Handlungsplanung

M. Helmert
G. Röger, P. Eyerich
Wintersemester 2008/2009

Universität Freiburg
Institut für Informatik

Projekt 3: Evaluation aktueller Planungssysteme und Problemformalisierung

Abgabe: 17. Februar 2008

In den bisherigen Projekten haben Sie JavaFF verwendet, um verschiedene Planungsaufgaben zu lösen. Da JavaFF speziell für die Lehre entwickelt wurde, lag der Schwerpunkt bei seiner Entwicklung auf leicht verständlichem Code und guter Erweiterbarkeit, was gewisse Abstriche bei der Performanz notwendig machte.

Damit Sie die Vorlesung nicht mit einem völlig falschem Bild von der Leistungsfähigkeit von Planungssystemen beenden, werden Sie in diesem Projekt zunächst verschiedene, aktuelle Planungssysteme evaluieren.

Da jedoch nicht nur die Qualität des Planungssystems, sondern auch die Qualität der Problemformalisierung einen großen Einfluss auf das letztendliche Ergebnis hat, können Sie im zweiten Teil noch einmal üben, informal gegebene Planungsaufgaben geeignet in PDDL zu formalisieren.

1 Evaluation von Planungsaufgaben

1.1 Der Internationale Planungswettbewerb

Der internationale Planungswettbewerb (International Planning Competition, IPC) findet alle zwei Jahre im Kontext der ICAPS-Konferenz (International Conference on Planning and Scheduling) statt. Dabei können Planungssysteme in verschiedenen Klassen gegeneinander antreten, wobei letztes Mal zwischen deterministischem Planen, Planen mit Unsicherheit und Planen mit Lernen unterschieden wurde. Bei dem deterministischen Teil gab es wiederum Wettbewerbe in den Bereichen sequentielles Planen, temporales Planen und Planen mit „weichen Zielen“ (Ziele, die von einem Plan erreicht werden sollen, aber nicht unbedingt müssen). Dabei wurde jeweils zwischen optimalem und suboptimalem Planen unterschieden.

Da wir uns in der Vorlesung nur mit deterministischem, sequentiellem Planen beschäftigt haben, wollen wir uns in diesem Projekt auf diese Planer beschränken. Werfen Sie daher zunächst einmal einen Blick auf die Webseite des deterministischen Teils der letzten IPC:

`http://ipc.informatik.uni-freiburg.de`

Für einen ersten Einstieg ist es sicherlich sinnvoll, sich mit den Wettbewerbsbedingungen (Competition Rules) vertraut zu machen. Lesen Sie daher zunächst die entsprechenden Informationen, besonders welche Features in den sequentiellen Tracks unterstützt werden mussten und nach welchen Kriterien die Planer bewertet wurden.

Auf der IPC-Webseite finden Sie auch den Sourcecode aller teilnehmenden Planer. Die einzelnen Planer entpacken sich jeweils in ein Verzeichnis, in dem Sie (neben vielen anderen Dateien) zwei Skripte `build` und `plan` vorfinden.

- Mit `build` kompiliert man den jeweiligen Planer.
- Mit `plan` lässt man den Planer laufen.
  Aufruf: `./plan <domain> <task> <output>`
  `<output>` gibt dabei an, wie die Dateien heißen sollen, in denen die gefundenen Pläne gespeichert werden. Bei anytime-Planern werden die Pläne in Dateien `<output>.1`, `<output>.2`,... ausgegeben, wobei der letzte vollständig erstellte Plan gewertet wird (es kann in seltenen Fällen passieren, dass der Planer zufällig genau während des Planschreibens abgebrochen wird).

Manche Planer gehen davon aus, dass die Domain- und die Problembeschreibung direkt im Verzeichnis des Planers stehen (siehe auch `http://ipc.informatik.uni-freiburg.de/PlannerSubmission`). Kopieren Sie die Dateien also im Zweifelsfall einfach dort hin.

1.2 Einige Hinweise zum Kompilieren der Planer

Die meisten Planer kompilieren auf Linuxsystemen recht problemlos (unter Windows vermutlich gar nicht). Kleine Änderungen sind aber meistens doch notwendig. Wir haben die Planer, die Sie evaluieren sollen, auf den Pool-Rechnern getestet. Dort bekommen Sie sie mit den folgenden Änderungen zum kompilieren.

- **LAMA** kompiliert auch ohne Änderungen, allerdings dauert das sehr lange, da das komplette python2.5 mitkompiliert wird. Wenn auf einem Rechner python2.5 bereits installiert ist, können Sie diesen Schritt ausschalten, indem Sie die Zeilen

  ```
  ./build-python
  export PATH=$(dirname "$0")/py2.5/bin:$PATH
  ```

  im `build`-Skript auskommentieren.

- Die **hsp***-Planer kompilieren nicht mit `g++-4.3`. Wir haben `g++-4.1` verwendet, es kann aber auch sein, dass Version 4.2 ebenfalls funktioniert. Sie können die Verwendung der Version 4.1 aktivieren, indem Sie im `build`-Skript jedes Vorkommen von `g++` durch `g++-4.1` ersetzen.

- Die meisten Probleme macht **Gamer**. Hier müssen Sie ebenfalls den `g++-4.1` verwenden. Sie erreichen das, indem Sie in der Datei `ground_src/Makefile` in der Zeile

  ```
  CC  =   g++ -O2 -m32 -I /usr/include/mingw/ -I $(TOPDIR)/include
  ```

  das `g++` wieder durch `g++-4.1` ersetzen.

  Außerdem funktioniert der Mechanismus mit dem `build` den korrekten Wert für `$JAVA_ROOT123` ermittelt, in vielen Fällen nicht. Setzen Sie daher im Zweifelsfall einfach manuell einen geeigneten Wert. Einen geeigneten Wert finden Sie zum Beispiel mit „`locate jni.h`“: Angenommen, `jni.h` befindet sich in Verzeichnis `/usr/lib/jvm/java-6-sun-1.6.0.07/include/`, dann ersetzen Sie die Zeilen

  ```
  currentDir=$(pwd)
  javaDir0=$(which java)
  javaDir="${javaDir0%%java}"
  cd $javaDir
  cd ..
  export JAVA_ROOT123=$(pwd)
  cd $currentDir
  ```

  durch die Zeile

  ```
  export JAVA_ROOT123="/usr/lib/jvm/java-6-sun-1.6.0.07/"
  ```

  Konsequenterweise sollten Sie dann auch alle Aufrufe von java-Programmen (`javac,jar,...`) mit dem kompletten Pfad (z.B. `$JAVA_ROOT123/bin/javac`) aufrufen, um eine Vermischung verschiedener Java-Versionen zu vermeiden. (Bei unseren Tests hat alles auch mit gemischten Versionen funktioniert, das bleibt aber Ihr Risiko…).

1.3 Benchmarks

Sie finden unter

`http://www.informatik.uni-freiburg.de/~ki/teaching/ws0809/aip/domains_p3.tgz`

einen Teil der Benchmarks, die beim Planungswettbewerb für die Evaluation der sequentiellen Planer verwendet wurden. Es handelt sich dabei um die Domänen

- Sokoban (verfolgt Sie wahrscheinlich inzwischen in Ihren Träumen),
- Peg-Solitär (siehe `http://de.wikipedia.org/wiki/Solitär_(Brettspiel)`) und
- Openstacks (siehe `http://ipc.informatik.uni-freiburg.de/Openstacks`).

Teilweise gibt es von den Domänen zwei Ordner mit Planungsinstanzen, einen mit leichteren Planungsaufgaben für die optimalen Planungssysteme und einen mit schwierigeren Aufgaben für die suboptimalen Planer.

Eventuell wundern Sie sich, dass in den Ordnern so viele Domain-Dateien sind. Bei Sokoban und Peg-Solitär sind diese alle gleich, bei Openstacks handelt es sich wirklich um unterschiedliche Dateien. Dies ist vor allem im Hinblick auf den folgenden Abschnitt relevant.

1.4 Probleme mit JavaFF

Für den letzten Planungswettbewerb wurde PDDL um die Anforderung (requirement) `:action-costs` erweitert, die in den Benchmarks auch verwendet wird. JavaFF unterstützt diese Anforderung jedoch noch nicht. Dies ist aber kein Problem, da JavaFF die ältere Anforderung `:fluents` unterstützt, die `:action-costs` „umfasst“. Um JavaFF auf den gegebenen Planungsaufgaben laufen zu lassen, müssen Sie daher in der Domainspezifikation diese Anforderung ersetzen und zudem bei der Definition der `:functions` das `- number` entfernen.

Aufgabe 1: Evaluation aktueller Planungssysteme

1.5 Punkte

(a) Recherchieren Sie, welche drei Planungssysteme beim suboptimalen Track am besten abgeschnitten haben und evaluieren Sie sie und JavaFF (in der ursprünglichen Konfiguration) analog zur Evaluationsmethode des Planungswettbewerbs. Damit Ihre Geduld nicht zu sehr strapaziert wird, ist es ausreichend, wenn Sie den Planern für jede Instanz nur 10 Minuten zur Verfügung stellen.

(b) Verfahren Sie analog für den optimalen Track (selbstverständlich ohne JavaFF).

Wichtiger Hinweis: Setzen Sie den Plannern ein Speicherlimit, das deutlich kleiner als der physikalische Speicher ist, da die Planer sonst die Computer zum Stillstand bringen können.

Hinweis: Wir empfehlen Ihnen, sich Skripte zu schreiben, die dann die Evaluierung übernehmen. Geben Sie diese Skripte am besten mit ab, eventuell können Sie mit guten Skripten Punktabzüge an anderer Stelle ausgleichen.

2 Problemformalisierung

Aus der Besprechung der Weihnachtsaufgabe wissen Sie, dass es für ein Problem unterschiedliche Formalisierungen geben kann, und dass die Planer sich mit verschiedenen Formalisierungen unterschiedlich schwer tun können. Die Herausforderung in der Handlungsplanung liegt daher nicht nur in der Entwicklung toller Planungsalgorithmen, sondern auch im Finden geeigneter Problemformalisierungen.

Aus diesem Grund sollen Sie im Rahmen dieses Projekts nochmal das Formalisieren von Planungsaufgaben üben. Welche Probleme Sie formalisieren, bleibt dabei Ihnen überlassen, Sie finden aber im Folgenden einige Inspirationen:

- Rush Hour: ein LKW-Verschiebepuzzle zu dem es im Internet genug Informationen geben sollte (z.B. unter `http://www-fs.informatik.uni-tuebingen.de/~reinhard/rush.html`).
- Klotski: ein Puzzle, das gewisse Ähnlichkeiten zu Rush Hour aufweist, aber etwas anderen Regeln besitzt (`http://en.wikipedia.org/wiki/Klotski`).
- Hamiltonpfadproblem: ein in der Graphentheorie wohlbekanntes und bei Komplexitätsbeweisen gerne verwendetes NP-vollständiges Problem (`http://en.wikipedia.org/wiki/Hamiltonian_path`)

- Türme von Hanoi: muss man sie wirklich noch vorstellen? Fall sie jemand nicht kennt, findet er hier einen offiziell lesenswerten Artikel: `http://de.wikipedia.org/wiki/Türme_von_Hanoi`
- Morgens aufstehen: Wie sehen Sie denn schon wieder aus? Sich zu kämmen und dann den Pullover anzuziehen, war wohl nicht wirklich sinnvoll. Und Ihre Zähne sind auch nicht wirklich sauber, weil Sie sie mal wieder VOR dem Frühstück geputzt haben...
- Alientiles: zum Abschluss nochmal ein Puzzle (`http://www.alientiles.com/`)

Aufgabe 2: Problemformalisierung

pro Domäne *1.5 Punkt* (für maximal 2 Domänen)

Formalisieren Sie 1-2 Domänen Ihrer Wahl. Sie können sich dabei an den oben vorgeschlagenen Ideen orientieren oder sich selbst Domänen ausdenken. Erstellen Sie für jede Domäne mindestens 5 Probleminstanzen und testen Sie, wie gut sich die suboptimalen Planer von Aufgabe 1 bei Ihren Problemen schlagen.

Bewertungskriterien: Neben der Korrektheit der Formalisierung ist es auch wichtig, dass die Probleme für die Planungssysteme interessant sind, d.h. nicht viel zu schwer, aber auch nicht trivial lösbar. Zudem fließt natürlich auch in die Bewertung ein, wie schwierig es ist, das Problem zu formalisieren.

Bonusaufgabe: Tücken der Problemformalisierung

1 Punkt

Finden Sie für eine Domäne zwei äquivalente Formalisierungen, bei denen sich die Planer aber unterschiedlich schwer tun (Sokoban gilt nicht, da in der Besprechung der Weihnachtsaufgabe auf zwei solche Formalisierungen eingegangen wurde). Verwenden Sie bei beiden Formalisierungen das gleiche PDDL-Fragment (also zum Beispiel nicht einmal `:adl` und einmal nur `:strips`). Worin liegt der Unterschied in Ihren Formalisierungen und warum denken Sie, ist die eine für die Planungssysteme leichter zu lösen?

3 Nachbereitung

Nach der Bearbeitung dieses Projekts sollten Sie ein recht gutes Gefühl für die Fähigkeiten aktueller Planungssysteme bekommen haben. Außerdem sollten Sie in der Lage sein, sequentielle Domänen, für die Sie nur eine informale Beschreibung besitzen, geeignet in PDDL zu formalisieren.