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 ≥ θ, for some 0 < θ ≤ 1. In earlier work we observed that one can reduce CPP to CP by finding a set of initial states whose probability ≥ θ, 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 < θ ≤ 1. We seek a plan π such that following its execution, the goal is achieved with probability ≥ θ. 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 Gefterner 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 Gefterner 2009; Albore, Palacios, and Gefterner 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 ≥ θ 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 ≤ 1 − θ 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 ≥ θ 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, b1, G), generalizes this framework, replacing the single initial state with a set of initially possible states, called the initial belief state b1. This initial state is often described by means of a formula ϕ1, such that b1 = {w | w |= ϕ1}. A plan is an action sequence π such that π(w1) ⊇ G for every w1 ∈ b1.
CPP extend CP by quantifying the uncertainty regarding \( b_I \) using a probability distribution \( b_{\pi|I} \). 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. \( b_{\pi_I} \) 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 \( \theta \), which specifies the required lower bound on the probability of achieving \( G \). A sequence of actions \( \pi \) is called a plan if the weight of the initial states from which \( \pi \) reaches the goal is at least \( \theta \).

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 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 \( CP = (V, A, b_{\pi_I}, G, \theta) \) is solvable iff there exists a solvable CP problem \( C = (V, A, b_I, G) \) such that \( b_{\pi_I}(\{w \in b_I\}) \geq \theta \). Moreover, \( \pi \) is a solution to CP iff it is a solution to \( P \).

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 tech-
niques 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 $T_1$'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 ($P, \text{conf-planner}$)

```
ψ_1 \leftarrow \text{RESTRICT}(P);
return conf-planner($P = (V, A, ψ_1, N_0, G, θ)$);
```

Algorithm 2 RESTRICT ($P$)

```
Q \leftarrow \text{SORT-CLAUSES}(P);
ψ_1 \leftarrow ψ_1;
while ($b_{π_1}(ψ_1) ≥ θ$) do
  $C \leftarrow \text{Extract-First}(Q)$;
  $ψ_1 \leftarrow \text{RESTRICT-CLAUSE}(C, ψ_1, P)$;
end while
return $ψ_1$;
```

Algorithm 3 SORT-CLAUSES ($P$)

```
return A sorted list of all non-unit clauses in $φ_1$ 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 $φ_1$ from the BN describing the initial belief state $N_0$. $φ_1$ is constructed as follows: for every multi-value variable $X$ whose initial value is uncertain, $φ_1$ 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 $φ_1 \models w$ iff $b_{π_1}(w) > 0$.

Next, we seek to simplify $φ_1$ by ignoring a set of states with probability at most $1 − θ$. Thus, at this stage we operate on the structure: $P = (V, A, φ_1, N_0, G, θ)$ 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 $φ_1$ as $ψ_1$. We also use: $b_{π_1}(φ)$ to denote $\Sigma_{ψ_1=wb_{π_1}(w)}$.

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

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

Algorithm 4 RESTRICT-CLAUSE ($C, ψ_1, P$)

```
while ($|C| > 1 ∧ b_{π_1}(ψ_1) ≥ θ$) do
  $C' \leftarrow C$;
  choose next proposition $p \in C'$ by the following order:
  1. $p \notin g$.
  2. $b_{π_1}(ψ_1 ∧ \neg p) ≥ θ$.
  3. Prefer not removing $p \in DRC(C)$ before other propositions.
  4. $b_{π_1}(p)$ - The initial probability of $p$. (Low to high).
  5. If all previous parameters equal - choose arbitrarily.
    $C' \leftarrow C \setminus p$;
    $ψ_1 \leftarrow ψ_1 \wedge C \cup C'$;
end while
return $ψ_1$;
```

For clarity, we use set notation, treating $ψ_1$ 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 $ψ_1$.  

2376
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.

<table>
<thead>
<tr>
<th>Task</th>
<th>$\theta = 0.25$</th>
<th>$\theta = 0.5$</th>
<th>$\theta = 0.75$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe-unl-35</td>
<td>PFF</td>
<td>PCC</td>
<td>PCC</td>
</tr>
<tr>
<td>Safe-cub-30</td>
<td>2.018</td>
<td>0.07/81</td>
<td>0.181</td>
</tr>
<tr>
<td>NC-safe-unl-35</td>
<td>3.07/10</td>
<td>0.05/39</td>
<td>0.36/59</td>
</tr>
<tr>
<td>Cube-cub-2-planes-log-4</td>
<td>11.35/12</td>
<td>3.83/10</td>
<td>3.83/10</td>
</tr>
<tr>
<td>Cube-cub-2-planes-log-3</td>
<td>11.032/12</td>
<td>0.06/77</td>
<td>0.29/77</td>
</tr>
<tr>
<td>Cube-cub-2-planes-log-2</td>
<td>2.23/62</td>
<td>0.08/64</td>
<td>0.08/64</td>
</tr>
<tr>
<td>Cube-cub-2-planes-log-1</td>
<td>7.39/60</td>
<td>0.04/62</td>
<td>0.04/62</td>
</tr>
<tr>
<td>Safe-unl-35</td>
<td>OOT</td>
<td>OOT</td>
<td>OOT</td>
</tr>
<tr>
<td>Safe-cub-30</td>
<td>OOT</td>
<td>OOT</td>
<td>OOT</td>
</tr>
<tr>
<td>NC-safe-unl-35</td>
<td>11.032/12</td>
<td>0.06/77</td>
<td>0.29/77</td>
</tr>
<tr>
<td>NC-cube-2-planes-log-1</td>
<td>2.23/62</td>
<td>0.08/64</td>
<td>0.08/64</td>
</tr>
<tr>
<td>Safe-unl-35</td>
<td>OOT</td>
<td>OOT</td>
<td>OOT</td>
</tr>
<tr>
<td>Safe-cub-30</td>
<td>OOT</td>
<td>OOT</td>
<td>OOT</td>
</tr>
<tr>
<td>NC-safe-unl-35</td>
<td>11.032/12</td>
<td>0.06/77</td>
<td>0.29/77</td>
</tr>
<tr>
<td>NC-cube-2-planes-log-1</td>
<td>2.23/62</td>
<td>0.08/64</td>
<td>0.08/64</td>
</tr>
<tr>
<td>Safe-unl-35</td>
<td>OOT</td>
<td>OOT</td>
<td>OOT</td>
</tr>
<tr>
<td>Safe-cub-30</td>
<td>OOT</td>
<td>OOT</td>
<td>OOT</td>
</tr>
<tr>
<td>NC-safe-unl-35</td>
<td>11.032/12</td>
<td>0.06/77</td>
<td>0.29/77</td>
</tr>
<tr>
<td>NC-cube-2-planes-log-1</td>
<td>2.23/62</td>
<td>0.08/64</td>
<td>0.08/64</td>
</tr>
</tbody>
</table>

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 $r$.
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 $q$ is relevant to $r$ and $r$ is relevant to $t$ then $q$ is relevant to $r$.
4. $p$ is relevant to $q$ if $p$ is relevant to $r$ and $r$ is relevant to $q$.

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 $\varphi \in \varphi_1$ is considered relevant to a proposition $p$ if $\exists q \in \varphi \lor \Theta q \in r(p)$.

PG consider $\varphi$ relevant to $p$ only if all propositions of $\varphi$ are in $r(p)$. 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 $\varphi_1$. Assume $p$ is relevant to some goal fact $q$ but $q$ is not. Assume also that $b_{rel}(p) = 0.5$ 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 taken into account the possibility to plan from all $p$ states, even though planning from all $q$ is infeasible.

Definition 3 (relevant clauses set) $\forall q \in G : r(g)$ contains all the clauses from $\varphi_1$ relevant to $g$:

$r(g)$ expresses the maximal amount of initial uncertainty relevant to a specific goal fact. $\text{Max}_{g \in G}(|r(g)|)$ 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 $r(g)$ 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 Top plan, 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 $r(g)$ and PG’s conformant width: First, we use probabilistic clause relevance, explained earlier. Second, PG compute the closure of $r(g)$, called $C_L$*, while we this costly computation, using $r(g)$ instead. For our heuristic usage, this provides a better tradeoff, and in fact, $r(g) = C_L$* 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)**  
\[
RL(C) = \max \{|rel(g)| \mid g \in G \land C \subseteq rel(g)\}
\]

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)**  
\[
Rad(C) = \#g \in G \mid C \subseteq rel(g)
\]

**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 \( X \in N_b_i \) be the corresponding node of a clause \( C_X \). We define:

\[
RelChildren(X) = \{Y \mid X \in \text{Parents}(Y) \land \exists g \in G \mid C_Y \subseteq rel(g)\}
\]

\[
PI(C_X) = |\text{RelChildren}(X)|
\]

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 \( N_b_i \) to check whether any of the literals in \( C_Y \) for each \( Y \in \text{RelChildren}(X) \) 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 \( \text{extreme} \). 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 \( \theta \) 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)**  
\[
RP(C) = \min \{b_{\pi_1}(p) \mid p \in C\}
\]

(\( By b_{\pi_1}(p) we mean The initial probability of p \)).

\( RP(C) \) 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 \( N_b_i \) 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)**  
\[
DRC(C) = \{p \in \{g \mid p \in C \} \mid \exists g \in G : p \text{ is relevant to } g\}
\]

If, for a clause \( C \): \( DRC(C) \subseteq C \), 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 \( N_b_i \) 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 \( \neg q \).

**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 cf2csm 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 $\theta$ 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-PP 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**


