A Relevance-Based Compilation Method for Conformant Probabilistic Planning

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

Conformant probabilistic planning (CPP) differs from conformant planning (CP) by two key elements: the initial belief state is probabilistic, and the conformant plan must achieve the goal with probability $\geq \theta$, for some $0 < \theta \leq 1$. In earlier work we observed that one can reduce CPP to CP by finding a set of initial states whose probability $\geq \theta$, for which a conformant plan exists. In previous solvers we used the underlying planner to select this set of states and to plan for them simultaneously. Here we suggest an alternative approach: start with relevance analysis to determine a promising set of initial states on which to focus. Then, call an off-the-shelf conformant planner to solve the resulting problem. This approach has a number of advantages. First, instead of depending on the heuristic function to select the set of initial states, we can introduce specific, efficient relevance reasoning techniques. Second, we can benefit from optimizations used by conformant planners that are unsound when applied to the original CPP. Finally, we are free to use any existing (or new) CP solver. Consequently, the new planner dominates previous solvers on almost all domains and scales to instances that were not solved before.

Introduction

In conformant probabilistic planning (CPP) we are given a set of actions – which like most past work, are assumed to be deterministic, a distribution over initial states, a goal condition, and a real value $0 < \theta \leq 1$. We seek a plan π such that following its execution, the goal is achieved with probability $\geq \theta$. Not many natural problems fit the frameworks of conformant planning (CP) and CPP, yet both serve as basic frameworks on which ideas for planning under uncertainty can be developed and tested. Indeed, important ideas developed in CP were later extended to the richer framework of contingent planning, including techniques for representing and reasoning about belief states (Hoffmann and Brafman 2005) and various translation schemes (Albore, Palacios, and Geffner 2009). The latter represent an important trend in research on planning under uncertainty, where simple planners are used to solve more complex problems (Yoon, Fern, and Givan 2007; Palacios and Geffner 2009; Albore, Palacios, and Geffner 2009).

This paper utilizes this reduction idea to develop an effective reduction scheme for solving CPP problems with deterministic actions, motivated by our observation 2013: One can solve CPP by finding a set of initial states with joint probability $\geq \theta$ for which a conformant plan exists. In our earlier work, we used this idea to build a translation-based CPP solver that has special actions that let the planner ignore a certain state in the solution. Each such action carries a cost equal to the probability of the ignored state. A plan with cost $\leq 1 - \theta$ is a solution to the original CPP problem. This technique has several weaknesses. First, the choice of which states to ignore is carried out by the underlying planner's heuristic function. It is not clear that this function is well suited for this task. Second, the planner need not minimize cost, but rather, ensure that a certain cost bound is met. Few current planners or search techniques support this optimization criterion. Finally, various optimization schemes used by CP solvers are unsound under this scheme.

We suggest an alternative, simpler approach where, first, dedicated relevance-based preprocessing analysis is conducted, determining a promising set of states with probability $\geq \theta$ to plan on. Then, this state set is given to an off-the-shelf conformant planner as its initial state. Besides addressing the above shortcomings of our older method, this method provides the flexibility of selecting the underlying conformant planner to suit the current planning domain. Our empirical evaluation shows that this approach dominates existing state-of-the-art planners on almost all problem instances.

In the next section, we provide some required background on CPP. Then, we discuss related work, followed by a formal description of our planner and its properties. We conclude with an empirical evaluation and a discussion of our results and potential future work.

Conformant Probabilistic Planning

We assume familiarity with the basic notation of classical planning domains via STRIPS with conditional effects: (V, A, I, G), corresponding to a set of *propositions*, *actions*, *initial world state*, and *goal*. A CP problem, (V, A, b_I, G), generalizes this framework, replacing the single initial state with a set of initially possible states, called the *initial belief state* b_I. This initial state is often described by means of a formula φ_I, such that $b_I = \{w | w \models \varphi_I\}$. A plan is an action sequence $\overline{a}$ such that $\overline{a}(w_I) \supseteq G$ for every $w_I \in b_I$.

CPP extend CP by quantifying the uncertainty regarding b_I using a probability distribution b_{π_I}. In its most general form, CPP allows for stochastic actions, but we leave this to future work, and assume all actions are deterministic. CPP tasks are 5-tuples $(V, A, b_{\pi_I}, G, \theta)$, corresponding to the *set of propositions*, *set of actions*, *initial belief state*, *goals*, and *acceptable goal satisfaction probability*. As before, G is a conjunction of propositions. b_{π_I} denotes a probability distribution over the world states, where $b_{\pi_I}(w)$ is the probability that w is the true initial world state.

In many settings, achieving G with certainty is impossible. CPP introduces the parameter θ, which specifies the required *lower bound* on the probability of achieving G. A sequence of actions $\overline{a}$ is called a *plan* if the weight of the initial states from which $\overline{a}$ reaches the goal is at least θ.

Some approaches (*PFF's*,for instance) to CPP require that the plan be executable in *all* initial states, even those from which it does not reach the goal. That is, each plan prefix must be conformant (with probability 1) with respect to the preconditions of the next action. This extra requirement may make sense in domains where executing an action without satisfying its preconditions may have catastrophic consequences. However, it seems to conflict with the very explicit success criterion of CPPs specified by the parameter θ, and we adapt this requirement as the only requirement from a plan for a CPP problem in our work.

A CPP specification language must provide a way to specify the initial distribution b_{π_I}. Following previous work (Domshlak and Hoffmann 2007) we assume that b_{π_I} is specified using a Bayes net (BN) $\mathcal{N}_{b_I}$. (We assume the reader is familiar with the notion of BN's.) BNs are typically defined over a set of multi-valued variables, and there are numerous formats used in the literature for their specification, all having the same semantics. Here, we adopt the notation used by the PFF planner (Domshlak and Hoffmann 2007) which contains two parts. First, a definition of an induced set of multi-valued variables, corresponding to a set of literals, only one of which can be true at a time (e.g., literals denoting possible locations of an object). And second, $\mathcal{N}_{b_I}$, which is a BN defined over this set of multi-valued variables.

Related Work

The best current CPP solvers are *Probabilistic FF* (PFF) (Domshlak and Hoffmann 2007) and PCBP (Taig and Brafman 2013), but each works well only on a subset of benchmark domains. Probabilistic-FF uses a time-stamped Bayesian Networks (BN) to describe probabilistic belief states, extending Conformant-FF's (Hoffmann and Brafman 2006) belief state encoding to model these BN. It uses SAT reasoning (following Conformant-FF) and weighted model-counting to determine whether goal probability is at least θ. In addition, it introduces approximate probabilistic reasoning into Conformant-FF's heuristic function. PFF performs well on many domains, but it is sensitive to the syntax of conditional effects (the order of literals in effect conditions), and requires complex probabilistic reasoning. To model the belief state during planning, it extends the original BN into a time stamped dynamic (and thus continually growing) BN, on which it performs probabilistic inference by compiling probabilistic queries into WMC queries. The algorithm we introduce reasons with a single, static BN that represents the initial state probability. Reasoning is performed at the pre-processing time only, in order to assess initial state restrictions. No probabilistic queries or data structures need be maintained at planning time, leading to significant saving of time and memory.

Closely related to our work are the *CLG+* planner (Albore and Geffner 2009) and the assumption-based planning approach introduced recently (Davis-Mendelow, Baier, and McIlraith 2013). Both attempt to solve planning problems with incomplete information in which goal achievement cannot be guaranteed from all states, by making assumptions that reduce the uncertainty, and planning under these assumptions. However, neither planner uses an explicit probabilistic semantics – a core part of CPP. In *CLG+*, assumptions are made online w.r.t. information gathered during planning and sensing, while our assumptions are made by an efficient analysis preprocessing phase. (Davis-Mendelow, Baier, and McIlraith 2013) consider the idea of planning under assumptions in general, but do not describe a particular strategy for selecting them, possibly allowing a domain expert to specify them.

Another line of related work by Tran et al. (Tran et al. 2009) and Nguyen et al. (Nguyen et al. 2011) considers methods for reducing the initial belief state by a preprocessing analysis phase while ensuring that the resulting plan applies to all original states. Thus, this is a completeness preserving, sound optimization method that can be used by our underlying conformant planner.

Relevance-Based Reduction

PCBP is the planner most relevant to our current work. It is a relevance-based translation-based planner that exploits the following Lemma:

Lemma 1 (Taig & Brafman, 2013) *A CPP* $\mathcal{CP} = (V, A, b_{\pi_I}, G, \theta)$ *is solvable iff there exists a solvable CP problem* $\mathcal{C} = (V, A, b_I, G)$ *such that* $b_{\pi_I}(\{w \in b_I\}) \geq \theta$. *Moreover,* $\overline{a}$ *is a solution to* $\mathcal{CP}$ *iff it is a solution to* $\mathcal{P}$.

PCBP augments the regular set of actions with special actions — one per possible initial state. Each such action allows the planner to ignore a possible state by essentially making the goal easy to achieve from that state. The cost of each such action equals the initial probability of the corresponding state, while all other actions have zero cost. A plan for the new problem with cost $\leq 1 - \theta$ represents a solution to the original CPP because the set of states from which it fails to reach the goal has probability $\leq 1 - \theta$.

PCBP performs well on a number of domains, but almost completely fails on others (such as logistics, grid, rovers). It suffers from a number of weaknesses — some inherent, and some tied to the limitation of existing translation methods and search methods. First, the choice of which states to ignore is essentially carried out by the planner's heuristic function, because it is implemented by the action selection mechanism. It is not clear that this function is well suited for this task. Second, the planner needs to solve a cost-bounded planning problem which few current planners or search techniques support. Third, there are various optimization that existing CP solvers carry out which are unsound in this scheme. For example, the completeness preserving pruning methods mentioned above, or T_0's relevance-analysis (Palacios and Geffner 2009) which identify clauses and propositions that can be safely ignored during planning. These techniques are unsound for CPPs. Finally, PCBP's reduction does not work if actions have cost and we seek a (truly) cost-optimal, or cost-efficient plan.

The RBPP Planner

To overcome these problems, we suggest a simpler, preprocessing approach. First, relevance-based analysis is carried out to identify those states which would be most profitable to ignore. Then, a CP problem is defined in which the initial state consists of all states that are not ignored. This is given to an off-the-shelf CP solver. This method addresses the above shortcomings of PCBP, and in addition gives us the flexibility to choose which conformant planner to use. While in this paper we do not attempt to provide an automated portfolio-based method, our empirical analysis indicates that different solvers have advantages in different domains, showing the potential for farther improvement by automatically selecting the underlying conformant planner which best suits the characteristics of the problem in hand.

Algorithm 1 RBPP (P, *conf-planner*)

```
$\psi_I \Leftarrow$ RESTRICT($P$);
**return** *conf-planner*($\widetilde{P} = (V, A, \psi_I, G)$);
```

Algorithm 2 RESTRICT (P)

```
$Q \Leftarrow$ SORT-CLAUSES($P$);
$\psi_I \Leftarrow \varphi_I$;
**while** ($b_{\pi_I}(\psi_I) \geq \theta$) **do**
    $C \Leftarrow$ *Extract-First*($Q$);
    $\psi_I \Leftarrow$ *RESTRICT-CLAUSE*($C, \psi_I, P$);
**end while**
**return** $\psi_I$;
```

Algorithm 3 SORT-CLAUSES (P)

```
**return** A sorted list of all non-unit clauses in $\varphi_I$ according to
the following parameters order:
1. RL(C) (high to low).
2. Rad(C) (high to low).
3. PI(C) (High to low).
4. RP(C). (low to high).
5. prefer, if exist, Clauses $C$ where: $C \cap G \neq \emptyset$.
6. If all previous parameters equal – choose arbitrarily.
```

First, our algorithm generates an initial belief-state formula φ_I from the BN describing the initial belief state $\mathcal{N}_{b_I}$. φ_I is constructed as follows: for every multi-value variable X whose initial value is uncertain, φ_I contains a *One-Of* clause C_X with one literal for every value of X possible (i.e., which has probability > 0). In addition, if $Pr(X = x|Parents(X) = y) = 0$ (i.e., $X = x$ is not possible when $Pa(X)$ have value y, an additional *Or* clause expressing $Pa(X) = y \rightarrow \neg(X = x)$ is added. This formula can be generated in linear time using a single top down traversal of the BN. The following is immediate:

Lemma 2 $\varphi_I \models w$ *iff* $b_{\pi_I}(w) > 0$.

Next, we seek to simplify φ_I by ignoring a set of states with probability at most $1 - \theta$. Thus, at this stage we operate on the structure: $P = (V, A, \varphi_I, \mathcal{N}_{b_I}, G, \theta)$ which denotes the original CPP together with the generated initial state formula. Algorithm 1 describes the high-level structure of RBPP – our *relevance-based probabilistic planner.* We denote the formula expressing the set of states valid after the simplification of φ_I as ψ_I. We also use: $b_{\pi_I}(\varphi)$ to denote $\Sigma_{\varphi \models w} b_{\pi_I}(w)$.

The main element – relevance-based analysis – is carried out by the *Restrict* procedure (Algorithm 2) which determines which initial states to ignore. This analysis is heuristic, and it is strongly motivated by the notions of conformant width and relevance developed by (Palacios and Geffner 2009) in the context of conformant planning.

First, *Restrict* sorts the clauses. Then, at each iteration it selects the first non-unit clause (i.e., a clause expressing uncertainty) in the list and restricts it by dropping some literal(s) using Algorithm 4. This results in a smaller clause satisfied by fewer states, hence ignoring those states satisfied by the dropped literal. The procedure is repeated until no further restrictions are possible given the probability bound. We note that each restriction is made only after we

Algorithm 4 RESTRICT-CLAUSE (C, ψ_I, P)

```
**while** ($|C| > 1 \wedge b_{\pi_I}(\psi_I) \geq \theta$) **do**
    $C' \leftarrow C$;
    choose next proposition $p \in C'$ by the following order:
1.      $p \notin g$.
2.      $b_{\pi_I}(\psi_I \wedge \neg p) \geq \theta$.
3.      Prefer not removing $p \in DRC(C)$ before other
        propositions.
4.      $b_{\pi_I}(p)$ - The initial probability of p. (low to high).
5.      If all previous parameters equal - choose arbitrarily.
    $C' \leftarrow C \setminus p$;
    $\psi_I \leftarrow \psi_I \setminus C \cup C'$;[1]
**end while**
**return** $\psi_I$;
```

check (using a standard BN software package) that the probability of the removed initial states does not exceed $1 - \theta$.

Restrict maximally restricts a clause before moving on to the next clause. We have found this priority to be most useful, as there appears to be greater advantage to making a single clause much smaller than to making multiple clauses smaller. Clauses are sorted based on a prioritized list of parameters, as shown in Algorithm 3. These parameters attempt to assess the impact of a clause on the difficulty of solving a problem. Future work could improve this by adapting the parameters and their priority to take into account their effect on the underlying conformant planner.

Choice of Clauses Our analysis is strongly motivated by the notion of *conformant relevance* (Palacios and Geffner

[1]For clarity, we use set notation, treating ψ as a set of clauses and treating clauses as a set of literals. Here, p is removed from C to obtain C', and C is replaced by C' to obtain the updated ψ_I.

	$\theta = 0.25$				$\theta = 0.5$				$\theta = 0.75$					
	t/l				t/l				t/l					
Task	PFF	PCBP	RBPP[T-0]	RBPP[CFF]	PFF	PCBP	RBPP[T-0]	RBPP[CFF]	PFF	PCBP	RBPP[T-0]	RBPP[CFF]	RBPP*[T-0]	RBPP*[CFF]
Safe-uni-70	2.3/18	**0.02/18**	0.07/18	0.1/18	5.5/35	**0.05/35**	0.1/35	1.13/35	9.6/53	6.4/70	**0.07/53**	1.44/**53**	0.07/53	1.03/53
Safe-cub-70	0.07/5	**0.03/5**	0.09/5	0.06/5	1.62/**12**	**0.04**/21	0.07/**12**	0.07/**12**	2.92/21	**0.04**/21	0.08/**12**	0.08/**12**	0.08/12	0.08/12
NC-Safe-uni-70	**1.15/5**	1.41/12	1.39/**5**	**1.13/5**	2.28/**12**	1.47/**12**	**1.39/12**	2.31/**12**	4.92/21	1.56/20	1.39/20	**1.3/20**	1.1/20	1.1/20
Cube-uni-corner-15	3.44/**26**	OOT	**0.11**/36	2.37/36	4.7/**33**	OOT	**0.07**/42	4.34/44	6.9/**38**	OOT	**0.07**/42	4.29/42	0.07/42	3.31/42
Cube-cub-corner-15	1.86/**20**	OOT	**0.06**/37	2.93/37	4.2/**28**	OOT	**0.09**/37	3.01/37	5.03/**33**	OOT	**0.1**/40	3.32/40	0.1/40	2.79/40
Cube-uni-center-15	OOT	OOT	**6.15/70**	OOT	OOT	OOT	**5.27/68**	OOT	OOT	OOT	**6.94/56**	OOT	5.88/56	OOT
Cube-cub-center-15	OOT	OOT	**6.02/64**	OOT	OOT	OOT	**5.72/61**	OOT	OOT	OOT	**3.87/61**	OOT	3.40/61	OOT
NC-cube-cub-15	5.22/**20**	OOT	**2.93**/37	10.42/37	2.94/**27**	OOT	**0.86**/37	2.63/37	5.74/**33**	OOT	**1.29**/37	2.37/37	1.19/37	2.21/37
NC-cube-uni-15	2.28/**26**	OOT	**0.08**/34	**0.08**/34	13.98/47	OOT	**3.06/34**	4.00/34	NP	NP	NP	NP	NP	NP
Push-Cube-uni-15	7.39/50	2.08/**42**	1.21/66	**0.06**/57	27.28/66	2.33/**45**	1.79/72	**0.09**/63	55.21/74	2.28/**46**	1.81/74	**0.11**/63	1.92/74	0.12/63
Push-Cube-cub-15	0.08/**15**	1.48/28	0.07/42	**0.05**/33	0.09/**17**	3.88/37	1.26/58	**0.05**/48	0.09/**18**	2.09/37	1.34/59	**0.09**/48	1.36/59	0.14/48
Bomb-50-50	0.01/0	0.01/0	0.01/0	0.01/0	0.1/**16**	9.02/50	**0.09**/50	6.13/50	0.2/**36**	8.4/90	**0.07**/50	6.22/50	0.06/50	5.78/50
Bomb-50-10	0.01/0	0.01/0	0.01/0	0.01/0	2.89/**22**	8.4/90	**0.04**/90	1.43/**22**	5.74/63	8.4/90	**0.04**/90	5.59/63	0.02/90	4.66/63
Bomb-50-5	0.01/0	0.01/0	0.01/0	0.01/0	1.94/**27**	8.6/95	**0.04**/95	1.83/**27**	8.02/**63**	9.14/95	**0.04**/95	7.07/67	0.04/95	7.81/67
Bomb-50-1	0.01/0	0.01/0	0.01/0	0.01/0	2.21/**31**	5.02/49	**0.03**/100	1.88/**31**	10.12/**71**	4.8/74	**0.05**/100	7.9/**71**	0.05/100	7.14/71
10-Log-2	3.73/**72**	OOT	4.51/85	**2.79**/77	3.17/79	OOT	4.97/84	**2.56/77**	**2.46/80**	OOT	3.13/**80**	5.3/87	4.9/86	5.48/87
10-Log-3	7.47/64	OOT	**5.16/57**	2.83/58	35.4/98	OOT	8.98/**77**	**3.23/77**	8.91/99	OOT	24.28/123	**4.12/91**	13.71/146	4.49/91
10-Log-4	8.35/75	OOT	8.41/**47**	**4.59/47**	34.2/81	OOT	13.29/78	**5.38/70**	12.09/**95**	OOT	36.53/111	**7.43/95**	95/180	50/150
15-Log-4	OOT	OOT	OOT	**15.02/98**	OOT	OOT	OOT	**28.31/138**	OOT	OOT	OOT	**32.5/154**	OOT	OOT
2-planes-log-3	19.81/84	OOT	14.09/**64**	**7.62/69**	57.98/97	OOT	31.24/104	**18.72/89**	**10.12**/112	OOT	229/134	13.7/**109**	OOT	OOT
2-planes-log-4	OOT	OOT	23.19/62	**12.64/57**	OOT	OOT	74.03/101	**17.91/87**	OOT	OOT	522/145	**47.52/107**	OOT	OOT
2-planes-C	OOT	OOT	OOT	**3.65/83**	OOT	OOT	OOT	**4.92/83**	OOT	OOT	OOT	**14.06/108**	OOT	15.22/108
grid-uni-2	**0.07/21**	OOT	4.33/75	3.15/47	**1.35/48**	OOT	OOT	3.18/53	6.11/69	OOT	OOT	**5.41/66**	OOT	6.47/65
grid-uni-3	16.01/**76**	OOT	**6.22**/86	41.14/86	15.8/**89**	OOT	**5.93**/96	127/103	82.24/123	OOT	**7.26/102**	OOT	OOT	166/105
grid-uni-4	28.15/**96**	OOT	**6.78**/111	134.21/115	**51.58/111**	OOT	OOT	247.58/117	**50.80/115**	OOT	OOT	721/146	OOT	OOT
grid-cub-3	OOT	OOT	18.14/40	**9.47/30**	OOT	OOT	31.14/43	**16.11/35**	OOT	OOT	**27.71/64**	39.42/76	OOT	90.05/88
grid-cub-4	OOT	OOT	**10.8/77**	83.84/92	OOT	OOT	**9.37/78**	114/92	OOT	OOT	**14.95/85**	200/106	26.11/112	243/106
Rovers-3	0.04/14	0.06/19	**0.02/12**	**0.02/12**	0.05/17	0.06/19	**0.02/12**	**0.02/12**	0.06/**18**	0.06/25	**0.02**/20	0.03/24	0.14/60	1.83/24
Rovers-7	5.72/65	4.12/54	**0.04**/47	**0.04/41**	5.48/75	4.14/54	OOT	**0.07/56**	6.55/83	10.74/88	OOT	**2.09/72**	OOT	7.14/72
C-Rovers-PP-7	10.54/65	OOT	OOT	**1.1/41**	15.4/75	OOT	OOT	**1.12/56**	39.1/77	OOT	OOT	**2.23/72**	OOT	OOT
C-Rovers-PPP-7-3-ID	3060/68	OOT	OOT	**3.31/47**	OOT	OOT	OOT	**11.34/71**	OOT	OOT	OOT	**127.53/96**	OOT	OOT
C-Rovers-PPP-NC-7	30.11/67	OOT	OOT	**3.12/41**	46.3/79	OOT	OOT	**4.14/57**	NP	NP	NP	NP	NP	NP

Table 1: Empirical results. t: time in seconds. l: plan length. Entries marked *OOT* means the search did not return after 30 minutes. 'NP'-no plan for this goal probability exists.

2009) (PG), and many of our definitions are adaptation of their concepts to the probabilistic case. Relevance is a transitive relation between facts expressing whether uncertainty about the value of one fact affects our knowledge about the value of the other. The relation propagates only through conditional effects of actions. Our relevance analysis focuses on facts relevant to subgoals, and while it is possible to consider also facts relevant to preconditions, we found the overhead of these computations costly.

Definition 1 (Conformant Relevance (PG 2009))
1. p is relevant to p.
2. p is relevant to q if there exist an action $a \in A$ with conditional effect $C \rightarrow q$ and $p \in C$.
3. If p is relevant to r and r is relevant to q then p is relevant to q.
4. p is relevant to q if p is relevant to $\neg r$ and r is relevant to $\neg q$.

We extend this relation to relevance between an initial non-unit clause and a (goal) fact as follows:

Definition 2 (probabilistic clause relevance) *A non unit clause $\varphi \in \varphi_I$ is considered relevant to a proposition p if $\exists q \in \varphi$ s.t $q \in rel(p)$.*

PG consider φ relevant to p only if all propositions of φ are in $rel(p)$. The latter yields fewer relevant propositions but cannot be extended to the probabilistic case. Suppose, for example, that we have *One-Of(p,q)* in φ_I. Assume p is relevant to some goal fact g but q is not. Assume also that $b_{\pi_I}(p) = 0.5$ and $\Theta = 0.4$. In the conformant case, this clause may be ignored, but in CPP, because we do not need to succeed with certainty, we must be take into account the possibility to plan from all p states, even though planning from all q is infeasible.

Definition 3 (relevant clauses set) $\forall g \in G : rel(g)$ *contains all the clauses from φ_I relevant to g;*

$rel(g)$ expresses the maximal amount of initial uncertainty relevant to a specific goal fact. $Max_{g \in G}\{|rel(g)|\}$ is closely related to the notion of *conformant width* introduced by PB. They show that the complexity of their reduction-based planning algorithm grows exponentially with this parameter. This effect on the problem's complexity stems from the fact that the solver must potentially consider initial states (or more accurately, initial state-sets) that correspond to all possible assignments to $rel(g)$ variables to compute the current probability of g, in the probabilistic case, and to determine whether g is valid, in the standard conformant case. And while this value was suggested by PG in the context of their analysis of the *T-0* planner, it appears relevant to the performance of other conformant planners, such as CFF, as our empirical analysis shows.

There are two technical differences between our $rel(g)$ and PG's conformant width: First, we use probabilistic clause relevance, explained earlier. Second, PG compute the closure of $rel(g)$, called $C_I(L)^*$. while we this costly computation, using $rel(g)$ instead. For our heuristic usage, this provides a better tradeoff, and in fact, $rel(g) = C_I(L)^*$ in almost all experimented benchmarks.

Given the above theoretical and empirical justifications for relation between $rel(g)$ and problem hardness, our most important parameter for assessing clauses is $RL(C)$:

Definition 4 (clause relevance level) $RL(C) = MAX\{|rel(g)|\}_{g\in G\wedge C\in rel(g)}$.

Relevance sets can also be used to understand how many goal facts are influenced by the uncertainty expressed by a clause. Restricting clauses that affect many goal facts is likely to be better, motivating the following definition:

Definition 5 (clause radius) $Rad(C) = \#g \in G\ s.t\ C \in rel(g)$.

Example: Consider the *logistics* domain with 10 cities,10 trucks, 10 packages, and one plane. Initially, there is one package and truck per city. There are 10 sub-goals specifying the final location of each package. Actions have no pre-conditions, but only conditional effects. The initial location of trucks and the plane are unknown. Each truck can be in one of three possible locations in a city. Each truck's location uncertainty is relevant only to one sub-goal (corresponding to the package in this city). Uncertainty regarding the plane is relevant to all goals. $RL(C) = 2$ for all clauses, while the radius of the clause expressing the plane's uncertainty is 10, and that of other clauses is 1. Restricting this clause will prevent the conformant planner from repeatedly reasoning about the plane's location. It is likely to yield a simpler plan, too, as there is no need for extra flights that will ensure a known location to the plane.

Relevance analysis ignores the initial probabilistic distribution. $PI(C_X)$ takes this information into account. It is a rough and tractable estimate of the effect evidences on X (e.g restrictions on C_X's literals) might have on our knowledge regarding C_Y's literals.

Definition 6 (Probabilistic Influence) *Let* $X \in \mathcal{N}_{b_I}$ *be the corresponding node of a clause* C_X. *We define:*
$RelChildren(X) = \{Y|X \in Parents(Y) \wedge \exists g \in G\ s.t\ C_Y \in rel(g)\}$
$PI(C_X) = |RelChildren(X)|$.

Potentially, restricting C_X might induce immediate restrictions on C_Y which, in turn, reduces the relevant uncertainty in the resulting conformant problem. Ideally, our definition should have been transitive, but again, to limit the cost of the restriction heuristic, we consider immediate parents only. In order to exploit this information and monitor such effects, after each restriction of a clause C_X s.t $PI(C_X) > 0$, we query $\mathcal{N}_{b_I}$ to check whether any of the literals in C_Y for each $Y \in RelChildren(X)$ has probability 0. In that case we restrict C_Y accordingly.

Example (continued): Suppose that the plane's initial location distribution is dependent on the initial distribution on the weather in some city *city*. Other than that, the weather does not influence other problem variables and does not appear in action descriptions. The relevance analysis will not identify the "weather" clause C_w as having any importance. However. since $PI(C_w) = 1$, while for all other clauses $PI(C) = 0$, the algorithm might choose to restrict C_w, setting the weather in *city* to *extreme*. Given this value there is 0 probability for the plane to be in *city* with the effect of restricting the clause expressing the plane's location uncertainty – C_p. This can help us in cases where θ and the initial distribution limits our ability to directly restrict C_p but allow the restriction of C_w.

Restrictions come with a cost: the probability mass we lose. We wish to restrict as many propositions as possible, so we prefer restrictions that carry lower probability "cost".

Definition 7 (clause restriction potential) *RP(C)* $= Min\{b_{\pi_I}(p) \mid p \in C\}$. *(By* $b_{\pi_I}(p)$ *we mean The initial probability of p).*

$RP(C)$ improves our ability to identify, in advance, clauses that are potentially more attractive for restriction. As such, it serves as a good tie-breaker. Note that if the node corresponding to C in $\mathcal{N}_{b_I}$ is not barren, this parameter is ignored.

Clause Restriction First, we make sure facts that can negatively affect the completeness or quality of solution are not removed, e.g., goal facts. Second, sometimes only a subset of facts are responsible for making the clause relevant to a goal and removing them can make the goal unreachable. Thus, we define:

Definition 8 (Direct Relevance causes) *DRC(C)* $= \{p \in C \mid \exists g \in G :$ *p is relevant to g*$\}$.

If, for a clause C: $DRC(C) \subsetneq C$ a plan might not exist from all assignments to C. Thus, we don't remove facts from $DRC(C)$ unless no other option exists.

Finally, we prefer to remove facts whose initial probability is smaller, to leave more probabilistic mass for farther restrictions but once again, if the node corresponding to $C's$ in $\mathcal{N}_{b_I}$ is not barren, the latter parameter is ignored.

Properties

Soundness: Our algorithm is sound if the underlying CP solver is sound. This follows from Lemma 1, provided we ensure that the probability of our new initial state $\geq \theta$. Each clause restriction is equivalent to ignoring a possible value of a variable. To accept such a restriction, we compute the probability of the set of ignored states, ensuring it does not exceed $1 - \theta$. When multiple values are ignored, we use the standard probabilistic semantics to compute their aggregated weight. (E.g., if we ignore $\neg p$ and then $\neg q$, we compute the weight of $\neg p$, and add the weight of $\neg q \wedge p$).

Completeness: Our algorithm is incomplete because we do not attempt to systematically examine all possible restrictions with weight $\leq 1-\theta$. Conceptually, it is a simple matter to add an outer loop that will make the algorithm complete, (e.g., as used by some greedy algorithms to provide theoretical completeness). Practically (and theoretically) this is a potentially super-exponential algorithm and we see little value in it (unlike soundness), as it will not help us scale up any better. As our experiments demonstrate, our algorithm does scale up better than earlier methods.

Complexity: There are three elements contributing to the complexity to our algorithm: the algorithm for selecting possible initial-state restrictions, the BN queries, and the CP solver. In theory, there are super-exponentially many possible restrictions to the initial state and the computation of each one's probability can be NP-hard. In practice, we use a heuristic low-order polynomial time algorithms to identify promising restrictions. For this we pay by sacrificing completeness. To check the validity of restrictions, we compute marginal distribution of variables in a BN, which is NP-hard (Cooper 1990). In practice, the queries we seek to compute are very simple, and are not likely to be the bottle-neck even when we go beyond current CPP benchmarks which feature very simple initial state BNs. Solving CP is a difficult problem (Haslum and Jonsson 1999), and this appears to be the more significant bottleneck in practice. This, of course, is to be expected, as we are solving a problem that is at least as difficult as CP. Thus, we have focused on making the reduction process simple and fast.

Empirical Evaluation

We implemented the algorithms and experimented with them on a variety of domains taken from *PFF* repository and the 2006 IPPC, as well as new, modified versions of these domains. Beyond our tools, we used a modification of PG's $cf2cs$ code for the relevance analysis and *NORSYS NETICA java api* program for the Bayesian net creation and reasoning. As the underlying conformant planner for RBPP we used both *T-0* and *CFF*. GC[LAMA] (Nguyen et al. 2012) is unable to handle most of the domains we tested on. The two *RBPP* variants were compared with state-of-the-art CPP planners: *PFF* and *PCBP*, We have not compared against *POND* (Bryce, Kambhampati, and Smith 2006) as *PFF* outperforms it. We note that in almost all experimented benchmarks there are very few (if any) preconditions with relevant uncertainty and thus, comparison to *PFF* is fair. Results are presented in Table 1. Each task was tested with three different θ value and two different initial state distributions: uniform (*uni*) and cubic(*cub*). In order to demonstrate the significance and generality of our main parameter, $RL(C)$, we included (in the case of $\theta = 0.75$) results for $RBPP^*$ which is identical to $RBPP$ except that $RL(C)$ is not used to evaluate potential restrictions. Results for this variant show that ignoring $RL(C)$ has a detrimental effect in the more complex benchmarks, both on plan quality and execution time and that in most benchmarks with high width, $RBPP^*$ fails to scale up. Results show dominance of *RBPP[T-0]* on almost all benchmarks, in many cases by an order of magnitude. An interesting observation is that in most cases *RBPP[CFF]* performs better than *PFF* which is a specially designed extension of Conformant-FF's. This demonstrates the effectiveness of the compilation approach over direct probabilistic reasoning, as used by *PFF*. On some of the simplest tasks such as *safe-70 PCBP* has a small advantage due to the overhead of relevance analysis relative to solution time in this problem. In more complex problems, such as *cube-15-corner* (an agent can initially be in any of the cube's 15^3 locations and needs to navigate to a corner) our planners run-time dominance is clear, although *PFF*'s plans are shorter. When, in the same setting, the agent needs to plan to the center of the cube, *RBPP[T-0]* produces good plans in a matter of seconds, while all other planners fail. Similar performance is seen in the *push-cube* domain (Taig and Brafman 2013). Here, a player must push a ball with unknown initial position to some location on a grid. It can select *(position, direction)* pairs, and if the ball is in *position*, it moves in *direction*. Solving this problem requires many actions, and *RBPP* is the clear winner. The *m-logistics-n* problem refers to logistics with m packages and m cities of size n. Here, too, the various *RBPP* versions are faster and generate shorter plans, when we set $m = 15$ only *RBPP[CFF]* scales up. The *2-planes* variation of logistics has 2 planes, instead of 10, as well as uncertainty on their initial location. Both *rovers* and *grid* have large probabilistic width, and our relevance analysis is crucial. Results here are mixed. As the uncertainty grows (*grid-4*,*rovers-7*) both *PFF* and *RBPP[T-0]* fail to scale up. *RBPP[CFF]* is the only planner that manages to solve all tasks, typically dominating other planners.

We also experimented on domains with no conformant plan (marked *NC*). For example, we modified *cube* such that some transitions between adjacent locations are blocked so the agent cannot reach the goal location if initially located beyond the blocked path. These experiments demonstrate the ability of RBPP's preprocessing phase to select initial states from which the goal is reachable.

Another important line of experiments is on the problems marked with *C*. In these problems the initial state distribution is more complex and involves dependencies. *2-planes-C* reflects the example given in this paper. In rovers-pp, object visibility from a waypoint depends on whether or not a rock sample is located at the waypoint. The probability of visibility is much higher if the latter is not the case. Rovers-PPP extends RoversPP by introducing the need to collect data about water existence. Each soil sample has a certain probability to be wet. For communicated sample data, an additional operator tests whether the sample was wet. The probability of being wet depends on sample location. In the instance marked "ID," one of the 3 samples must be wet, increasing dependencies to a level that only *RBPP[CFF]* can handle. In the instance marked *NC*, there is no guarantee that one of the 2 samples is wet, and thus, there is no conformant plan. Here, *RBPP[CFF]* scales better than *PFF* by an order of magnitude. Overall *RBPP[CFF]* appears less sensitive to the conformant width than RBPP[T-0].

Conclusion and Future work

We presented a new approach to CPP where relevance analysis and heuristics are used before planning to identify states that can likely be ignored in planning and use conformant planning to solve the problem of the remaining states. Our empirical evaluation shows that this method is the strongest CPP algorithm currently, with better coverage and scaling than previous approaches.

Presently, we make no attempt to automatically select the underlying conformant solver. In future work, we intend to attempt to identify problem features that can help predict which planner will perform better, and to use these within a portfolio-based solver. In addition, we intend to extend the approach presented here to handle CPP domain with stochastic actions. A first step is to simply consider deterministic version of these actions, and to be able to verify that the solution addresses a sufficiently large probability mass.

Acknowledgments The authors were supported in part by ISF grant 933/13 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.

Bryce, D.; Kambhampati, S.; and Smith, D. E. 2006. Planning graph heuristics for belief space search. *J. Artif. Intell. Res. (JAIR)* 26:35–99.

Cooper, G. F. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artif. Intell.* 42(2-3):393–405.

Davis-Mendelow, S.; Baier, J. A.; and McIlraith, S. A. 2013. Assumption-based planning: Generating plans and explanations under incomplete knowledge. In *AAAI*.

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 Jonsson, P. 1999. Some results on the complexity of planning with incomplete information. In *ECP*, 308–318.

Hoffmann, J., and Brafman, R. I. 2005. Contingent planning via heuristic forward search witn implicit belief states. In *ICAPS*, 71–80.

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

Nguyen, H.-K.; Tran, D.-V.; Son, T. C.; and Pontelli, E. 2011. On improving conformant planners by analyzing domain-structures. In *AAAI*.

Nguyen, H.-K.; Tran, D.-V.; Son, T. C.; and Pontelli, E. 2012. On computing conformant plans using classical planners: A generate-and-complete approach. In *ICAPS*.

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

Taig, R., and Brafman, R. I. 2013. Compiling conformant probabilistic planning problems into classical planning. In *ICAPS*.

Tran, D.-V.; Nguyen, H.-K.; Pontelli, E.; and Son, T. C. 2009. Improving performance of conformant planners: Static analysis of declarative planning domain specifications. In *PADL*, 239–253.

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