IPC 2004 Probabilistic Planning Track: FAQ 0.1

Michael L. Littman
Department of Computer Science
Rutgers University
Piscataway, NJ 08854 USA
mlittman@cs.rutgers.edu

Håkan L. S. Younes
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
lorens@cs.cmu.edu

Abstract

The 2004 International Planning Competition, IPC 2004, will include a probabilistic planning track for the first time. This document provides some of the high level decisions that have been made concerning how the competition will be run.

The 2004 International Planning Competition, IPC 2004, will include a probabilistic planning track for the first time. This document lays out some of the high level decisions that have been made concerning how the competition will be run. The details are still in flux at the time of this writing, although we hope to have them spelled out more precisely by the beginning of Summer 2003.

The overriding goal of the probabilistic planning track is to bring together two communities converging on a similar set of research issues and aid them in creating comparable tools and approaches. One community consists of Markov decision process (MDP) researchers interested in developing algorithms that apply to powerfully expressive representations of environments. The other consists of planning researchers incorporating probabilistic and decision theoretic concepts into their planning algorithms. Cross fertilization has begun, but the probabilistic planning track promises a set of shared benchmarks and evaluation metrics that could crystallize efforts in this domain of study.

This document represents a snapshot of the ongoing development of the IPC 2004 probabilistic track. For the latest developments, please visit: `http://www.cs.rutgers.edu/~mlittman/topics/ipc04-pt.html`.

Frequently Asked Questions

There are many issues to be ironed out as part of establishing the new track, but there are some issues that are mostly decided at this point. Here are some clarifying questions and their answers.

What domain description language will be used to represent probabilistic domains?

That's a great first question. We are creating a new domain description language, sketched in the second section of this document. It is modeled on PDDL 2.1, the domain description language for deterministic domains that has been used in the IPC in the past. Syntactically, this language has a STRIPS-like flavor, but includes probabilistic constructs.

By basing the domain description language on PDDL, we remain in the spirit of the existing programming competition, which we expect to help further bring the communities together.

But, I like DBN representations. Is there some way for me to participate?

Do not fear. The representation is sufficiently powerful to support a direct translation from the kind of conditional probability tables used in dynamic Bayesian network (DBN) representations (Dean & Kanazawa 1989), even if they include context-specific independence (Boutilier *et al.* 1996).

For groups that have DBN-based planners and don't feel they will be able to make drastic modifications, it is worth pointing out that there are techniques that make it possible to generate a DBN representation from a (propositionalized) STRIPS-like representation with only a polynomial increase in representation and plan size (Littman 1997).

Was that a shameless plug?

No, we're a little ashamed.

Since the domain description language is based on PDDL, does this mean the representation is relational?

Yes, you're very observant. Although representations with explicit objects are not a traditional feature of MDP-based domain description languages, algorithms that exploit these features have begun to appear.

We expect that many groups will propositionalize the domains because they cannot directly plan with parameterized operators. Most of the test domains will allow for relatively straightforward propositionalization, so the relational representation should not be seen as an impediment to entry for interested groups. We simply feel that relational MDPs are an exciting direction worth supporting and want to give researchers interested in relational issues an opportunity to explore this type of representation.

I don't know how to write a parser for such an elegant, syntactically rich language. What resources will be available for me?

Thank you, it is a nice language, isn't it? We intend to provide software in C++ for a plan validator and very simple planner (essentially a parser). We are also considering

writing some conversion tools from our domain description language to a simple propositional format. This could be used to prepare domains for input to various existing planning algorithms.

Wait a second. Doesn't PDDL 2.1 support numbers? How can we propositionalize when there are numbers?

Numbers will only be used in a very limited way to express rewards; further details are given in the next section. None of the domains for the probabilistic track will have numeric variables as part of the state space, so the domains we use will propositionalize.

Will problems have a single initial state, or a probability distribution over possible initial states?

We don't think this matters conceptually, since the initial state can always be set so that it produces a probabilistic transition to a set of states immediately following the first action. However, it is syntactically convenient to allow explicit initial distributions, so we include this feature.

Will there be continuous variables or simulated physics?

Not at this time, no. Although these would be critical for representing many important domains (like billiards, say), we are not aware of any planners that can exploit representations of this kind. We hope the community is able to move in this direction in the future.

I haven't seen anything about partial observability. Will it be supported?

Planning in partially observable domains is very important and it is a direction we believe the community should move in. At this time, however, there are a greater number of planning algorithms that can make use of complete information, so the IPC 2004 will feature complete observability exclusively (MDPs, not POMDPs).

Our domain description language does support partial observability, since fluents can be explicitly marked as `:unobservable`. The hidden fluents will not be available for decision making.

No test domains in the competition will include unobservable fluents, so partially observability is not supported in IPC 2004.

How will the competition be run?

The probabilistic track will follow the same procedure as the classical track of IPC 2004 (see `http://ipc.icaps-conference.org/`). The current plan is as follows:

- Test domains will be distributed and all experiments will be run by competitors prior to the ICAPS 2004 conference, not on site.
- Our intention is to provide an extra room at the conference in which the results can be viewed in detail throughout the conference, after a ceremony announcing results and recognizing the "winners".
- In addition, we plan to distribute a handout containing abstracts describing the competing planners.
- These events could be centered around a separate competitors workshop.

Sounds good. But, what will we use for test problems?

Good question. The organizers of the classical track are moving toward more practical problems, but this first instantiation of the probabilistic planning track will be more about realistic expectations than realistic problems.

There will definitely be a noisy blocks world and a noisy logistics problem. Other problems that showcase the probabilistic representation will also be included, but the details are not yet available.

In future years, once the foundation has been laid, problems of practical interest should be introduced, for example planning in a Mars rover with continuous resource management.

How will plans be represented?

In the classical track, a plan is a series of operators. A successful plan is one that, when applied to the initial state, achieves the goal.

Life is not so easy in the probabilistic track. While there are many proposals for plan representations in nondeterministic environments (straightline plans, plan trees, policy graphs, triangle tables, etc.), none is considered a widely accepted standard. In addition, even simple plans are challenging to evaluate exactly in a non-deterministic environment, as all possible outcomes need to checked and combined.

For these reasons, we plan to evaluate planners by sampling or simulation. That is, the plan validator will be a server, and individual planning algorithms will be clients. Planners connect to the validator, receive an initial state, and return an operator. This dialog continues until a terminating condition is reached at which point the validator evaluates the performance of the planner. This entire process is repeated several times and results averaged over the multiple runs.

How will plans be scored, then?

The proposal to evaluate plans using a client–server model means that the distinction between a planner and an executor has significantly blurred. It also means that computation is no longer a one-time preprocessing cost, but something integrated with action selection itself.

Planning quality, therefore, needs to be a combination of expected utility and running time. To a first approximation, we'll have three speed categories (real-time, intermediate, and deliberative) and each planner should try to have the maximum expected utility within the time bounds of its intended category.

Will the domains focus on a more MDP-like decision theoretic reward criterion or a more AI-planning-like goal satisfaction criterion?

In many ways, this is a false distinction. The probability of reaching a goal is equivalent to expected reward if a reward of $+1$ is issued upon goal achievement and all other transitions have $+0$ rewards.

But, what if you have more general rewards being accumulated during execution? Would a goal-oriented planner still be able to do something interesting?

We were just getting to that. Mathematically, general reward problems can be cast as goal-achievement problems. Essentially, each transition with a reward can be viewed as a probabilistic transition to a goal state (proportional to the reward), a probabilistic transition to a non-goal sink state (proportional to the difference between the reward and the maximum possible reward), and a probabilistic transition to the next state (proportional to the original transition probability). Majercik & Littman (2002) provide this argument in more depth and give citations for papers on this topic.

Based on this mapping, it ought to be possible to write a converter that creates a goal-oriented problem from a reward-based problem. Whether this results in a competitive planner is an open question, however.

Note that one or more of the test domains will use only a goal-type performance objective. A description of the `(reward)` fluent appears in the second section.

What about nondeterministic planning?

Depending on the number of groups that are interested, all the test domains will also be evaluated in "nondeterministic" mode. Although the nondeterministic planning community is making progress independently of the probabilistic planning and MDP communities, the IPC 2004 organizers were not able to identify an individual to spearhead a separate nondeterministic track. As such, we hope to include this community as part of the probabilistic track. We will make sure that several test domains remain sensible when detailed probability values are ignored.

Is there room for reinforcement-learning methods?

We welcome reinforcement-learning approaches to the domains we will use in the competition. At this time, we are not planning a special subtrack in reinforcement learning (no domain model provided). Contact the organizers (`probplan-panel@cs.rutgers.edu`) if you might be interested so we can gauge the appeal of something like this. There is a good chance that the line separating learning from planning will blur considerably in the next decade, so it is our hope that the competition will move the community in this direction.

Will there be opportunities about other kinds of learning?

Yes, because we will provide the formal descriptions for some of the domains in advance, there will be an opportunity for groups to learn about these domains in advance. For at least one domain, we hope to have an explicit parameterized generator for problems, which will be available well in advance of the competition. In this case, approaches that try to generalize planning strategies from solving small instances will have an opportunity to benefit from this information.

What is the competition timeline?

- By July 2003, develop and circulate the specification for the domain description language. At the same time, distribute several domains and problems, and an initial validator.
- Starting in July 2003, we will respond to feedback from the research community on the initial release of the domain description language, fix any important problems with the domains, problems, and evaluation metrics. Initial statements of interest will be due from the research groups wishing to participate in IPC-4.
- January 2004, several test domains with automated problem generators will be distributed to the research community for use by groups interested in studying "learning to plan". These domains will be used in the competition with previously unreleased problems.
- In April 2004, IPC-4 will be held in Vancouver. Planners will be run on the competition problems one week prior to the conference, with analysis and results to be announced at the conference. Results and test domains will be released publicly.

Is there something else I should be asking?

Almost certainly, but we've run out of answers at this point. The most important thing is that there is a lot of work still to be done and we'll need your help. If you would like to participate in any way, please contact `mlittman@cs.rutgers.edu`. We are maintaining a list to discuss the design of the competition (`probplan-panel`, currently 13 people signed up) and one to make announcements to anyone interested in the area (`probplan-announce`, currently 87 people signed up). There will also be a list specific to groups who will enter planners in the competition. We hope you will be able to contribute!

Probabilistic PDDL

We present extensions to PDDL 2.1 allowing for the modeling of MDP domain models. Familiarity with PDDL 2.1 syntax (Fox & Long 2002) is assumed as we only present the syntax of new language constructs here.

Probabilistic Effects

An important aspect of MDP domain models is that actions can have probabilistic effects. We adopt a model of stochastic actions that is a variation of *factored probabilistic STRIPS operators* (Dearden & Boutilier 1997). A stochastic action a consists of a precondition ϕ and a consequence set $C = \{c_1, \ldots, c_n\}$. Each consequence c_i has a trigger condition ϕ_i with a corresponding effects list $\mathcal{E}_i = \langle p^i_1, E^i_1; \ldots; p^i_{k_i}, E^i_{k_i} \rangle$, where E^i_j is a set of literals and $p^i_j \in [0, 1]$ is a probability associated with the jth effect set. We require that $\sum_{j=1}^{k_i} p^i_j = 1$ and that consequences with mutually consistent trigger conditions have *commutative* effects. The latter means that the successor state is the same after applying an action to a state s regardless of the order in which the acting effect sets of enabled consequences are applied to s.

The precondition of a stochastic action serves as a factored trigger condition common to all consequences in C. An action therefore has no effects if it is executed in a state where its precondition does not hold. The full semantics of stochastic actions is provided elsewhere (Younes 2003).

We can specify a stochastic action by extending the PDDL syntax for action effects with a probabilistic construct similar to that used by Bonet & Geffner (2001). Figure 1

shows the proposed extension. A new requirements flag, `:probabilistic-effects`, is used to indicate that support for probabilistic effects is required.

There is a clear correspondence between the syntax and the representation of stochastic actions introduced above. An effects list is specified as

(probabilistic p_1^i E_1^i ... $p_{k_i}^i$ $E_{k_i}^i$).

The above statement also represents a consequence with a trigger condition $\phi_i = true$. Consequences with non-trivial trigger conditions are specified using conditional effects:

(when ϕ_i (probabilistic p_1^i E_1^i ... $p_{k_i}^i$ $E_{k_i}^i$))

Figure 2 shows a domain description with stochastic actions. A statement such as

```
(probabilistic 0.9 (and (clear ?x) ...))
```

with the probabilities not adding up to 1 is meant as a syntactic sugar for

```
(probabilistic 0.9 (and (clear ?x) ...)
               0.1 (and)),
```

where `(and)` represents an empty effect set.

Numeric effects can be used in combination with probabilistic effects, although this could result in a stochastic process with an infinite state space. This could be remedied by introducing a bounded integer type, `(integer` *low high*`)`, in addition to the standard PDDL type, `number`, for functional expressions. This would provide a straightforward way of ensuring a finite state space. For example,

```
(:functions (power ?x) - (integer 0 10))
```

would effectively define an integer state variable $power_x \in [0, 10]$ for each object x in the domain. This is not a language feature we expect to introduce for the 2004 competition, however, so numeric effects will not be used in the probabilistic track except to express rewards (explained next).

Rewards and Goals

Markovian rewards can be encoded using fluents. We reserve the functional expression `(reward)` to represent the total accumulated reward since the start of execution. The accumulated reward is not considered part of the state space, so `(reward)` will never occur in action preconditions and effect conditions. The reward is of course zero in the initial state. A new requirements flag, `:rewards`, is introduced to signal that support for rewards is required. Rewards will be encoded as action effects as shown in Figure 3.

Figure 4 defines a probabilistic domain with rewards. The reward is $+100$ for opening the door that the tiger is not behind it, and there is a -100 reward for opening the door that the tiger is behind. The figure also includes a problem definition. We have used the plan metric introduced in PDDL 2.1 to specify that the objective is to maximize the accumulated reward. The problem definition also shows how to specify a probability distribution over initial states. The statement

```
(:init (probabilistic 0.5 (tiger-on-left)))
```

means that the probability is 1/2 that the tiger is behind the left door in the initial state. Predicates not mentioned are assumed false in the initial state (closed-world assumption).

Regular PDDL goals will be used to express goal-type performance objectives. A goal statement `(:goal` ϕ`)` for a probabilistic planning problem encodes the objective that the probability of achieving ϕ should be maximized. We can transform this into a reward-oriented problem, as discussed earlier, by making all states satisfying ϕ absorbing, assigning a reward of $+1$ to transitions into such a state, and assigning zero reward to all other transitions.

Acknowledgments

David Andre, Bob Givan, Carlos Guestrin, Terran Lane, Nicolas Meuleau, Ron Parr, Christian Shelton, and others asked some of the questions.

References

Bonet, B., and Geffner, H. 2001. GPT: A tool for planning with uncertainty and partial information. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence Workshop on Planning under Uncertainty and Incomplete Information*, 82–87.

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI 96)*, 115–123.

Dean, T., and Kanazawa, K. 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3):142–150.

Dearden, R., and Boutilier, C. 1997. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence* 89(1–2):219–283.

Fox, M., and Long, D. 2002. PDDL2.1: An extension to PDDL for expressing temporal planning domains. Technical Report 20/02, University of Durham, Durham, UK.

Littman, M. L. 1997. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 748–754. AAAI Press/The MIT Press.

Majercik, S. M., and Littman, M. L. 2002. Contingent planning under uncertainty via stochastic satisfiability. *Artificial Intelligence*. In press.

Younes, H. L. S. 2003. Extending PDDL to model stochastic decision processes. In *Proceedings of the ICAPS-03 Workshop on PDDL*.

```
<effect>        ::= <d-effect>
<effect>        ::= (and <effect>*)
<effect>        ::=:conditional-effects (forall (<typed list(variable)>) <effect>)
<effect>        ::=:conditional-effects (when <GD> <d-effect>)
<d-effect>      ::=:probabilistic-effects (probabilistic <prob-eff>+)
<d-effect>      ::= <a-effect>
<prob-eff>      ::= <probability> <a-effect>
<a-effect>      ::= (and <p-effect>*)
<a-effect>      ::= <p-effect>
<p-effect>      ::= (not <atomic formula(term)>)
<p-effect>      ::= <atomic formula(term)>
<p-effect>      ::=:fluents (<assign-op> <f-head> <f-exp>)
<probability>   ::= Any rational number in the interval [0, 1].
```

Figure 1: PDDL extension for probabilistic effects.

```
(define (domain fuzzy-blocks-domain)
   (:requirements :probabilistic-effects)
   (:predicates (on ?x ?y) (ontable ?x) (clear ?x) (handempty) (holding ?x))
   (:action pick-up
            :parameters (?x)
            :precondition (and (clear ?x) (ontable ?x) (handempty))
            :effect (probabilistic 0.9 (and (not (ontable ?x)) (not (clear ?x))
                                            (not (handempty)) (holding ?x))))
   (:action put-down
            :parameters (?x)
            :precondition (holding ?x)
            :effect (and (not (holding ?x)) (clear ?x) (handempty) (ontable ?x)))
   (:action stack
            :parameters (?x ?y)
            :precondition (and (holding ?x) (clear ?y))
            :effect (and (not (holding ?x)) (clear ?x) (handempty)
                         (probabilistic 0.95 (and (not (clear ?y)) (on ?x ?y))
                                        0.05 (ontable ?x))))
   (:action unstack
            :parameters (?x ?y)
            :precondition (and (on ?x ?y) (clear ?x) (handempty))
            :effect (probabilistic 0.9 (and (holding ?x) (clear ?y) (not (clear ?x))
                                            (not (handempty)) (not (on ?x ?y))))))
```

Figure 2: Blocks-world domain with actions having probabilistic effects.

```
<p-effect>   ::=:rewards (increase (reward) <number>)
<p-effect>   ::=:rewards (decrease (reward) <number>)
```

Figure 3: Syntax for rewards as action effects.

```
(define (domain tiger-domain)
   (:requirements :negative-preconditions :conditional-effects
                  :probabilistic-effects :rewards)
   (:predicates (tiger-on-left) (hear-tiger-on-left))
   (:action listen
            :effect (and (when (tiger-on-left)
                               (probabilistic 0.85 (hear-tiger-on-left)
                                              0.15 (not (hear-tiger-on-left))))
                         (when (not (tiger-on-left))
                               (probabilistic 0.85 (not (hear-tiger-on-left))
                                              0.15 (hear-tiger-on-left)))))
   (:action open-left-door
            :effect (and (when (not (tiger-on-left)) (increase (reward) 100))
                         (when (tiger-on-left) (decrease (reward) 100))))
   (:action open-right-door
            :effect (and (when (tiger-on-left) (increase (reward) 100))
                         (when (not (tiger-on-left)) (decrease (reward) 100)))))


(define (problem tiger-problem)
   (:domain tiger-domain)
   (:init (probabilistic 0.5 (tiger-on-left)))
   (:metric maximize (reward))
```

Figure 4: Domain and problem definitions with rewards.