Otto-Friedrich-Universität Bamberg

Professur für Angewandte Informatik,
insbesondere Kognitive Systeme

Kognitive Systeme

Seminararbeit

Learning from Planning Experience – Learning Track of International Planning Competition

Autor: Claudia Fischer

Dozent: Prof. Dr. Ute Schmid

Datum: 17.07.2012

Inhaltsverzeichnis

Inhaltsverzeichnis I

1. Einleitung 1

2. Grundlagen 2
   - 2.1. Automatisches Planen 2
   - 2.2. Maschinelles Lernen 2
   - 2.3. Verstärkendes Lernen 3

3. Ansätze zur Verbesserung des AP 5
   - 3.1. Planungssuchkontrollwissen 5
   - 3.2. Makro-Aktionen 7

4. Wettbewerb 10
   - 4.1. Domänen 10
   - 4.2. Probleme 12
   - 4.3. Regeln und Ablauf 12
   - 4.4. Wertung 13

5. PbP2.quality 14
   - 5.1. Allgemeines 14
   - 5.2. Architektur 14
   - 5.3. Auswahl der Planer und Makro-Aktionen 16
   - 5.4. Praktische Anwendung 17
     - 5.4.1. Allgemeines 17
     - 5.4.2. Ausführen des Lernprogramms 18
     - 5.4.3. Ausführen des Planprogramms 20

6. Fazit 22
   - 6.1. Zusammenfassung 22
   - 6.2. Ausblick 22
   - 6.3. Praktische Anwendung von PbP2.quality 23

Anhang i
- Abbildungsverzeichnis i
- Listings i
- Tabellenverzeichnis i
- Abkürzungsverzeichnis ii
- Quellenverzeichnis ii

1. Einleitung

Die künstliche Intelligenz (kurz KI) ist ein Teilgebiet der Informatik und befasst sich mit der Entwicklung von „intelligenten“ Computern. Der Preisträger des letztjährigen Turing Awards (der dem Nobel-Preis in der Informatik entspricht) leistete wichtige Beiträge im Bereich der KI, was zeigt, dass die KI aktuell ein sehr wichtiges Thema darstellt.

Automatisches Planen (engl. *Automated Planning*, kurz AP) ist ein Teilgebiet der KI und beschäftigt sich mit dem automatischen Erstellen von Aktionsfolgen, die vorgegebene Problemstellungen lösen.

Angefangen mit Beweisen von Theoremen und den ersten Versuchen mit Robotern in den fünfziger Jahren, wurden AP-Systeme im letzten Jahrzehnt auch bei realen Problemen (z. B. bei der Steuerung von Unterwasserfahrzeugen) eingesetzt (nach [Jim09]).

Jedoch hat die AP nach wie vor mit zwei Problemen zu kämpfen (nach [Jim09]):

- Automatische Planer benötigen eine präzise Beschreibung der Planungsaufgabe (beinhaltet Aktionsmodell, Spezifikation des Status der Umgebung und der Ziele, die zu erreichen sind). Dies ist für reale Probleme fast nicht machbar.
- Standard-Planer scheitern sehr häufig am Hervorbringen von Lösungen guter Qualität, da AP-Probleme meist eine PSPACE-vollständige Komplexität besitzen und Standard-Planer sich auf solche Komplexitätsdimensionen nicht erweitern können.

Maschinelles Lernen (engl. *Machine Learning*, kurz ML) ist ein gutes Werkzeug, um das AP hinsichtlich dieser beiden Probleme zu verbessern.[1] Seit 2005 gibt es zudem regelmäßig internationale Wettbewerbe, auf denen Planungssysteme, die dieses Konzept verwenden, gegeneinander antreten (nach [Jim09]).

[1] Näheres zur Verbesserung von AP durch ML findet sich in „*Learning-assisted automated planning: looking back, taking stock, going forward*“ von Zimmermann und Kambhampati (2003)

2. Grundlagen

In den folgenden Kapiteln werden die hier thematisierten Begriffe AP, ML sowie das verstärkende Lernen, einem weiteren Ansatz des ML, erläutert. Dabei wird sich stark an [Jim09] orientiert.

2.1. Automatisches Planen

Unter Planen versteht man, eine Folge von Aktionen zu finden, die ein gegebenes Problem bzw. eine Aufgabenstellung löst. Beim automatischen Planen geschieht dies automatisiert durch Programme.

Eine AP-Aufgabenstellung besteht aus einer Domäne und einem Problem. Die Domäne wird durch eine Reihe von Zuständen der Umgebung und eine Reihe von Aktionen, die Übergänge zwischen Zuständen darstellen, repräsentiert. Das Problem wird durch eine Reihe von Fakten, die den Startzustand angeben, sowie eine Reihe von Fakten, die die Ziele der AP-Aufgabe angeben, definiert. Die Lösung dieser Aufgabe ist eine Folge von Aktionen bzw. eine Folge von Zustandsübergängen. Je besser die Aktionsmodelle definiert sind (durch Experten), desto besser sind auch die AP-Systeme.

Es kann vorkommen, dass der Zustandsübergangsgraph, der alle möglichen Zustandsübergange grafisch dargestellt, so groß wird, dass es schwierig wird, einen Lösungspfad zu finden. Die Komplexität der Problemlösungsalgorithmen steigt exponentiell in der Anzahl der Problemvariablen (Objekte und Prädikate), was Pläne mit einer höheren Anzahl von Aktionen nicht zulässt. Seit der Einführung des Planungsgraphen können starke domänenunabhängige Heuristiken entwickelt werden, die in polynomischer Zeit berechnet werden können. Beispielsweise können in einigen Sekunden 100 Aktionspläne erzeugt werden.

Eine sehr große Anzahl von Objekten und Aktionsparameter stellen bei Planern immer noch ein großes Problem dar (was Rechenzeit und die Lesbarkeit des dabei entstehenden Suchbaumes angeht).

2.2. Maschinelles Lernen

Maschinelles Lernen ist der Wissenserwerb eines Systems bzw. eines Programmes mittels Erfahrung. ML beinhaltet vier Kernpunkte, um einen automatischen Prozess zu verbessern. Diese werden im Folgenden aufgezählt und erläutert.

Wissensrepräsentation

Zum einen muss die Art des Wissens, welches erlernt wird, definiert und zum anderen die Darstellung des erlernten Wissens festgelegt werden. Dazu gehören die Repräsentationssprache sowie der Merkmalsraum:

- Als Repräsentationssprache dient vorwiegend die Prädikatenlogik. Die Beschreibungslogik und die temporale Logik werden seltener verwendet.
- Im AP enthält der Merkmalsraum die Domänenprädikate, mit denen die Aktionen, Zustände und Ziele definiert werden.

Erfahrungsextraktion

Bei der Erfahrungsextraktion geht es um die Frage, wie Lernbeispiele erfasst werden, selbstständig vom Planungssystem oder von einem externen Agenten, z. B. einem menschlichen Experten.

Lernalgorithmus

Der Lernalgorithmus erkennt Muster in den extrahierten Erfahrungen. Dabei wird zwischen drei Arten von Lernen unterschieden:

- Beim induktiven Lernen werden bekannte Beispiele verallgemeinert und daraus Muster erstellt.
- Beim analytischen Lernen werden die Lernbeispiele durch bereits vorhandenes Wissen sowie eine deduktive Beweisführung (vom Allgemeinen auf das Spezielle schließen) erklärt.
- Hybrid induktiv-analytisches Lernen verwendet beide Lerntypen.

Nutzung des erlernten Wissens

Das erlernte Wissen ist von den drei oberen Punkten (Wissensrepräsentation, Erfahrungsextraktion und Lernalgorithmus) abhängig.

Ist das Wissen unvollständig oder sogar fehlerhaft, kann eine direkte Nutzung den Planungsprozess negativ beeinträchtigen.

2.3. Verstärkendes Lernen

Verstärkendes Lernen (engl. *Reinforcement Learning*, kurz RL) wird hauptsächlich dann eingesetzt, wenn das Umgebungsmodell unbekannt ist.

Das System lernt durch Belohnungen bzw. Verstärkungen, was in bestimmten Situationen eine gute bzw. eine schlechte Aktion ist und kann sich so für ein Problem eine optimale Lösungsstrategie aufbauen.

Ein wichtiger Aspekt beim RL ist das sogenannte „*exploration-exploitation dilemma*“ (engl. für Erkundungs-Verwertungs-Dilemma), bei dem es darum geht, abzuwägen, wann neue und wann bereits erfolgreich verwendete Aktionen ausgeführt werden sollen.

Beim RL kann zwischen modellfreiem und modellbasiertem RL unterschieden werden. Modelbasiert bedeutet hier, dass ein Umgebungsmodell (Übergangs- und Belohnungsmodell) benötigt wird. Zur Berechnung von heuristischen Funktionen wird die dynamische Programmierung verwendet. RL wird auch als heuristische Echtzeitsuche bezeichnet.

Die modellfreie RL ist dagegen nicht auf ein Umgebungsmodell angewiesen. Die folgenden zwei Ansätze können verwendet werden:

- Das Lernen von Aktionsmodellen und das modellbasierte RL werden kombiniert. Das System erlernt ein Übergangsmodell und wendet die dynamische Programmierung an.
- Das reine modellunabhängige RL wird bei Domänen mit hoher Unsicherheit (engl. *Uncertainty*) angewandt, da es in solchen Domänen schwieriger ist, ein Umgebungsmodell zu erlernen als eine Strategie für das Erreichen von Zielen.

3. Ansätze zur Verbesserung des AP

Wie bereits in Abschnitt 1 erläutert, gibt es beim AP zwei große Schwierigkeiten. Zum einen wird eine präzise Beschreibung der Planungsaufgabe benötigt, was für reale, komplexe Probleme sehr aufwändig ist. Weiterhin soll die Leistungsfähigkeit von Planern gesteigert werden.

Abbildung 3.1 zeigt grafisch den Zusammenhang der Ansätze mit AP.

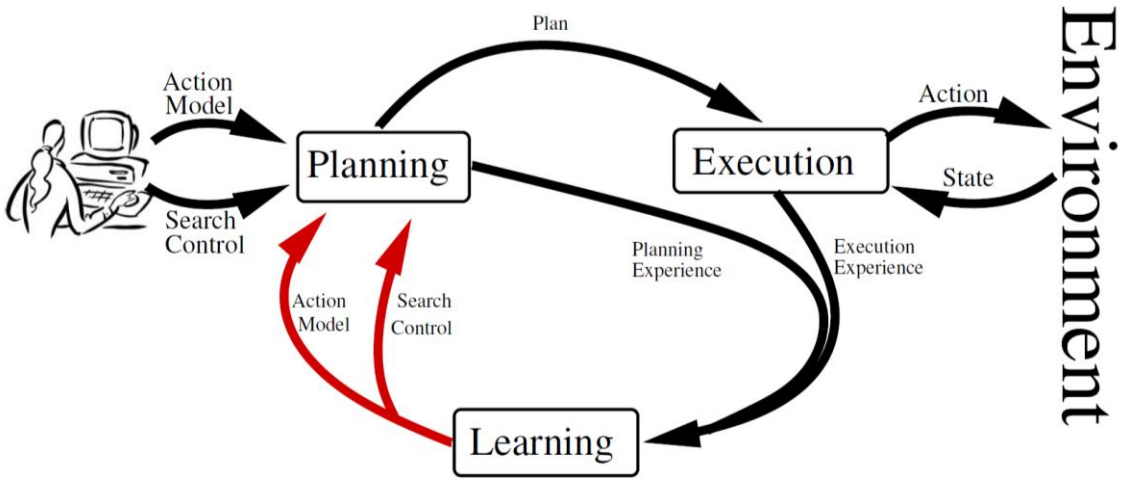

Abbildung 3.1: Verwendung von ML zur Planungsprozessverbesserung aus [Jim09]

In diesem Kapitel werden Konzepte vorgestellt, die die Effizienz von Planern verbessern. Dabei wird sich an [Jim09] orientiert. Anschließend wird die Technik „Makro-Aktionen“ detaillierter erläutert.

3.1. Planungssuchkontrollwissen

Planungssuchkontrollwissen ist das Wissen über eine AP-Aufgabe, das über das Domänenmodell hinaus geht. Die Nutzung solchen Wissens ermöglicht dem Planer eine effizientere Berechnung einer Lösung.

Kontrollwissen, das für eine spezielle Domäne erstellt wurde, wird domänenspezifisches Kontrollwissen (engl. *Domain-specific Control Knowledge*, kurz DCK) genannt.

Makro-Aktionen

Aktionen, die häufig hintereinander ausgeführt werden, werden zu einer Makro-Aktion (engl. *Macro-Action*) zusammengefügt. Eine Makro-Aktion kann nun als eigenständige Aktion ausgeführt werden. Dieses Thema wird in Kapitel 3.2 noch ausführlicher erläutert.

Allgemeine Strategien

Eine allgemeine Strategie enthält für einen Planungskontext (bestehend aus dem aktuellen und dem Zielzustand) die bevorzugten Aktionen, die auszuführen sind, um dem Ziel näher zu kommen.

Die in Listing 3.1 dargestellte allgemeine Strategie legt den Block TOP auf den Block BUTTON, falls es das Ziel ist, dass TOP auf BUTTON ist, BUTTON frei ist und falls BUTTON „*well placed*“ ist. Ein Block ist *well placed*, wenn er bezüglich dem Ziel auf dem richtigen Block bzw. auf dem Tisch liegt, und, falls andere Blöcke unter ihm liegen, diese ebenfalls *well placed* sind.

```
 1  stack(TOP,BOTTOM) :- state_holding(TOP), goal_on(TOP,BOTTOM),
 2                       state_clear(BOTTOM), wellplaced(BOTTOM).
 3  putdown(BLOCK) :- state_holding(BLOCK).
 4  unstack(TOP,BOTTOM) :- state_clear(TOP), state_armempty().
 5  pickup(BLOCK) :- state_clear(BLOCK), state_armempty().
 6
 7
 8  ;;; Representation of the wellplaced concept
 9  ingoal(Block) :- goal_ontable(Block).
10  ingoal(Top) :- goal_on(Top,Bottom).
11
12  wellplaced(Block) :- state_ontable(Block), goal_ontable(Block).
13  wellplaced(Block) :- state_on(Block,Bottom), not(ingoal(Bottom)).
14  wellplaced(Block) :-
    state_on(Block,Bottom),goal_on(Block,Bottom),wellplaced(Bottom)
```

Listing 3.1: Allgemeine Strategie für Blockworld aus [Jim09]

Allgemeine heuristische Funktionen

Eine heuristische Funktion dient dazu, die Suche einer Problemlösung in die richtige Richtung zu lenken. Sie schätzt, ausgehend vom aktuellen Knoten, die Kosten für den billigsten Pfad zum Zielknoten. Dabei werden die Kosten stets unterschätzt.

Ein Beispiel für eine Heuristik ist im folgenden Listing dargestellt.

```
1  goal_on(A,B).
2  on(A,B)?
3  + yes: [0]
4  +--no: on(C,A) AND on(D,B)?
5         +--yes: [6]
6         +--no: on(C,A) OR on(D,B)?
7                +--yes: [4]
8                +--no: [1]
```

Listing 3.2: Domänenspezifische Heuristik für Blocksworld aus [Jim09]

Dabei werden die Kosten abgeschätzt, die je nach Anordnung der Blöcke nötig sind, um das Ziel on(A,B) zu erreichen. Die Kosten entsprechen der dafür benötigten Anzahl von Zügen.

Hierarchische Zerlegungsmethoden

Eine Möglichkeit, die Komplexität eines Problems zu verringern, ist, das Problem mittels hierarischer Zerlegungsmethoden in kleinere Teilprobleme zu zerlegen. Die Summe der Kosten dieser Teilprobleme ist bei einer sinnvollen Zerlegung geringer als die Kosten des ursprünglichen Problems. Ein Ansatz zur Modellierung von solchen Zerlegungen ist das sogenannte *Hierarchical Task Network* (kurz HTN). Ein HTN-Planer enthält ein Aktions-

modell, so wie es bei STRIPS verwendet wird, sowie ein Aufgabenmodell (*engl. Task Model*). Das Aufgabenmodell beschreibt eine Menge von Methoden, in denen definiert wird, wie bestimmte Aufgaben in Teilaufgaben (engl. *Subtasks*) zerlegt werden sollen. Listing 3.3 enthält die Aktion `unstack` sowie die Methode für den Task `move-block`., der einen Block zu seiner endgültigen Position bewegt. Dies geschieht mittels der beiden Teilaufgaben `method-for-moving-x-from-y-to-z` sowie `method-for-moving-x-from-table-to-z`. In beiden Teilaufgaben wird überprüft (vgl. `check`, `check2` und `check3`), ob der Block direkt zu seiner endgültigen Position bewegt werden kann, und sichergestellt, dass er danach nicht mehr bewegt wird.

```
 1  (:operator (!unstack ?top ?bottom)
 2             ;preconds
 3             ((clear ?top) (on ?top ?bottom))
 4             ;effects
 5             ((holding ?top) (clear ?bottom)))
 6
 7  (:method (move-block ?x ?z) ;head
 8             method-for-moving-x-from-y-to-z
 9             ;preconds
10             (:first (on ?x ?y))
11             ; Decomposition
12             ((!unstack ?x ?y) (!stack ?x ?z)
13             (!assert ((dont-move ?x)))
14             (!remove ((stack-on-block ?x ?z)))
15             (check ?x) (check2 ?y) (check3 ?z))
16
17             method-for-moving-x-from-table-to-z
18             ;preconds
19             nil
20             ; Decomposition
21             ((!pickup ?x) (!stack ?x ?z)
22             (!assert ((dont-move ?x)))
23             (!remove ((stack-on-block ?x ?z)))
24             (check ?x)))
```

Listing 3.3: Hierarschische Zerlegungsmethode in Blocksworld aus [Jim09]

3.2. Makro-Aktionen

Wie bereits im vorherigen Abschnitt erwähnt, ist eine Makro-Aktion eine spezielle Aktion, die aus einer Folge von Aktionen, die häufig hintereinander ausgeführt werden, zusammengesetzt ist. Mittels Makro-Aktionen wird die Geschwindigkeit des Planungsprozesses erhöht sowie die Tiefe des Suchbaumes des Planungsalgorithmus reduziert, was darauf zurückzuführen ist, dass durch die Anwendung von Makro-Aktionen mehrere Standard-Aktionen ausgeführt werden und somit seltener eine Entscheidung getroffen werden muss, welche Aktion nun ausgeführt werden soll. Jedoch steigt bei einer hohen Anzahl von Makro-Aktionen die Anzahl der Verzweigungen im Suchbaum. da die Anzahl der möglichen Aktionen erhöht wird (Standard-Aktionen plus Makro-Aktionen).

Ein Beispiel für eine Makro-Aktion ist in Listing 3.4 illustriert. `unstack-putdown` ist aus den beiden Aktionen `unstack` und `putdown` zusammengesetzt.

```
;;; Primitive actions
(:action unstack
  :parameters (?x - block ?y - block)
  :precondition (and (on ?x ?y) (clear ?x) (handempty))
  :effect (and (holding ?x)
               (clear ?y)
               (not (clear ?x))
               (not (handempty))
               (not (on ?x ?y))))

(:action putdown
  :parameters (?x - block)
  :precondition (holding ?x)
  :effect (and (not (holding ?x))
               (clear ?x)
               (handempty)
               (ontable ?x)))

;;; Macro-action
(:action unstack-putdown
  :parameters (?x - block ?y - block)
  :precondition (and (on ?x ?y) (clear ?x) (handempty))
  :effect (and (clear ?y)
               (not (on ?x ?y))
               (ontable ?x)))
```

Listing 3.4: Makro-Aktion unstack-putdown der Blocksworld-Domäne aus [Jim09]

Wissensrepräsentation

Makro-Aktionen werden ebenso wie Standard-Aktionen mittels Prädikatenlogik dargestellt.

Aktionen werden wie folgt zu Marko-Aktionen zusammengefasst (*i* und *j* stellen dabei zwei Aktionen dar) (aus [Jim09]):

- Parameter: $par(a_{i,j}) \subseteq (par(a_i) \cup par(a_j))$
- Vorbedingungen: $pre(a_{i,j}) \subseteq \big(pre(a_i) \cup pre(a_j)\big) \setminus add(a_i)$
- Positive Effekte: $add(a_{i,j}) \subseteq \big(add(a_i) \cup add(a_j)\big) \setminus del(a_j)$
- Negative Effekte: $del(a_{i,j}) \subseteq \big(del(a_i) \cup del(a_j)\big) \setminus add(a_j)$

Die Parameter beider Aktionen werden zusammengefügt. Die Vorbedingungen werden ebenfalls zusammengefügt, jedoch ohne die Zustände, die positive Effekte von *j* sind. Die positiven Effekte der Makro-Aktion enthalten die postitiven Effekte von *i* und *j* abzüglich der negativen Effekte von *j*. Die neuen negativen Effekte ergeben sich aus der Vereinigungsmenge der negativen Effekte von *i* und *j* ohne die positiven Effekte von *j*.

Lernbeispiele

Als Lernbeispiele können alle Aktionsfolgen dienen, die Lösungen darstellen.

Lernalgorithmus

Ein Lernalgorithmus für Marko-Aktionen durchläuft die Lösung eines Planes und erfasst die am häufigsten darin vorkommenden Aktionsteilfolgen.

Dabei werden zwei Parameter definiert:

- *l* (Länge der Makro-Aktion, also die Anzahl der enthaltenen Standard-Aktionen. Der Wert von *l* kann zwischen Zwei und der Länge des längsten Lösungsplanes liegen. Das Optimum liegt zwischen diesen Werten.)
- *k* (Anzahl der Aktionen die in einem Lösungsplan maximal übersprungen werden dürfen. So können nicht relevante Zwischen-Aktionen ignoriert werden.)

Die Komplexität des Algorithmus beträgt $\binom{l+k}{l} n$, wobei *n* die Anzahl der Aktionen in einem Plan ist.

Nutzung des erlernten Wissens

Marko-Aktionen können vom jedem Standard-Planer verwendet werden, da sie als Standard-Aktionen in das Aktionsmodell integriert werden können.

4. Wettbewerb

Der internationale Planungswettbewerb (engl. *International Planning Competition*, kurz IPC) findet zusammen mit der ICAPS-Konferenz (engl*. International Conference on Automated Planning and Scheduling*) alle zwei Jahre statt. Hier treten Planungssysteme in unterschiedlichen Klassen gegeneinander an. Die IPC-2011 war in die drei Teile „Deterministisches Planen“, „Planen und Lernen“ sowie „Unsicherheit“ aufgeteilt (nach [IPC11]).

In den folgenden beiden Abschnitten werden die verschiedenen Wettbewerbsdomänen, die Wettbewerbsregeln sowie der Sieger der IPC-2011, der Planer „PbP2.quality“, vorgestellt. Dabei wird sich inhaltlich an [IPC11] und [Ger11] orientiert.

4.1. Domänen

Insgesamt gibt es neun unterschiedliche Wettbewerbsdomänen, in denen sich die teilnehmenden Planer-Systeme beweisen müssen (nach [IPC11]).

Barman

Diese Domäne besteht aus einem Roboter-Barkeeper, der Getränkespender, Gläser und einen Shaker steuert. Ziel ist es, einen Plan der Aktionen des Roboters zu finden, der vorgegebene Getränke serviert.

Blocksworld

In der bekanntesten Domäne *Blocksworld* gibt es einen Tisch, eine Roboterhand und eine Menge von Klötzen. Das Ziel ist es, die Klötze wie gewünscht anzuordnen.

Depots (engl. für Lager)

In dieser Kombination aus Blocksworld- und Transportdomäne müssen Kisten mittels Laster vom einem zum anderen Lager transportiert werden. Die Anzahl der Kisten, die auf- bzw. abgeladen werden dürfen, hängt von der Anzahl der Paletten an jedem Standort ab.

Gripper (engl. für Greifer)

Gripper besteht aus einer Anzahl von Robotern mit je zwei Greifarmen und einer Reihe von Räumen, in denen sich Bälle befinden. Ziel dieser Domäne ist es, einen Plan zu entwickeln, Bälle von einem bestimmten Raum in einen anderen zu befördern.

Parking

Diese Domäne enthält *N* Parkmöglichkeiten sowie *2*(N-1)* Autos. Auf jeder Parkmöglichkeit können maximal zwei Autos stehen. Ausgehend von einer Parkanordnung müssen die Autos nun nach einer anderen Anordnung geparkt werden.

Rover

Eine Menge von Rovern mit unterschiedlichen Ausrüstungen befindet sich auf einem Planeten. Diese Rover müssen nun mit speziellen Sammeloperatoren Boden- und Steilproben analysieren sowie Objekte fotografieren die Oberfläche untersuchen und ihre Ergebnisse einem Lander schicken.

Satellite

Die Domäne *Satellite* umfasst eine Anzahl von Satelliten mit einer jeweils unterschiedlichen Werkzeugausstattung. Das Ziel ist nun, eine vorgegebene Menge von Bildern zu machen, wobei sich die Satelliten entsprechend ihren vorhandenen Werkzeugen aufteilen müssen.

Spanner (engl. für Schraubenschlüssel)

Zu Beginn dieser Domäne ist ein Arbeiter mit einer Menge von Schraubschlüsseln in einem Schuppen. Außerhalb des Schuppens ist ein Tor, an welchem eine Anzahl von Muttern festgezogen werden müssen. Ein Schraubenschlüssel kann nur für jeweils eine Mutter verwendet werden. Der Arbeiter muss Schraubenschlüssel aufheben und die Muttern anziehen. Dies ist die einzige Domäne mit einem gerichteten Suchraum, da der Arbeiter den Schuppen verlassen, ihn danach jedoch nicht mehr betreten kann, um z.B. neue Schraubenschlüssel zu holen.

TPP (Traveling Purchase Problem)

Das Ziel dieser Domäne ist es, bestimmte Märkte auszuwählen, auf denen es vorgegebene Güter gibt. Darüber hinaus muss eine minimale Route gefunden werden.

Domänendefinitionen

Jede Domäne ist in PDDL definiert. Im Folgenden ist ein Ausschnitt der Domänenbeschreibung von *Barman* ([IPC11]) dargestellt:

```
 1  (define (domain barman)
 2  (:requirements :strips :typing)
 3  (:types hand level beverage dispenser container - object
 4         ingredient cocktail – beverage
 5                 shot shaker - container)
 6
 7  (:predicates  (ontable ?c - container)
 8                (holding ?h - hand ?c - container)
 9                (handempty ?h - hand)    (empty ?c - container)
10                (clean ?c - container)
11                (dispenses ?d - dispenser ?i - ingredient)
12                (shaker-empty-level ?s - shaker ?l - level)
13                (shaker-level ?s - shaker ?l - level)
14                (unshaked ?s - shaker)
15                (shaked ?s - shaker)
16                ;;; weitere Prädikate
17                 )
18
```

```
19  (:action clean-shaker
20     :parameters (?h1 ?h2 - hand ?s - shaker)
21     :precondition (and (holding ?h1 ?s)
22                        (handempty ?h2)
23                        (empty ?s))
24     :effect (and (clean ?s)))
```

Listing 4.1: Ausschnitt aus der Domänenbeschreibung von Barman [ICP11]

Die Definition enthält die Bezeichnung der Domäne, die erlaubten Konstanten sowie einen Teil der zu verwendenden Prädikate und Aktionen. Alle vollständigen Domänendefinitionen sind unter [IPC11] zu finden.

4.2. Probleme

Wie bereits in Abschnitt 2.1 erwähnt, besteht ein Problem aus einem Start- und einem Zielzustand. Beides wird durch Fakten repräsentiert.

Für jede Domäne gibt es einen Problemgenerator sowie bereits vorhandene Problemstellungen (Problem-Test-Set).

4.3. Regeln und Ablauf

Als Modellierungssprache muss der STRIPS-Repräsentationsformalismus von PDDL verwendet werden. Es sind sowohl induktive als auch deduktive Lerntechniken zulässig.

Der Ablauf ist in die Lern- und Testphase gegliedert:

Lernphase

Die Lernphase beginnt, wenn alle Teilnehmer ihren Planer-Quellcode eingereicht haben.

Jeder Teilnehmer erhält anschließend für jede Wettbewerbsdomäne folgende Komponenten:

- Datei mit Domänendefinition
- Problemgenerator
- Problem-Test-Set

Jedes Lernsystem generiert für jede Domäne ein Verzeichnis, das die erlernte DCK enthält. Die Teilnehmer haben insgesamt zwei Wochen Zeit, die resultierende DCK sowie die verwendeten Trainingsdaten einzureichen.

Testphase

Jeder Planer wird in jeder einzelnen Domäne mit und ohne die DCK bewertet. Die Bewertung findet auf den lokalen Rechnern der Ausrichter statt.

Jeder Planer hat maximal 15 Minuten Zeit, ein Problem zu lösen.

4.4. Wertung

Eine Wertung findet in den Kategorien Planungszeit, Planqualität und Lernqualität statt.

Planungszeit

*T** ist die minimale Zeit, die ein Planer für ein Problem benötigt hat. Die Punkte für einen Planer, der das Problem in der Zeit *T* gelöst hat, werden mittels der Formel

$$\frac{1}{1 + \log_{10}(\frac{T}{T^*})}$$

berechnet. Die Wertung für die Planungszeit ist die Summe der Punkte für alle Probleme, die ein Planer gelöst hat.

Planqualität

*N** ist die minimal benötigte Anzahl von Aktionen, die ein Planer für ein Problem benötigt. Ein Planer, der einen Lösungsweg mit *N* Aktionen liefert, erhält dafür $\frac{N^*}{N}$ Punkte. Die Summe der Punkte für alle gelösten Probleme ergibt die Wertung für die Planungsqualität.

Lernqualität

Weiterhin bekommt der Planer, dessen Leistungsfähigkeit durch das am meisten gesteigert wurde, eine Auszeichnung. Dabei werden das Ergebnis des Planers ohne DCK sowie die Differenz aus der Punktezahl mit und ohne DCK bewertet, um so ausschließen zu können, dass Planer ohne DCK bewusst verschlechtert werden.

5. PbP2.quality

In diesem Abschnitt wird der Planer PbP2.quality, welcher den Planungswettbewerb im Jahre 2011 gewann, vorgestellt. Dieser gehört zusammen mit dem PbP2.speed zum Planer-System PbP2, weshalb im Folgenden stets von PbP2 gesprochen wird.

Zunächst wird ein allgemeiner Überblick gegeben und anschließend auf seinen Aufbau sowie Funktionalität eingegangen, wobei sich jeweils an [Ger11] orientiert wurde.

5.1. Allgemeines

PbP steht für *Portfolio-based Multi-Planner* und ist ein Multi-Planer-System, das für eine gegebene Domäne eine Menge von am besten geeigneten Planern auswählt. Dabei werden für jeden Planer eine Anzahl von Makro-Aktionen, die bereits in Abschnitt 3.2 erläutert wurden, generiert sowie dessen Parameterkonfigurationen optimiert. PbP2 ist eine überarbeitete und erweiterte Fassung und in zwei unterschiedlichen Varianten verfügbar. PbP2.quality legt den Schwerpunkt auf die Planqualität und PbP2.speed auf die Geschwindigkeit, mit der das System lernt und plant.

5.2. Architektur

In PbP2 sind insgesamt acht Planer integriert, die in Tabelle 5.1 aufgelistet sind.

Tabelle 5.1: In PbP2 integrierte Planer aus [Ger11]

Planer	Autoren, Datum
FD	Helmert, 2006
Lama	Richter & Westphal, 2008
LPG-td	Gerevini, Saetti & Serina, 2005
Macro-FF	Botea, Enzenberger, Müller & Schaeffer, 2005
Marvin	Coles & Smith, 2007
Metric-FF	Hoffmann & Nebel, 2001
SGPlan5	Chen, Wah & Hsu, 2006
YAHSP	Vidal, 2004

Eine Übersicht über die im PbP2-System enthaltenen Komponenten ist in Abbildung 5.1 dargestellt.

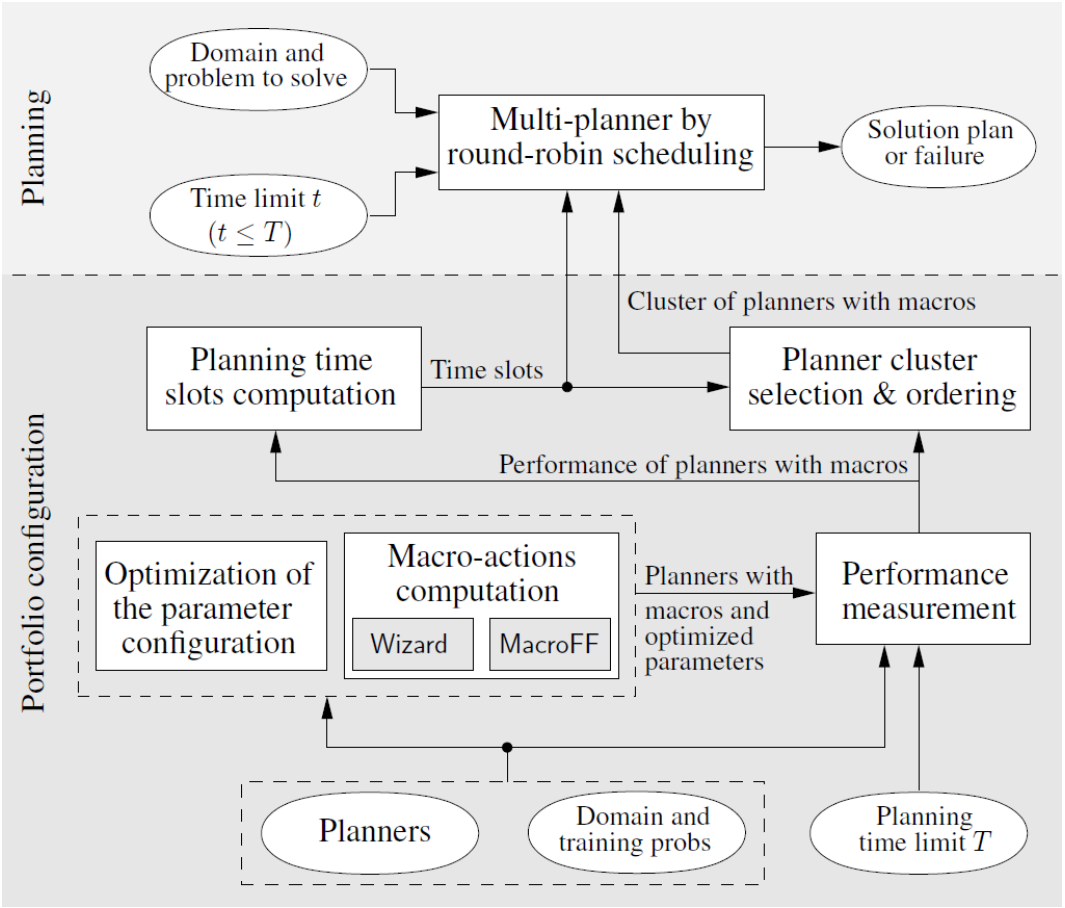

Abbildung 5.1: Architektur von PbP2 aus [Ger11]

Die enthaltenen sechs Hauptkomponenten werden im Folgenden aufgezählt und grob erläutert.

Berechnung der Makro-Aktionen

Für jeden in Tabelle 5.1 enthaltenen Planer werden mehrere Mengen von Makro-Aktionen berechnet. Dabei kommen *MacroFF* und *Wizard* zum Einsatz:

- MacroFF: Bei diesem System werden die Makro-Aktionen berechnet, indem aus den Lösungen von Trainingsdaten die Folgen von Aktionen erfasst werden, die häufig hintereinander vorkommen und den Suchaufwand signifikant verringern, Es werden bis zu fünf Sätze Makro-Aktionen erstellt.
- Wizard: Hier wird ein evolutionärer Algorithmus (engl. *genetic Algorithm*) verwendet. Dabei werden maximal zwei Mengen von Makro-Aktionen produziert.

Optimierung der Parameterkonfiguration

Mittels des PbP2 können parametrisierte Planer konfiguriert werden. Von den in Tabelle 5.1 enthaltenen Planern ist dies jedoch lediglich im Planer LPG-td verwirklicht.

Leistungsbewertung

Die Leistungsbewertung ist der Teil des Planer-Systems, der des meisten Rechenaufwands bedarf. Hierbei wird nun jeder integrierte Planer einmal mit und einmal ohne die erlernten Makro-Aktionen und optimierten Parameter (wie bereits erwähnt, aktuell nur LPG-td) für die Zeit *T* ausgeführt. Als Eingabe dienen hier Trainingsdaten. Die Planer können anschließend hinsichtlich der Problemanzahl, die innerhalb von *T* gelöst werden konnte, der für die Lösung eines Problems benötigte CPU-Zeit sowie der Qualität der berechneten Lösungen beurteilt und verglichen werden.

Berechnung der Planungszeitfenster

Für jeden in PbP2 enthaltenen Planer wird ein Planungszeitfenster berechnet, in dem den jeweiligen Planern während der Lernphase die CPU zugeteilt wird.

Auswahl und Anweisen des Planer-Clusters

Nun wird ein Cluster von Planern, jeweils mit den berechneten Makro-Aktionen und der Standard- oder optimierten Parameterkonfiguration, ausgewählt. Die Reihenfolge der Planer wird durch die Größe der zugewiesenen Zeitfenster festgelegt.

Auswahl eines Planers mittels Round-Robin-Scheduling-Verfahren

Aus diesem Planer-Cluster wird nun mittels eines Round-Robin-Scheduling-Algorithmus und den jeweils errechneten CPU-Zeitfenstern ein Planer ausgewählt und ausgeführt.

5.3. Auswahl der Planer und Makro-Aktionen

Nachdem eine kurze Einführung in die wichtigsten Komponenten des PbP2 gegeben wurde, soll im Folgenden genauer auf die Auswahl des Planer-Clusters und der dazugehörigen Makro-Aktionen eingegangen werden.

Alle Planer werden mit und ohne optimierte Parameterkonfiguration sowie mit jeder zuvor berechneten Makro-Aktionen-Menge in jeder Testdomäne einmal ausgeführt.

Anschließend werden die Ergebnisse ausgewertet und so für jede Domäne das beste Cluster aus Planern und jeder Planer mit jeweils einer Menge von Makro-Aktionen ausgewählt. Dabei werden die Planer zu verschiedenen Clustern zusammengefügt und mittels Round-Robin auf die gleichen Trainingsprobleme ausgeführt.

Die Auswertung wird mittels des Wilcoxon-Vorzeichen-Rang-Tests (engl. *Wilcoxon signed-rank Test*), einem System zur statischen Analyse, durchgeführt.

Die Leistungsbewertung wird entweder bezüglich der Zeit (PbP2.speed) oder der Planqualität (PbP2.quality) durchgeführt.

5.4. Praktische Anwendung

In diesem Abschnitt soll PbP2.quality praktisch angewendet werden. Zunächst wird Allgemeines über die Installation erläutert und anschließend die Ergebnisse der Anwendung zusammengefasst.

5.4.1. Allgemeines

Das dafür benötigte Betriebssystem ist Linux. Der Quellcode des Planers kann unter [PbP2q] heruntergeladen werden.

Folgende Programme müssen installiert werden:

- Flex, ein Scanner -Generator.
- Bison, ein Parser-Generator.

Um den Planer verwendet zu können, müssen das Lern- und das Planprogramm zunächst mittels des Shellskripts `build` erstellt werden. Der entsprechende Befehl sieht folgendermaßen aus (dabei muss man sich im Verzeichnis des Planers befinden):

```
1 ./build
```

Listing 5.1: Befehl, um das Shellskript build auszuführen

Das Format des Befehls, um das Lernprogramm zu starten, sieht wie folgt aus:

```
1 ./learner -o <domain file> -t <training folder> -k <DCK folder>
```

Listing 5.2: Befehlsformat für das Lernprogramm

Mit der Option `-o <domain file>` wird der Pfad der Datei, die die Domänendefinition enthält, definiert. Mittels der zweiten Option `-t <training folder>` wird das Verzeichnis für die Trainingsprobleme festgelegt. Die letzte Option `-k <DCK folder>` legt das Verzeichnis für die erlernte DCK fest.

Das Befehlsformat zum Ausführen des Planers ist in Listing 5.3 dargestellt.

```
1 ./planner -o <domain file> -f <problem file> -p <plan file>
      -k <DCK folder>
```

Listing 5.3: Befehlsformat für das Planprogramm

Die ersten drei Optionen (Angabe der Domänendefinition des Problems sowie der Datei für den Lösungsplan) müssen angegeben werden. `-k <DCK folder>` ist optional. Ohne die Option wird der Planer ohne DCK ausgeführt.

Als Domäne wurde *Barman* gewählt, bei welcher PbP2.quality während der IPC 2011 recht gut abgeschnitten hat. Die Ressourcen können unter [IPC11] heruntergeladen werden.

Nachdem die `build`-Shell ausgeführt wurde, konnte der Planer ausgeführt werden. In den folgenden beiden Abschnitten werden die Ergebnisse der praktischen Anwendung von PbP2.quality erläutert.

5.4.2. Ausführen des Lernprogramms

Insgesamt wurden vier Trainingsprobleme verwendet, die aus dem Problem-Test-Set stammen, das unter [IPC11] heruntergeladen werden kann.

Da das Lernprogramm einige Zeit benötigte, um durchzulaufen, wurde es über Nacht ausgeführt.

Im Verzeichnis `results` sind die resultierenden Planungszeiten (`runtimes`) sowie die Längen der Pläne (`qualities`) zu finden. Die folgenden beiden Abbildungen zeigen jeweils einen Ausschnitt aus den Dateien `barman.qualities` und `barman.runtimes`.

```
data/planners/planner_sgplan.sh#data/learning/planner_sgplan.sh/domain.pddl#
pfile01-001.pddl 21.845
pfile01-002.pddl 33.495
pfile02-006.pddl 13.214
pfile02-007.pddl 27.356
data/planners/planner_Marvin.sh#data/learning/planner_Marvin.sh/domain.pddl#
pfile01-001.pddl 1800.00
pfile01-002.pddl 1800.00
pfile02-006.pddl 1800.00
pfile02-007.pddl 1800.00
data/planners/planner_YAHSP.sh#data/learning/planner_YAHSP.sh/domain.pddl#
pfile01-001.pddl 1800.00
pfile01-002.pddl 1800.00
pfile02-006.pddl 1800.00
pfile02-007.pddl 1800.00
data/planners/planner_lpg_q.sh#data/learning/planner_lpg_q.sh/domain.pddl#
pfile01-001.pddl 1800.00
pfile01-002.pddl 1800.00
pfile02-006.pddl 1800.00
pfile02-007.pddl 1800.00
data/planners/planner_downward.sh#data/learning/planner_downward.sh/domain.pddl#
pfile01-001.pddl 1800.00
pfile01-002.pddl 136.218
pfile02-006.pddl 1800.00
pfile02-007.pddl 1800.00
```

Abbildung 5.2: Ausschnitt aus barman.runtimes

Wie in Abbildung 5.3 zu sehen ist, war SGPlan5 der einzige Planer, der alle Trainingsprobleme lösen konnte. Der Wert `1800.00` zeigt an, dass der jeweilige Planer das Problem nicht lösen konnte. Der Downward-Planer konnte das zweite Problem in der vorgegebenen Zeit lösen.

```
data/planners/planner_sgplan.sh#data/learning/planner_sgplan.sh/domain.pddl#
pfile01-001.pddl 366
pfile01-002.pddl 369
pfile02-006.pddl 402
pfile02-007.pddl 397
data/planners/planner_Marvin.sh#data/learning/planner_Marvin.sh/domain.pddl#
pfile01-001.pddl 1800.00
pfile01-002.pddl 1800.00
pfile02-006.pddl 1800.00
pfile02-007.pddl 1800.00
data/planners/planner_YAHSP.sh#data/learning/planner_YAHSP.sh/domain.pddl#
pfile01-001.pddl 1800.00
pfile01-002.pddl 1800.00
pfile02-006.pddl 1800.00
pfile02-007.pddl 1800.00
data/planners/planner_lpg_q.sh#data/learning/planner_lpg_q.sh/domain.pddl#
pfile01-001.pddl 1800.00
pfile01-002.pddl 1800.00
pfile02-006.pddl 1800.00
pfile02-007.pddl 1800.00
data/planners/planner_downward.sh#data/learning/planner_downward.sh/domain.pddl#
pfile01-001.pddl 1800.00
pfile01-002.pddl 591
pfile02-006.pddl 1800.00
pfile02-007.pddl 1800.00
```

Abbildung 5.3: Ausschnitt aus barman.qualities

Abbildung 5.4 zeigt den letzten Teil der Konsolenausgabe des Lernprogramms, den Wilcoxon Test. Da lediglich ein Planer alle Probleme gelöst hat, ist dieser auch der einzige, der zum Cluster hinzugefügt wird.

```
*********************************************************
Applying Wilcoxon test (1 planners, QUALITY, alpha=0.05)
*********************************************************

Group 0:
  (00) data/planners/planner_sgplan.sh#data/learning/planner_sgplan.sh/domain.pddl#

Keeping only group 0

*********************************************************
Running times using round robin:
*********************************************************
Run only data/planners/planner_sgplan.sh#data/learning/planner_sgplan.sh/domain.pddl#

Done! :)
```

Abbildung 5.4: Output Wilcoxon Test

Leider konnte keine zufriedenstellende DCK erzeugt werden. Möglichweise fehlten einige Komponenten wie Bibliotheken oder es waren zu wenige Trainingsprobleme. Bei der IPC haben die Teilnehmer zwei Wochen Zeit, ihre DCK einzureichen.

Der Inhalt des erzeugten DCK-Verzeichnisses zeigt die nachfolgende Abbildung.

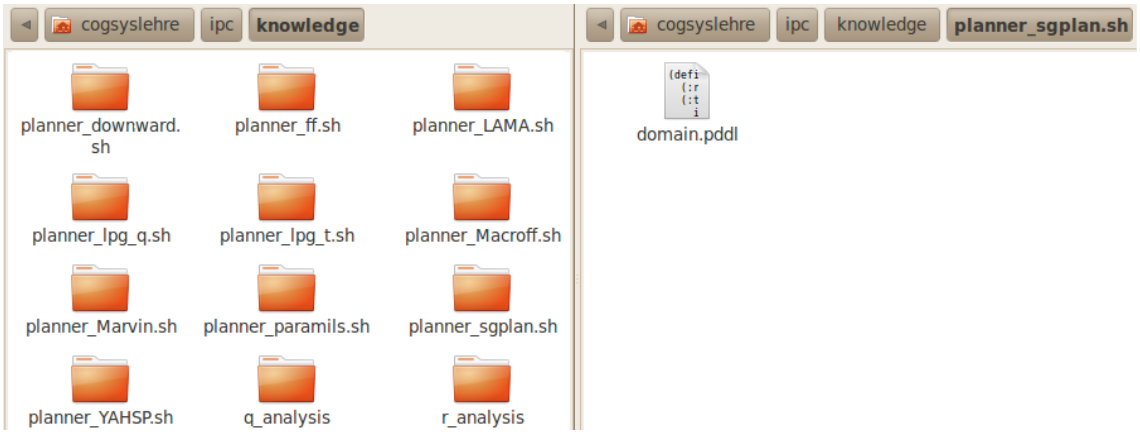

Abbildung 5.5: Erzeugtes DCK-Verzeichnis für die Barman-Domäne

Für jeden Planer wurde ein Ordner erstellt, jedoch enthalten diese lediglich jeweils die pddl-Datei der Barman-Domäne. Dies änderte sich auch nicht, als das Lernprogramm mit zehn Trainingsproblemen ausgeführt wurde.

5.4.3. Ausführen des Planprogramms

Das Planprogramm wurde ohne DCK (ohne die Option `-k`) ausgeführt. Als Ausgangsprobleme wurden `pfile01-001.pddl` und `pfile01-002.pddl` verwendet. Wie die beiden folgenden Abbildungen zeigen, bestehen die Planlösungen aus 366 bzw. 369 Aktionen. Dies entspricht auch den Zahlen in Abbildung 5.3. Für den Plan für das Problem in `pfile01-001.pddl` benötigte der Planer ca. sechs Minuten. Bei der IPC 2011 hatten die Planer für jedes Problem maximal 15 Minuten Zeit, der Planer kann somit auch ohne DCK dieses Zeitlimit einhalten.

```
356.357: (GRASP RIGHT SHOT25) [1]
357.358: (REFILL-SHOT SHOT25 INGREDIENT7 RIGHT LEFT DISPENSER7) [1]
358.359: (POUR-SHOT-TO-CLEAN-SHAKER SHOT25 INGREDIENT7 SHAKER1 RIGHT L0 L1) [1]
359.360: (CLEAN-SHOT SHOT25 INGREDIENT7 RIGHT LEFT) [1]
360.361: (FILL-SHOT SHOT25 INGREDIENT8 RIGHT LEFT DISPENSER8) [1]
361.362: (POUR-SHOT-TO-USED-SHAKER SHOT25 INGREDIENT8 SHAKER1 RIGHT L1 L2) [1]
362.363: (GRASP LEFT SHAKER1) [1]
363.364: (LEAVE RIGHT SHOT25) [1]
364.365: (SHAKE COCKTAIL16 INGREDIENT7 INGREDIENT8 SHAKER1 LEFT RIGHT) [1]
365.366: (POUR-SHAKER-TO-SHOT COCKTAIL16 SHOT1 LEFT SHAKER1 L2 L1) [1]
```

Abbildung 5.6: Die letzten zehn Aktionen der Lösung von pfile01-001.pddl

```
359.360: (FILL-SHOT SHOT1 INGREDIENT5 RIGHT LEFT DISPENSER5) [1]
360.361: (POUR-SHOT-TO-CLEAN-SHAKER SHOT1 INGREDIENT5 SHAKER1 RIGHT L0 L1) [1]
361.362: (CLEAN-SHOT SHOT1 INGREDIENT5 RIGHT LEFT) [1]
362.363: (FILL-SHOT SHOT1 INGREDIENT8 RIGHT LEFT DISPENSER8) [1]
363.364: (POUR-SHOT-TO-USED-SHAKER SHOT1 INGREDIENT8 SHAKER1 RIGHT L1 L2) [1]
364.365: (CLEAN-SHOT SHOT1 INGREDIENT8 RIGHT LEFT) [1]
365.366: (GRASP LEFT SHAKER1) [1]
366.367: (LEAVE RIGHT SHOT1) [1]
367.368: (SHAKE COCKTAIL4 INGREDIENT8 INGREDIENT5 SHAKER1 LEFT RIGHT) [1]
368.369: (POUR-SHAKER-TO-SHOT COCKTAIL4 SHOT1 LEFT SHAKER1 L2 L1) [1]
```

Abbildung 5.7: Die letzten zehn Aktionen der Lösung von pfile01-002.pddl

6. Fazit

Sowohl bei der Zusammenfassung als auch beim Ausblick wird sich an [Jim09] orientiert.

6.1. Zusammenfassung

In den neunziger Jahren konnte mittels ML die Leistungsfähigkeit im Bereich des AP gesteigert werden. Nachdem das Konzept der domänenunabhängigen Heuristiken auftauchte, sank jedoch das Interesse für ML. Heute wird es wieder häufiger bei realen Problemen eingesetzt. Einige Beispiele sind im Folgenden aufgezählt.

Verwendung von ML bei Problemen im Alltag

- Logistik: Sortierung von Gegenständen nach unterschiedlichen Kriterien
- Klassifizierung und Clustern von Objekten, Bildern, etc.
- Mustererkennung
- Computerspiele (Gegner, etc.)
- Lernende autonome Roboter
- Suchmaschinen (z.B. CAPTCHA)
- Automatische Rechtschreibkorrektur und Übersetzung

Durch die stetige Verbesserung der ML-Algorithmen und immer neuen Ideen und Ansätzen können immer komplexere Probleme mittels ML gelöst werden.

6.2. Ausblick

Im Folgenden werden einige im Bereich des AP noch offenen Punkte erläutert.

Wissensrepräsentation für mehrere Domänen

Aktuell gibt es keine effektive Wissensrepräsentation, die für mehrere Domänen gültig ist. Sobald die Kernelemente einer Domäne nicht vollständig beschrieben sind, ist es schwierig dafür Wissen zu generieren.

Die standardisierte Repräsentation SAS+, die anstelle von PDDL verwendet werden kann, ist noch nicht ausgereift. Zwar gibt es bereits Möglichkeiten, Domänen mittels abstrakter Strukturen wie Schleifen und Hierarchien zu beschreiben, diese können jedoch nicht von Standard-Planern verwendet werden.

Selbstständige Erfassung von Lernbeispielen

Mittels der selbstständigen Erfassung von Lernbeispielen kann die Lerneffizienz gesteigert werden. Aktuell werden Probleme von Zufallsgeneratoren erstellt, wodurch die Lösbarkeit des Problems nicht garantiert werden kann. Weiterhin sind die Parameter der Problemgeneratoren domänenspezifisch. *Random Walks* ist ein Ansatz, mit dem automa-

tisch und domänenunabhängig Problemstellungen generiert werden können. Jedoch ist diese Strategie noch nicht ausgereift.

Lernalgorithmen

Ein weiterer offener Punkt ist die Anwendung und Kombination von neuen ML-Algorithmen. Bisher wurden hauptsächlich Algorithmen zur Klassifizierung und Regression entwickelt. Der Bereich des relationalen Lernens ist dagegen kaum erforscht.

Richtige Nutzung des erlernten Wissens

Je gieriger (engl. *greedier*) erlerntes Wissen angewendet wird, desto empfindlicher sind die Algorithmen bezüglich des erlernten Wissens. Ist das Wissen „schädlich“, kann es den Planer möglicherweise weiter vom Ziel entfernen. Es muss also eine Möglichkeit gefunden werden, Planer gegen solches Wissen resistent zu machen.

Ein möglicher Ansatz ist die Bewertung von erlerntem Wissen. Dessen Qualität kann aktuell lediglich dadurch überprüft werden, indem es verwendet wird um Trainingsprobleme zu lösen und anschließend die Performanz bewertet wird.

6.3. Praktische Anwendung von PbP2.quality

Das Planprogramm des PbP2.quality konnte erfolgreich getestet werden. Die Erzeugung der DCK war enttäuschend. Möglichweise fehlte dabei Wissen über den Planer. Das Ausführen das Lernprogrammes, also der Befehl ist zwar intuitiv aufgebaut, jedoch haben eventuell Bibliotheken oder Ähnliches gefehlt, so dass das Multi-Planer-System nicht alle notwendigen Komponenten zur Verfügung hatte. Möglicherweise waren es auch immer noch zu wenige Trainingsprobleme.

Anhang

Abbildungsverzeichnis

Abbildung 3.1: Verwendung von ML zur Planungsprozessverbesserung aus [Jim09] 5
Abbildung 5.1: Architektur von PbP2 aus [Ger11] 15
Abbildung 5.2: Ausschnitt aus barman.runtimes 18
Abbildung 5.3: Ausschnitt aus barman.qualities 19
Abbildung 5.4: Output Wilcoxon Test 19
Abbildung 5.5: Erzeugtes DCK-Verzeichnis für die Barman-Domäne 20
Abbildung 5.6: Die letzten zehn Aktionen der Lösung von pfile01-001.pddl 20
Abbildung 5.7: Die letzten zehn Aktionen der Lösung von pfile01-002.pddl 21

Listings

Listing 3.1: Allgemeine Strategie für Blockworld aus [Jim09]..... 6
Listing 3.2: Domänenspezifische Heuristik für Blocksworld aus [Jim09]..... 6
Listing 3.3: Hierarschische Zerlegungsmethode in Blocksworld aus [Jim09] 7
Listing 3.4: Makro-Aktion unstack-putdown der Blocksworld-Domäne aus [Jim09] 8
Listing 4.1: Ausschnitt aus der Domänenbeschreibung von Barman [ICP11] 12
Listing 5.1: Befehl, um das Shellskript build auszuführen.....17
Listing 5.2: Befehlsformat für das Lernprogramm.....17
Listing 5.3: Befehlsformat für das Planprogramm.....17

Tabellenverzeichnis

Tabelle 5.1: In PbP2 integrierte Planer aus [Ger11] 14

Abkürzungsverzeichnis

AP	Automated Planning
DCK	Domain-specific Control Knowledge
HTN	Hierarchical Task Networks
KI	Künstliche Intelligenz
ML	Machine Learning
PbP	Portfolio-based Multi-Planner
PbP2	Portfolio-based Multi-Planner2
PDDL	Planning Domain Definition Language
RL	Reinforcement Learning
STRIPS	Stanford Research Institute Problem Solver
TPP	Traveling Purchase Problem

Quellenverzeichnis

[Ger11] Alfonso E. Gerevini, Alessandro Saetti, Mauro Vallati: *PbP2: Automatic Configuration of a Portfolio-based Multi-Planner*, Freiburg 2011.

[IPC11] IPC2011: Planning and Learning Part - IPC2011, *http://www.plg.inf.uc3m.es/ipc2011-learning/*, 2011, abgerufen am 01.05.2012

[Jim09] Sergio Jiménez, Tomás de la Rosa, Susana Fernández, Fernando Fernández, Daniel Borrajo: *A Review of Machine Learning for Automated Planning*, In The Knowledge Engineering Review, Cambridge 2009

[PbP2q] Planning and Learning research Group: PbP2.quality Planner, *http://www.plg.inf.uc3m.es/~plgweb/learning-Multiplanq.tgz*, abgerufen am 01.07.2012