5th International Planning Competition: Non-deterministic Track Call For Participation

Blai Bonet
Departamento de Computación
Universidad Simón Bolívar
Caracas, Venezuela
bonet@ldc.usb.ve

Robert Givan
Electrical & Computer Engineering
Purdue University
West Lafayette, IN 47907
givan@ecn.purdue.edu

Abstract

The 5th International Planning Competition will be colocated with ICAPS-06. This IPC edition will contain a track on non-deterministic and probabilistic planning as the continuation of the probabilistic track at IPC-4. The non-deterministic track will evaluate systems for conformant, non-deterministic and probabilistic planning under different criteria. This document describes the general goals of the track, the planning tasks to be addressed, the representation language and the evaluation methodology.

Introduction

The 5th International Planning Competition (IPC-5) will be colocated with the 16th International Conference on Automated Planning and Scheduling, ICAPS-06, to be held in The English Lake District, UK, during June 6–10, 2006. The IPC is a biannual event where planning systems are evaluated across several problems and domains of different difficulty addressing specific planning subtasks. The main goals of the competition are to assess the current state-of-the-art in planning, to evaluate and motivate research in the field, and to identify lines for future research.

The IPC started in 1998 when Drew McDermott and the comittee organized the first IPC at AIPS-98, and created the PDDL language. The competition was a big success in the planning community whose results have had a big impact on research. Since then, IPC had taken place at AIPS-2000, organized by Faheim Bacchus, at AIPS-2002, organized by Derek Long and Maria Fox, and at AIPS-2004, organized by Stefan Edelkamp and Jörg Hoffmann.

The 4th edition of IPC included a track on probabilistic planning for the first time, organized by Michael Littman and Håkan Younes, where 7 research groups from several countries met to evaluate their systems. The results of the competition included a specification language for probabilistic planning (PPDDL), an interactive system for planner evaluation, a set of benchmark problems, and some important learned lessons.

In response to this success, IPC-5 will include a new edition of the probabilistic planning track, generalized to include separate subtracks for nondeterministic planning. Moreover, due to the number of research groups and planning systems dealing with more general forms of non-determinism in planning, non-deterministic planning subtracks that will cover the areas of non-deterministic conformant planning, non-deterministic planning (i.e. conditional planning with full observability), and probabilistic planning (i.e. conditional probabilistic planning with full observability).

As done in the classical track of IPC, we believe that planners that offer different guarantees on the quality of their solutions should be evaluated differently; otherwise the comparisons are not meaningful. Hence, planners within each group will be further categorized by the guarantees they provide, as much as possible given the number of participants.

The rest of this document is organized as follows. Sect. 2 gives a brief background on the different planning tasks included in the competition as well as the form of the solutions. Sect. 3 presents the extensions and restrictions upon the PPDDL language to be used. Sect. 4 focuses on the evaluation aspects of the competition, mainly how different planners are evaluated, while Sect. 5 includes brief thoughts about the nature of the competition problems. Finally, Sect. 6 includes tentative schedule of major timelines, and the appendix includes BNF grammars for the planning languages.

Tasks and Form of Solutions

The competition will focus only on planning problems for *goal reachability* with unit costs, as they are the simplest generalization of classical planning to the non-deterministic setting, and also as the majority of existing planners fall into this category. Problems of this type can be described by models of the form:

M1. a finite state space (set of states) S,
M2. a set $S_0 \subseteq S$ of initial states,
M3. a set $S_G \subseteq S$ of goal states,
M4. sets $A(s)$ of applicable actions for each $s \in S$, and
M5. a non-deterministic transition function $F(s, a) \subseteq S$ for all states $s \in S$ and actions $a \in A(s)$.

Models of M1–M5 are described using a high-level planning language based on propositional logic in which the states are valuations for the propositional symbols, the set of initial and goal states are described by logical formulae, and the set of applicable actions (operators) and the transition function are described by means of action schemata.

The form of a solution and the optimality criteria depend on the particular planning task as follows.

Conformant Planning

The problem of conformant planning is that of deciding whether there exists a *linear* sequence of actions that will achieve the goal from any initial state and any resolution of the non-determinism in the problem (Goldman & Boddy 1996; Smith & Weld 1998).

Given a model for M1–M5, we say that $s_0, a_0, \ldots, a_{n-1}, s_n$ is a *trajectory* generated by actions $a_0, \ldots, a_{n-1}$ when

C1. $s_0 \in S_0$,

C2. $a_k \in A(s_k)$ for $0 \leq k < n$, and

C3. $s_{k+1} \in F(s_k, a_k)$ for $0 \leq k < n$.

The plan $a_0, \ldots, a_{n-1}$ is a (valid) solution to the model if each trajectory under $a_0, \ldots, a_{n-1}$ is such that $s_n \in S_G$. The plan is optimal if its length is minimal.

Non-deterministic Planning

Non-deterministic planning with full observability refers to deciding whether there exists a conditional plan that achieves the goal for a model satisfying M1–M5. The main difference from conformant planning is that solutions are policies (partial functions) mapping states into actions, rather than linear sequences of operators.

Let $\pi : S \rightarrow \cup_{s \in S} A(s)$ be a policy for model M1–M5, S_π the domain of definition of π, and $S_\pi(s)$ the set of states *reachable* from s using π,[1] then we say that:

a) π is *closed* with respect to s iff $S_\pi(s) \subseteq S_\pi$,

b) π is *proper* with respect to s iff a goal state can be reached using π from all $s' \in S_\pi(s)$,

c) π is *acyclic* with respect to s iff there is no trajectory $s = s_0, \pi(s_0), \ldots, s_n$ with i and j such that $0 \leq i < j \leq n$ and $s_i = s_j$, and

d) π is closed (resp. proper or acyclic) with respect to $S' \subseteq S$ if it is closed (resp. proper or acyclic) with respect to all $s \in S'$,

A policy π is a valid solution for the non-deterministic model iff π is *closed and proper* with respect to the set of initial states S_0. A valid policy π is assigned a (worst-case scenario) cost V_π equal to the longest trajectory starting at $s_0 \in S_0$ and ending at a goal state. For acyclic policies with respect to S_0, the cost V_π is always well defined, i.e. $< \infty$. A policy π is optimal for model M1–M5 if it is a valid solution of minimum V_π value.

The competition will only include non-deterministic domains that admit acyclic solutions, and thus optimal solutions always have finite cost.

Probabilistic Planning

Probabilistic planning problems can be described by models M1–M5 extended with

M6. transition probabilities $0 < P_a(s'|s)$, for $s' \in F(s,a)$ and $a \in A(s)$, such that $\sum_{s' \in F(s,a)} P_a(s'|s) = 1$.

[1] A state s' is reachable from s using π if there is some trajectory $s = s_0, \pi(s_0), \ldots, \pi(s_{n-1}), s_n = s'$ ending in s'.

In this case, solutions are also policies π that map states into actions. As in the non-deterministic case, definitions (a)–(d) can be used to characterize the properties of π. A policy π is a valid solution if it is closed and proper with respect to S_0. The cost V_π assigned to a *valid* π is defined as the expected cost incurred by the policy when it is applied from the initial states, i.e. V_π is defined as $\sum_{s \in S_0} V_\pi(s_0)/|S_0|$ where the function $V_\pi(\cdot)$ is the *unique* solution to the Bellman equation giving $V_\pi(s)$ for states $s \notin S_G$:

$$V_\pi(s) \;=\; 1 + \sum_{s' \in F(s,\pi(s))} P_a(s'|s) V_\pi(s'),$$

where $V_\pi(s)$ is taken to be zero for $s \in S_G$. We can then take as *optimal* any policy for a probabilistic model M1–M6 that is a valid policy with minimum V_π value.

Language

For this edition of IPC, there is no need to introduce a new description language or to significantly modify PPDDL. Thus, the official language for the competition is PPDDL with minor extensions required to model non-deterministic effects. On the other hand, the original PPDDL specification is too ample for the competition needs, and thus only a subset of it will be actually used.

For those tasks where an explicit solution plan is required, for example conformant planning, a language for describing such plans is required.

This section describes the extensions and subset of PPDDL to be used in the competition as well as the output language.

The formal definition of PPDDL and its semantics is given in (Younes & Littman 2004). We extend it with an additional non-deterministic statement, the counterpart of the `probabilistic` statement for non-deterministic models, of the form:

`(oneof` e_1 e_2 `...` e_n`)`

where the e_k's are PPDDL effects. The semantics is that when executing such effect, one of the e_i is chosen and applied to the current state.

For the competition, PPDDL specifications will be based on the `:adl` requirement, i.e. STRIPS with arbitrary conditions and conditional effects, yet no existential quantification, disjunctions or negative literals will be permitted in the preconditions of operators nor in the conditions for conditional effects. However, general formulae will be allowed in the descriptions of the goals. As mentioned in the PPDDL manual, all effects will be order independent and non-conflicting (interfering), see (Younes & Littman 2004) for details. Additionally, in order to ease the development of parsers for PPDDL, all operator schemata will be such that non-determinism inside conditional effects and/or nested conditional effects will not be allowed (this is similar to the 1ND normal form of (Rintanen 2003) with additionally no nested conditional effects). This restriction still leaves PPDDL universal (Rintanen 2003).

Output Language

Conformant planners or planners that claim guarantees of properness and/or optimality of solutions will be required to output the solution policy into a file in a suitable representation language. This section describes such language. Other output languages will be considered on request, as needed, but the competition staff may not have the resources to support additional output languages.

The file contains three sections separated by '%%':

```
<n> <atom-list>
%%
<m> <action-list>
%%
<plan>
```

where <n> is an integer, possibly 0, denoting the size of <atom-list> which is a space-separated list of atoms such as '(on A B)', <m>, possibly 0, is the size of <action-list> which is a space-separated list of operators 'such as (move A C B)', and <plan> is the representation of the plan.

For conformant planning problems, <plan> must be of the form:

```
linear <k> <integer-list>
```

where <k> is the size of <integer-list> which is a space-separated list of integers in $[0, m-1]$ each denoting the action with such index.

For example, the following file denotes the conformant plan '(pick B);(putdown B)':

```
0
%%
4 (pick A) (putdown A) (pick B) (putdown B)
%%
linear 2 2 3
```

For non-deterministic and probabilistic problems, <plan> can be either an explicit or a factored representation of the policy. In the first case, <plan> is of the form:

```
policy <k> <map-list>
```

where <k> is the size of <map-list> which is a space-separated list of variable-sized elements. The elements of <map-list> define a partial function mapping states into actions. Each element is of the form:

```
<l> <atom-list> <action-index>
```

where <l> is the size of <atom-list> which is a space-separated list of integers in $[0, n-1]$, each denoting the atom with such index, and <action-index> is an integer in $[0, m-1]$ denoting the action with such index. Such element defines the mapping from the unique state that makes all and only all atoms in <atom-list> true into the action with appropriate index.

For example, the file:

```
4 (on A B) (clear A) (clear B) (on B A)
%%
4 (pick A) (putdown A) (pick B) (putdown B)
%%
policy 2 2 0 1 0 2 3 2 2
```

denotes the policy π such that $\pi(s)$ and $\pi(s')$ are '(pick A)' and '(pick B)' respectively, and $s = \{$(on A B), (clear A)$\}$ and $s' = \{$(on B A), (clear B)$\}$.

Factored representation of policies are supported in the form of Free Algebraic Decision Diagrams (FADDs) (Bryant 1992). An FADD is like an Free Binary Decision Diagram whose leaves are tagged with reals. In our case, we use FADDs with leaves tagged with integers in $[0, m-1]$ denoting actions.

For a factored policy, <plan> is of the form:

```
factored <k> <fadd-elements>
```

where <k> is the size <fadd-elements> which is a space-separated list of variable-sized elements that define the FADD. Each FADD element is either an internal element or a leaf element. An internal element is of the form:

```
I <atom> <left> <right>
```

where <atom> is an integer in $[0, n-1]$ denoting the atom with such index, and <left> and <right> are integers in $[0, k-1]$ denoting the FADD elements with such indexes. The <left> branch corresponds to states when <atom> is true and the <right> branch to states when <atom> is false. A FADD leaf is of the form:

```
L <action>
```

where <action> is an integer in $[0, m]$: if <action> is less than m, it denotes the action with such index, else it denotes the *undefined* action. Undefined actions are needed in factored representations since not all the valuations of atoms stand for valid states of the problem and/or since the policy doesn't need to be complete in order to be a proper and closed policy. The FADD elements should be listed in *inverse topological order* of the DAG associated to the FADD.

For example, the file:

```
4 (on A B) (clear A) (clear B) (on B A)
%%
4 (pick A) (putdown A) (pick B) (putdown B)
%%
factored 3
L 0
L 2
I 1 0 1
```

denotes a policy π' such that $S_{\pi'} \supseteq S_\pi$ and that π' agrees with π in S_π, the domain of definition of π, where π is the policy defined above. The corresponding FADD is depicted in Fig. 1.

The appendix includes BNF grammars for the version of the PPDDL language to be used in the competition as well as the output language.

Evaluation

The non-deterministic track of IPC-5 will include:

- Conformant planning track
- Non-deterministic planning track: proper subtrack and general subtrack,
- Probabilistic planning track

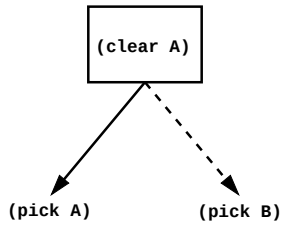

Figure 1: FADD associated with the factored policy in the example in text.

All conformant planner entries will be required to output a valid plan that will be verified by the competition software. The general case for the non-deterministic track, as well as the entire probabilistic track, will use the server-based evaluation of the planning system as was done in IPC-4. Indeed, the same client-server architecture of IPC-4 will be used, perhaps with few changes on the protocol and the evaluation function. In this case, the planner is not required to produce an explicit solution, instead it connects with a server and sends the actions to be executed in a dynamic environment. The planner is evaluated over a number of random 'trials'.

The other cases of conformant planning and the proper subtracks for non-deterministic require planners to produce an explicit solution for the problem. Such solution is checked for properness and its cost is computed. At the end, the planners are ranked by quality of solution and time to compute. For this case, we plan to distribute the properness verifier and cost computation software so competitors can prepare in advance. The software will read the PPDDL specification of the problem and the plan description, and then output whether the plan is proper and its cost.

Prior to the actual competition, competitors will be asked to categorize their planners into one (or multiple) tracks. Any planner that enters the conformant planning track or the proper subtrack of the nondeterministic track must *offer* such guarantee, and the competition software will attempt to check such guarantees. If any such guarantee is found to be false, as for example, if the properness verifier detects an improper policy, then the planner is disqualified from the corresponding domain, and may at the organizers' discretion be disqualified from the entire track.

We are open to offering other tracks upon request, if there is sufficient interest and sufficient time available. In particular we are willing to offer versions of the above tracks that allow for automated domain analysis and/or machine learning via planner interaction with a problem generator and/or PDDL description for a learning period prior to the evaluation. Please inform the organizers of any interests your group has in this or other directions prior to the registration deadline.

Tentative Schedule

- 12/2005: Call for Participation
- 01/31/2006: Deadline for registration
- 02/15/2006: Release of plan verifiers and example domains
- 02/28/2006: Release of final version of languages and server.
- 03/20–26/2006: Mock competition on example domains
- 04/20–26/2006: Competition.

Plan verifiers for the conformant track and the nondeterministic proper track will accept plans as input and return true or false according to whether the plan meets the requirements. Example domains will be either single PDDL files or programs generating a range of related PDDL files (problem generators).

References

Bryant, R. E. 1992. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys* 24(3):293–318.

Goldman, R. P., and Boddy, M. S. 1996. Expressive planning and explicit knowledge. In Drabble, B., ed., *Proc. 3rd International Conf. on Artificial Intelligence Planning Systems*. Edinburgh, Scotland: AAAI Press.

Rintanen, J. 2003. Expressive equivalence of formalisms for planning with sensing. In Giunchiglia, E.; Muscettola, N.; and Nau, D., eds., *Proc. 13th International Conf. on Automated Planning and Scheduling*, 185–194. Trento, Italy: AAAI Press.

Smith, D., and Weld, D. 1998. Conformant graphplan. In Mostow, J., and Rich, C., eds., *Proc. 15th National Conf. on Artificial Intelligence*, 889–896. Madison, WI: AAAI Press / MIT Press.

Younes, H., and Littman, M. 2004. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. http://www.cs.cmu.edu/ lorens/papers/ppddl.pdf.

Appendix A: BNF for PPDDL

The following BNF is adapted from the PPDDL document (Younes & Littman 2004). It describes the version of PPDDL used in the competition.

⟨domain⟩ ::= `(define (domain` ⟨NAME⟩`)`
`(:requirements :adl)`
[⟨types⟩] [⟨constants⟩] [⟨predicates⟩]
⟨action⟩*`)`

⟨types⟩ ::= `(:types` ⟨NAME⟩*`)`

⟨constants⟩ ::= `(:constants` ⟨typed-list⟩`)`

⟨typed-list⟩ ::=

⟨predicates⟩ ::= `(:predicates` ⟨NAME⟩*`)`

⟨action⟩ ::= `(:action` ⟨NAME⟩ [⟨param⟩] ⟨body⟩`)`

⟨param⟩ ::= `(:parameters` ⟨typed-list⟩`)`

⟨body⟩ ::= [⟨prec⟩] [⟨effect⟩]

⟨prec⟩ ::= `(:precondition` ⟨p-formula⟩`)`

⟨effect⟩ ::= `(:effect` {⟨nd-eff⟩ | ⟨det-eff⟩}`)`

⟨nd-eff⟩ ::= ⟨prob⟩ | ⟨one-of⟩

⟨prob⟩ ::= `(probabilistic` ⟨p-eff⟩$^+$`)`

⟨p-eff⟩ ::= ⟨RATIONAL⟩ ⟨det-eff⟩

⟨one-of⟩ ::= `(oneof` ⟨det-eff⟩$^+$ `)`
⟨det-eff⟩ ::= ⟨ATOM⟩ | `(not` ⟨ATOM⟩ `)` |
`(and` ⟨det-eff⟩$^+$ `)` |
`(when` ⟨p-formula⟩ ⟨simple-eff⟩ `)` |
`(forall` ⟨typed-list⟩ ⟨det-eff⟩ `)`
⟨simple-eff⟩ ::= ⟨ATOM⟩ | `(not` ⟨ATOM⟩ `)` |
`(and` ⟨det-eff⟩$^+$ `)` |
`(forall` ⟨typed-list⟩ ⟨simple-eff⟩ `)`
⟨p-formula⟩ ::= ⟨ATOM⟩ |
`(not (=` ⟨NAME⟩ ⟨NAME⟩ `))` |
`(and` ⟨p-formula⟩* `)` |
`(forall` ⟨typed-list⟩ ⟨p-formula⟩ `)`
⟨formula⟩ ::= ⟨ATOM⟩ |
`(not (=` ⟨NAME⟩ ⟨NAME⟩ `))` |
`(not` ⟨formula⟩ `)` |
`(and` ⟨formula⟩* `)` |
`(or` ⟨formula⟩* `)` |
`(forall` ⟨typed-list⟩ ⟨formula⟩ `)` |
`(exists` ⟨typed-list⟩ ⟨formula⟩ `)`
⟨problem⟩ ::= `(define (problem` ⟨NAME⟩ `)`
`(:domain` ⟨NAME⟩ `)`
`(:requirements :adl)`
[⟨objects⟩]
[⟨init⟩] ⟨goal⟩ `)`
⟨goal⟩ ::= `(:goal` ⟨formula⟩ `)`
⟨init⟩ ::= `(:init` $tokeninit - el^*$ `)`
⟨init-el⟩ ::= ⟨ATOM⟩ |
`(probabilistic` ⟨p-init-el⟩$^+$ `)` |
`(oneof` ⟨ATOM⟩$^+$ `)`
⟨p-init-el⟩ ::= ⟨RATIONAL⟩ ⟨ATOM⟩

Appendix B: BNF for Output Language

⟨file⟩ ::= ⟨atoms⟩ `%%` ⟨actions⟩ `%%` ⟨plan⟩
⟨atoms⟩ ::= ⟨INT⟩ ⟨ATOM⟩*
⟨actions⟩ ::= ⟨INT⟩ ⟨ACTION⟩*
⟨plan⟩ ::= ⟨linear⟩ | ⟨policy⟩ | ⟨factored⟩
⟨linear⟩ ::= `linear` ⟨INT⟩ ⟨INT⟩*
⟨policy⟩ ::= `policy` ⟨INT⟩ ⟨map⟩*
⟨map⟩ ::= ⟨INT⟩ ⟨INT⟩* ⟨INT⟩
⟨factored⟩ ::= `factored` ⟨INT⟩ ⟨fadd⟩*
⟨fadd⟩ ::= ⟨internal⟩ | ⟨leaf⟩
⟨internal⟩ ::= `I` ⟨INT⟩ ⟨INT⟩ ⟨INT⟩
⟨leaf⟩ ::= `L` ⟨INT⟩