OTTO FRIEDRICH UNIVERSITÄT BAMBERG

Learning from Planning Experience

Michael Reuß

02.07.2012

Seminararbeit

Kognitive Systeme

Matrikelnummer: 1694263

Dozent: Prof. Dr. Ute Schmid

Learning from Planning Experience

1 Einleitung

Das Lernen aus Planungserfahrung erweist sich in der Praxis als äußerst schwierig. Das Fachgebiet der automatisierten Planung (AP) aus dem Bereich der künstlichen Intelligenz beschäftigt sich mit dieser Problemstellung. In der letzten Dekade wurden AP Systeme erfolgreich auf reale Anwendungsfälle wie z.B. für die Kontrolle von autonomen Unterwasserfahrzeugen angewandt. Trotz derlei Beispiele ist die Anwendung auf reale Probleme immer noch kompliziert, vor allem im Hinblick auf zwei Definitionsprobleme:

- Genaue Beschreibung der Planungsaufgabe: Die Beschreibung einer Aufgabe muss die möglichen Aktionen, den Status der Umgebung und die zu erreichenden Ziele genau repräsentieren. Dies ist jedoch nicht immer möglich, da in der realen Welt nicht alle Auswirkungen einer Aktion auf die Umgebung erfasst werden kann oder bestimmte Teile nicht komplett beschrieben wurden bzw. beschrieben werden können.
- Die Vielfalt an Möglichkeiten ist zu groß: Im Allgemeinen ist die Suche nach einem Lösungsplan ein PSpace-vollständiges Problem. D.H. die Suche ist nicht in Polynomialzeit machbar. Moderne Planer versuchen deshalb durch verschiedene Methoden, z.B. Heuristiken, den Suchraum so zu verkleinern, dass sie eine Lösung in überschaubaren Zeiträumen finden können.

2 Grundlagen

Grundlegende Gedanken zum Automatischen Planen und Maschinellen Lernen (ML) sollen hier vorgestellt werden.

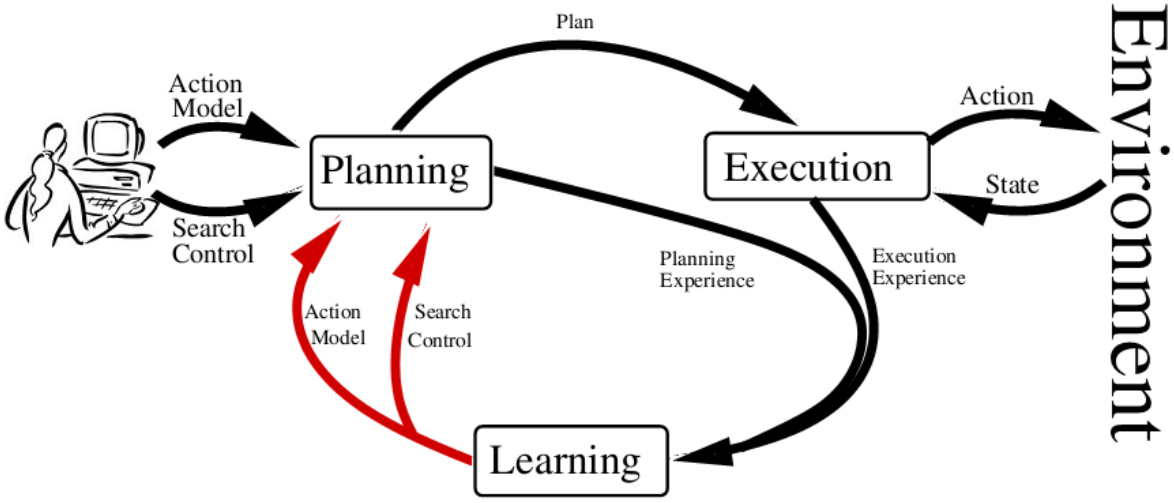

Abbildung 1 Lernen im Zusammenspiel mit Planen und Ausführen

2.1 Automatic planning

Automatisches Planen versucht Ansätze zu finden die verschiedene Problemklassen lösen können. Ein Arbeitspaket im AP ist hierbei aus zwei Elementen definiert:

- Die Domäne: Eine Beschreibung der Zustände der Umgebung und einer Menge der durchführbaren Aktionen um zwischen diesen Zuständen zu wechseln.
- Das Problem: Eine Menge an Zustandsvariablen die einen Startzustand und weitere Menge die den Zielzustand repräsentieren.

Die Lösung eines solchen Arbeitspakets ist dabei eine Abfolge an Aktionen, die einer entsprechenden Menge an Zustandsänderungen entsprechen um vom Start- in den Zielzustand zu gelangen.

2.2 Machine Learning

Maschinelles Lernen bezeichnet die künstliche Generierung von Wissen. Dabei wird versucht nicht einfach Lösungen für Beispiele auswendig zu lernen sondern Gesetzmäßigkeiten aus diesen Beispielen zu erkennen sodass diese auch für andere Probleme, welche nicht Teil der Beispiele waren, angewendet werden können. Hierbei spielen Faktoren wie die Wissensrepräsentation, das extrahieren von Wissen, der Lernalgorithmus sowie die Ausnutzung des gelernten wichtige Rollen.

2.3 Reinforcement Learning

Das bestärkende Lernen ist streng genommen ein Teil des maschinellen Lernens und beschreibt Methoden, die durch Wertungsmechanismen einer bestimmten Sequenz oder Handlungsfolge einen Nutzen zuschreiben. Es lässt sich grob in zwei Arten, dem Modelbasierenden und Modelfreien Lernen unterteilen.

- Modellbasierend: RL benötigt hierbei eine Beschreibung der Umgebung in Form von Zustandsübergängen und einem Belohnungssystem. Hierbei werden Standardalgorithmen zur Findung von Heuristik-Funktonen, welche eine optimale Lösung finden sollen benutzt.
- Modellfrei: Modellfrei wird keine explizite Beschreibung gebraucht. Es kann hier darauf zurückgegriffen werden mit entsprechenden Methoden ein Zustandsübergangsmodel zu lernen und auf diesem weiterzuarbeiten oder ganz darauf zu verzichten. In diesem Fall werden entsprechende Belohnungswerte nicht auf Zustände sondern auf einzelne Aktionen angewandt.

3 Probleme

Die in der Einleitung umrissenen Definitionsprobleme bilden sie Grundprobleme des automatischen Planens. Die beiden Probleme werden fachlich in Learning Planning Action Models und Learning Planning Search Control Knowledge unterschieden.

Learning Planning Action Models

Die Notwendigkeit eine komplette und korrekte Beschreibung der Aktionen in einer Umgebung ist des Größte Problem des Automatischen Lernens.

Learning Planning Search Control

Mit einhergehender Größe des Models stellt sich immer mehr die Frage wie in der Vielfalt an möglichen Aktionen und Zuständen eine Lösung effizient zu suchen ist.

4 Lösungsansätze

Für die Probleme lassen sich diverse Lösungsansätze aus unterschiedlichen Richtungen bilden. Für jedes der beiden Probleme sollen hier vier Ansätze kurz dargestellt werden.

4.1 Learning Planning Action Models

Das Erstellen eines Action Models von Grund auf ist eine komplexe und Zeit intensive Aufgabe. Eine Möglichkeit dieser Aufgabe gegenüberzutreten ist es, dies durch Techniken des maschinellen Lernens erledigen zu lassen. Nachfolgend werden einige dieser Techniken kurz umrissen. Eine Klassifizierung findet hierbei anhand der stochastischen Einwirkung der Aktionen und der Überwachbarkeit der Umgebung statt.

4.1.1 Deterministische Effekte in einer voll überwachbaren Umgebung

Im einfachsten Fall kann sind die ausführbaren Aktionen deterministisch und die komplette Umgebung bekannt. Dies kann beispielsweise in einer abgegrenzten Blocksworld Umgebung der Fall sein. Das Lernen findet hierbei durch Beobachtung der Änderungen von Zustandsvariablen statt.

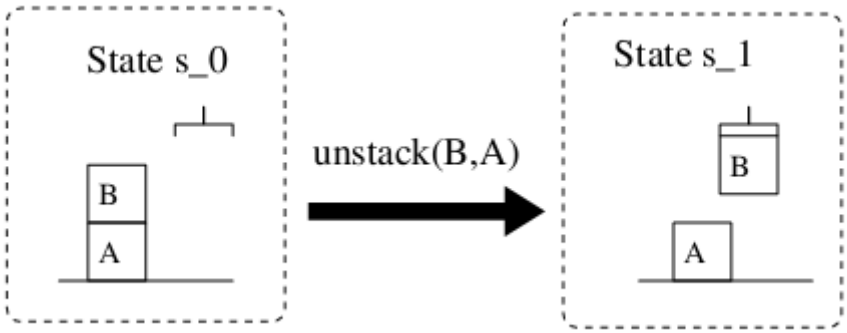

Abbildung 2 Überwachbarer Effekt von unstack(B,A)

4.1.2 Deterministische Effekte in einer zum Teil überwachbaren Umgebung

Hierbei sind die Auswirkungen von Aktionen auch deterministisch, jedoch ist der Status der Umgebung nicht vollständig beobachtbar, weshalb man die möglichen Folgezustände nur vermuten kann. Gelernt wird mittels Inference oder Induction.

- Inference: Ähnlich dem Erfüllbarkeitsproblem der Aussagenlogik (SAT, vom englischen satisfiability) das die Frage stellt ob eine aussagenlogische Formel erfüllbar ist. Das Action Model kann als die Erfüllbarkeit einer logischen Formel angesehen werden. Z.B. als Disjunktion alle möglichen Konjunktionen der vorhergehenden und nachfolgenden Zustande der betrachteten Aktion

```
(or ;;; pre-state 1 post-state 1
    ;;; pre-state 1 post-state 2
    (and (Unstack B A)
         (DelClear B) (DelOn B A) (DelOntable A)
         (AddClear A) (AddOn A B) (AddOntable B))
    ;;; pre-state 1 post-state 3
    (and (Unstack B A)
         (DelEmptyhand) (DelClear B) (DelOn B A)
         (AddHolding B) (AddClear A))

     ...

    ;;; pre-state 5 post-state 5
    (and (Unstack B A))
)
```

Abbildung 3 Beispielformel für die Modellierung der Aktion unstack einer Blocksworld Umgebung mit zwei Blöcken

- Induction: Man kann den gesamten Suchraum an Action Models durchsuchen und das am besten passende für die beobachteten Beispiele auswählen. Dies ist ähnlich dem Inductive Logic Progamming (ILP). Es werden für jede Aktion der Domäne zwei logische Programme eingeführt. Jeweils eins das alle Voraussetzungen und eins das die Effekte der betrachteten Aktion beinhaltet.

```
preconditions_unstack(TOP,BOTTOM):- object(TOP),object(BOTTOM),
                                    emptyhand(), clear(TOP), on(TOP,BOTTOM).

effects_unstack(TOP,BOTTOM):- object(TOP),object(BOTTOM),
                              del_emptyhand(),del_clear(TOP), del_on(TOP,BOTTOM),
                              add_holding(TOP),add_clear(BOTTOM).
```

Abbildung 4 Repräsentation der Aktion unstack einer Blocksworld Umgebung als zwei logische Programme

4.1.3 Probabilistische Effekte in einer voll überwachbaren Umgebung

Die Auswirkungen der Aktionen sind nicht deterministisch sondern weisen stochastische Effekte auf. Die Umgebung dagegen ist vollständig beobachtbar. Gelernt wird hierbei mittels Beobachtung der Zustandsvariablen, jedoch ist das Lernen von Aktionen mit stochastischen Effekten trotzdem kompliziert, da die Zustandsänderungen überlappen und die Auswirkungen von Aktionen somit nicht klar abgegrenzt werden können. Einige Lerntechniken, wie z.B. Stochastic Logic Programms (Muggleton, 1995; Cussens, 2001) und Markov Logic Networks (Richardson and Domingos, 2006), weisen zwei grundsätzliche Schritte auf:

- Structural Learning: In diesem Schritt führen ILP Algorithmen First-Order-Logic Formeln ein, welche Beziehungen der grundsätzlichen Fakten der Lernbeispiele generalisieren.
- Parameter Estimation: Dieser Schritt schätzt die mit der eingeführten Formel verbundenen Wahrscheinlichkeiten. Schritt eins gibt oftmals mehrere Regeln der Form C→P aus, wobei P die prognostizierte Regel und C Bedingungen sind unter welchen P gilt. In diesem Fall wird die Wahrscheinlichkeit prob(P|C) ausgerechnet. Wenn die eingeführte Regel disjunktiv ist wird diese Wahrscheinlichkeit direkt vom Lernbeispiel übernommen. Wenn sie jedoch überlappen werden weitere Berechnungen benötigt.

- Bayesian Estimation: Dem Bayes Theorem folgend:

$$prob(P|C) = prob(C|P)\frac{prob(P)}{prob(C)}$$

- Maximum Likelihood Estimation: Es wird davon ausgegangen dass das Lernbeispiel einer gegebenen Wahrscheinlichkeitsverteilung folgt und dann iterativ die Parameter dieser angepasst werden um das passende zu finden.

4.1.4 Probabilistische Effekte in einer zum Teil überwachbaren Umgebung

In diesem Szenario ist weder die Umgebung vollständig beobachtbar noch sind die Auswirkungen der Aktionen deterministisch. Hierfür gibt es bislang noch wenig Studien und momentan auch keine generelle Strategie dieses Problem anzugehen. Vorläufige Arbeiten (QUELLE) versuchen mittels Weighted Maximum Satisfiability dem Problem zu begegnen.

4.2 Learning Planning Search Control Knowledge

Die Performance eines Planers kann signifikant verbessert werden wenn man Domänen spezifisches Wissen ausnutzt was nicht explizit mit der Domäne modelliert wurde. Dieses Kapitel beschreibt ML Methoden welche die Geschwindigkeit oder die Qualität des Planungsprozesses verbessern indem sie dieses Wissen bereitstellen.

4.2.1 Macro Actions

Unter dem bilden von Makroaktionen versteht man das zusammenfassen von Aktionen um den resultierenden Suchraum zu verkleinern. Bei zu vielen neuen Aktionen besteht jedoch das Problem das der Suchraum zwar nicht tiefer jedoch breiter wird und dadurch der eigentliche Nutzen geringer wird.

4.2.2 Generalized Policies

Generalisierte Richtlinien besagen grob welche Aktionen in welchen Zuständen sinnvoll anzuwenden sind. Mittels einer akkuraten generellen Richtlinie ist es möglich jedes Problem einer bestimmten Domäne ohne jegliche Suche zu lösen, einfach indem wiederholt immer wieder die passende Aktion auf den entsprechenden Zustand ausgeführt wird. Im Falle der Blocksworld Domäne wäre ein Beispiel für eine solche Richtline erst alle Blöcke

4.2.3 Generalized Heuristics

Heuristiken werden in AP dazu benutzt die Suche nach einem Plan zu fokussieren. Allgemein kann eine Heuristik als eine Schätzung des Abstandes eines bestimmten Zustandes zu einem anderen in welchen Ziele erfüllt werden bezeichnet werden. Beispielsweise ist eine einfache Heuristik im Problem der Wegfindung die Euklidische Distanz zwischen zwei Punkten. Eine Heuristik ist generalisiert wenn sie für alle Zustände und in verschiedenen Domänen anwendbar ist. Die verbreitetste Form einer solchen Heuristik besteht aus der Relaxation der AP Aufgabe in dem Sinn, das negative Effekte einer Aktion vernachlässigt und nur positive gewertet werden. Diese Methode ist Domänen unabhängig, kann jedoch nicht bestimmte Besonderheiten in der jeweiligen Domäne abdecken.

4.2.4 Decomposition Models

Eine weitere Strategie der Problemkomplexität zu begegnen ist es das jeweilige Problem in kleinere Sub-Probleme zu zerlegen. Wenn eine effektive Zerlegung gefunden ist, ist die Lösung der kleineren Probleme einfacher als das komplette Problem an sich zu lösen. Hierarchical Task Networks (HTN) (Hogg, C. et al. 2009) ist eines der am besten untersuchten Herangehensweisen für eine solche Zerlegung. Ein HTN Planer besteht hierbei aus einem Action Model, ähnlich dem STRIPS im klassischen Planen, und einem sog. Task Model. Das Task Model beschreibt eine Menge an Funktionen welche darlegen wie sich eine Aufgabe in kleinere zerlegen lässt. Die Aufgabe von HTN besteht nun darin das ursprüngliche Problem soweit in kleinere Sub-Probleme zu zerlegen, bis es nur noch aus einer Abfolge einfacher Aktionen besteht.

5 International Planning Competition

5.1 Wettbewerb

Der Wettbewerb wird von ICAPS zweimal jährlich veranstaltet und zielt weniger darauf ab den besten Planer zu ermitteln, als vielmehr möglichst viele Daten zu sammeln, diese zu präsentieren und die Daten sowie die Interpretation der Ergebnisse der forschenden Gemeinschaft zur Verfügung zu stellen.

Regeln und Ablauf

Der Wettbewerb teilt sich in zwei Phasen, der Lern- und der Testphase, welche nacheinander stattfinden auf. Die zwei wöchige Lernphase beginnt nachdem alle Teilnehmer ihren Code den Organisatoren zugesandt haben. Ab diesem Zeitpunkt darf der Code nichtmehr geändert werden (Code Freeze). Die Teilnehmer erhalten dann für jede Domäne:

a. Eine Domänendefinition
b. Einen Problemgenerator
c. Einige Testprobleme. Ziel hierbei ist es, das der Planer soviel Domänenspezifisches Wissen lernt sodass er diese Probleme gut lösen kann. Die Zielproblemeder Testphase werden dabei nicht veröffentlicht.

Nach Ablauf der Lernphase werden in der Testphase die einzelnen teilnehmenden Planer evaluiert. Hierzu lassen die Organisatoren die Planer auf ihren lokalen Maschinen für jede Domäne einmal mit, sowie ohne Domänenspezifisches Wissen die Zielprobleme bearbeiten. Die evaluierten Werte jedes Planers mit bzw. ohne des erlerntem Wissens lassen dabei Rückschlüsse auf den Einfluss des gelernten zu.

Wertung

Die Auswertung des Wettbewerbs erfolgt aus den Ergebnissen in drei Messwerten während des Testphase:

Planungszeit

- **T*** sei die minimal benötigte Zeit die ein Planer für die Problemlösung brauchte. (Sollte kein Planer zu einer Lösung kommen wird diese Bewertung ignoriert.)
- Ein Planer der ein Problem in einer Zeit **T** löste erhält nun die Wertung $\mathbf{1/(1 + \log_{10}(T/T^*))}$.
- Der Planer der ein Problem am schnellsten löste, also in **T***, erhält somit die Wertung 1.
- Planer die ein Problem nicht lösen konnten erhalten die Wertung 0.
- Alle Zeiten unterhalb 1 Sekunde erhalten dieselbe Wertung.
- ➔ Die Gesamtwertung ist die Summe aus den in allen Problemstellungen erreichten Wertungen.

Planungsqualität

- **N*** sei die minimal benötigte Anzahl an Aktionen die ein planer benötigte um ein Problem zu lösen. (Sollte kein Planer zu einer Lösung kommen wird diese Bewertung ignoriert.)
- Ein Planer der ein Problem mit einer Anzahl Aktionen **N** löste erhält nun die Wertung **N*/N**
- Der Planer der ein Problem am besten löste, also in **N***, erhält somit die Wertung 1.
- Planer die ein Problem nicht lösen konnten erhalten die Wertung 0.
- ➔ Die Gesamtwertung ist die Summe aus den in allen Problemstellungen erreichten Wertungen.

Lerneffekt

Die Organisatoren vergeben einen extra Preis an den Planer der den Lerneffekt erzielen konnte. Hierbei wird darauf geachtet das das dem Wettbewerb zugrundliegende Ziel, einen guten Planer zu erstellen, nicht verletzt wird. Organizers will reward the planner that demonstrates the best performance improvement through learning. Please note that the spirit of the competition is to make the best planners possible. Die Teilnehmer sollen demnach ihre ungelernten Planer nicht absichtlich schlecht machen. Sollte dies bei einem Teilnehmer vermutet werden, so ist dieser nicht mehr berechtigt den Preis zu erhalten. Um die Entscheidung zu stützen wird ein *Pareto Ranking* auf Grundlage der Wertung des ungelernten Planers und der Delta-Wertung mit erlerntem Wissen verwendet.

Zusätzlich wird ein extra Preis an jedem Planer verliehen der **(1.)** Den Gewinner des Sequentiellen Teilwettbewerbs mit erlerntem Wissen schlägt und **(2.)** ohne dieses Wissen gegen ihn verliert.

5.2 Planner

An der IPC 11 nahmen insgesamt acht unterschiedliche Planer teil, sieben werden hier nur kurz aufgeführt und dann der Bootstrap-Planner aufgrund seines Ansatzes Heuristiken zu lernen genauer erläutert. Die Teilnehmenden Planer sind:

- Optimize-and-Learn: A Parameter Tuning Framework applied for the Divide-and-Evolve Method used to AI Planning
- FD-Autotune: Domain-Specific Configuration using Fast Downward (quality)
- FD-Autotune: Domain-Specific Configuration using Fast Downward (speed)
- ParLPG: Generating Domain-Specific Planners through Automatic Parameter Configuration in LPG
- PbP2: Automatic Configuration of a Portfolio-based Multi-Planner (quality)
- PbP2: Automatic Configuration of a Portfolio-based Multi-Planner (speed)
- CBL- Case Based Planning for Learning
- Bootstrap-Planner

Der Bootstrap-Planner ist ein iterativer Planer der versucht seine Heuristiken durch lernen zu verbessern. Ausgehend von einer initialen Heuristik, versucht der Bootstrap Planner durch sukzessives Lösen von mehreren, umfangreicher werdenden Problemen diese Heuristik zu verbessern.

Der Lernalgorithmus des Bootstrap-Planners könnte folgendermaßen vereinfacht beschrieben werden:

Iterate:

For each E of T do: Solve(E)

If CountSolved(E)>50:

For each Solved(E) do:

Backpropagation

T: Menge Trainingseinheiten

E: Trainingseinheit

Solve: Planungsaufgabe lösen

CountSolved(E): Anzahl gelöste Trainingseinheiten

Solved(E): Gelöste Trainingseinheit

Der Planer startet mit einer Menge an Trainingseinheiten e, einer initialen Heuristik H und einem Zeitlimit t. Nur wenn innerhalb des Zeitlimits t genug Trainingseinheiten erfolgreich absolviert wurden, lernt der Planer eine neue Heuristik Hi und geht in die nächste Iteration i mit einer größeren Menge an Trainingseinheiten über. Sollte der Planner nicht genügend Trainingsheinheiten im gegeben Zeitlimit gelöst haben so wird das Zeitlimit erhöht.

Die neue Heuristik wird dabei wie folgt gelernt:

Für jeden Status s einer gelösten Trainingseinheit werden die Zustandsvariablen und der Abstand zum Zielzustand gesammelt und an den Lernalgorithmus übergeben.

Der Lernalgorithmus ist ein Backpropagation Algorithmus aus dem Bereich der künstlichen neuronalen Netze. Hier in einer Ausführung mit drei versteckten und einem Ausgabeneuron(dargestellt in Abbildung 5). Der Algorithmus wird beendet nach 500 Epochen oder wenn der mittlere quadratische Fehler MSE <0,005 wird. (MSE vom englischen mean squared error)

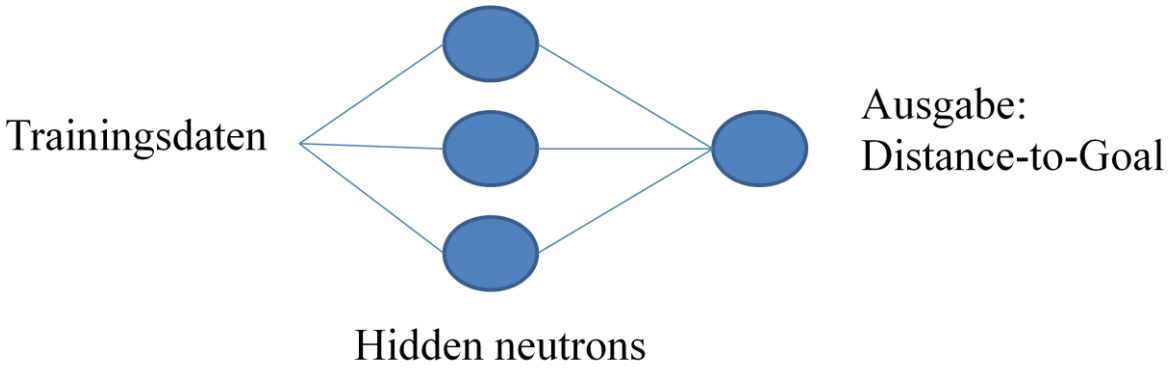

Abbildung 5 Vereinfachte Darstellung des Backpropagation Netz des Bootstrap Planners

Weiterhin ist der Bootstrap Planner auf dem Fast Downward Planer (Helmert, M. 2006) aufgebaut. Der hierbei benutzte Suchalgorithmus ist greedy best first search mit einer alternierenden Liste und jeweils einer Liste für die gelernte und der präferierten Heuristik. Diese Entscheidungen wurden getroffen sodass der Planer auch für große Probleme skalieren kann. Die finale gelernte Heuristik wird für die Lösung der Testprobleme genommen.

6 Abschluss

Automatisches Planen findet immer mehr Verwendung im Alltag. Beispiele hierfür wären die anfangs erwähnten autonomen Unterwasser Roboter, jedoch auch andere Beispiele wie Mustererkennung oder der Einsatz von AP Techniken in Spielen.

Das erlernte Wissen, wie das hier kurz dargestellte des Bootstrap Planners, ist für Menschen oftmals schwer verständlich und nicht auf einfache Weise darzustellen. Im Hinblick auf die International Planning Competition zeigt die Zweitplatzierung in den Disziplinen Quality- und Time-learning dass der Bootstrap Planer mittels seines Ansatzes er in der Lage ist, selbstständig zu lernen. Wenngleich er gegen nicht lernende (sondern gut parametrisierte) Planer chancenlos war.

7 Quellen

Diese Seminararbeit basiert hauptsächlich auf:

Sergio Jimnénez, Tomás de la Rosa, Susana Fernández, Fernando Fernández, Daniel Borrajo (2009)
A Review of Machine Learning for Automated Planning
http://scalab.uc3m.es/~sjimenez/publications/sergio-ker11/sergio-ker11.pdf (Stand 02.07.2012)

Weitere:

Cussens, J. (2001). Parameter estimation in stochastic logic programs, Machine Learning, 44(3):245-271.

Helmert, M. (2006), The Fast Downward planning system Journal of Artificial Intelligence Research 26:191–246

Hogg, C., Kuter, U., Muñoz-Avila, H. (2009), Learning hierarchical task networks for nondeterministic planning domains, International Joint Conference on Artificial Intelligence IJCAI-09

International Conference on Automated Planning and Scheduling (ICAPS)
http://www.icaps-conference.org/ (Stand 02.07.2012)

Muggleton, S. (1995), Stochastic logic programs, International Workshop on Inductive Logic Programming

Richardson, M. and Domingos, P. (2006), Markov logic networks, Machine Learning, 62:107–136

Shahab Jabbari Arfaee, Robert C. Holte, Sandra Zilles
Bootstrap Planner: An Iterative Approach to Learn Heuristic Functions for Planning Problems
http://www.plg.inf.uc3m.es/ipc2011-learning/Planners?action=AttachFile&do=get&target=p06.pdf
(Stand 02.07.2012)

The Seventh International Planning Competition, 2011, Learning Track
http://www.plg.inf.uc3m.es/ipc2011-learning (Stand 02.07.2012)