}_1} \vee \text{Burg}_0 \vee \text{Graben}_0 \vee \overline{e_{2,1}} \vee \text{Burg}_1) \wedge$

$(\overline{\text{baueBurg}_1} \vee \text{Burg}_0 \vee \text{Graben}_0 \vee e_{2,1} \vee \overline{\text{Burg}_1})$

Anfangsbedingungen und Ziel:
Jeder Variable des Anfangszustandes p_{i,t_0} und jeder Zielbedingung entspricht eine atomare Klausel.

Einander ausschliessende Aktionen:
Zu jedem Zeitpunkt ausser t_0 darf maximal und muss minimal eine Aktion stattfinden. Das wird mit $\binom{A}{2} + 1 = \frac{1}{2}(A^2 + A) + 1$ Klauseln beschrieben.

Bäume:
Jeder Pfad durch jeden der Bäume (nach der Anpassung des ST, die ganz am Anfang stattfindet) wird durch eine Klausel pro Zeitpunkt ausgedrückt.

Wenn wir zum Beispiel graben und noch keinen Graben haben, werden wir mit Wahrscheinlichkeit $d_{1,1}$ im nächsten Schritt einen Graben haben.

Das ergibt die Implikation
$\text{grabeGraben}_1 \wedge \overline{\text{Graben}_0} \wedge d_{1,1} \Rightarrow \text{Graben}_1$,
in Klauselschreibweise also
$\overline{\text{grabeGraben}_1} \vee \text{Graben}_0 \vee \overline{d_{1,1}} \vee \text{Graben}_1$

Insgesamt enthält die fertige KNF maximal $2P + (1 + \binom{A}{2})N + 2N \sum_{i=1}^{A} L_i$ Klauseln (L_i ist die Anzahl der Blätter im Baum von Aktion a_i).

Für die Komplexität der KNF ergibt sich damit: Die Grösse der Klauseln wird von der Pfadlänge der Bäume dominiert. Die Anzahl der Klauseln ist ein Polynom niederer Ordnung der Problemgrösse.

2 Der Problemlöser von MAXPLAN

MAXPLAN baut auf dem bewährten DPLL-Algorithmus für Erfüllbarkeitsprobleme (Davis-Putnam-Logemann-Loveland Verfahren für Erfüllbarkeitstests, entwickelt 1962) auf, welches noch immer der beste Algorithmus ist, den die Autoren kennen.

Zuerst müssen alle die KNF erfüllenden Belegungen gefunden werden. Dann wird für jede mögliche Belegung der Wahlvariablen die Gesamtwahrscheinlichkeit (für alle Belegungen der anderen Variablen) berechnet, die KNF zu erfüllen.

Diese entspricht der Erfolgswahrscheinlichkeit des durch die Belegung der Wahlvariablen gewählten / beschriebenen Planes. So kann der Plan mit den besten Erfolgsaussichten gefunden werden.

Die Suche nach den die KNF erfüllenden Belegungen baut einen Binärbaum auf, in dem jeder Knoten einer Variable entspricht. Es ist wichtig, nur die „interessanten” Knoten zu expandieren.

Der Wert eines Knotens ist dabei der durch die Wahrscheinlichkeit gewichtete Durchschnitt der Unterbäume, bzw. deren Maximum, wenn es um eine Wahlvariable geht.

Einige Begriffsdefinitionen zum Problemlöser:

- **Aktive Variable:** Variable, der noch kein Wahrheitswert zugewiesen wurde
- **Aktive Klausel:** Klausel, die noch nicht durch zugewiesene Variablen erfüllt wird
- **Aktuelle KNF Teilformel:** Wird durch die Menge noch aktiver Klauseln und der darin vorkommenden Variablen beschrieben
- **Wert einer KNF Teilformel:** Maximum der Erfolgswahrscheinlichkeiten $\sum_S(\ldots)$ (wobei S die Menge aller die KNF erfüllenden Belegungen der Zufallsvariablen ist) für alle möglichen Pläne D:

  $$\rightarrow \quad \max_D \sum_S \Big(\prod_{v_i \in CH} \pi_i^{v_i} * (1-\pi_i)^{1-v_i} \Big)$$

  (D sind alle möglichen Belegungen der Wahlvariablen, π_i ist die Wahrscheinlichkeit, dass die Zufallsvariable v_i wahr ist.)

2.1 Eliminierung unnötiger Berechnungen

2.1.1 DPLL-Pruningregeln

DPLL entscheidet mit folgenden 3 Regeln, wann ein Unterbaum nicht mehr weiterverfolgt werden muss:

- **Unwichtige Variablen belegen** (**IRR**, irrelevant variable elimination):

  Einer aktiven Variable, die in keiner aktiven Klausel mehr vorkommt, wird der Wert True zugewiesen.

- **Atomare Klauseln erfüllen** (**UNIT**, unit propagation):

  Einer Variablen, die allein eine aktive Klausel bildet, wird der passende Wert (der diese aktive Klausel wahr macht) zugewiesen.

- **Eindeutige Variablen entscheiden** (**PURE**, purification):

  Einer aktiven Variable, die in allen aktiven Klauseln nur noch negiert bzw. nur noch nicht negiert vorkommt, wird der passende Wert zugewiesen, mit dem alle diesen Klauseln erfüllbar bleiben.

2.1.2 Heuristiken für die Planbewertung

Wenn kein Pruning möglich ist, muss an einer Variablen verzweigt werden (bevorzugt an einer Wahlvariable). Der Wert des Verzweigungsknotens wird aus dem Wert der Unterbäume gemessen (bei Wahlvariablen das Maximum aus den Werten der Unterbäume, sonst der durch die Wahrscheinlichkeit gewichtete Durchschnitt). Die richtige Wahl der Verzweigungsvariablen in dieser Situation ist wichtig für die Effizienz der Berechnung.

Drei Verzweigungsheuristiken wurden getestet, Testproblem war Sandburg-67 (gemessen wurde nur die Zeit zur Bewertung eines festen Planes: Abwechselnd graben und bauen, letzter Schritt ist bauen):

- **Zufall (random):**

  Wähle die nächste aktive Variable aus einer zu Beginn festgelegten zufälligen Ordnung der Variablen.

- **Gier (sats most clauses):**

  Wähle die aktive Variable, die die meisten Klauseln erfüllen kann.

- **Zeit (time-ordered):**

  Wähle die aktive Variable, die als nächste im Plan auftaucht.

2.2 Optimierung des Planbewerters

Wie erwartet benötigt man bei wachsendem Planhorizont exponentiell mehr Zeit. Messungen zeigen aber, dass man mit der Zeit-Heuristik in einem vorgegebenen Zeitraum die längsten Pläne (für Sandburg-67) findet:

Heuristik	max. Schritte in	
	1 Minute	10 Min.
Zufall	8	10
Gier	11	13
Zeit	15	19

Messungen mit verschiedenen Kombinationen der Pruningregeln (jeweils in Kombination mit der Zeit-Heuristik) ergeben, dass das Weglassen von Regeln die Berechnung stark verlangsamt.

Die Rechenzeit bei Verwendung von nur PURE / UNIT, nur UNIT und ganz ohne Pruning steigt extrem an. Die Berechnung von 5-Schritt-Plänen brauchte bei eingeschränktem Pruning jeweils schon Tage.

Da ein einfacher Algorithmus mit dynamischer Programmierung (**ENUM**) bei der Planbewertung nur zur Planlänge lineare Zeit braucht, wurde **Memoisierung** der Teilformeln und ihrer Werte in MAXPLAN eingebaut.

Es wurden also Berechnungsergebnisse aufbewahrt, um die wiederholte Berechnung gleicher Teilformeln zu vermeiden. Nun konnten in der gleichen Zeit statt 18 ganze 110 Schritte bewertet werden!

Scheinbar behinderte jedoch das Pruning die Wirkung der Memoisierung. Also wurden Vergleichsmessungen mit eingeschränktem Pruning durchgeführt, „kein Pruning” bzw. „nur UNIT” zeigte sich als beste Wahl (bei anderen Tests übertraf „nur UNIT” „kein Pruning”; alle Tests mit Memoisierung und Zeit-Heuristik).

Pruning-regeln	Planlänge in 20 Sekunden	Rechenzeit für 100 Schritte
alle DPLL	≈ 50	≈ 140s
ohne IRR	≈ 60	≈ 90s
nur UNIT	≈ 95	≈ 25s
keine	≈ 100	≈ 20s

Der so optimierte MAXPLAN wurde schliesslich am kompletten Sandburg-67 Problem bis zu einem Planhorizont von 10 Schritten getestet.

Der beste Plan mit 10 Schritten ist G-B-G-B-B-G-B-G-B-B (B: baueBurg, G: grabeGraben). Er hat eine Erfolgswahrscheinlichkeit von etwa 97%. Auf einer Sparc Ultra 1-140 mit 128 MB RAM findet MAXPLAN diesen Plan in etwa 12 Sekunden.

3 Vergleich von MAXPLAN mit anderen Planern

MAXPLAN wurde mit 3 weiteren Planern verglichen:

- **BURIDAN**, ein klassischer KI-Planer, Partial-order Planen für probabilistische Probleme.
- **ENUM**, Planaufzählung und dynamische Programmierung für die Planbewertung.
- **POMDP**, Lösung teilweise beobachtbarer MDP (Markov Decision Process) mit dynamischer Programmierung (inkrementelles Pruning).

Minimax-Planer kamen nicht in Frage, da auch Probleme gelöst werden sollten, für die es keinen 100% erfolgversprechenden Plan gibt.

Der Vergleich mit BURIDAN erwies sich als schwierig, da bei BURIDAN die (nicht automatisierbare) Wahl einer zum Problem passenden Methode entscheidend für die Leistung ist. In jedem Fall war jedoch MAXPLAN über 10 mal schneller bei der Lösung des mit BURIDAN beschriebenen **„Bombe-im-Klo” Problemes**[1]:

MAXPLAN benötigte 0.3 bis 0.7 Sekunden (alle Kombinationen von Pruningregeln ausser kein Pruning benötigten nur 0.3 Sekunden, zum Beispiel nur UNIT Pruning).

BURIDAN (auf einer etwas langsameren Maschine) benötigte je nach Bewertungsmethode 6.9 bis 6736 Sekunden (Vorwärts 6.9 Sekunden, Rückwärts 6736 Sekunden, Abfrage 64.9 Sekunden, Netzwerk 152 Sekunden).

3.1 Welcher Planer skaliert besser?

ENUM und POMDP müssen alle 2^P Zustände aufzählen und benötigen zur Grösse des Zustandsraumes exponentielle Resourcen. ENUM bewertet alle A^N Pläne und braucht also auch zum Planhorizont exponentielle Zeit. **Dennoch ist MAXPLAN nicht immer schneller!**

MAXPLAN versucht nur den nötigen Teil des Problemraumes zu erforschen, ohne die Optimalität aufs Spiel zu setzen. Bei Sandburg-67 kann es aber die Unbeobachtkeit des Zustandes nicht ausnutzen. POMDP kann das und braucht zur Planlänge lineare Zeit, MAXPLAN exponentielle.

Die erreichte Plantiefe in Abhängigkeit von der verfügbaren Zeit:

Zeit: (Sek.)	POMDP	ENUM	MAXPLAN
	Schritte		
0.1	6	13	7
10	50	16	9

[1]Eins von 2 oder mehr Paketen enthält eine Bombe, um Bomben zu entschärfen kann man sie ins Klo werfen

MAXPLAN kann einfache Probleme ausnutzen:

Bei dem **Bomben-entschärfen Prolem** kann MAXPLAN für 1 bis 10 Pakete die Lösung in etwa konstanter Zeit (etwa $\frac{1}{10}$ Sekunde) finden. Der optimale Plan besteht immer aus zwei Schritten.

Bei **Bomben-entschärfen** gibt es die Aktionen Untersuchen und Entschärfen, wobei Untersuchen alle Bomben findet und Entschärfen alle Bomben zugleich entschärft, falls man weiss, in welchen Paketen die Bomben sind.

POMDP und ENUM benötigen zur Anzahl der Pakete trotzdem exponentielle Zeit. ENUM braucht ab 6 Paketen mehr Zeit als MAXPLAN, POMDP bereits bei 1 Paket. In $\frac{1}{2}$ Sek. schaffen POMDP und ENUM 5 bzw. 7 Pakete.

4 Zukünftige Verbesserungen

Skalierung:
Ausser den hier erwähnten wurde noch das rutschiger-Griff Problem und Varianten des Bombe-im-Klo Problems (bei der das Klo verstopfen kann) untersucht, aber es muss noch weiter geforscht werden.

Bomben-entschärfen zeigte, dass MAXPLAN *einfache* Lösungen auch in einem wachsendem Zustandsraum effizient finden kann.

KNF:
Die aus dem Problem gewonnene KNF Formel ist nicht optimal, die Speicherung der KNF kann auch eleganter gelöst werden.

Verzweigungsheuristik:
Die Zeit-Heuristik wertet alle Variablen eines Zeitschrittes gleich. Die Klauseln eines Zeitschrittes bilden aber einen **natürlichen Block**, d.h. sie kommen nur in einem Zeitschritt und dessen direkten Nachbarn vor. Das Problem kann in diese natürlichen Teilformeln zerlegt werden (Teile-und-herrsche Strategie).

Memoisierung:
Alle Teilformeln und ihre Werte zu speichern ist unnötig und verbraucht zuviel Speicher (bei Sandburg-67 reicht der Speicher des Testrechners nur für 15 Schritte).

Eine bestimmte Menge Speicher als Cache für Teilformeln zu reservieren und eine gute Strategie finden, welche Teilformeln gespeichert werden (z.B. abhängig von ihrer Grösse, Komplexität und Benutzungshäufigkeit) wäre ein Ausweg.

4.1 Zukünftige Erweiterungen

Verstärkungslernen:
Verstärkungslernen kann eine bessere Verzweigungs- und Cache-Heuristik finden. Dabei wird es aber oft an Trainingsdaten mangeln, so dass das Problem entsteht, einige „typische" Probleme zu finden, an denen das System trainieren kann.

Näherungsverfahren:
In der Praxis ist es nützlich, schnell eine ungefähre Lösung zu finden und sich in der verbleibenden Zeit bis zur Abfrage des Ergebnisses der optimalen Lösung weiter zu nähern.

Belief Networks:
Die Variablen der KNF Darstellung können als Knoten eines Belief Network gesehen werden – entsprechende Lösungsverfahren für Belief Networks sind also auf MAXPLAN Probleme anwendbar.

Beobachtbarkeit berücksichtigen:
MAXPLAN geht von unbeobachtbaren Problemwelten aus – es sollte ggf. auch möglich sein, den Plan der (*beobachtbaren*) Entwicklung der Dinge anzupassen, statt eine feste Folge von Schritten im Voraus zu planen.

Pläne mit Schleifen berücksichtigen:
Auch Pläne mit Schleifen, die wiederholte Aktionen kompakt darstellen, sind eine denkbare Erweiterung.

5 Zusammenfassung

MAXPLAN ermöglicht effizientes Planen in nicht vollständig deterministischen Welten und verbindet damit Vorteile von klassischen Planungstechniken mit denen von operations research Planern.

Dies wird erreicht, indem das Planproblem in ein Erfüllbarkeitsproblem umgewandelt wird und der Plan mit den maximalen Erfolgsaussichten (der die maximale Erfüllungswahrscheinlichkeit bietet) gesucht wird. Eine Reihe von Heuristiken beschleunigen die Berechnung typischer Probleme wesentlich, während im schlechtesten Fall natürlich weiterhin exponentielle Zeit benötigt wird.

Die Untersuchungen von Majercik und Littman zeigen, dass MAXPLAN sich in der vorliegenden Version mit etablierten Planern messen kann, aber noch immer eine Reihe von zukünftigen Verbesserungen eine weitere Steigerung der Effizienz versprechen.

6 Literatur

MAXPLAN: A New Approach to Probabilistic Planning (1998) Stephen M. Majercik and Michael L. Littman, Dept. of Computer Science, Duke University, Durham, NC 27708-0129. ©American Accociation for Artificial Intelligence (www.aaai.org).