

# Fast Probabilistic Planning Through Weighted Model Counting

**Carmel Domshlak**

Technion - Israel Institute of Technology  
Haifa, Israel

**Jörg Hoffmann**

Max Planck Institute for Computer Science  
Saarbrücken, Germany

## Abstract

We present a new algorithm for probabilistic planning with no observability. Our algorithm, called Probabilistic-FF, extends the heuristic forward-search machinery of Conformant-FF to problems with probabilistic uncertainty about both the initial state and action effects. Specifically, Probabilistic-FF combines Conformant-FF's techniques with a powerful machinery for weighted model counting in (weighted) CNFs, serving to elegantly define both the search space and the heuristic function. Our evaluation of Probabilistic-FF on several probabilistic domains shows an unprecedented, several orders of magnitude improvement over previous results in this area.

## Introduction

In this paper we address the problem of *probabilistic planning with no observability* (Kushmerick *et al.* 1995), also known in the AI planning community as conditional (Majercik & Littman 2003) or conformant (Hyafil & Bacchus 2004) probabilistic planning. In such problems we are given an initial *belief state* in the form of a probability distribution over the world states  $W$ , a set of actions having (possibly) probabilistic effects, and a set of alternative goal states  $W_G \subseteq W$ . A solution to such a problem is a *single* sequence of actions that transforms the system into one of the goal states with probability exceeding a given threshold  $\theta$ . The basic assumption of the problem is that the system cannot be observed at the time of plan execution. Such a setting is useful in controlling systems with uncertain initial state and non-deterministic actions, if sensing is expensive or unreliable. Non-probabilistic conformant planning may fail due to non-existence of a plan that achieves the goals *with 100% certainty*. Even if there is such a plan, that plan does not necessarily contain information about what actions are most useful to achieve (only) *the requested threshold*  $\theta$ .

The two most prominent steps made in the direction of conformant probabilistic planning are the probabilistic extensions of partial-order planning (Kushmerick *et al.* 1995), and to fixed-length planning with problem reformulation into either probabilistic SAT (Majercik & Littman 1998) or probabilistic CSP (Hyafil & Bacchus 2004). The state-of-the-art performance of probabilistic planners has been ad-

vancing much more slowly than that of deterministic planners, scaling from 5-10 step plans for problems with  $\approx 20$  world states to 15-20 step plans for problems with  $\approx 100$  world states. Since probabilistic planning is inherently harder than its deterministic counterpart (Littman, Goldsmith, & Mundhenk 1998), such a difference in evolution rates is not surprising. Yet, here we show that dramatic improvements in probabilistic planning *can* be obtained.

We introduce Probabilistic-FF, a new probabilistic planner based on *heuristic forward search* in the space of *implicitly represented* probabilistic belief states. The planner is based on combining the idea of lazy CNF-based belief state representation introduced in the (non-probabilistic) conformant planner Conformant-FF (Brafman & Hoffmann 2004) with recent techniques for probabilistic reasoning using weighted model counting (WMC) in propositional CNFs (Sang, Beame, & Kautz 2005). This synergetic combination allows us to elegantly extend both Conformant-FF's search space and heuristic function to the probabilistic setting.

The algorithms we present cover probabilistic initial belief states given as Bayes networks, deterministic and probabilistic actions, conditional effects, and standard action preconditions. By the time of submission of this paper, our ongoing implementation supports probabilistic belief states, as well as deterministic actions with preconditions and conditional effects. Implementing the presented techniques for probabilistic effects will be our next step. So far, we can offer stunning results in domains with complex probabilistic initial states: in contrast to the SAT and CSP based approaches mentioned above, Probabilistic-FF can find *100-step plans for instances with billions of world states*. This success is based on consequent exploitation of problem structure, through our belief state representation and heuristic function – both of which are heavily based on the synergy between WMC and Conformant-FF's CNF-based techniques. In fact, while without probabilities one could imagine to successfully replace the CNFs with BDDs, *with probabilities*, this seems hopeless.

We detail our planning model, then in turn describe the belief state representation by Bayes networks, the encoding of Bayes networks as weighted CNFs, and the heuristic function; we give our empirical results, and conclude.

## Problem Modeling

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  $\mathcal{X}$  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  $w \supseteq G$ . Actions  $a$  are pairs  $(pre(a), E(a))$  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  $w - del(e) + add(e)$ .

Our probabilistic planning setting extends the above with (i) probabilistic uncertainty about the initial state, and (ii) an additional type of action having probabilistic effects.<sup>1</sup> In general, probabilistic planning tasks are quadruples  $(A, P_I, G, \theta)$ , corresponding to the *action set*, *initial belief state*, *goals*, and *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,  $P_I$ , where  $P_I(w)$  describes the likelihood of  $w$  being the initial world state. The action set  $A$  consists of two different types of actions.  $A_d \subseteq A$  is a set of *deterministic actions* that have precisely the same structure and semantics as the actions in STRIPS with conditional effects.  $A_p \subseteq A$  is a set of *probabilistic actions*. Similarly to  $A_d$ , actions  $a \in A_p$  are pairs  $(pre(a), E(a))$ , but the effect set  $E(a)$  for such  $a$  has different structure and semantics. Each *probabilistic effect*  $e \in E(a)$  is a pair  $(con(e), \Lambda(e))$  of a propositional condition and a set of *probabilistic outcomes*. Each probabilistic outcome  $\varepsilon \in \Lambda(e)$  is a triplet  $(p(\varepsilon), add(\varepsilon), del(\varepsilon))$ , where *add* and *delete* lists are as before, and  $p(\varepsilon)$  is the probability that outcome  $\varepsilon$  occurs as a result of effect  $e$ . Naturally, we require that probabilistic effects define probability distributions over their outcomes, that is,  $\sum_{\varepsilon \in \Lambda(e)} p(\varepsilon) = 1$ . Without loss of generality, we assume that the conditions of probabilistic effects  $E(a)$  are mutually exclusive and exhaustive (Kushmerick, Hanks, & Weld 1995). Unconditional probabilistic actions are modeled as having a single probabilistic effect  $e$  with  $con(e) = \emptyset$ . As before, if  $a$  is not applicable in  $w$ , then the result of applying  $a$  to  $w$  is undefined. If  $a$  is applicable in  $w$ , then there exists exactly one effect  $e \in E(a)$  such that  $con(e) \subseteq w$ , and for each  $\varepsilon \in \Lambda(e)$ , applying  $a$  to  $w$  results in  $w + add(\varepsilon) - del(\varepsilon)$  with probability  $p(\varepsilon)$ .

Considering the initial belief state, practical considera-

<sup>1</sup>Our formalism covers the features of the previously proposed languages for conformant probabilistic planning (Kushmerick *et al.* 1995, Majercik & Littman 1998, Hyafil & Bacchus 2004).

tions force us to limit our attention only to compactly representable probability distributions  $P_I$ . While there are numerous alternatives for compact representation of structured probability distributions, Bayes networks (BNs) (Pearl 1988) to date is by far the most popular such representation model.<sup>2</sup> As excellent introductions to BNs abound (e.g., see (Jensen 1996)), it suffices to briefly define our notation. A BN  $\mathcal{N} = (\mathcal{G}, \mathcal{P})$  represents a probability distribution as a directed acyclic graph  $\mathcal{G}$ , where its set of nodes  $\mathcal{V}$  stands for random variables (assumed discrete in this paper), and  $\mathcal{P}$ , a set of tables of conditional probabilities (CPTs) - one table  $T_X$  for each node  $X \in \mathcal{V}$ . For each possible value  $x \in Dom(X)$  (where  $Dom(X)$  denotes the domain of  $X$ ), the table  $T_X$  lists the probability of the event  $X = x$  given each possible value assignment to all of its immediate ancestors (parents)  $Pa(X)$  in  $\mathcal{G}$ . Thus, the table size is exponential in the in-degree of  $X$ . Usually, it is assumed either that this in-degree is small (Pearl 1988) or that the probabilistic dependence of  $X$  on  $Pa(X)$  induces a significant local structure allowing a compact representation of  $T_X$  (Boutilier *et al.* 1996).

We assume that the initial belief state  $P_I$  is described by a BN  $\mathcal{N}_I$  over our set of propositions  $\mathcal{X}$ . In Probabilistic-FF we allow  $\mathcal{N}_I$  to be described over the true multi-valued variables underlying our problem. This significantly simplifies the process of specifying  $\mathcal{N}_I$  since the STRIPS propositions  $\mathcal{X}$  do not correspond to the true random variables underlying problem specification.<sup>3</sup> Specifically, let  $\bigcup_{i=1}^k \mathcal{X}_i$  be a partition of  $\mathcal{X}$  such that each proposition set  $\mathcal{X}_i$  uniquely corresponds to a multi-valued variable underlying our problem. That is, for every world state  $w$  and every  $\mathcal{X}_i$ , if  $|\mathcal{X}_i| > 1$ , then there is *exactly one* proposition  $q \in \mathcal{X}_i$  that holds in  $w$ . The variables of the BN  $\mathcal{N}_I$  describing our initial belief state  $P_I$  are  $\mathcal{V} = \{X_1, \dots, X_k\}$ , where  $Dom(X_i) = \mathcal{X}_i$  if  $|\mathcal{X}_i| > 1$ , and  $Dom(X_i) = \{q, \neg q\}$  if  $\mathcal{X}_i = \{q\}$ . It is not hard to see that the semantics of the actions  $a \in A$  can be specified in terms of  $\mathcal{V}$  in a straightforward manner. If  $|\mathcal{X}_i| > 1$ , then no action  $a$  can add a proposition  $q \in \mathcal{X}_i$  without deleting some other proposition  $q' \in \mathcal{X}_i$  (and vice versa). Thus, we can consider  $a$  as setting  $X_i = q$ . If  $|\mathcal{X}_i| = 1$ , then adding and deleting  $q \in \mathcal{X}_i$  has the standard semantics of setting  $X_i = q$  and  $X_i = \neg q$ , respectively.

Finally, since our actions transform (probabilistic) belief states to belief states, achieving  $G$  with certainty is typically unrealistic. Hence,  $\theta$  specifies the required *lower bound* on probability of achieving  $G$ . A sequence of actions  $\mathbf{a}$  is called a *plan* if we have  $P_{\mathbf{a}}(G) \geq \theta$  for the belief state  $P_{\mathbf{a}}$ , obtained by applying  $\mathbf{a}$  to the initial belief state  $P_I$ .

**Example 1** Say we have a robot and a block that physically can be at one of two locations. This information is captured by the propositions  $r_{L_1}, r_{L_2}$  for the robot, and

<sup>2</sup>While BNs are our choice here, our framework can support other models as well, e.g. stochastic decision trees.

<sup>3</sup>In short, specifying  $\mathcal{N}_I$  directly over  $\mathcal{X}$  would require identifying the multi-valued variables anyway, followed by connecting all the propositions corresponding to a multi-valued variable by a complete DAG, and then normalizing the CPTs of these propositions in a certain manner.

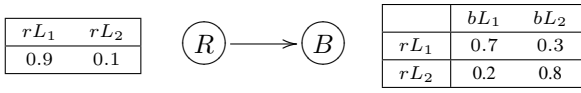


Figure 1: Bayes network  $\mathcal{N}_I$  for Example 1.

$bL_1, bL_2$  for the block, respectively. The robot can either move from one location to another, or do it while carrying the block. If the robot moves without the block, then its move is guaranteed to succeed. This provides us with a pair of (symmetrically defined) deterministic actions  $A_d = \{\text{move-right}, \text{move-left}\}$ . The action *move-right* has an empty precondition, and a single conditional effect  $e$  with  $\text{con}(e) = \{rL_1\}$ ,  $\text{add}(e) = \{rL_2\}$ , and  $\text{del}(e) = \{rL_1\}$ . If the robot tries to move while carrying the block, then this move succeeds with probability 0.7, while with probability 0.2 the robot ends up moving without the block, and with probability 0.1 this move of the robot fails completely. This provides us with a pair of (again, symmetrically defined) probabilistic actions  $A_p = \{\text{move-b-right}, \text{move-b-left}\}$ . The action *move-b-right* has an empty precondition, and two conditional effects specified as follows:

$E(a)$	$\text{con}(e)$	$\Lambda(e)$	$p(\varepsilon)$	$\text{add}(\varepsilon)$	$\text{del}(\varepsilon)$
$e$	$rL_1 \wedge bL_1$	$\varepsilon_1$	0.7	$\{rL_2, bL_2\}$	$\{rL_1, bL_1\}$
		$\varepsilon_2$	0.2	$\{rL_2\}$	$\{rL_1\}$
		$\varepsilon_3$	0.1	$\emptyset$	$\emptyset$
$e'$	$\neg rL_1 \vee \neg bL_1$	$\varepsilon'_1$	1.0	$\emptyset$	$\emptyset$

Now, say the robot is known to be initially at one of the two possible locations with probability  $P(rL_1) = 0.9$  and  $P(rL_2) = 0.1$ . Suppose there is a correlation in our belief about the initial locations of the robot and the block. We believe that, if the robot is at  $rL_1$ , then  $P(bL_1) = 0.7$  (and  $P(bL_2) = 0.3$ ), while if the robot is at  $rL_2$ , then  $P(bL_1) = 0.2$  (and  $P(bL_2) = 0.8$ ). The initial belief state BN is defined over two variables  $R$  (“robot”) and  $B$  (“block”) with  $\text{Dom}(R) = \{rL_1, rL_2\}$  and  $\text{Dom}(B) = \{bL_1, bL_2\}$ , respectively, and it is depicted in Figure 1.

## Belief States as Bayes Networks

Probabilistic-FF constitutes a forward search in belief space. The search states are belief states (that is, probability distributions over the world states  $w$ ), and the search is restricted to belief states reachable from the initial belief state  $P_I$  through some sequences of actions  $\mathbf{a}$ . A key decision one should make is the actual representation of the belief states. Let  $P_I$  be our initial belief state captured by  $\mathcal{N}_I$ , and let  $P_{\mathbf{a}}$  be a belief state resulting from applying to  $P_I$  a sequence of actions  $\mathbf{a}$ . One of the well-known problems in the area of decision-theoretic planning is that the description of  $P_{\mathbf{a}}$  in terms of only  $\mathcal{X}$  (that is,  $\mathcal{V}$ ) is getting less and less structured as the number of actions in  $\mathbf{a}$  increases. To overcome this limitation, we represent belief states  $P_{\mathbf{a}}$  as a BN  $\mathcal{N}_{\mathbf{a}}$  that explicitly captures sequential application of a starting from  $P_I$ . Below we formally specify the structure of such a BN  $\mathcal{N}_{\mathbf{a}}$ , assuming that all the actions  $\mathbf{a}$  are applicable in the corresponding belief states of their application (and later showing that Probabilistic-FF makes sure this is indeed the case.) Figure 2 illustrates this construction of  $\mathcal{N}_{\mathbf{a}}$  on our running example with  $\mathbf{a} = \langle \text{move-b-right}, \text{move-left} \rangle$ .

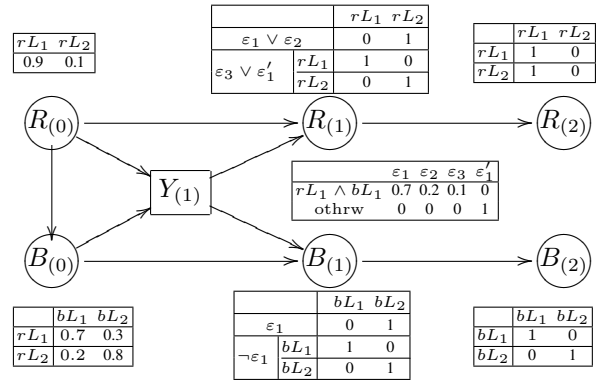


Figure 2: Bayes network  $\mathcal{N}_{\mathbf{a}}$  for our running example.

Let  $\mathbf{a} = \langle a^1, \dots, a^m \rangle$  be a sequence of actions, numbered according to their appearance on  $\mathbf{a}$ . For  $0 \leq t \leq m$ , let  $\mathcal{V}_{(t)}$  be a replica of our state variables  $\mathcal{V}$ , with  $X_{(t)} \in \mathcal{V}_{(t)}$  corresponding to  $X \in \mathcal{V}$ . The variable set of  $\mathcal{N}_{\mathbf{a}}$  is the union of  $\mathcal{V}_{(0)}, \dots, \mathcal{V}_{(m)}$ , plus some additional variables that we introduce for the probabilistic actions in  $\mathbf{a}$ .

First, for each  $X_{(0)} \in \mathcal{V}_{(0)}$ , we set the parents  $\text{Pa}(X_{(0)})$  and conditional probability tables  $T_{X_{(0)}}$  to simply copy these of  $X$  in  $\mathcal{N}_I$ . Now, suppose that  $a^t$  is a deterministic action  $a \in A_d$ , and let  $E_X(a) \subseteq E(a)$  be the conditional effects of  $a$  that add and/or delete propositions associated with the domain of a variable  $X \in \mathcal{V}$ . If  $E_X(a) = \emptyset$ , then we set  $\text{Pa}(X_{(t)}) = \{X_{(t-1)}\}$ , and

$$T_{X_{(t)}}(X_{(t)} = x \mid X_{(t-1)} = x') = \begin{cases} 1, & x = x', \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Otherwise, we set

$$\text{Pa}(X_{(t)}) = \{X_{(t-1)}\} \cup \bigcup_{e \in E_X(a)} \{X'_{(t-1)} \mid \text{con}(e) \cap \text{Dom}(X) \neq \emptyset\}, \quad (2)$$

and specify  $T_{X_{(t)}}$  as follows. Let  $x_e \in \text{Dom}(X)$  be the value that effect  $e \in E_X(a)$  provides to  $X$ . For each  $\pi \in \text{Dom}(\text{Pa}(X_{(t)}))$ , if there exists  $e \in E_X(a)$  such that  $\text{con}(e) \subseteq \pi$ , then we set

$$T_{X_{(t)}}(X_{(t)} = x \mid \pi) = \begin{cases} 1, & x = x_e, \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Otherwise, we set

$$T_{X_{(t)}}(X_{(t)} = x \mid \pi) = \begin{cases} 1, & x = \pi[X_{(t-1)}], \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

It is not hard to verify that Eqs. 3-4 are consistent, and, together with Eq. 1, capture the semantics of the conditional deterministic actions.

Alternatively, suppose that  $a^t$  is a probabilistic action  $a \in A_p$ . For each such action we introduce a discrete variable  $Y_{(t)}$  with  $\text{Dom}(Y_{(t)}) = \bigcup_{e \in E(a)} \Lambda(e)$ ,

$$\text{Pa}(Y_{(t)}) = \bigcup_{e \in E(a)} \{X_{(i-1)} \mid \text{con}(e) \cap \text{Dom}(X) \neq \emptyset\}, \quad (5)$$

and, for each  $\pi \in \text{Dom}(\text{Pa}(Y_{(t)}))$ , we set

$$T_{Y_{(t)}}(Y_{(t)} = \varepsilon \mid \pi) = \begin{cases} p(\varepsilon), & \text{con}(e(\varepsilon)) \subseteq \pi, \\ 0, & \text{otherwise} \end{cases}, \quad (6)$$

where  $e(\varepsilon)$  denotes the effect  $e$  of  $a$  such that  $\varepsilon \in \Lambda(e)$ . We refer to the set of all such variables created for  $a$  as  $\mathcal{Y}$ . Now, similarly to the case of deterministic actions, let  $E_X(a) \subseteq E(a)$  be the (now probabilistic) effects of  $a$  that affect a variable  $X \in \mathcal{V}$ . The case of  $E_X(a) = \emptyset$  is similar to this for deterministic actions, that is, we set  $Pa(X_{(t)}) = \{X_{(t-1)}\}$ , and set  $T_{X_{(t)}}$  according to Eq. 1. Alternatively, if  $E_X(a) \neq \emptyset$ , let  $x_\varepsilon \in \text{Dom}(X)$  be the value provided to  $X$  by  $\varepsilon$ ,  $e(\varepsilon) \in E_X(a)$ . Recall that the outcomes of effects  $E(a)$  are all mutually exclusive. Hence, we set  $Pa(X_{(t)}) = \{X_{(t-1)}, Y_{(t-1)}\}$ , and

$$T_{X_{(i)}}(X_{(i)} = x \mid X_{(i-1)} = x', Y_{(i-1)} = \varepsilon) = \begin{cases} 1, & e(\varepsilon) \in E_X(a) \wedge x = x_\varepsilon, \\ 1, & e(\varepsilon) \notin E_X(a) \wedge x = x', \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

Here as well, it is not hard to verify that Eqs. 6-7 capture the precise semantics of our probabilistic actions and frame axioms. This accomplishes our construction of  $\mathcal{N}_a$  over the variables  $\mathcal{V}_a = \mathcal{V} \cup_{t=0}^m \mathcal{V}_{(t)}$ .

**Theorem 1** *Let  $(A, \mathcal{N}_I, G, \theta)$  be a probabilistic planning task, and  $\mathbf{a}$  be an  $m$ -step sequence of actions applicable in  $P_I$ . Let  $P$  be the probability distribution induced by  $\mathcal{N}_a$  on its variables  $\mathcal{V}_a$ .*

1. *The belief state  $P_a$  corresponds to the marginal distribution of  $P$  on  $\mathcal{V}_{(m)}$ , that is,  $P_a = P(\mathcal{V}_{(m)})$ .*
2. *If  $G_{(m)}$  is the (possibly partial) assignment provided by  $G$  to  $\mathcal{V}_{(m)}$ , then the probability that  $\mathbf{a}$  achieves  $G$  is equal to  $P(G_{(m)})$ , that is,  $P_a(G) = P(G_{(m)})$ .*

At this point, it is worth bringing attention to the fact that all the variables in  $\mathcal{V}_{(1)}, \dots, \mathcal{V}_{(m)}$  are completely deterministic. Moreover, the CPTs of these variables are all *compactly representable* (not exponential in the number of parents) due to the local structure induced by a large amount of context-specific independence (Boutilier *et al.* 1996). Hence, the description complexity of  $\mathcal{N}_a$  is linear in the size of the input, and in the number of actions in  $\mathbf{a}$ .

**Lemma 1** *Let  $(A, \mathcal{N}_I, G, \theta)$  be a probabilistic planning task, and  $\mathbf{a}$  be a sequence of actions from  $A$ . Then, we have  $|\mathcal{N}_a| = O(|\mathbf{a}| \cdot (|\mathcal{N}_I| + \alpha))$ , where  $\alpha$  is the largest description size of an action in  $A$ .*

## Belief States as Weighted CNFs

Given our representation of belief states as BNs, next we should select a mechanism for reasoning about these BNs. Computing the probability of a query in a BN is well known to be #P-complete (Roth 1996). While numerous exact algorithms for inference with BNs have been proposed in the literature (e.g., see (Darwiche 2000; Dechter 1999; Zhang & Poole 1994)), all these algorithms do not scale well on large networks exhibiting high density and tree-width. Recent advances in the area, however, show that practical exact inference in BNs is far from being hopeless. The main observation that guides these advances is that real-world domains typically exhibit a significant degree of deterministic

dependencies and context-specific independencies. Targeting this property of BNs already resulted in developing powerful inference techniques (Chavira & Darwiche 2005; Sang, Beame, & Kautz 2005). The general principle of these techniques is to (i) compile a BN  $\mathcal{N}$  into a *propositional logic formula*  $\phi(\mathcal{N})$ , (ii) associate some literals of  $\phi(\mathcal{N})$  with numerical *weights* derived from the CPTs of  $\mathcal{N}$ , and (iii) perform an efficient *weighted model counting* on  $\phi(\mathcal{N})$  by reusing (and adapting) certain techniques that appear powerful in enhancing backtracking DPPL-style search for SAT.

We have already shown that the BNs representing our belief states exhibit a large amount of both deterministic nodes and context-specific independence. Together with the fact that queries of our interest correspond to computing probability of “evidence”  $G_{(m)}$  in  $\mathcal{N}_a$ , this makes the model-counting techniques especially attractive for our purposes. Taking this route, in Probabilistic-FF we compile our BNs to weighted CNFs following the encoding scheme proposed in (Sang, Beame, & Kautz 2005), and answer probabilistic queries using Cachet (Sang *et al.* 2004), one of the most powerful (if not the most powerful) system to date for weighted model counting in CNFs.

For a detailed description of the weighted CNF encoding we refer the reader to (Sang, Beame, & Kautz 2005). In general, this encoding is based on two types of propositional variables: *state variables* for the values of the BN variables, and *chance variables*, encoding the entries of conditional probability tables. For each variable  $X_{(t)} \in \mathcal{V}_{(t)}$ , we need up to  $\gamma \cdot (|\text{Dom}(X_{(0)})| - 1)$  chance variables, where  $\gamma$  is the number of rows in  $T_{X_{(t)}}$ . Clauses involving both state and chance variables encode the structure of the CPTs, while clauses involving state variables only encode the “exactly one” relationship between the state variables capturing the value of a single BN variable.

Together with the weighted chance variables, the clauses of the encoding provide the input for a weighted model counting procedure. A simple recursive DPPL-style procedure basic-WMC underlying Cachet (Sang *et al.* 2004) is depicted in Figure 3. The weight of a (partial) variable assignment is the product of weights of the literals in that assignment. The weight of a formula  $\phi$  is the sum of weights of all complete satisfying assignments to the variables of  $\phi$ , and this is exactly the result of  $\text{basic-WMC}(\phi)$ . Likewise, Theorem 3 in (Sang, Beame, & Kautz 2005) shows that if  $\phi$  is a weighted CNF encoding of a BN  $\mathcal{N}$ , and  $P(Q|E)$  is a general query with respect to  $\mathcal{N}$ , then we have  $P(Q|E) = \frac{\text{basic-WMC}(\phi \wedge Q \wedge E)}{\text{basic-WMC}(\phi \wedge E)}$ . Note that, in our special case of empty evidence,  $P(Q|E)$  reduces to  $P(Q) = \text{basic-WMC}(\phi \wedge Q)$ , that is, a single call to the basic-WMC procedure.

**Lemma 2** *Let  $(A, P_I, G, \theta)$  be a probabilistic planning task with a BN  $\mathcal{N}_I$  describing  $P_I$ , and  $\mathbf{a}$  be an  $m$ -step sequence of actions. The probability  $P_a(G)$  that  $\mathbf{a}$  achieves  $G$  is:*

$$P_a(G) = \text{WMC}(\varphi(\mathbf{a}) \wedge G_{(m)}). \quad (8)$$

Besides the fact that weighted model counting is attractive for the kinds of BNs arising in our context, in what follows we show that weighted CNF representation of belief states goes extremely well with the ideas underlying

```

procedure basic-WMC( $\phi$ )
1 if  $\phi = \emptyset$  return 1
2 if  $\phi$  has an empty clause return 0
3 select a variable  $v \in \phi$ 
4 if  $v$  is a chance variable
5   return basic-WMC( $\phi|_{v=1}$ )  $\cdot$  weight( $v$ ) +
6     basic-WMC( $\phi|_{v=0}$ )  $\cdot$  (1 - weight( $v$ ))
7 else
8   return basic-WMC( $\phi|_{v=1}$ ) + basic-WMC( $\phi|_{v=0}$ )

```

Figure 3: Basic DPPL-style weighted model counting.

Conformant-FF (Brafman & Hoffmann 2004), a state-of-the-art approach for (non-probabilistic) conformant planning. In what follows, we outline the Conformant-FF approach to belief space search, and then explain the extensions made for Probabilistic-FF.

Conformant-FF does a forward search in a non-probabilistic belief space in which each belief state corresponds to a *set of world states* considered to be possible. The main trick of Conformant-FF is the use of CNF formulas for an *implicit* representation of belief states. Implicit, in this context, means that formulas  $\varphi(\mathbf{a})$  encode the semantics of executing action sequence  $\mathbf{a}$  in the initial belief state, with propositional variables corresponding to facts with *time-stamps*. Any actual knowledge about the belief states has to (and can) be *inferred* from these formulas; most particularly, a fact  $p$  is known to be true in a belief state iff  $\varphi(\mathbf{a}) \rightarrow p(m)$ , where  $m$  is the time endpoint of the formula. The only knowledge computed by Conformant-FF about belief states are these known facts, as well as (symmetrically) the facts that are known to be false. This suffices to do STRIPS-style planning, i.e., to determine applicable actions and goal belief states. This computation of only a partial knowledge constitutes a *lazy* kind of belief state representation, in comparison to other approaches that use explicit enumeration (Bonet & Geffner 2000) or BDDs (Bertoli *et al.* 2001) to fully represent belief states.

The basic idea underlying Probabilistic-FF is (i) to extend Conformant-FF’s belief state formulas to model our BNs, (ii) to use weighted model-counting to determine whether the probability of the goals in a belief state is high enough, and (iii) to adapt the heuristic function of Conformant-FF to the probabilistic setting. Note the synergetic effect: Probabilistic-FF can re-use *all* of Conformant-FF’s technology to recognize facts that are true or false with probability 1. This fully serves to determine applicable actions, as well as detect whether part of the goal is already known.

In more detail, given a probabilistic planning task  $(A, P_I, G, \theta)$ , a belief state  $P_{\mathbf{a}}$  corresponding to some executable  $m$ -step action sequence  $\mathbf{a}$ , and a proposition  $q \in \mathcal{X}$ , we say that  $q$  is *known* in  $P_{\mathbf{a}}$  if  $P_{\mathbf{a}}(q) = 1$ , *negatively known* in  $P_{\mathbf{a}}$  if  $P_{\mathbf{a}}(q) = 0$ , and *unknown* in  $P_{\mathbf{a}}$ , otherwise. We begin by determining whether  $q$  is known, negatively known, or unknown at time  $m$ . Re-using the Conformant-FF machinery, this classification requires up to two SAT tests of  $\varphi(\mathcal{N}_{\mathbf{a}}) \wedge \neg q(m)$  and  $\varphi(\mathcal{N}_{\mathbf{a}}) \wedge q(m)$ , respectively. The information provided by this classification is used threefold. If a subgoal  $g$  is negatively known at time  $m$ , then we have  $P_{\mathbf{a}}(G) = 0$ . On the other extreme, if all the subgoals of  $G$  are known at time  $m$ , then we have  $P_{\mathbf{a}}(G) = 1$ . Finally, if some subgoals of  $G$  are known and the rest are unknown at

time  $m$ , then we accomplish evaluating  $\mathbf{a}$  by testing whether

$$P_{\mathbf{a}}(G) = \text{WMC}(\varphi(\mathcal{N}_{\mathbf{a}}) \wedge G(m)) \geq \theta. \quad (9)$$

Note also that having the sets of all (positively/negatively) known propositions at all time steps up to  $m$  allows us *significantly* simplify the CNF formula  $\varphi(\mathcal{N}_{\mathbf{a}}) \wedge G(m)$  by inserting into it the corresponding values of known propositions.

After evaluating the considered action sequence  $\mathbf{a}$ , if we get  $P_{\mathbf{a}}(G) \geq \theta$ , then we are done. Otherwise, the forward search continues, and the actions that are applicable in  $P_{\mathbf{a}}$  (and thus used to generate the successor belief states) are those whose preconditions are all known in  $P_{\mathbf{a}}$ .

## Heuristic Function

Like the belief state representation, Conformant-FF’s heuristic goal distance estimation process translates naturally to the new context. For the estimation, Conformant-FF relaxes the planning task by ignoring all delete lists, and by ignoring all but one effect condition of each effect. Under these simplifying assumptions, original FF’s machinery (Hoffmann & Nebel 2001) can be extended with a DAG – called “implication graph” – capturing the dependencies between unknown facts, as well as with a reasoning machinery that uses this implication graph to infer when unknown facts become known. The extended machinery is sound and complete for relaxed tasks, and yields a heuristic function that is highly informative across a range of challenging domains (Brafman & Hoffmann 2004).

In the Probabilistic-FF context, we need to extend Conformant-FF’s machinery with the abilities to determine when the goal set is sufficiently *likely* (rather than known to be true for sure). To achieve that, we must introduce into relaxed planning some effective reasoning about the probabilistic initial state and effects of probabilistic actions. It turns out that such a reasoning can be obtained by a certain natural *weighted* extension of the implication graph and the reasoning done with it. In what follows, we explain the extended process in detail, starting with brief descriptions of the machinery used in original FF, and in Conformant-FF.

The heuristic of FF is based on the notion of *relaxed plan*, which is a plan that achieves the goals while assuming that all delete lists of actions are empty. The heuristic value  $h(w)$  that FF provides to a world state  $w$  encountered during the search is the length of the relaxed plan to  $w$ . Formally, relaxed plans are computed as follows. Starting from  $w$ , FF builds a *relaxed planning graph* as a sequence of alternating proposition layers  $P(t)$  and action layers  $A(t)$ , where  $P(0)$  is the same as  $w$ ,  $A(t)$  is the set of all actions whose preconditions are contained in  $P(t)$ , and  $P(t+1)$  is obtained from  $P(t)$  by including the add effects (with fulfilled conditions) of the actions in  $A(t)$ . The relaxed planning graph is constructed either until it reaches a propositional layer  $P(m)$  that contains all the goals, or until the construction reaches a fixpoint  $P(t) = P(t+1)$  without reaching the goals. The latter case corresponds to (all) situations in which a relaxed plan does not exist, and because existence of a relaxed plan is a necessary condition for existence of a real plan, the state  $w$  is excluded from the search space by setting  $h(w) = \infty$ .

In the former case of  $G \subseteq P(m)$ , a relaxed plan is a subset of actions in  $A(1), \dots, A(m)$  that suffices to achieve the goals (under ignoring the delete lists), and it can be extracted by a simple backchaining loop: For each goal in  $P(m)$ , select an action in  $A(1), \dots, A(m)$  that achieves this goal, and iterate the process by considering those actions' preconditions and the respective effect conditions as new subgoals. The length of the relaxed plan (aka heuristic value  $h(w)$ ) is the number of actions selected in this backchaining process.

In Conformant-FF, this construction is extended with additional fact layers  $uP(t)$  containing the facts *unknown* at time  $t$ , and the reasoning is extended to when such unknown facts become known in the relaxed planning graph. As the complexity of this type of reasoning is prohibitive, Conformant-FF further relaxes the planning task by ignoring not only the delete lists, but also all but one of the unknown conditions of each action effect. That is, if action  $a$  appears in layer  $A(t)$ , and for effect  $e$  of  $a$  we have  $con(e) \subseteq P(t) \cup uP(t)$  and  $con(e) \cap uP(t) \neq \emptyset$ , then  $con(e) \cap uP(t)$  is arbitrarily reduced to contain exactly one literal, and reasoning is done as if  $con(e)$  had this reduced form from the beginning. This relaxation converts implications  $\bigwedge_{c \in con(e) \cap uP(t)} c(t) \rightarrow q(t+1)$  that the (unknown) effects induce between unknown propositions into their 2-projections that take the form of binary implications  $c(t) \rightarrow q(t+1)$ , with arbitrary  $c \in con(e) \cap uP(t)$ . The set of all these binary implications forms a DAG. Thus, checking whether a proposition  $q$  becomes known at time  $t$  can be done by a backchaining over the implication edges that end in  $q(t)$ , followed by a SAT check to see if the initial belief state formula implies the disjunction of the reachable tree leafs. Note that this process over-estimates the sets of propositions that become known at a given time point; Since it is easier to achieve a fact in relaxed planning, reasoning about when and what becomes known under relaxation is a non-sound but complete approximation of this reasoning without relaxation. Hence, if Conformant-FF discovers that there is no relaxed plan to a world state  $w$ , then  $w$  can be safely excluded from the search.

We now explain our extension of this reasoning process to the probabilistic setting. Some parts are inherited from Conformant-FF, and the new parts are dedicated to processing probabilistic belief states and probabilistic actions. We begin with our procedure for building a *probabilistic relaxed planning graph* (or PRPG, for short), then we show how one can extract a (probabilistic) relaxed plan from the PRPG.

Figure 4 depicts the main routine for building the PRPG. Layers  $-n$  to  $-1$  of PRPG correspond to the  $m$ -step action sequence a leading to the considered belief state  $P_a$ . The negative indices of the layers are used to simplify the presentation of reasoning about the past. The PRPG is initialized with an empty implication set  $Imp$ , and  $P(-m)$  and  $uP(-m)$  containing the propositions that are known (hold with probability 1) and unknown (hold with probability less 1 but greater than 0) in the initial belief state. The **for** loop (lines 3-19) then builds the sets  $P$  and  $uP$  for the subsequent time steps, incrementally extending the implication set  $Imp$ . At each iteration  $-m \leq t \leq -1$ , the sets  $P(t+1)$  and  $uP(t+1)$  are made to contain the proposi-

```

procedure build-PRPG ( $\langle a^{-m}, \dots, a^{-1} \rangle, A, \varphi(\mathcal{N}_I), G, \theta$ )
1   $Imp := \emptyset, P(-m) := \{q \mid q \text{ is known in } \varphi(\mathcal{N}_I)\},$ 
2   $uP(-m) := \{q \mid q \text{ is unknown in } \varphi(\mathcal{N}_I)\}$ 
3  for  $t := -m \dots -1$  do
4     $P(t+1) := \{q \mid q \text{ is known after } a^t\}$ 
5     $uP(t+1) := \{q \mid q \text{ is unknown after } a^t\}$ 
6     $Imp \cup = \{(q(t), q(t+1)) \mid q \in uP(t) \cap uP(t+1)\}$ 
7    for all  $e \in E(a^t)$  s.t.  $con(e) \subseteq P(t) \cup uP(t)$  do
8      if  $a^t \in A_d$  and  $con(e) \cap uP(t) \neq \emptyset$  then
9        select  $c \in con(e) \cap uP(t)$ 
10        $Imp \cup = \{(c(t), q(t+1)) \mid q \in add(e) \cap uP(t+1)\}$ 
11      elseif  $a^t \in A_p$ 
12        for all  $\varepsilon \in \Lambda(e), q \in add(\varepsilon) \cap uP(t+1)$ 
13          introduce new fact  $\omega_q^\varepsilon(t)$  with  $weight(\omega_q^\varepsilon(t)) = p(\varepsilon);$ 
14           $Imp \cup = \{(\omega_q^\varepsilon(t), q(t+1))\}$ 
15          if  $con(e) \cap uP(t) \neq \emptyset$  then
16            select  $c \in con(e) \cap uP(t)$ 
17             $Imp \cup = \{(c(t), \omega_q^\varepsilon(t))\}$ 
18          endif
19      endif
20   $t := 0$ 
21  while  $get-P(t, G) < \theta$  do
22     $A(t) := \{a \mid a \in A, pre(a) \subseteq P(t)\}$ 
23    build-timestep( $t, A(t)$ )
24    if  $P(t+1) = P(t)$  and
25       $uP(t+1) = uP(t)$  and
26       $\forall q \in uP(t+1) : WImpleafs(q(t+1)) = WImpleafs(q(t))$  then
27      return FALSE
28    endif
29     $t := t + 1$ 
30  return TRUE

```

Figure 4: Building a probabilistic relaxed planning graph.

tions that are known/unknown after applying the action  $a^t$ , respectively. The implication set  $Imp$  is extended as follows. First, we add an implication  $q(t) \rightarrow q(t+1)$  for every proposition  $q$  that is unknown at both  $t$  and  $t+1$ . Then we consider each effect  $e \in E(a^t)$  that has no negatively known conditions at time  $t$ . If action  $a^t$  is deterministic and not all of its conditions are known at time  $t$  (lines 8-10), then for each considered effect  $e$  we add an implication  $c(t) \rightarrow q(t+1)$  for an arbitrarily chosen unknown condition of  $e$ , and each unknown at  $t+1$  add effect  $q$  of  $e$ . Alternatively, if action  $a^t$  is probabilistic (lines 12-18), then we consider each probabilistic outcome  $\varepsilon$  of  $e$ . For each (unknown at  $t+1$ ) add effect  $q$  of  $\varepsilon$ , we introduce a chance proposition  $\omega_q^\varepsilon(t)$  with  $weight(\omega_q^\varepsilon(t)) = p(\varepsilon)$ , and add an implication  $\omega_q^\varepsilon(t) \rightarrow q(t+1)$ . In addition, if effect  $e$  is conditional and not all of its conditions are known at  $t$ , then we add an implication  $c(t) \rightarrow \omega_q^\varepsilon(t)$  for an arbitrarily chosen unknown condition of  $e$ .

The next **while** loop (lines 21-29) constructs the relaxed planning graph from layer 0 onwards by iterative invocation of the build-timestep procedure that expands PRPG by a single level at each call. This construction is controlled by two termination tests. If the goal is achieved at layer  $t$  with probability higher than  $\theta$ , then a relaxed plan can be extracted. Otherwise, if the graph reaches a fix point, then we know that no relaxed (and thus, no real) plan exists. The structure  $WImpleafs(q(t))$  used in this termination test corresponds to the set of *weighted* (in a manner described below) leafs of the  $Imp$  subgraph rooted at  $q(t)$ .

```

procedure build-timestep ( $t, A(t)$ )
1   $P(t+1) := P(t), uP(t+1) := uP(t)$ 
2   $Imp \cup = \{(q(t), q(t+1)) \mid q \in uP(t)\}$ 
3  for all  $a \in A_d \cap A(t)$ , all  $e \in E(a)$  do
4    if  $con(e) \subseteq P(t)$  then  $P(t+1) \cup = add(e)$  endif
5    if  $con(e) \subseteq P(t) \cup uP(t)$  and  $con(e) \cap uP(t) \neq \emptyset$  then
6       $uP(t+1) \cup = add(e)$ 
7      select  $c \in con(e) \cap uP(t)$ 
8       $Imp \cup = \{(c(t), q(t+1)) \mid q \in add(e)\}$ 
9    endif
10 for all  $a \in A_p \cap A(t)$ , all  $e \in E(a)$  do
11   for all  $\varepsilon \in \Lambda(e)$ , all  $q \in add(\varepsilon)$  do
12      $uP(t+1) \cup = \{q\}$ 
13     introduce new fact  $\omega_q^\varepsilon(t)$  with  $weight(\omega_q^\varepsilon(t)) = p(\varepsilon)$ ;
14      $Imp \cup = \{(\omega_q^\varepsilon(t), q(t+1))\}$ 
15     if  $con(e) \cup uP(t) \neq \emptyset$  do
16       select  $c \in con(e) \cap uP(t)$ 
17        $Imp \cup = \{(c(t), \omega_q^\varepsilon(t))\}$ 
18     endif
19 for all  $q \in uP(t+1) \setminus P(t+1)$  do
20    $WImpleafs(q(t+1)) := build-w-impleafs(q(t+1), Imp)$ 
21    $DS := \{l \mid l \in WImpleafs(q(t+1)) \wedge weight_{q(t+1)}(l) = 1\}$ 
22   if  $\varphi(\mathcal{N}_I) \rightarrow \bigvee_{l \in DS} l$  then  $P(t+1) \cup = \{q\}$  endif
23  $uP(t+1) \setminus = P(t+1)$ 

```

```

procedure build-w-impleafs ( $q(t), Imp$ )
1  for all state variables  $v \in Imp_{q(t)}$  do  $weight(v) := 1$ 
2  for all  $v \in Imp_{q(t)}$  in a reverse topological order of  $Imp_{q(t)}$  do
3     $weight_{q(t)}(v) := weight(v)$ 
4    if  $(\max \{weight_{q(t)}(u) \mid (v, u) \in Imp_{q(t)}\} < 1)$  then
5       $weight_{q(t)}(v) * = \min \{1 - \delta, \sum_{(v, u) \in Imp_{q(t)}} weight_{q(t)}(u)\}$ 
6  return the set of all leafs  $l$  of  $Imp_{q(t)}$  annotated with  $weight_{q(t)}(l)$ 

```

```

procedure get-P ( $t, G$ )
1  if  $G \not\subseteq P(t) \cup uP(t)$  then return 0 endif
2  if  $G \subseteq P(t)$  then return 1 endif
3  return  $WMC(\varphi(\mathcal{N}_I) \wedge \bigwedge_{g \in G} \neg P(t) \bigvee_{l \in WImpleafs(g(t))} l)$ 

```

Figure 5: Building a new layer in the PRPG, and computing the goal likelihood at time step  $t$ .

The build-timestep procedure is shown in Figure 5. The first **for** loop (lines 3-9) proceeds over all effects of all deterministic actions “applicable” at time  $t$ . Effect whose condition is known at time  $t$  (possibly) make some additional facts to be known at time  $t+1$ . Effects whose condition depends on the actual initial state and/or outcome of probabilistic actions (possibly) add some additional such facts at time  $t+1$  (together with the appropriate binary implications in  $Imp$ ). The second **for** loop (lines 10-18) proceeds over all possible outcomes of all probabilistic actions “applicable” at layer  $t$ , providing all the propositions in the add lists of these outcomes with processing similar to this in lines 14-17 of build-PRPG. Now, let  $Imp_{q(t)}$  be the subgraph of  $Imp$  formed by  $q(t)$  and all its predecessors in  $Imp$ . The third **for** loop (lines 19-22) checks whether an unknown proposition  $q$  becomes known at  $t+1$ . Specifically, it is checked whether the initial state formula implies one of the leafs  $l$  of  $Imp_{q(t+1)}$  such that there exists a path in  $Imp_{q(t+1)}$  from  $l$  to  $q(t+1)$  with *all* edges of the path being contributed by deterministic actions only. The latter is determined using the build-w-impleafs procedure (Figure 5).

The build-w-impleafs procedure constructs a set

$WImpleafs(q(t))$  of all leafs of  $Imp_{q(t)}$  weighted with respect to  $q(t+1)$ . Specifically, build-w-impleafs performs a BFS-style, top-down *weight propagation* from  $q(t)$  down to the leafs of  $Imp_{q(t)}$ . Together with acyclicity of  $Imp$ , the BFS-style traversal ensures that each variable  $v$  is processed only after processing all variables  $u$  such that  $Imp_{q(t)}$  contains implication  $v \rightarrow u$ . When processed,  $v$  is assigned with a weight  $weight_{q(t)}(v)$  derived from its own “global” weight  $weight(v)$  and the weights  $weight_{q(t)}(u)$  of its immediate successors  $u$  in  $Imp_{q(t)}$ . We have:

- (1)  $weight_{q(t)}(v)$  is an *upper bound* on the probability of achieving  $q$  at time  $t$  by a sequence of actions responsible for a path from  $v$  to  $q(t)$  in  $Imp$ ;
- (2)  $weight_{q(t)}(v) = 1$  iff there exists a path from  $v$  to  $q(t)$  in  $Imp_{q(t)}$  where all edges are due to deterministic actions. (Otherwise, we always have  $weight_{q(t)}(v) \leq 1 - \delta$ , for some arbitrary small  $0 < \delta \ll 1$ .)

**Theorem 2** *Let  $(A, \mathcal{N}_I, G, \theta)$  be a probabilistic planning task, and  $\mathbf{a}$  be a sequence of actions applicable in  $P_I$ . If build-PRPG( $\mathbf{a}, A, \varphi(\mathcal{N}_I), G, \theta$ ) returns FALSE, then there is no relaxed plan for  $(A, \mathcal{N}_I, G, \theta)$  that starts with  $\mathbf{a}$ .*

The proof of Theorem 2 is based on three observations. Let  $T > 0$  be the last layer of the PRPG upon the termination of build-PRPG. First, for every  $-m \leq t \leq T$ , the sets  $P(t)$  and  $uP(t)$  contain all (and only all) propositions that are known (respectively unknown) after executing all the actions in the action layers up to  $A(t-1)$ . Second, if build-PRPG returns FALSE, then the corresponding termination criterion would hold in all future iterations. Finally, if the PRPG does contain a relaxed plan for  $(A, \mathcal{N}_I, G, \theta)$ , then we have  $get-P(T, G) \geq \theta$ , and thus (together with the second observation) build-PRPG cannot return FALSE.

To see the latter, consider the details of the procedure get-P as in Figure 5. This procedure is used to compute the goal likelihood at time  $t$ . First, if one of the subgoals is negatively known at time  $t$ , then the overall probability of achieving the goal is 0. On the other extreme, if all the subgoals are known at time  $t$ , then the probability of achieving the goal is trivially 1. This leaves us with the main case in which we are uncertain about some of the subgoals, and this uncertainty is either due to dependence of these subgoals on the actual initial world state, or due to achieving these subgoals using probabilistic actions (or due to both.) The uncertainty about the initial state is fully captured by our weighted CNF  $\varphi(\mathcal{N}_I)$ . Likewise, the new chance propositions  $\omega_q^\varepsilon(t')$  introduced in build-PRPG and build-timestep for the add lists of probabilistic outcomes are “chained up” in  $Imp$  to the propositions on these add lists, and “chained down” in  $Imp$  to the (relaxed) conditions of these outcomes, if any. Therefore, if some add effect  $q$  of a probabilistic action at time  $t' < t$  is relevant to achieving a subgoal  $g \in G$  at time  $t$ , then the corresponding chance variable  $\omega_q^\varepsilon(t')$  will appear in  $Imp_{g(t)}$ , and its weight will be properly propagated by build-w-impleafs( $g(t), Imp$ ) down to the leafs of  $Imp_{g(t)}$ . (For *unconditional* probabilistic actions,  $\omega_q^\varepsilon(t')$  is in already a leaf in  $Imp$ .)

**Example 2** To illustrate the treatment of probabilistic effects, consider the following simple throw-a-dice example. The only propositions are  $q_1, \dots, q_6$  with the obvious meaning. The initial state is empty,  $G = \{q_6\}$ . We have a single, probabilistic, action throw-a-dice that has a single unconditional effect with six equiprobable outcomes  $\varepsilon_1, \dots, \varepsilon_6$ , where  $\text{add}(\varepsilon_i) = \{q_i\}$  and  $\text{del}(\varepsilon_i) = \emptyset$ . Figure 6 depicts an excerpt of the implication graph constructed by  $t = 3$ . The probability of throwing a six in three throws (that is, the probability of  $q_6$  after a sequence of three actions throw-a-dice) is equal to  $\text{WMC}(\omega_{q_6}^{\varepsilon_6}(0) \vee \omega_{q_6}^{\varepsilon_6}(1) \vee \omega_{q_6}^{\varepsilon_6}(2))$ .

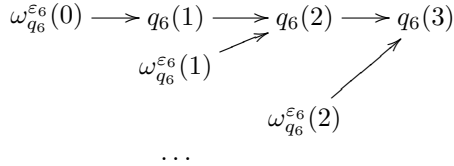


Figure 6: Implication graph for *throw-a-dice* example.

If build-PRPG succeeds at a level  $T$ , then a relaxed plan can be extracted from this PRPG using the extract-PRPlan procedure (Figure 7). The sets  $G(1), \dots, G(T)$  used in this procedure store goals and subgoals arising at layers  $1 \leq t \leq T$ . First, the **for** loop in lines 1-7 processes the goals that remain unknown at  $T$ . For each such goal  $g$ , and each<sup>4</sup> leaf  $l$  of the implication subgraph  $\text{Imp}_{g(T)}$ , the procedure selects an *arbitrary* path from  $l$  to  $g(T)$ , and adds to the relaxed plan all the actions responsible for the edges of this path. The next **for** loop (lines 9-21) accomplishes the construction of the relaxed plan in a way similar in spirit to Conformant-FF. Specifically, for each goal/subgoal  $g(t)$ , if there is a *deterministic* action  $a$  in  $A(t-1)$  that guarantees to always achieve  $g$ , then one such action is added to the relaxed plan. Otherwise, the relaxed plan is extended with a set of all actions responsible for a set of implication paths from all leafs in  $DS \subseteq \text{WImpleafs}(g(t))$  (similar to this in line 21 of build-timestep) to  $g(t)$ .

In summary, if build-PRPG returns TRUE, then we set  $h(\mathbf{a})$  to the number of actions selected by extract-PRPlan. Otherwise, we set  $h(\mathbf{a}) = \infty$ , and the corresponding belief state  $P_{\mathbf{a}}$  is excluded from the search.

## Empirical Results

We implemented the techniques described in the previous sections in C, starting from the Conformant-FF code. We call the implemented system Probabilistic-FF. With  $\theta = 1.0$ , Probabilistic-FF behaves exactly like Conformant-FF. Otherwise, Probabilistic-FF uses Cachet (Sang, Beame, & Kautz 2005) for the weighted model counting. The experiments were run on a PC running at 1.2GHz with 1GB main memory and 1024KB cache running Linux.

<sup>4</sup>In attempt to minimize the length of the extracted relaxed plan, one can consider only a minimal subset  $L$  of  $\text{Impleafs}(g(T))$ , constructed by a greedy iterative removal of the leaves until the goal probability gets too low. This minimization, however, can be costly due to the required numerous calls to model counting.

```

procedure extract-PRPlan ( $G, \theta$ )
1 for each  $g \in G \setminus P(T)$  do
2   for each  $l \in \text{WImpleafs}(g(T))$ 
3     pick an arbitrary path  $\rho$  in  $\text{Imp}$  from  $l$  to  $g(T)$ 
4     for  $i := 0 \dots T-1$  do
5       if some edge in  $\rho$  is due to an effect  $e$  of an action  $a \in A(i)$ :
6         select  $a$  at time  $i$ 
7         sub-goal( $(\text{pre}(a) \cup \text{con}(e)) \cap P(i)$ )
8   sub-goal( $G \cap P(T)$ )
9 for  $t := T, \dots, 1$  do
10  for all  $g \in G(t)$  do
11    if  $\exists a \in A(t-1) \cap A_d, \exists e \in E(a)$  such that
12       $g \in \text{add}(e), \text{pre}(a) \subseteq P(t-1), \text{con}(e) \subseteq P(t-1)$  then
13        select one such pair  $a$  and  $e$ 
14        sub-goal( $\text{pre}(a) \cup \text{con}(e)$ )
15    else
16       $DS := \{l \mid l \in \text{WImpleafs}(g(t)) \wedge \text{weight}_{g(t)}(l) = 1\}$ 
17      for each  $l \in DS$ 
18        pick a path  $\rho$  in  $\text{Imp}$  from  $l$  to  $g(t)$  such that
19           $\forall q \in \rho : \text{weight}_{g(t)}(q) = 1$ 
20        for  $i := 0 \dots t-1$  do
21          if some edge in  $\rho$  is due to an effect  $e$  of an action  $a \in A(i)$ :
22            select  $a$  at time  $i$ 
23            sub-goal( $(\text{pre}(a) \cup \text{con}(e)) \cap P(i)$ )

```

```

procedure sub-goal ( $P$ ): for all  $p \in P, G(\min\{t \mid p \in P(t)\}) \cup = \{p\}$ 

```

Figure 7: Extracting a probabilistic relaxed plan.

As said, at the current stage of development support is implemented only for deterministic effects. This makes comparison with previous planners (Maxplan (Majercik & Littman 1998) and CPP (Hyafil & Bacchus 2004)) particularly difficult – apart from the very different input languages – since the experiments done by these authors focused exclusively on rather simple or even fully known initial states, and probabilistic effects. Instead, our experiments focus on highly non-trivial uncertain initial states, but deterministic effects. As a concrete means of comparison, we ran Maxplan on some of our benchmarks. In all cases, Maxplan ran out of time ( $> 1$  hour) on instances far from a size where Probabilistic-FF’s runtime becomes even measurable. We couldn’t run CPP due to technical problems. Some more details on all this are given below. (From the comparison between Maxplan and CPP (Hyafil & Bacchus 2003), it seems very likely that CPP would not scale into regions challenging for Probabilistic-FF.) Note that Maxplan and CPP both plan for a given length bound, returning the best (maximal goal likelihood) plan of that length; to plan for a given  $\theta$ , these planners would (theoretically) have to be run with iteratively increased bound.

As a rougher means of comparison, note that the performance limit of Maxplan and CPP is, generally speaking, reached in solving instances with around 100 world states and 15-20 steps plans. In contrast, with Probabilistic-FF we can find 100-step plans for instances with billions of world states. The results are in Table 1. We show the relevant search data, for different instances, and for different settings of the goal threshold  $\theta$ . As a coarse measure of instance size, we specify the numbers of ground actions and ground facts, as well as an estimate of the number of world states.

		$\theta = 0.25$	$\theta = 0.5$	$\theta = 0.75$	$\theta = 1.0$
Instance	#actions / #facts / #world states	$t/tWMC/ S /l$	$t/tWMC/ S /l$	$t/tWMC/ S /l$	$t/ S /l$
Safe-uni-70	70 / 71 / 140	2.21/0.37/19/18	6.19/2.15/36/35	13.47/6.64/54/53	7.72/71/70
Safe-cub-70	70 / 70 / 138	0.58/0.02/6/5	1.30/0.12/13/12	2.57/0.44/22/21	7.49/70/69
Cube-uni-11	6 / 66 / 1331	1.31/0.30/105/18	1.52/0.26/127/24	2.37/0.38/165/28	5.69/257/30
Cube-cub-11	6 / 66 / 1331	0.16/0.03/24/5	0.46/0.09/51/9	0.66/0.15/74/13	5.65/257/30
Bomb-50-1	51 / 102 / $> 2^{51}$	0.07/0.01/1/0	2.64/0.60/662/31	7.29/1.84/1192/71	4.38/1325/99
Bomb-50-5	255 / 110 / $> 2^{55}$	0.07/0.01/1/0	1.93/0.44/468/27	6.09/1.54/998/67	3.86/1131/95
Bomb-50-10	510 / 120 / $> 2^{60}$	0.07/0.01/1/0	1.16/0.24/248/22	4.93/1.20/778/62	3.26/911/90
Bomb-50-50	2550 / 200 / $> 2^{100}$	0.10/0.01/1/0	0.32/0.02/17/16	0.41/0.06/37/36	0.48/51/50
Log-2	3440 / 1040 / $> 20^{10}$	1.26/0.01/117/54	1.59/0.01/152/62	2.21/0.01/205/69	2.51/295/78
Log-3	3690 / 1260 / $> 30^{10}$	3.60/0.01/159/64	10.56/0.01/328/98	6.00/0.01/336/99	5.23/364/105
Log-4	3960 / 1480 / $> 40^{10}$	3.29/0.01/138/75	11.54/0.01/391/81	8.14/0.01/377/95	10.48/554/107
Grid-2	2040 / 825 / $> 36^{10}$	0.22/0.01/39/21	1.91/0.14/221/48	10.05/0.06/1207/69	9.65/1207/69
Grid-3	2040 / 841 / $> 36^{10}$	22.61/0.09/1629/76	20.77/2.22/1119/89	108.71/11.53/3974/123	93.84/3974/123
Grid-4	2040 / 857 / $> 36^{10}$	38.39/0.13/2167/96	72.28/1.26/2541/111	71.80/0.53/2541/115	281.08/6341/155
Rovers-7	393 / 97 / $> 6^3 * 3^8$	0.03/0.00/37/18	0.03/0.00/37/18	0.03/0.00/37/18	0.03/0.00/37/18
RoversP-7	393 / 133 / $> 6^3 * 3^8$	3.60/0.00/942/65	3.06/0.00/983/75	3.29/0.00/1008/83	3.82/1008/83
RoversPP-7	393 / 133 / $> 6^3 * 3^8$	9.77/0.01/948/65	14.26/0.10/989/75	14.70/0.12/994/77	18.49/1014/83
RoversPPP-7	395 / 140 / $> 6^3 * 3^8$	28.46/2.36/950/67	45.61/1.14/996/79	0.02/proved unsolvable	0.02/proved unsolvable

Table 1: Empirical results. Times  $t$  in seconds ( $tWMC$  is time spent in WMC, which is none if the goal request is 1.0), search space size  $|S|$  (number of calls to the heuristic function), plan length  $l$ .

Our first three domains are probabilistic versions of traditional conformant benchmarks: “Safe”, “Cube”, and “Bomb”. In Safe, each of  $n$  combinations of a safe must be tried to make sure the safe is open. We implemented two probability distributions over the  $n$  combinations, a uniform one (“Safe-uni”) and a distribution that declines according to a cubic function (“Safe-cub”). Table 1 shows that Probabilistic-FF can solve this very efficiently even with  $n = 70$ . In Safe-cub, less combinations must be tried since a lot of “weight” is contained in the combinations at the start of the cubic distribution (the last combination has weight 0 and so needs not be tried even with  $\theta = 1.0$ ). Observe that, rather than resulting in a performance decline, moving from  $\theta = 1.0$  to  $\theta < 1.0$  (from Conformant-FF to cases that require weighted model counting) even *improves* performance (except for  $\theta = 0.75$  in Safe-uni) since the plans become shorter. In Cube, the task is to move into a corner of a 3-dimensional grid. Again, we implemented uniform and cubic distributions (over the initial position in each dimension), and again, Probabilistic-FF scales well, easily solving instances on a  $11 \times 11 \times 11$  grid, becoming faster with decreasing  $\theta$ . (We also tried a version of Cube where the task is to move into the grid *center*; like Conformant-FF, Probabilistic-FF is bad at doing so, reaching its performance limit at  $n = 7$ .) Our version of Bomb contains  $n$  bombs and  $m$  toilets, where each bomb may be armed or not armed *independently*, resulting in huge numbers of initially possible world states. In the depicted instances, each bomb has a 0.02 chance of being armed. Dunking a bomb in a toilet clogs the toilet, which must then be flushed. Like Conformant-FF, Probabilistic-FF scales to large  $n$ , and becomes faster as  $m$  increases. And again, we obtain the nice pattern of *improved* performance as we move from non-probabilistic ( $\theta = 1.0$ ) into probabilistic planning (specifically,  $\theta \leq 0.5$ ; for  $\theta \leq 0.25$ , the initial state is good enough already). We

remark that, for all of Safe, Cube, and Bomb, Probabilistic-FF’s plans are optimal (the shortest possible).

Our next three domains are adaptations of benchmarks from deterministic planning: “Logistics”, “Grid”, and “Rovers”. We assume that the reader is familiar with these domains. Each “Log- $x$ ” instance contains 10 cities, 10 airplanes, and 10 packages, where each city has  $x$  locations. The packages are with chance 0.88 at the airport of their origin city, and uniformly at any of the other locations in that city. The effects of all loading and unloading actions are conditional on the (right) position of the package. Note that higher values of  $x$  increase not only the space of world states, but also the initial uncertainty. We again observe the nice scaling properties discussed above.<sup>5</sup> Grid is the complex grid world run in the AIPS’98 planning competition, featuring locked positions that must be opened with matching keys. “Grid- $x$ ” here is a modification of instance nr. 2 (of 5) run at AIPS’98, with a  $6 \times 6$  grid, 8 locked positions, and 10 keys of which 3 must be transported to a goal position. Each lock uniformly has  $x$  possible shapes, and each of the 3 goal keys uniformly has  $x$  possible initial positions. The effects of pickup-key, putdown-key, and open-lock actions are conditional. The scaling behavior is similar as in the previous domains. Our Rovers instances, finally, are modifications of instance nr. 7 (of 20) run at the AIPS’02 planning competition, with 6 waypoints, 3 rovers, 2 objectives, and 6 rock/soil samples. From “Rovers” to “RoversPPP” we modify the instance/domain as follows. “Rovers” is the original AIPS’02 domain and instance for compari-

<sup>5</sup>The hardest Logistics instance used to test CPP (Hyafil & Bacchus 2004) has 4 cities of size 2, 1 airplane, and 2 packages. There, CPP solves 13 steps in 6400 seconds on a 2.4GHz machine, and runs out of 3GB memory for 14 steps. While CPP incorporates non-deterministic effects, we believe that this – in comparison to our results – still says something about scaling in Logistics.

son. In “RoversP”, each sample is with chance 0.8 at its original waypoint, and with chance 0.1 at each of two others; each objective may be visible from 3 waypoints with uniform distribution (this is a probabilistic adaptation of the domain from (Bryce & Kambhampari 2004)). “RoversPP” enhances this by *conditional* probabilities in the initial state, stating that whether or not an objective is visible from a waypoint depends on whether or not a rock sample (intuition: a large piece of rock) is located at the waypoint. The probability of visibility is much higher if the latter is not the case (concretely, the visibility of each objective depends on the locations of two rock samples; if a rock sample is present then the visibility probability drops to 0.1). “RoversPPP”, finally, introduces the need to collect data about water existence. Each of the soil samples has a certain likelihood (below 1) to be “wet”. An additional operator tests, for communicated sample data, if the sample was wet; if so, a fact “know-that-water”, that is contained in the goals, is set to true. The probability of being wet depends on the location of the sample. Goal probabilities  $\theta = 0.75$  and  $\theta = 1.0$  can not be achieved, which is proved by *the heuristic function*, providing the correct answer in split seconds.

As for Maxplan and CPP, due to technical problems (not on our side) with the CPP distribution, that were not resolved by the time of writing this paper, it was impossible to run CPP on our benchmarks. We ran Maxplan in Safe, Cube, and Bomb. In Safe with 10 combinations, we obtained a plan with 5 steps (goal likelihood 0.5) in 270 seconds. Setting the bound to 6, Maxplan ran out of time (killed after 1 hour). In Cube, Maxplan easily solved a  $3 \times 3 \times 3$  grid up to 6, where the goal is achieved with certainty. It found plans within a few hundred seconds for  $4 \times 4 \times 4$  and  $5 \times 5 \times 5$  grids with a bound of 5, but with bound 6 ran out of time. In Bomb, finally, 10 bombs 10 toilets could only be done up to bound 3 (270 seconds), and 10 bombs 1 toilet failed at bound 8.

## Conclusion

We developed a synergetic combination of Conformant-FF with recent techniques for probabilistic reasoning. The resulting planner scales well in several adaptations of known planning benchmarks, constituting an advance of several orders of magnitude to the state of the art in probabilistic planning with no observability and deterministic actions. We do not claim that this solves the (this particular) problem once and for all: Probabilistic-FF inherits strengths *and* weaknesses from FF and Conformant-FF, like domains where FF’s heuristic function yields bad estimates, or domains (e.g. the mentioned Cube-center variant) where Conformant-FF gets lost due to heuristic inaccuracy combined with a high degree of uncertainty. We *do* believe that the synergy between WMC and Conformant-FF’s CNF-based techniques constitutes a major step forward.

Next, of course, we will implement the presented treatment of probabilistic effects, and evaluate its performance. In the long term, we intend to work towards applicability in real-world settings, particularly in the space application settings that our Rovers domain hints at, in medication-type treatment planning domains, and in the power supply restoration domain (e.g., (Bertoli *et al.* 2002)).

## References

- Bertoli, P.; Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2001. MBP: a model based planner. In *IJCAI’01 Workshop on Planning under Uncertainty*.
- Bertoli, P.; Cimatti, A.; Slaney, J.; and Thiébaux, S. 2002. Solving power supply restoration problems with planning via symbolic model-checking. In *ECAI’02*, 576–580.
- Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *AIPS’00*.
- Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *UAI’96*, 115–123.
- Brafman, R. I., and Hoffmann, J. 2004. Conformant planning via heuristic forward search: A new approach. In *ICAPS’04*, 355–364.
- Bryce, D., and Kambhampari, S. 2004. Heuristic guidance measures for conformant planning. In *ICAPS’04*, 365–374.
- Chavira, M., and Darwiche, A. 2005. Compiling Bayesian networks with local structure. In *IJCAI’05*, 1306–1312.
- Darwiche, A. 2000. Recursive conditioning. *AIJ* 125(1-2):5–41.
- Dechter, R. 1999. Bucket elimination: A unified framework for reasoning. *AIJ* 113:41–85.
- Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.
- Hyafil, N., and Bacchus, F. 2003. Conformant probabilistic planning via CSPs. In *ICAPS’98*, 205–214.
- Hyafil, N., and Bacchus, F. 2004. Utilizing structured representations and CSPs in conformant probabilistic planning. In *ECAI’04*, 1033–1034.
- Jensen, F. 1996. *An Introduction to Bayesian Networks*. Springer Verlag.
- Kushmerick, N.; Hanks, S.; and Weld, D. 1995. An algorithm for probabilistic planning. *AIJ* 78(1-2):239–286.
- Littman, M.; Goldsmith, J.; and Mundhenk, M. 1998. The computational complexity of probabilistic planning. *JAIR* 9:1–36.
- Majercik, S., and Littman, M. 1998. MAXPLAN: A new approach to probabilistic planning. In *AIPS’98*, 86–93.
- Majercik, S., and Littman, M. 2003. Contingent planning under uncertainty via stochastic satisfiability. *AIJ* 147(1-2):119–162.
- Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- Roth, D. 1996. On the hardness of approximate reasoning. *AIJ* 82(1-2):273–302.
- Sang, T.; Beame, P.; and Kautz, H. 2005. Solving Bayes networks by weighted model counting. In *AAAI’05*.
- Sang, T.; Bacchus, F.; Beame, P.; Kautz, H.; and Pitassi, T. 2004. Combining component caching and clause learning for effective model counting. In *SAT’04*.
- Zhang, N., and Poole, D. 1994. A simple approach to Bayesian network computations. In *AI’94*.