Problem Solving Using Classical Planners
AAAI Technical Report WS-12-12

Using Classical Planners to Solve Conformant Probabilistic Planning Problems

Ran Taig and **Ronen I. Brafman**
Computer Science Department
Ben Gurion University of The Negev
Beer-Sheva, Israel 84105
taig,brafman@cs.bgu.ac.il

Abstract

Motivated by the success of the translation-based approach for conformant planning, introduced by Palacios and Geffner, we present two variants of a new compilation scheme from conformant probabilistic planning problems (CPP) to variants of classical planning. In CPP, we are given a set of actions – which we assume to be deterministic in this paper, a distribution over initial states, a goal condition, and a value $0 < p \leq 1$. Our task is to find a plan π such that the goal probability following the execution of π in the initial state is at least p. Our first variant translates CPP into classical planning with resource constraints, in which the resource represents probabilities of failure. The second variant translates CPP into cost-optimal classical planning problems, in which costs represents probabilities. Empirically, these techniques show mixed results, performing very well on some domains, and poorly on others. This indicates that compilation-based technique are a feasible and promising direction for solving CPP problems and, possibly, more general probabilistic planning problems.

Introduction

An important trend in research on planning under uncertainty is the emergence of planners that utilize an underlying classical, deterministic planner to solve more complex problems. Two highly influential examples are the replanning approach (Yoon, Fern, and Givan 2007) in which an underlying classical planner is used to solve MDPs by repeatedly generating plans for a determinized version of the domain, and the translation-based approach for conformant (Palacios and Geffner 2009) and contingent planning (Albore, Palacios, and Geffner 2009), where a problem featuring uncertainty about the initial state is transformed into a classical problem on a richer domain. Both approaches have drawbacks: replanning can yield bad results given dead-ends and low-valued, less likely states. The translation-based approach can blow-up in size given complex initial belief states and actions. In both cases, however, there are efforts to improve these methods, and the reliance on fast, off-the-shelf, classical planners seem to be very useful.

This paper continues this trend, leveraging the translation-based approach of (Palacios and Geffner 2009) to handle a quantitative version of conformant planning, in which there is a probability distribution over the initial state of the world, although actions remain deterministic. The task now is to attain the goal condition with certain probability, rather than with certainty. More generally, conformant probabilistic planning (CPP) allows for stochastic actions, but as in earlier work, we focus on the simpler case of deterministic actions. Unlike our earlier work (Brafman and Taig 2011), which reduces CPP into a Metric Planning problem, we offer two closely related reductions of CPP into more common variants of classical planning.

The key to understanding our translation is the observation that a successful solution plan for a deterministic CPP problem is a conformant plan w.r.t. a (probabilistically) sufficiently large set of initial states. Hence, a possible solution method would be to ”guess” this subset, and then solve the conformant planning problems it induces. Both our methods essentially generate a conformant planning problem with additional operators that allow the planner to select the set of states on which it will ”plan”. Without additional constraints, the planner can simply guess an empty initial state, or a singleton, and plan for it. We offer two alternative methods for enforcing these constraints. The first method uses resources constraints. In this method, the planner is provided with some ”probabilistic” resource it can spend. If θ is the desired probability of success, then the planner has $1 - \theta$ units of ”probability” to spend. The planner can spend this resource on ignoring initial states. It pays $Pr(s)$ to ignore a state s. An optimal plan for this problem would thus be the shortest plan that satisfies the resource constraints and attains the goal.

A second variant of this translation scheme reduces the problem to cost-optimal planning. As in the resource-constrained planning reduction, we add actions that tell the planner to ”ignore” an initial state. An action that ignores state s costs $Pr(s)$; all other actions have zero (or ϵ cost). Consequently, the cost-optimal plan is a plan with the highest probability of success. Thus, while the first method attempts to find a good, valid plan quickly, the second method attempts to find the most probable plan, possibly sacrificing planning time.

Semantically, both methods described above map CPP into resource-constrained or cost-optimal conformant planning. Practically, this is done utilizing the $K_{T,M}$ translation from conformant to classical planning (Palacios and Geffner

2009).

To assess our proposed translation schemes, we conduct a preliminary comparison between them, *PFF* (Domshlak and Hoffmann 2007) and *PTP* (Brafman and Taig 2011), which are the state of the art in CPP. The results are mixed, showing no clear winner. We believe they indicate that the transformation approach is a promising technique for solving probabilistic planning problems, worthy of additional attention.

Background

Conformant Probabilistic Planning

The probabilistic planning framework we consider adds probabilistic uncertainty to a subset of the classical ADL language, namely (sequential) STRIPS with conditional effects. Such STRIPS planning tasks are described over a set of propositions $\mathcal{P}$ as triples (A, I, G), corresponding to the *action set*, *initial world state*, and *goals*. I and G are sets of propositions, where I describes a concrete initial state w_I, while G describes the set of goal states $w \supseteq G$. An action a is a pair $(pre(a), E(a))$ of the *precondition* and the *(conditional) effects*. A conditional effect e is a triple $(con(e), add(e), del(e))$ of (possibly empty) proposition sets, corresponding to the effect's *condition*, *add*, and *delete* lists, respectively. The precondition $pre(a)$ is also a proposition set, and an action a is *applicable* in a world state w if $w \supseteq pre(a)$. If a is not applicable in w, then the result of applying a to w is undefined. If a is applicable in w, then all conditional effects $e \in E(a)$ with $w \supseteq con(e)$ occur. Occurrence of a conditional effect e in w results in the world state $w \cup add(e) \setminus del(e)$, which we denote by $a(w)$. We will use $\overline{a}(w)$ to denote the state resulting from the sequence of actions $\overline{a}$ in world state w.

If an action a is applied to w, and there is a proposition q such that $q \in add(e) \cap del(e')$ for (possibly the same) occurring $e, e' \in E(a)$, the result of applying a in w is undefined. Thus, actions cannot be self-contradictory, that is, for each $a \in A$, and every $e, e' \in E(a)$, if there exists a world state $w \supseteq con(e) \cup con(e')$, then $add(e) \cap del(e') = \emptyset$. Finally, an action sequence $\overline{a}$ is a *plan* if the world state that results from iterative execution of $\overline{a}(w_I) \supseteq G$.

Our probabilistic planning setting extends the above with probabilistic uncertainty about the initial state. In its most general form, CPP covers stochastic actions as well, but we leave this to future work. Conformant probabilistic planning tasks are 5-tuples (V, A, b_I, G, θ), corresponding to the *propositions set*, *action set*, *initial belief state*, *goals*, and *acceptable goal satisfaction probability*. As before, G is a set of propositions. The initial state is no longer assumed to be known precisely. Instead, we are given a probability distribution over the world states, b_I, where $b_I(w)$ describes the likelihood of w being the initial world state.

There is no change in the definition of actions and their applications in states of the world. But since we now work with belief states, actions can also be viewed as transforming one belief state to another. The likelihood $[b, a]\,(w')$ of a world state w' in the belief state $[b, a]$, resulting from applying action a in belief state b, is given by $[b, a]\,(w') = \sum_{a(w)=w'} b(w)$.

We will also use the notation $[b, a]\,(\varphi)$ to denote $\sum_{a(w)=w', w' \models \varphi} b(w)$, and we somewhat abuse notation and write $[b, a] \models \varphi$ for the case where $[b, a]\,(\varphi) = 1$.

For any action sequence $\overline{a} \in A^*$, and any belief state b, the new belief state $[b, \overline{a}]$ resulting from applying $\overline{a}$ at b is given by $[b, \overline{a}] = \begin{cases} b, & \overline{a} = \langle\rangle \\ [b, a]\,, & \overline{a} = \langle a \rangle, a \in A \\ [[b, a]\,, \overline{a}']\,, & \overline{a} = \langle a \rangle \cdot \overline{a}', a \in A, \overline{a}' \neq \emptyset \end{cases}$..

In such setting, achieving G with certainty is typically unrealistic. Hence, θ specifies the required *lower bound* on the probability of achieving G. A sequence of actions $\overline{a}$ is called a *plan* if we have $b_{\overline{a}}(G) \geq \theta$ for the belief state $b_{\overline{a}} = [b_I, \overline{a}]$. Because our actions are deterministic, this is essentially saying that $\overline{a}$ is a *plan* if $Pr(\{w : \overline{a}(w) \models G\}) \geq \theta$, i.e,. the weight of the initial states from which the plan reaches the goal is at least θ.

Related Work

The best current probabilistic conformant planner is *Probabilistic FF* (PFF) (Domshlak and Hoffmann 2007). The basic ideas underlying Probabilistic-FF are:

1. Define time-stamped Bayesian Networks (BN) describing probabilistic belief states.
2. Extend Conformant-FF's belief state to model these BN.
3. In addition to the SAT reasoning used by Conformant-FF (Hoffmann and Brafman 2006), use weighted model-counting to determine whether the probability of the (unknown) goals in a belief state is high enough.
4. Introduce approximate probabilistic reasoning into Conformant-FF's heuristic function.

PFF results were partially improved by the *PTP* planner (Brafman and Taig 2011). Our work is close in spirit to *PTP* . *PTP* compiles CPP into a metric planning problem in which the numeric variables represent the probabilities of various propositions and actions update this information. For every variable p, they maintain a numeric variable Pr_p that holds the probability of p in the current state. They also maintain variables of the form p/t that capture conditional knowledge. If an action adds p/t, then the value of Pr_t is increased by the probability of t. Similar information about the probability of the goal is updated with a Pr_{goal} variable, and the goal in this metric planning problem is: $Pr_{goal} \geq \theta$. We compare our methods to theirs below. The main a-priori advantage of our approach is the reduction to planning problems that are more "classical." Cost-optimal planning has received much attention in recent years, and we hope to take advantage of improvements in this area to improve our ability to solve the CPP problem.

An earlier attempt to deal with probabilities by reducing it to action costs appears in (Jiménez et al. 2006) in the context of probabilistic planning problems where actions have probabilistic effects but there is no uncertainty about the initial state. The probabilistic problem is compiled into a classical problem where each possible effect e is now represented by a unique action and the cost associated with this action is set to be $1 - Pr_e$. That value captures the amount of risk the planner takes when choosing that action, which equals the probability the effect won't take place if the original action would have been executed. This value needs, of course, to be minimized by the cost-optimal planner. Our second compilation scheme uses related ideas but deals with uncertainty about the initial state, and comes with correctness guarantees.

Closely related to our work is the *CLG+* planner (Albore and Geffner 2009). This planner attempts to solve contingent planning problems in which goal achievement cannot be guaranteed. Thus, gradually, this planner makes assumptions that reduce the uncertainty, and allow it to plan. This is achieved by adding special actions, much like ours, that "drop" a tag, i.e., assume its value is impossible. These actions are associated with a high cost. The main difference, of course, with our planner is that the cost we associate with assumption-making actions reflects the probability of the states ruled out, allowing us to model probabilistic planning problems as cost-optimal planning. Furthermore, our planner may decide (depending on the search procedure used) to come up with a sub-optimal plan, albeit one that meets the desired probabilistic threshold, even when a full conformant plan exists. This flexibility allows us to trade-off computational efficiency with probability of success.

Resource constrained classical planning

Resource constrained planning is a well known extension of classical planning that models problems in which actions consume resources, such as time, energy, etc., and the agent must achieve the goal using some initial amount of resources. Here we follow the formulation of (Nakhost, Hoffmann, and Müller 2010) and (Haslum and Geffner 2001) where a constrained resource planning task extends a simple classical planning task with a set R of resource identifiers as well as two functions:

- $i : R \to \mathbb{R}_{\geq 0}$, i(r) is the initial level of resource $r \in R$.
- $u : (A \times R) \to \mathbb{R}_{\geq 0}$, for each action $a \in A$ and each resource $r \in R$, $u(a, r)$ is the amount of r consumed by an execution of a.

A state $\bar{s}$ is a pair (s, rem) where $rem \in {\mathbb{R}_{\geq 0}}^{|R|}$ holds the remaining amount of each resource when reaching s. To execute action a in $\bar{s}$, its preconditions must hold in s, and for every resource r, its value in rem must be at least as high as the amount of this resource consumed by a.

The Translation Approach

We present here a modified version of the translation-based method of (Palacios and Geffner 2009), adapted to our settings. The essential idea behind the translation approach to conformant planning implemented in the T_0 planner is to reason by cases. The different cases correspond to different conditions on the initial state, or, equivalently, different sets of initial states. These sets of states, or conditions, are captured by tags. That is, a tag is identified with a subset of b_I. Below we abuse notation often, treating a tag as the set of initial states it defines.

With every proposition p, we associate a set of tags T_p. We require that this set be *deterministic* and *complete*. We say that T_p is *deterministic* if for every $t \in T_p$ and any sequence of actions $\bar{a}$, the value of p is uniquely determined by t, the initial belief state b_I and $\bar{a}$. We say that T_p is *complete* w.r.t. an initial belief state b_I if $b_I \subseteq \bigcup_{t \in T_p} t$. That is, it covers all possible relevant cases. We say that a set of tags is *disjoint* when for every $t \neq t' \in T_p$ we have that $t \cap t' = \emptyset$. We say that a set of tags is ***DCD*** if it is deterministic, complete, and disjoint.

Once we compute the tags required for a proposition p, (see below) we augment the set of propositions with new propositions of the form p/t, where t is one of the possible tags for p. p/t holds the current value of p given that the initial state satisfies the condition t. The value of each proposition p/t is known initially – it reflects the value of p in the initial states represented by t, and since we focus on deterministic tags only, then $p/t \lor \neg p/t$ is a tautology throughout. Our notation p/t differs a bit from the Kp/t notation of Palacios and Geffner. The latter is used to stress the fact that these propositions are actually representing knowledge about the belief state. However, because of our assumption that tags are deterministic, we have that $\neg Kp \to K\neg p$. To stress this and remove the redundancy, we use a single proposition p/t instead of two propositions $Kp/t, K\neg p/t$.

The actions are transformed accordingly to maintain our state of knowledge. Given the manner tags were selected, we always know how an action would alter the value of some proposition given any of its tags. Thus, we augment the description of actions to reflect this. If the actions are deterministic (which we assume in this paper), then the change to our state of knowledge is also deterministic, and we can reflect it by altering the action description appropriately.

The resulting problem is a classical planning problem defined on a larger set of variables. The size of this set depends on the original set of variables and the number of tags we need to add. Hence, an efficient tag generation process is important. A trivial set of tags is one that contains one tag for each possible initial state. Clearly, if we know the initial state of the world, then we know the value of all variables following the execution of any set of actions. However, we can often do much better, as the value of each proposition at the current state depends only on a small number of propositions in the initial state. This allows us to use many fewer tags (=cases). In fact, the current value of different propositions depends on different aspects of the initial state. Thus, in practice, we select different tags for each proposition. We generate the tags for p by finding which literals are relevant to its value using the following recursive definition:
1.p is relevant to p.
2. If q appears (possibly negated) in an effect condition c for action A such that $c \to r$ and r contains p or $\neg p$ then q is relevant to p.
3.If r is relevant to q and q is relevant to p then r is relevant to p.

Let C_p denote the set containing all the propositions relevant to p. The set of tags consisting of one tag for every possible assignment to C_p is DCD. This set can be reduced farther, while remaining DCC, if we remove any tag that corresponds to an assignment to C_p which has probability 0 in the initial state.

New Translation Schemes for CPP

Let $P = (V, A, b_I, G, \theta)$ be the CPP given as input. Recall that T_p is the set of tags for p. We use T to denote the entire set of tags (i.e., $\cup T_p$). We will also assume a special distinguished tag, the empty set. We now present two compilation methods for P.

Variant 1: CPP as resource constrained classical planning (RCCP)

The basic motivation for this method is the understanding that we can solve the problem by identifying a set of initial states whose joint probability is greater or equal to θ, such that we have a conformant plan from this set of states to the goal. This plan is a solution to the CPP problem. We make the identification of this set of states part of the planning process. That is, the planner essentially decides which states to ignore. By default, the other states are the states it plans for. The joint probability of ignored states cannot exceed $1-\theta$, and the planner treats this probability as a resource it can spend. This resource is consumed by special actions that essentially tell the planner to ignore a state (or set of states). Such an action consumes as much resource as the probability of the states ignored. Technically, the ignored states make it easier for the planner to obtain knowledge. Typically, we say that the agent knows φ at a certain belief state, if φ holds in all world states in this belief state. In the compilation approach such knowledge is added by applying ”inference” actions – actions that do not change the state of the world, but rather deduce new information from existing information – known as *merge* actions. Once a state has been ”ignored” by an ”ignore” action, the merge actions effectively ignore it, and deduce the information as if this state is not possible.

The Translation Scheme. Given P, we generate the following resource constrained planning problem $\widetilde{P} = (\widetilde{V}, \widetilde{A}, \widetilde{I}, \widetilde{G}, \widetilde{R}, i, u)$:

Variables: $\widetilde{V} = \{p/t \mid p \in V, t \in T_p\} \cup \{DROP_t | t \in T\}$. The first set of variables are as explained above. By convention, we will use p to denote $p/\{\}$. These variables denote the fact that p holds unconditionally, i.e., in all possible worlds. The second set of variables – $Drop_t$ – denotes the fact that we can ignore tag t.

Initial State: $\widetilde{I} = \{l/t \mid l$ is a literal, and $t, I \vDash l\}$. All valid assumptions on the initial worlds captured by the special variables. Note that all $DROP_t$ propositions are false.

Resources: We define single resource so that $\widetilde{R} = \{\text{”}UISP\text{”}\}$ (for *unneeded initial states probability*).

Resource initialization: $i(UISP) = 1 - \theta$. The planner is limited by the unneeded states' joint probability.

Goal State : $\widetilde{G} = G$. The goal must hold for all initial states. Recall that what we call knowledge is not real knowledge, because we allow ourselves to overlook the ignored states.

Actions: $\widetilde{A} = \widetilde{A}_1 \cup \widetilde{A}_2 \cup \widetilde{A}_3 \cup \widetilde{A}_4$ where:

- $\widetilde{A}_1 = \{\widetilde{a} \mid a \in A\}$: Essentially, the original set of actions.
  - $pre(\widetilde{a}) = pre(a)$. That is, to apply an action, its preconditions must be known. Recalling that our notion of knowledge is modulo ignored states, this means that we allow for plan failure that stems from the execution of an action without a proper precondition. This is unlike previous CPP solvers.
  - For every conditional effect $(c \rightarrow p) \in E(a)$ and for every $t \in T$, $\widetilde{a}$ contains: $\{c/t \mid c \in con\} \rightarrow \{p/t\}$. That is, for every possible tag t, if the condition holds given t, so does the effect.
- $\widetilde{A}_2 = \{\{p/t \mid t \in T_p\} \rightarrow p$ |p is a precondition of some action $a \in A$ or $p \in G\}$. These are known as the *merge* actions. They allow us to infer from conditional knowledge about p, given certain sets of tags, absolute knowledge about p. That is, if p holds given t, for an appropriate set of tags, then p must hold everywhere.
- $\widetilde{A}_3 = \{Drop_t \mid t \in T\}$ where: $pre(Drop_t) = \{\}, eff(Drop_t) = \{Drop_t\}$. That is, the $Drop_t$ action let's us drop tag t, making $DROP_t$ true.
- $\widetilde{A}_4 = \{Assume_{p/t}$ |p is a precondition of some action $a \in A$ or $p \in G, t \in T_p\}$ $pre(Assume_{p/t}) = \{DROP_t\}, eff(Assume_{p/t}) = \{p/t\}$.

  That is, we can assume whatever we want about what is true given a dropped tag.
- For each action $Drop_t \in \widetilde{A}_3$, we set the resource function as follows: $u(UISP, Drop_t) = Pr_I(t)$. All other actions consume 0 resource.

  Thus, essentially, using the $DROP_t$ action, the planner decides to ”pay” some probability for dropping all initial states that correspond to this tag. Once it drops a tag, it can conclude whatever it wants given this tag. Thus, achieving the goal given a dropped tag is trivial. Because we limit the resource level to $1-\theta$, for all other states, whose weight is at least θ, the plan must truly achieve the goal.

  If tags are DCD, our methods are sound and complete. Soundness follows from the soundness result for conformant planning (Palacios and Geffner 2009). However, there is a subtle point about the semantics of success in CPP. In (Hoffmann and Brafman 2006) a successful plan was required to be applicable to all possible initial states, and successful on a subset with probability of θ or more. We believe that a more sensible definition requires that the plan be applicable and successful on a set state with probability θ, but agnostic as to what happens on other states. In principle, this gives us more flexibility, as any plan that works under the former criteria works under the latter. However, somewhat paradoxically, this requires more from the relevance relation used to compute the set of tags. Whereas previously, in Step 2 of the recursive relevance relation, we did not need to consider the preconditions of an action, but only its conditional effects (because preconditions had to hold with probability 1), now we need to add preconditions of relevant actions to the set of relevant propositions. This change can cause the width of a planning problem to increase.

Completeness is less immediate. Intuitively, if the granularity of the set of tags is too coarse, we may not be able to express the assumptions under which a plan exists. That is, either we can ignore too small a set of states, for which no plan exists, or we ignore too large a set of states, whose probability exceeds $1 - \theta$.

It turns out that DCD suffices for completeness. The proof is omitted due to space limitations, but the following example illustrates the intuition. Suppose our goal is $p \wedge q$. p is the only proposition relevant to p and q is the only proposition relevant to q. Furthermore, assume that initially, the probability of p is 0.5 and the probability of q is 0.5 and the required success probability if 0.75. Imagine that there exists a plan π that achieves the goal from all states except the state satisfying $p \wedge q$ (and thus, meets the success criterion). In our language, we only have tags of the form "drop p", "drop $\neg p$", "drop q", "drop $\neg q$". Any one of them "costs" 0.5, and thus cannot be used in a successful plan. We claim that from the relevance relation, it follows that π achieves the goal from all states. To see this, notice that, by assumption, π achieves p from $p \wedge \neg q$. But q is not relevant to p, and hence π must achieve p from $p \wedge q$, as well. Symmetrically, this holds for q as well. More generally, we have:

Lemma 1 *Let $\widetilde{\pi}$ be a valid plan that solves $\widetilde{P}$. The plan π obtained from $\widetilde{\pi}$ by dropping all actions of type $\widetilde{A}_2 \cup \widetilde{A}_3 \cup \widetilde{A}_4$ is a valid solution plan for P. (Soundness)*

Lemma 2 *For every plan π that solves P there exists a plan π^* that solves $\widetilde{P}$ such that π^* extends π with actions from $\widetilde{A}_2 \cup \widetilde{A}_3 \cup \widetilde{A}_4$ only. (completeness)*

Variant 2: CPP as cost-optimal Planning

CPP is a decision problem – find a plan with suitable success probability. Rather than solve this problem directly, we solve the optimization problem of finding a plan with the highest probability of success. Practically, this is often a harder problem, and in certain applications we may not care about finding such a solution because it may be much longer to generate, and possibly much longer to execute then an alternative, short solution that is not optimal. Nevertheless, much effort has been invested in cost-optimal planning, and we hope to leverage the improvements in this area.

The underlying ideas are essentially the same as the previous method, except that we attempt to find a plan that uses a minimal amount of the resource. Since there is a single resource, we can simply model the resource consuming actions as action with a cost that equals their resource consumption. Actions that do not consume resource have cost 0.[1] We use $\widehat{P}$ to denote the optimal-cost planning problem induced from P using this method. A solution to $\hat{P}$ will make the least costly assumptions possible, and hence will work on the (probabilistically) largest set of initial states.

[1]In practice, we set the cost of all other actions to some very small ϵ (0.001) because we found cost-optimal planner have difficulty handling 0-cost actions.

The following results hold for this translation method, again, under the assumption of deterministic, complete, and disjoint set of tags.

Lemma 3 *Let $\hat{\pi}$ be a valid plan that solves $\widehat{P}$. The plan π obtained from $\hat{\pi}$ by dropping all actions of type $\widetilde{A}_2 \cup \widetilde{A}_3 \cup \widetilde{A}_4$ is the solution plan for P with the largest probability of success.*

Lemma 4 *Let π be a solution plan for P. Then, the solution to $\widehat{P}$ achieves G with probability of at least θ (After dropping all non original actions).*

Example

We illustrate the ideas behind our first method using an example adapted from (Palacios and Geffner 2009) and (Brafman and Taig 2011). In this problem we need to move an object from the origin to a destination on a linear grid of size 4. There are two actions: $pick(l)$ picks an object from location l if the hand is empty and the object is in l. If the hand is full, it drops the object being held in l. $put(l)$ drops the object at l if the object is being held. All effects are conditional, and there are no preconditions. Formally, the actions are as follows:

$\mathbf{pick(l)}$: $\neg hold, at(l) \rightarrow hold \wedge \neg at(l), hold \rightarrow \neg hold \wedge at(l)$.

$\mathbf{put(l)}$: $hold \rightarrow \neg hold \wedge at(l)$

Consider an instance P of this domain in which the hand is initially empty with certainty, and the object is initially at either l_1 or l_2 or l_3, and it needs to be moved to l_4 with a probability of 0.5. More specifically:

$I = \{Pr[\neg hold] = 1, Pr[at(l_1)] = 0.2, Pr[at(l_2)] = 0.4, Pr[at(l_3)] = 0.4, Pr[at(l_4)] = 0\}, G = \{Pr[at(l_4)] \geq 0.5\}$.

It is easy to see that the goal can be achieved by considering two possible original object locations only, unlike in conformant planning where we must succeed for all three possible initial locations. The tags : $T_L = \{at(l_1), at(l_2), at(l_3)\}$ for $L \in \{hold, at(l_4)\}$. Note that T_L is indeed disjoint, deterministic, and complete for L. Based on these tags our algorithm outputs the following resource constrained planning task (Method 1):

$\hat{P} = \{\hat{V}, \hat{R}, \hat{A}, \hat{I}, \hat{G}, i, u\}$ as follows:

$\hat{V} = \{L/t \mid L \in \{at(l), hold\}, t \in \{l_1, l_2, l_3\}\} \cup \{DROP_t \mid t \in \{at(l_1), at(l_2), at(l_3)\}\}$.

$\hat{I} = \{at(l)/at(l) \mid l \in \{l_1, l_2, l_3\}\}$. $\hat{R} = \{r\}$ and $i(r) = 0.5$ – the complementary probability of Θ. We now show an example of the modified actions on the first conditional effect in $pick(l)$:

Original: $\neg hold, at(l) \rightarrow hold$

Modification: For each $l' \in \{l_1, l_2, l_3,$empty tag$\}$ we add the following conditional effects to the modified action: $\neg hold/at(l'), at(l)/at(l') \rightarrow hold \wedge \neg at(l), hold/at(l')$;

In addition we add (examples):

Merge action :$at(l_4)/at(l_1) \wedge at(l_4)/at(l_2) \wedge at(l_4)/at(l_3) \rightarrow at(l_4)$.

$Drop_{at_{l_3}}$: No pre-conditions, adds the variable $DROP_{at_{l_3}}$,$u(Drop_{at_{l_3}}, r) = 0.4$.

$ASSUME_{at_{l_4}/at_{l_3}}$: Single precondition: $DROP_{at_{l_3}}$,

		$\theta = 0.25$					$\theta = 0.5$					$\theta = 0.75$					$\theta = 1.0$				
Instance	#actions/#facts/#states	t/l					t/l					t/l					t/l				
		PFF	PTP	PRP	PFF(m)	PACP(m)	PFF	PTP	PRP	PFF(m)	PACP(m)	PFF	PTP	PRP	PFF(m)	PACP(m)	PFF	PTP	PRP	PFF(m)	PACP(m)
Safe-uni-70	70/71/140	2.65 /18	0.87/18	0.55/70	4.88/18	0.41/18	5.81/35	0.85/35	0.55/70	32.07/35	0.51/35	10.1/53	0.9/53	0.59/70	69.12/53	0.59/53	5.1/70	0.88/70	0.65/70	5.1/70	0.65/70
Safe-cub-70	70/70/138	0.88/5	0.9/5	0.55/70	1.05/5	0.72/5	1.7/12	0.94/12	0.55/70	2.62/12	0.93/12	3.24/21	0.95/21	0.55/70	6.71/21	1.19/21	4.80/70	0.96/70	0.55/70	4.80/70	0.55/70
Cube-uni-15	6/90/3375	4.25/26	2.4/33	0.8/30	11.21/27	0.3/26	6.35/34	2.49/45	0.84/34	11.44/34	0.37/34	9.20/38	2.65/50	0.7/38	15.25/38	0.41/38	31.2/42	2.65/50	0.73/42	31.2/42	0.73/42
Cube-cub-11	6/90/3375	0.3/5	1.17/12	0.61/7	7.2/14	0.94/14	0.9/9	1.31/15	0.7/14	11.32/26	1.05 /26	1.43/13	1.41/21	0.7/15	13.44/34	1.27/34	28.07/31	3.65 /36	1.1/30	28.07/31	1.1/30
Bomb-50-50	2550/200/$> 2^{100}$	0.01/0	0.01/0	0.1/0	0.1/0	0.1/0	0.10/16	3.51/50	36.1/50	–	–	0.25/36	3.51/50	36.1/50	–	–	0.14/51/50	3.51/50	36.1/50	–	–
Bomb-50-10	510/120/$> 2^{60}$	0.01/0	0.01/0	0.1/0	0.1/0	0.1/0	0.89/22	1.41/90	2.05/66	–	–	4.04/62	1.41/90	2.8/42	–	–	1.74/90	1.46/90	1.45/90	–	–
Bomb-50-5	255/110/$> 2^{55}$	0.01/0	0.01/0	0.1/0	0.1/0	0.1/0	1.70/27	1.32/95	1.35/47	–	–	4.80/67	1.32/95	1.15/71	–	–	2.17/95	1.32/95	0.9/95	–	–
Bomb-50-1	51/102/$> 2^{51}$	0.01/0	0.01/0	0.1/0	0.1/0	0.1/0	2.12/31	0.64/99	0.85/50	–	–	6.19/71	0.64/99	0.7/74	–	–	2.58/99	0.64/99	0.7/99	–	–
Log-2	3440/1040/$> 20^{10}$	0.90/54	–	–	–	–	1.07/62	–	–	–	–	1.69/69	–	–	–	–	1.84/78	–	–	–	–
Log-3	3690/1260 /$> 30^{10}$	2.85/64	–	–	–	–	8.80/98	–	–	–	–	4.60/99	–	–	–	–	4.14/105	–	–	–	–
Log-4	3960/1480/$> 40^{10}$	2.46/75	–	–	–	–	8.77/81	–	–	–	–	6.20/95	–	–	–	–	8.26/107	–	–	–	–

Table 1: Empirical results. t: time in seconds. l: plan length. *PFF*(m) and *PACP*(m): performance on modified problems (see text).

:

Single effect:at_{l_4}/at_{l_3}. Possible solution plan is: $\langle$ pick(l_1),put(l_4),pick(l_2),put(l_4),$Drop_{at_{l_3}}$, $ASSUME_{at_{l_4}/at_{l_3}}$, $MERGE_{at(l_4)}\rangle$. Dropping all the inference and assumption actions, we get a plan for the original CPP.

Empirical Evaluation

We implemented the algorithms as follows. Our input problem is stripped of probabilistic information and transformed into a conformant planning problem. This is fed to the *cf2cs* program, which is a part of the *T-0* planner, and computes the set of tags. Using this set of tags, we generate the new compiled problem. Currently, we have a somewhat inefficient tool for generating the new domains, which actually uses part of the *T-0*'s domain generation code and another tool we implemented that creates the described compilations. The compiled problem is then fed into Metric-FF. The output is then returned after we removed all special actions we added. For our first variant, Metric-FF is used as a a resource-constrained planner. Resources are modeled as numeric variables and suitable pre-conditions are added to actions which consume resources to make sure no resource is consumed behind his initial amount. We refer to the resulting planner as *PRP*. For the second variant *Metric-FF* is used as a cost optimal planner. *Metric-FF* and the other *FF* extensions are currently the only efficient classical planners that support and function well with a large amount of conditional effects. The major drawback of using *Metric-FF* is that it is not an optimal planner so, our implementation does not really return a minimal cost solution, and hence is not complete. Unfortunately, FD (Helmert 2006) fails to even pass the translation to SAS phase, apparently due to the large number of conditional effects. Moreover, none of its more competitive admissible heuristics supports conditional effects. We refer the resulting second planner *PACP*. Table 1 shows the results of our experimental evaluation.

On the bomb, cube, and safe domains, *PRP* works as good or better than *PFF*, with few exceptions, such as bomb-50-50 and bomb-50-10 and cube-11 for lower values of θ. On logistics, the translation method fail completely, unable to solve the classical planning problem. On some of these domains, a full conformant plan is shorter (because ignoring each subgoal would require making assumptions, each of which requires two actions, whereas the subgoal can be achieved with a single "real" action.) For this reason, *PRP* prefers the fully conformant plan.

PACP finds the most probable solution. Since all domains tested by *PFF* (with deterministic actions) have conformant plans, we also created modified versions of these domains that have no conformant solution by removing certain actions, yet ensuring that the problem is solvable with probability $\sim \Theta$ (according to the specific experiment). To make the comparison with *PFF* fair, we also give results for the performance of *PFF* on the modified versions. The results for both on *cube* and *safe* shows that *PACP* dominates *PFF* by an order of magnitude, as it seems to have more difficulty when problems have no conformant solutions.

In logistics, PACP fails because the classical planner returns a highly suboptimal plan in which the goal is solved by ignoring all the possible worlds. Whether a cost-optimal planner would solve this problem, or simply fail to return an answer remains to be seen. For *PRP*, the problem appears to be the heuristic guidance provided by *Metric-FF* whose relaxation-based heuristic effectively ignores resource constraints. Finally, two additional domains that we have not tested our planners on are *rovers* and *grid*. These domains have high conformant width, implying that a complete translation scheme would yield very large problem instances. *T-0* is able to deal with these domains by using various simplifications, and integrating these simplifications with our numeric techniques remains an important challenge.

Summary

We described two new, closely related translation schemes for CPP that build upon the techniques of (Palacios and Geffner 2009) and their extension to CPP (Brafman and Taig 2011). Our planners perform well on some domains, whereas in others they face fundamental problems in using the underlying planners. We believe that improvements in resource-constrained classical planning, together with better translation techniques may make the compilation approach a viable method for solving probabilistic planning problems.

Acknowledgments

The authors were partly supported by ISF Grant 1101/07, the Paul Ivanier Center for Robotics Research and Production Management, and the Lynn and William Frankel Center for Computer Science.

References

Albore, A., and Geffner, H. 2009. Acting in partially observable environments when achievement of the goal cannot be guaranteed. In *ICAPS'09 Planning and Plan Execution for Real-World Systems Workshop*.

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *IJCAI*, 1623–1628.

Brafman, R. I., and Taig, R. 2011. A translation based approach to probabilistic conformant planning. In *ADT*.

Domshlak, C., and Hoffmann, J. 2007. Probabilistic planning via heuristic forward search and weighted model counting. *J. Artif. Intell. Res. (JAIR)* 30:565–620.

Haslum, P., and Geffner, H. 2001. Heuristic planning with time and resources. *Artificial Intelligence* 129(1-2):5–33.

Helmert, M. 2006. The fast downward planning system. *JAIR* 26:191–246.

Hoffmann, J., and Brafman, R. I. 2006. Conformant planning via heuristic forward search: A new approach. *Artif. Intell.* 170(6-7):507–541.

Jiménez, S.; Coles, A.; Smith, A.; and Madrid, I. 2006. Planning in probabilistic domains using a deterministic numeric planner. In *The 25th PlanSig WS*.

Nakhost, H.; Hoffmann, J.; and Müller, M. 2010. Improving local search for resource-constrained planning. In *SOCS*.

Palacios, H., and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *JAIR* 35:623–675.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. Ff-replan: A baseline for probabilistic planning. In *ICAPS*, 352–.