Planning in probabilistic domains using a deterministic numeric planner

Sergio Jiménez
Departamento de Informática.
Universidad Carlos III de Madrid.
Avda. de la Universidad, 30. Leganés (Madrid). Spain.
sjimenez@inf.uc3m.es

Andrew Coles and **Amanda Smith**
Department of Computer and Information Sciences.
University of Strathclyde.
26 Richmond Street, Glasgow, G1 1XH, UK
firstname.lastname@cis.strath.ac.uk

Abstract

In the probabilistic track of the IPC5 —the last International Planning Competitions— a probabilistic planner based on combining deterministic planning with replanning —FF-REPLAN— outperformed the other competitors. This probabilistic planning paradigm discarded the probabilistic information of the domain, just considering for each action its nominal effect as a deterministic effect. Thus, in certain domains, the plans proposed by this approach are not robust, so replanning occurs too much frequently. This paper describes a new approach to solve probabilistic planning problems, also based on deterministic planning and replanning; but without rejecting the probabilistic information of the domain. In this approach, the probabilistic domain is compiled into a new deterministic domain and the probabilistic information is translated to an action 'cost model', used by a numeric planner to improve the robustness of the plans found, reducing the frequency with which replanning occurs.

Introduction

In 2004, the Probabilistic Planning Competition took place for the first time as a separate track alongside the Fourth International Planning Competition (IPC4)[1]. The main objective of this event was to establish a common representation language (Younes *et al.* 2005), and to define benchmarks and methodologies to evaluate and compare the performance of probabilistic planners. The participant planners had to solve sets of problems from different probabilistic domains. The effects of executing the non-deterministic actions was simulated using software provided by the organisers. In 2006 the competition took place for a second time and, as in 2004, the probabilistic planner FF-REPLAN outperformed the other competitors. FF-REPLAN is based solely on deterministic planning, discarding any probabilistic information when proposing plans: it is reliant on replanning should an unexpected state be reached during plan execution. The fact that FF-REPLAN outperformed other planners in general terms, does not mean it is more suitable to solve any probabilistic problem than specialised probabilistic planning techniques, such as solving Markov Decision Processes (MDPs) (Boutilier, Dean, & Hanks 1999). A more detailed analysis of the competition results reveals that:

- In all domains FF-REPLAN finds solutions to many more problems than any other planner. The replanning approach is much more robust to unexpected states than planners that finding policies (total functions mapping states into actions: $\pi = S \rightarrow A$).
- In all domains FF-REPLAN is competitive in terms of computation time with the state-of-the-art probabilistic planners. Replanning using the deterministic planner FF (Hoffmann & Nebel 2001) is sufficiently fast to allow FF-REPLAN to compete with the other probabilistic planning paradigms.
- In several domains FF-REPLAN is not able to find the shortest plans to solve problems. Specifically, in the domains: *Blocksworld*, *Drive* and *Pitchcatch* from the IPC5, FF-REPLAN takes more turns to solve a problem than other probabilistic planners. In these domains the plans proposed by FF are not as robust as they could be, so the execution of those plans fails more often, meaning that a greater number of actions must be executed to solve each problem.

The aim of the approach presented in this work is to develop a probabilistic planning paradigm also based on combining deterministic planning and replanning techniques — thereby being able to deal with unexpected states and to compete with the state-of-the-art planers in terms of computation time— but without discarding the actions' non-nominal effects [2] and the probabilistic information of the domain, so more robust plans can be found. In the paper we describe how to transform a probabilistic planning domain into a deterministic one compiling the probabilities associated with the actions' effects into a 'cost model'.

The organisation of the paper is as follows: first we describe how the probabilistic domain is compiled into a deterministic one, how the probability values are transformed into a cost model and how this information is handled to solve probabilistic planning problems. Then we demonstrate the performance of our approach in several domains from the

[1] http://ipc.icaps-conference.org/

[2] The nominal effects of an action are those that represent the most likely outcome of the action

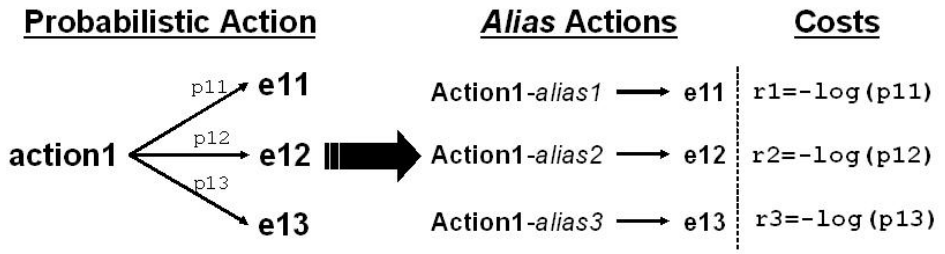

Figure 1: Transforming the probabilistic action *action1* into three *Alias Actions*

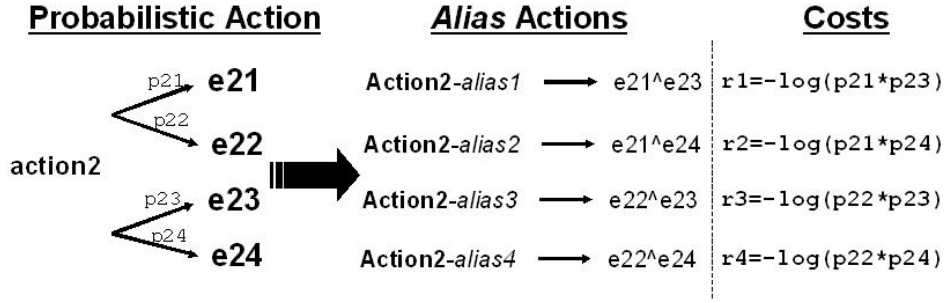

Figure 2: Transforming the probabilistic action *action2* into four *Alias Actions*

probabilistic track of the IPC5. Finally, we discuss some related work, conclusions and future work.

Probabilities as a Planning Action Cost Model

This section describes how a probabilistic domain is processed to solve probabilistic planning problems using a standard deterministic numeric planner and without discarding the probabilities associated with the actions' effects.

From Probabilistic to Deterministic

Every potential effect of a probabilistic action is transformed into an *Alias Action*. An *Alias Action* is a deterministic action representing exactly one effect of a probabilistic action with an associated cost value denoting the risk of the action failing. Figure 3 shows the *Alias Action* resulting from compiling the nominal effect of the probabilistic action PICK-UP from the *Blocksworld* domain of the probabilistic track of the IPC5.

In PPDDL (Younes *et al.* 2005), the potential different effects of an action are described as trees. These trees may branch in multiple paths, with probability values associated to them. The sum the probability values associated to all the branches must be 1. When the effects of the actions are described by just one tree, as happens in Figure 1 the translation process is immediate as every leaf in the tree will generate one *Alias Action*. In the example shown in Figure 1, the sum of the probability values $p11 + p12 + p13$ must be 1.

When the effects of the actions are described by more than one tree (for example, in Figure 2, the potential effects of the `action2` are described by two trees), it is necessary to compute all possible combinations of the different effects

```
(:action pick-up-ALIAS-0
   :parameters (?b1 - block ?b2 - block)
   :precondition (and (not (= ?b1 ?b2))
                      (emptyhand)
                      (clear ?b1)
                      (on ?b1 ?b2))
   :effect (and (holding ?b1)
                (clear ?b2)
                (not (emptyhand))
                (not (clear ?b1))
                (not (on ?b1 ?b2))
                (increase risk 0.287682)))
```

Figure 3: *Alias Action* corresponding to the nominal effect of the action PICK-UP from the *Blocksworld* domain

the action can cause. Each combination of effects generates one *Alias Action*. In the example shown in Figure 2 $p21 + p22$ must be equal to 1 and $p23 + p24$ must be equal to 1.

The Action Cost Model

Intuitively, a metric indicating the robustness of the actions should allow a numeric planner to find robust solutions. As a first approach, this metric represents the risk of failing to achieve some effects when the action is executed by attaching a cost to the action. The cost of applying an action can be defined by the expression:

$$risk_i = 1 - prob_i$$

where $prob_i$ is the probability of the effect i being true when the action has been executed. Using this way, minimising the product of these, as a metric to guide search, would lead the planner towards robust solutions.

Generally, however, numeric planners do not deal very efficiently with minimising products of values; so using this theoretically correct metric will not in practice lead to good experimental results in terms of quality and time. Instead, numeric planners are better designed to minimise the sum of cost values. Thus, based on a property of logarithms, $log(a) + log(b) = log(a * b)$, we propose the following definition for the metric:

$$risk_i = -log(prob_i)$$

Using this expression, minimising the sum of the negative logarithms of the success probabilities results in the product of the failure probabilities being minimised.

Planning with costs

The deterministic planner used in this work is LPG (Gerevini & Serina 2002), version LPG-TD-1.0 with the `quality` option to find plans minimising the risk metric. With this option selected, LPG incrementally finds the best plan in terms of cost action that it can derive within an user-specified CPU-time limit. This planner is used to find a plan with the compiled probabilistic domain consisting of deterministic *Alias Actions*. We call this plan the PAC-PLAN (ProbabilitiesAreCosts-Plan). Once the PAC-PLAN is found, it is translated into an equivalent sequence of actions containing just the original actions from the probabilistic domain. To solve a probabilistic problem, this plan is executed step-by-step. When a plan action cannot be executed it means that a previous action modified the state in a non-desired way: in these cases a new PAC-PLAN is computed to try to solve the probabilistic problem from the new current state.

This approach is non-planner-dependent an any other planner able to handle metric minimisation can be used instead of LPG.

Experiments

This section describes experiments to demonstrate that using the cost model we get from compiling the domain probabilistic information leads the planning process towards more robust solutions. First we describe the test domains and then we report the results of our experiments.

The Domains

We have tested our approach with probabilistic planning problems used in the IPC5 to evaluate the performance of the participant probabilistic planners. The domains chosen are those whose language constructs are supported by the current version of the planner. The result of executing actions has been simulated with the software provided by the competition organisers.

Blocksworld In this domain, the blocks are in some initial configuration, on other blocks or on the table and a goal configuration is specified. A gripper can be holding one block, holding a tower of them or be empty. When trying to perform an action, the gripper can fail resulting in blocks falling down.

Elevators In this domain there are coins scattered all over a building that have to be collected. To collect them one can move over the building. The movements can be horizontally, in the same floor, or can consist on taking an elevator to move to a different floor. Moving horizontally is subject to the risk of falling down into the elevator shaft and finishing on a different floor. This situation does not lead to an unrecoverable dead end, simply to an unexpected state.

Exploding Blocksworld This is a version of the classic *Blocksworld* domain where blocks can explode when they are put down. The explosion of a block can affect to the table or other blocks.

Tireworld This domain involves driving a car between two locations. Every time the car covers a stretch of the journey one of its tyres may become flat with a given probability. When the car has a flat tyre it has to be replaced by a spare one, however, spare tyres cannot be found in all locations.

Tree This domain involves moving from a starting level to an upper one. Operators allow to increase just one level at a time. The higher the level reached the more operators there are to get the next one, however the operators also become less reliable.

Zeno Travel This is a transportation domain involving the transportation of people planes. A plane has two different modes of movement: fast and slow. The fast movement consumes more fuel than the slow movement. With some probability actions can fail to execute, and not cause any new effects, so they have to be repeated.

Results

Two configurations of the proposed system are used for evaluation:

- One in which LPG-TD-1.0 with the `quality` option is used, minimising the metric: $risk_i = -log(prob_i)$ and with 30 min as the time limit to find the best solution in terms of the metric.
- One in which the `speed` option is used. In this mode, LPG-td finds a solution (of any quality) as quickly as possible. This configuration serves as a control to allow comparison: the metric is ignored during search, so any differences in performance can be ascribed to the use of the risk metrics.

As in the Probabilistic Planing Competition, the planner attempts each problem 30 times. The planner is allocated a total of 30 minutes to solve all 30 runs of a given problem; if this time limit is exceeded the remaining runs are said to have failed. All results are generated using a machine with a 3.4 GHz Pentium D processor and 2GB of RAM. Three different measurements are recorded:

1. Number of solved problems. The number of times the planner has succeed on solving a problem.
2. Time. The average time the planner needs to solve a problem.

3. Turns. The average number of actions the planner needs to execute to solve the problem.

On problems that the planner fails to solve the values for time and turns are discarded and are not included in the computation of the averages. The values obtained for these measurements in the aforementioned domains are shown in Figures 4 to 9. Note that all results for time taken and number of turns are presented on a logarithmic scale.

In the *Blocksworld* domain, results for which are shown in Figure 4, the quality version of the planner achieves greatly improved coverage compared to the speed version. On all mutually solved problems the quality version requires fewer turns to reach the goal state. The quality version does, however, often take longer to solve problems due to the time spent in finding a more-optimised plan.

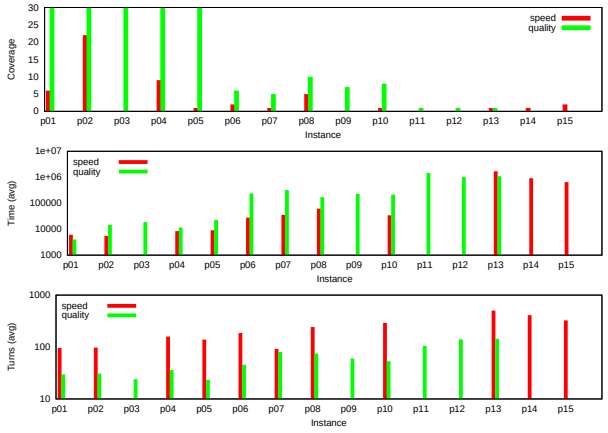

Figure 4: Results obtained running both versions of the planner in the IPC 5 *Blocksworld* domain

The results obtained in the *Elevators* domain are shown in Figure 5. The observations support the expected pattern that the quality version generally takes slightly longer to solve problems, but is more robust, requiring fewer turns to solve problems. Here we observe that the planner is producing more robust plans, meaning replanning will occur less frequently, but it is forced to spend more time in order to generate those plans. The version of PAC-PLAN reasoning about cost consistently solves more of the presented problems than the speed version. Indeed there are two problems, problems 9 and 10 that are solved by the quality version and not by the speed version. In the case of problem 9 the planner is able to solve most of the problems; on problem 10, however, only 2 of the problems are solved.

In the *Exploding Blocksworld* domain, results for which are shown in Figure 6, the quality version again achieves greater coverage than the speed version. The number of turns used by the quality version is lower than that required by the speed version. The speed version does, however, solve the problems more quickly.

The results in the *Tireworld* domain, shown in Figure 7, show that the quality version of the planner is using fewer turns to solve problems. The speed version is still solving many of the problems more quickly than the quality version;

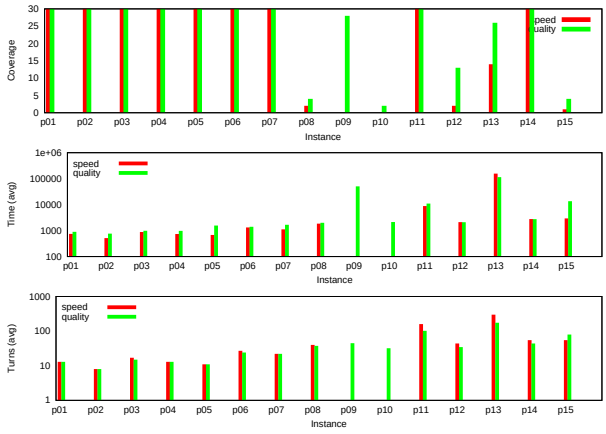

Figure 5: Results obtained running both versions of the planner in the IPC 5 *Elevators* domain

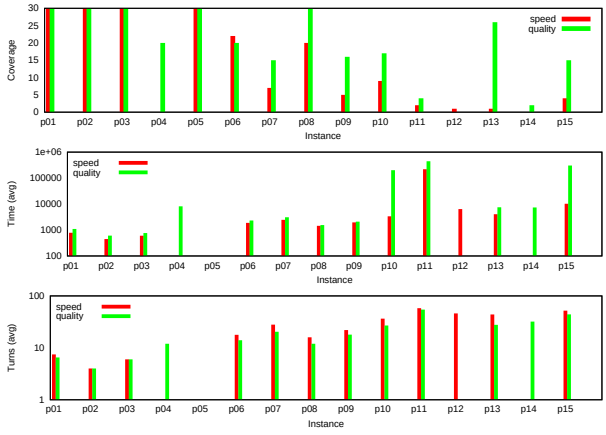

Figure 6: Results obtained running both versions of the planner in the IPC 5 *Exploding Blocksworld* domain

it is, however, pleasing to note that the quality version does successfully solve 5 of the problems more quickly than the speed version. The reduction in the number of turns used is sufficient, in one third of the problems attempted, to overcome the increased effort in optimising the plan. The quality version of the planner is able to generate a lower risk plan, reducing the frequency with which cars have to take detours in order to change a tire, or with which they fail. The relative coverage varies on the different problems with neither version clearly emerging as consistently able to solve more problems.

The results for the *Tree* domain, shown in Figure 8, are especially positive. The quality version of the planner again shows that the number of turns can be reduced, in some problems by an order of magnitude, by reasoning about the probabilities during the planning process. Clear benefits are also seen on several problems in terms of time taken: in this domain the extra effort expended attempting to optimise the plan with respect to achieving minimal risk is actually giving an improvement in the time taken to solve more than half

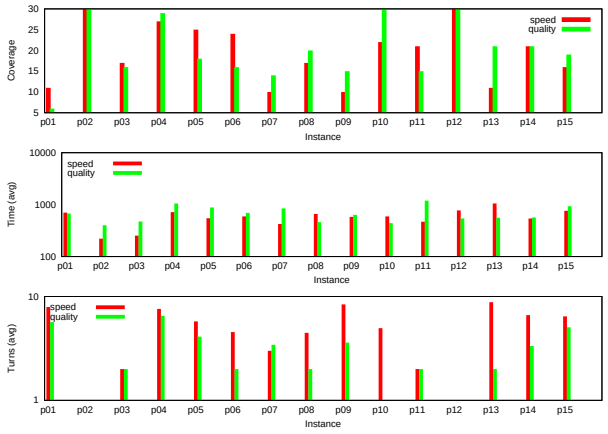

Figure 7: Results obtained running both versions of the planner in the IPC 5 *Tireworld* domain

of the mutually solved problems. Improved coverage is also seen for the quality version.

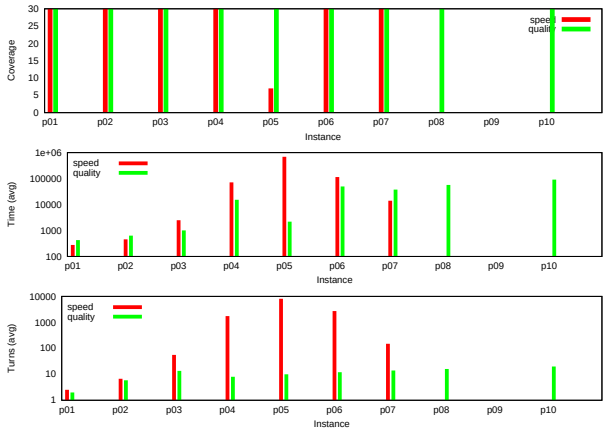

Figure 8: Results obtained running both versions of the planner in the IPC 5 Tree domain

In final domain to be considered, the *Zeno Travel* domain, the speed version outperforms the quality version both in terms of coverage and time taken to solve problems (see Figure 9). This is because the planning phase of this problem is expensive. Asking the planner to minimise the risk metric in an already difficult problem results in a lot of time being spent during the planning phase. Despite the number of turns being smaller when the quality version can solve a problem, this benefit is not sufficient to make the quality version as robust as the speed version. The overheads incurred by spending extra time planning are not overcome by a sufficient reduction in the number of turns, leading to fewer problems being solved. The easier planning problems posed in the other domains considered require less time to optimise, as in this case the difficulty of the induced deterministic planning problem increases the time taken to optimise the problem increases more steeply.

Analysing the obtained results it can be seen that:

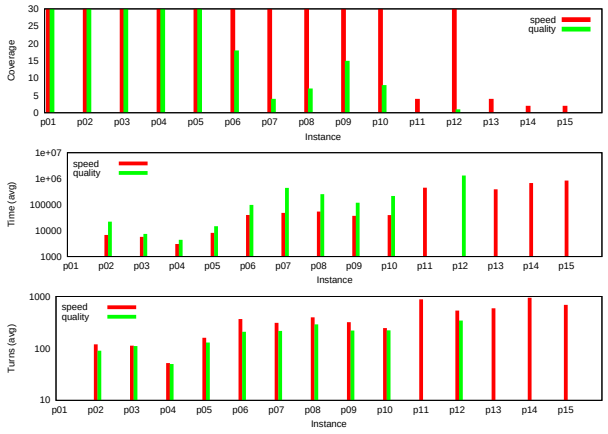

Figure 9: Results obtained running both versions of the planner in the *Zeno Travel* domain

- The number of problems solved by the planner is generally improved when the probabilities are taken into account.
- On problems solvable without optimising to improve the chance of the plan succeeding, the overall time required to solve problems is still generally lower if the planner does not reason about probabilities during planning; instead finding a potentially poor solution very quickly. In these cases, the cost of optimisation is not outweighted by the improved plan quality. This is also affected by the fact that much of planning research is concentrated on satisfycing planners, and optimisation of plans with respect to a given metric is not as widely researched [3].
- Reasoning about probabilities generally decreases the number of turns required to solve problems. In minimising the risk that a plan will fail the planner is able to generate a plan that is less likely to fail, or at least that will fail fewer times.

Related Work

The problem of probabilistic planning has been widely researched. In literature several different techniques to solve these problems are documented; however, these techniques can be grouped in three main approaches:

1. Reinforcement Learning (Kaelbling, Littman, & More 1996). Reinforcement Learning techniques allow systems to solve problems in worlds where the domain dynamics are not completely known. This technique attempts to learn optimal policies to achieve particular goals by trying actions and observing the results.
2. Solving Markov Decision Processes (Boutilier, Dean, & Hanks 1999). These systems transform probabilistic planning domains into non deterministic state-transition systems with probabilities assigned to transitions. The plan-

[3] Although optimal planning does attract research interest most optimal planners optimise a fixed, author chosen, metric—usually makespan

ning goals are seen as utility functions and solving probabilistic planning problems becomes the task of finding a policy that maximises this utility functions.

3. Extending classical planning algorithms. The work presented in this paper lies in this category. The first attempts to face probabilistic planning problems followed this approach, most were extensions to partial order planning (Onder & Pollack 1999). In recent years this line of work has been resurrected through the development of planners based on the Graphplan framework. An example is the probabilistic planner PARAGRAPH (Little & Thiébaux 2006), which participated in IPC5.

When deciding on the best approach to use in solving a given probabilistic planning problem several facts must be taken into account:

- The initial knowledge of the world. Both, planners based on solving MDPs and planners based on extending classical planning algorithms assume an initial perfect knowledge of the domain theory. They suppose that all the potential action effects and the probabilities associated with them are known 'a priori'. This is a very restrictive assumption as it is not true in most realistic domains.
- The need for a critical speed response. Planners using replanning can have response times that are too long for certain situations. Finding a new plan to overcome an unexpected state can take too long in some domains. Planners based on finding policies do not encounter this problem.
- The reversibility of the actions effects. When the effects caused by actions executions are easily reversible a simple approach, as FF-REPLAN, based on using a deterministic planner together with replanning techniques can be sufficient.
- Changeability of goals. Approaches based on finding policies can not deal with changeable goals domains. In these cases systems based on replanning or plan repairing are more suitable.
- Need for generalisation and transference of the solutions. Solutions expressed as policies to find particular goals are not easily generalised to solve similar problems. Further, the knowledge acquired solving one problem is not easy to transfer to other applications or communicate to humans.

Conclusions and Future Work

In this paper we have presented a new approach to solve probabilistic planning problems with a deterministic standard numeric planner. This approach, although based on a deterministic planner, does not discard the non-nominal action effects and the probability values of the actions effects so it is able to get good performance in domains where the actions effects are not easily reversible.

According to the results of the Probabilistic Planning Competitions IPC4 and IPC5, the paradigm based on combining deterministic planning and replanning seems to be the best approach to solve probabilistic planning problems. Indeed our planner outperforms all other planners in the competition that take action probabilities into account. This statement is, however, not generally true for all domains as this approach outperformed the other competitors mainly because the kind of test benches used to evaluate the planners. In the competition test benches, the world dynamics of all the domains is 'a priori' completely specified, there is no need for a quick speed response and in most of the used domains the effects of actions are reversible.

In the future we plan to test our approach with different definitions of the actions cost model. We also plan to study how metrics can be combined with our cost model to get good performance in planning domains that involve maximising any other feature besides robustness, such as `fuel` in the *Zeno Travel* domain.

One weakness of our approach is that because replanning is used to handle unexpected states, the response speed can be slow. Replanning means that our planner requires more time to take some decisions than systems based on policies. We would like to integrate plan repairing techniques in our system to reduce the number times the planner must replan.

When developing real planning applications, having 'a priori' a perfect description of the dynamics of a probabilistic domain is unusual. Another important line of future work is dealing with planning domains with incomplete action model. We plan to integrate relational learning algorithms with this paradigm of probabilistic planner and learn the likely outcome of actions from executions models.

References

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11:1–94.

Gerevini, A., and Serina, I. 2002. LPG: a planner based on local search for planning graphs. *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'02), AAAI Press, Toulouse, France*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Kaelbling, L. P.; Littman, M.; and More, A. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Little, I., and Thiébaux, S. 2006. Concurrent probabilistic planner in the GraphPlan framework. *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS '06), The English Lake District, Cumbria, UK*.

Onder, N., and Pollack, M. E. 1999. Conditional, probabilistic planning: A unifying algorithm and effective search control mechanisms. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*.

Younes, H.; Littman, M. L.; Weissman, D.; and Asmuth, J. 2005. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research* 24:851–887.