Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence

On Improving Conformant Planners by Analyzing Domain-Structures*

Khoi Nguyen, Vien Tran, Tran Cao Son, and **Enrico Pontelli**
Computer Science Department
New Mexico State University
MSC CS, PO Box 30001
Las Cruces, NM 88003, USA
knguyen—vtran—tson—epontell@cs.nmsu.edu

Abstract

The paper introduces a novel technique for improving the performance and scalability of best-first progression-based conformant planners. The technique is inspired by different well-known techniques from classical planning, such as landmark and stratification. Its most salient feature is that it is relatively cheap to implement yet quite effective when applicable. The effectiveness of the proposed technique is demonstrated by the development of new conformant planners by integrating the technique in various state-of-the-art conformant planners and an extensive experimental evaluation of the new planners using benchmarks collected from various sources. The result shows that the technique can be applied in several benchmarks and helps improve both performance and scalability of conformant planners.

Introduction

The concept of *ordered landmarks,* introduced in (Hoffmann, Porteous, and Sebastia 2004), has played an important role in the development of various state-of-the-art planning systems. It has been extensively studied, and variants and heuristics based on it have been proposed (e.g., action landmarks (Zhu and Givan 2004; Vidal and Geffner 2006)). To the best of our knowledge, these extensions and variations have been considered mainly in the context of classical planning, focusing on the generation of good heuristics in different settings, such as satisfycing and/or optimal planning (e.g., (Hoffmann, Porteous, and Sebastia 2004; Karpas and Domshlak 2009; Richter and Westphal 2010)).

In this paper, we investigate the idea of using landmarks in conformant planning (Smith and Weld 1998), specifically, to address the problem of dealing with the high degree of uncertainty in conformant planning. In PDDL, uncertainty is often specified by `one-of`-clauses or `or`-clauses. The size of the initial belief state of a conformant planning problem depends directly on the number and size of these clauses in the initial state specification, and it is often exponential in the number of objects in the problem instances; e.g., one of the hardest instances in the IPC 2008, `coins-30`, has 25 `one-of`-clauses which generate 10^{25} possible initial states.

Dealing effectively with the potentially huge size of the initial belief state is one of the main challenges for the scalability of conformant planners. A variety of techniques have been developed to address this issue. Different representations of belief states are used in (Brafman and Hoffmann 2004; To, Pontelli, and Son 2009; To, Son, and Pontelli 2010). Several heuristics are developed in (Bryce, Kambhampati, and Smith 2006). (Tran et al. 2009) proposes the `one-of`-combination technique to reduce the size of the initial belief state, effective in various problems. While it is useful for planners employing an explicit disjunctive representation of belief states, as in CPA (Tran et al. 2009) and DNF (To, Pontelli, and Son 2009), a significant amount of work is required to apply this technique to other planners, due to the different representations they use. Likewise, the `one-of`-relaxation technique in (To, Son, and Pontelli 2010) is useful in CNF but is difficult to use in other planners.

Our motivation in this work comes from these observations and the fact that classical planners can be used for different purposes in non-classical planning, as in the translation approach for conformant planning in (Palacios and Geffner 2009) and the replanning approach in probabilistic planning (Yoon, Fern, and Givan 2007). Our goal is to apply techniques which are useful in classical planning (e.g., landmarks and stratification), and utilize classical planners in conformant planning. To achieve this, we propose the notion of *viable landmarks* and discuss the complexity of the problem of computing them. To address the computational challenge of the problem, we propose to approximate viable landmarks and develop an algorithm for computing these approximations. To test the effectiveness of the new technique, we integrate it in different planners and compare them with state-of-the-art planners. The results show that the new technique is useful in several problems and improves the performance and scalability of other planners.

Background: Conformant Planning Problem

A conformant planning problem P is specified by a tuple $\langle F, O, I, G\rangle$, where F is a set of propositions, O a set of action descriptions, I a set of formulae describing the initial state of the world, and G a formula describing the goal. (F, O) are referred to as a planning domain.

A *literal* is a proposition $p \in F$ or its negation $\neg p$. $\bar{\ell}$ denotes the complement of the literal ℓ, and it is defined as $\bar{\ell} = \neg\ell$, where $\neg\neg p = p$ for $p \in F$. For a set of literals L, $\overline{L} = \{\bar{\ell} \mid \ell \in L\}$, and L is often used to represent $\wedge_{\ell \in L} \ell$.

*Partially supported by the NSF grant IIS-0812267

A set of literals X is *consistent* if there exists no $p \in F$ such that $\{p, \neg p\} \subseteq X$. A *state* s is a consistent and *complete* set of literals, i.e., s is consistent, and for each $p \in F$, either $p \in s$ or $\neg p \in s$. A *belief state* is a set of states. A set of literals X satisfies a literal ℓ, denoted by $X \models \ell$, (resp. a set of literals Y) iff $\ell \in X$ (resp. $Y \subseteq X$). For a belief state S, $S \models \ell$ iff $u \models \ell$ for every $u \in S$.

Each action a in O is associated with a precondition, denoted by $pre(a)$, and a set of conditional effects of the form $\psi \rightarrow \ell$ (denoted by $a : \psi \rightarrow \ell$), where $pre(a)$ and ψ are sets of literals and ℓ is a literal. We write $a : \psi \rightarrow \ell_1, \ldots, \ell_k$ to denote $\{a : \psi \rightarrow \ell_1, \ldots, a : \psi \rightarrow \ell_k\}$.

The initial state I is a set of literals, `one-of` clauses (each of the form $\texttt{one-of}(\psi_1, \ldots, \psi_n)$), and `or` clauses (each of the form $\texttt{or}(\psi_1, \ldots, \psi_m)$), where each ψ_i is a set of literals.

A set of literals X satisfies the `one-of` clause $\texttt{one-of}(\psi_1, \ldots, \psi_n)$ if there exists some i, $1 \leq i \leq n$, such that $\psi_i \subseteq X$ and for every $j \neq i, 1 \leq j \leq n, \overline{\psi_j} \cap X \neq \emptyset$. X satisfies the `or` clause $\texttt{or}(\psi_1, \ldots, \psi_m)$ if there exists some $1 \leq i \leq m$ such that $\psi_i \subseteq X$.

By $ext(I)$ we denote the set of all states satisfying every literal in I, every `one-of` clause in I, and every `or` clause in I. E.g., if $F=\{g, f, h\}$ and $I=\{\texttt{or}(g, h), \texttt{one-of}(f, h)\}$ then $ext(I) = \{\{g, h, \neg f\}, \{g, \neg h, f\}, \{\neg g, h, \neg f\}\}$.

A goal G is a collection of literals and `or` clauses.

Given a state s and an action a, a is executable in s if $pre(a) \subseteq s$. A conditional effect $a : \psi \rightarrow l$ is applicable in s if $\psi \subseteq s$. The set of effects of a in s, denoted by $e_a(s)$, is defined as: $e_a(s) = \{l \mid a : \psi \rightarrow l \in O \text{ is applicable in } s\}$. The execution of a in a state s results in a successor state $succ(a, s)$, where $succ(a, s) = (s \cup e_a(s)) \setminus \overline{e_a(s)}$ if a is executable in s, and $succ(a, s) = \textbf{failed}$, otherwise. Using this function, we define $\widehat{succ}$ for computing the state resulting from the execution of a sequence of actions $\alpha = [a_1, \ldots, a_n]$: $\widehat{succ}(\alpha, s) = s$ if $n = 0$; and $\widehat{succ}(\alpha, s) = succ(a_n, \widehat{succ}(\beta, s))$ if $n > 0$, where $\beta = [a_1, \ldots, a_{n-1}]$ and $\widehat{succ}(\gamma, \textbf{failed}) \overset{def}{=} \textbf{failed}$ for any sequence of actions γ. For a belief state S and action sequence α, let $\widehat{succ}^*(\alpha, S) = \{\widehat{succ}(\alpha, s) \mid s \in S\}$ if $\widehat{succ}(\alpha, s) \neq \textbf{failed}$ for every $s \in S$; and $\widehat{succ}^*(\alpha, S) = \textbf{failed}$, otherwise. α is a *solution* of P iff $\widehat{succ}^*(\alpha, ext(I)) \neq \textbf{failed}$ and G is satisfied in every state belonging to $\widehat{succ}^*(\alpha, ext(I))$.

Viable Landmarks in Conformant Planning

In this section, we define the notion of a viable landmark for conformant planning problems. By definition, the search for a conformant plan is done in the space of belief states. A straightforward generalization of a landmark to conformant planning would be a literal which is true in one of the belief states generated during the execution of any solution from the initial belief state. This is quite restrictive though.

Example 1. *Consider $P = \langle \{f, g, h\}, O, I, \{h\} \rangle$ where $I = \{\texttt{one-of}(f, g), \texttt{one-of}(h, \neg h)\}$ and O contains $a : f, \neg g \rightarrow \neg f, \neg g$; $a : \neg f, g \rightarrow f, g$; $b : \neg f, \neg g \rightarrow h$; $b : f, g \rightarrow h$; $c : f \rightarrow \neg f$; and $c : g \rightarrow \neg g$. Furthermore, $pre(a) = pre(b) = pre(c) = \top$.*

It is easy to see that $[a, b]$ is a solution of P. Yet, there exists no literal ℓ different from h that is true during the execution of $[a, b]$. In other words, this problem has only a trivial landmark, h, according to the proposed generalization.

It is easy to see that $[c, b]$ is another solution of P in Example 1 and for every state $s \in succ^*(c, ext(I))$, $\neg f \in s$, and $\neg g \in s$. In other words, the uncertainty with regards to the clause $\texttt{one-of}(f, g)$ has been removed from $succ^*(c, ext(I))$. Moreover, $succ^*(c, ext(I))$ has fewer elements than $succ^*(a, ext(I))$. This means that for conformant planners employing the cardinality heuristic, the solution $[c, b]$ is likely to be returned as the first solution of P. This motivates us to define the notion of a viable landmark for a conformant problem as a belief state which (*i*) can easily be reached or predicted; and (*ii*) from which there is a sequence of actions that reaches the goal. In the above example, $succ^*(c, ext(I))$ would be a viable landmark, which can be reached by the sequence $[c]$ and from which there is the sequence $[b]$ that reaches the goal. For simplicity of the presentation, we will focus on reducing the uncertainty caused by the `one-of` clauses in this paper—the results can be easily extended to the `or`-clauses.

Definition 1. *Let S be a belief state, Ω be a belief state, and o be an `one-of` clause. We say that S is a* convergent point *of Ω for o, denoted by $\Omega \rightsquigarrow S[o]$, if for every literal ℓ appearing in o, $S \models \ell$ or $S \models \overline{\ell}$, and for every state $s \in \Omega$, there exists an action sequence α_s such that $\widehat{succ}(\alpha_s, s) \in S$.*

In Example 1, $succ^*(c, ext(I))$ is a convergent point of $ext(I)$ for $\texttt{one-of}(f, g)$. A convergent point is only useful for the search of a solution if there exists a action sequence α that leads to this point from all possible initial states. We therefore define the notion of an abstract point as follows.

Definition 2. *Let S be a belief state, Ω be a belief state, α be an action sequence, and o be an `one-of` clause. We say that S is an* abstract point *of Ω for o w.r.t. α, denoted by $\Omega \overset{\alpha}{\rightsquigarrow} S[o]$, if $\Omega \rightsquigarrow S[o]$ and $\widehat{succ}^*(\alpha, \Omega) \subseteq S$. We call α an* abstract path *for o from Ω to S. S is an* abstract point *of Ω for o if there exists an action sequence α such that $\Omega \overset{\alpha}{\rightsquigarrow} S[o]$.*

We have that $ext(I) \overset{[c]}{\rightsquigarrow} succ^*(c, ext(I))[\texttt{one-of}(f, g)]$ for Example 1. The following proposition holds.

Proposition 1. *Let S be a belief state, Ω be a belief state, α be an action sequence, and o be an `one-of` clause. Then, $\Omega \overset{\alpha}{\rightsquigarrow} S[o]$ implies $\Omega \overset{\alpha}{\rightsquigarrow} \widehat{succ}^*(\alpha, \Omega)[o]$.*

We will often say that α is an abstract path for a planning problem $P = \langle F, O, I, G \rangle$ if there exists an `one-of`-clause o in I such that $ext(I) \overset{\alpha}{\rightsquigarrow} \widehat{succ}^*(\alpha, ext(I))[o]$.

Definition 3. *Let $P = \langle F, O, I, G \rangle$ be a conformant planning problem. A belief state S is called a* viable landmark *of P if S is an abstract point of $ext(I)$ for some `one-of` clause in I and the problem $P' = \langle F, O, S, G \rangle$ has a solution.*

This achieves our objective of having $succ^*(c, ext(I))$ as a viable landmark of P for Example 1. The next proposition follows from the definition of a viable landmark.

Proposition 2. *Let $P = \langle F, O, I, G \rangle$ be a conformant planning problem and S be a viable landmark of P. Then, there exists an action sequence α such that for every solution β of the problem $P' = \langle F, O, S, G \rangle$, $\alpha \circ \beta$ is a solution of P.*

β in Prop. 2 is referred to as a *goal path* w.r.t. S. Prop. 2 indicates that we could solve a conformant planning problem $P = \langle F, O, I, G\rangle$ by (*i*) finding a viable landmark S of P; and (*ii*) solving the problem $P' = \langle F, O, S, G\rangle$. Prop. 1 implies that instead of S, we can find an action sequence α and solve the problem $P'' = \langle F, O, \widehat{succ}^*(\alpha, ext(I)), G\rangle$. By definition of a viable landmark, we know that there exists an `one-of` clauses o in I such that every literal ℓ in o is known to be either true or false in S (or in $\widehat{succ}^*(\alpha, ext(I))$). It means that the uncertainty in o has been removed. In general, there is no guarantee that the search for a solution from the new belief state S (or $\widehat{succ}^*(\alpha, ext(I))$) will be easier than the search from the initial belief state $ext(I)$. However, if α only changes the value of literals in o, then P'' is *closer* to a classical planning problem than P, and thus it could be easier than P, since classical planning has a lower complexity than conformant planning (Baral, Kreinovich, and Trejo 2000).

An obstacle in applying the above idea is that finding a viable landmark of P is not a simple task. Let *RLandmark* be the problem of determining if a belief state S is a viable landmark of a conformant planning problem $P = \langle F, O, I, G\rangle$. Since determining if S is a viable landmark requires checking for a plan from S to G, the complexity of *RLandmark* is at least as hard as the conformant planning problem (i.e., Σ_2^P (Baral, Kreinovich, and Trejo 2000)).

The above complexity result shows that attempting to find an arbitrary viable landmark and using it in the search for a solution is likely not a good idea. This result is also in-line with the complexity of the LANDMARK problem (Hoffmann, Porteous, and Sebastia 2004). However, Prop. 1 implies that a viable landmark is associated with an abstract point and an action sequence (an abstract path). So, we can approximate a viable landmark by an abstract point.

Approximating Abstract Points

Let us explore ways to quickly identify possible viable landmarks. We assume that we are confronted with a problem that has goal G from a belief state Ω that satisfies o. To motivate this process, let us look at the following example.

Example 2. *Let us consider an instance of the cube problem encoded by* $P = \langle\{1, 2, 3\}, O, \{\texttt{one-of}(1, 2, 3)\}, \{2\}\rangle$ *where* O *contains* $l : i \rightarrow i - 1, \neg i$ *for* $i = 2, 3$ *and* $r : i \rightarrow i + 1, \neg i$ *for* $i = 1, 2$*; and* $pre(l) = pre(r) = \top$. i *indicates the location, and* l *and* r *stand for* $left$ *and* $right$. *The initial belief state* $\Omega = ext(I)$ *consists of* 3 *states,* $s_1 = \{1, \neg 2, \neg 3\}$, $s_2 = \{\neg 1, 2, \neg 3\}$, *and* $s_3 = \{\neg 1, \neg 2, 3\}$.

We can check that each $\{s_i\}$ *is an abstract point of* Ω *for* `one-of`$(1, 2, 3)$. *Consider* $\{s_3\}$. *We observe that for* $\alpha_1 = [r, r]$ $\alpha_2 = [r]$, *and* $\alpha_3 = []$, *it holds that* $\widehat{succ}(\alpha_i, s_i) = s_3$. *Furthermore, for* $\alpha = \alpha_1 \circ \alpha_2 \circ \alpha_3$, *we also have that* $\widehat{succ}(\alpha, s_i) = s_3$. *It is easy to see that* $\{s_3\}$ *(resp.* $\{s_1\}$*) is a viable landmark of* P *as there exists the goal path* $[l]$ *(resp.* $[r]$*) from* $\{s_3\}$ *(resp.* $\{s_1\}$*) to the goal.*

The example shows that an abstract point of a belief state of Ω for an `one-of` clause o may be one of the states in Ω.

Definition 4. *Let* $\Omega = \{s_1, \ldots, s_n\}$ *be a belief state and* o *an* `one-of` *clause. We say that* o *is* closed-convergent *in* Ω *if there exists an* $s_i \in \Omega$ *and an action sequence* α *such that* $\widehat{succ}(\alpha, s_j) = s_i$ *for every* j *such that* $1 \leq j \leq n$.

We can check that `one-of`$(1, 2, 3)$ (Example 3) is closed-convergent in $ext(I)$. It is interesting to point out that if o is closed-convergent in Ω then the uncertainty caused by o could be completely removed after the execution of the abstract path α. This is certainly the ideal case. The next example shows that, sometimes, the convergent point lies outside of the belief state itself.

Example 3. *Let us consider a variant of the cube problem encoded by* $P = \langle\{1, 2, 3, 4\}, O, \{\texttt{one-of}(1, 2), \neg 3, \neg 4\}, \{4\}\rangle$ *where* O *contains* $a : 1 \rightarrow 3, \neg 1$*;* $b : 2 \rightarrow 3, \neg 2$*;* $c : 3 \rightarrow 4$, *and* $pre(a) = pre(b) = pref(c) = \top$.

The initial belief state $\Omega = ext(I)$ *consists of two states,* $s_1 = \{1, \neg 2, \neg 3, \neg 4\}$ *and* $s_2 = \{\neg 1, 2, \neg 3, \neg 4\}$.

Let $s_3 = \{\neg 1, \neg 2, 3, \neg 4\}$. *It is easy to check that* $\{s_3\}$ *is an abstract point of* Ω *for* `one-of`$(1, 2)$. *We observe that for* $\alpha_1 = [a]$ $\alpha_2 = [b]$, *it holds that* $\widehat{succ}(\alpha_i, s_i) = s_3$ *and, for* $\alpha = \alpha_1 \circ \alpha_2$, *we also have that* $\widehat{succ}(\alpha, s_i) = s_3$.

It is easy to see that $\{s_3\}$ *is a viable landmark of* P *as there exists the goal path* $[c]$ *from* $\{s_3\}$ *to the goal.*

The above example leads to the following definition that generalizes the notion of closed-convergent.

Definition 5. *Let* $\Omega = \{s_1, \ldots, s_n\}$ *be a belief state and* o *be an* `one-of` *clause. We say that* o *is* open-convergent *in* Ω *if there exist a belief state* S *and an action sequence* α *such that* $\widehat{succ}(\alpha, s_i) \in S$ *for every* j *such that* $1 \leq j \leq n$.

Clearly, if o is open-convergent in Ω then we know that there exists some S and α such that $\Omega \overset{\alpha}{\rightsquigarrow} S[o]$.

In the rest of this section, we will develop a greedy algorithm for identifying possible abstract points of a planning problem. Before we discuss the idea in details, let us introduce the following notation. Let $o = (l_1, \ldots, l_n)$ be an `one-of` clause. A set of literals δ is an interpretation of o if (*i*) for every literal $l \in \delta$, $l \in lit(o) \cup \overline{lit(o)}$; and (*ii*) for each i, $\{l_i, \overline{l_i}\} \cap \delta \neq \emptyset$ and $\{l_i, \overline{l_i}\} \setminus \delta \neq \emptyset$. For a state s, let $s|_o = s \cap (lit(o) \cup \overline{lit(o)})$. For a belief state Ω, let $\Omega|_o = \{s|_o \mid s \in \Omega\}$.

Proposition 3. *Let* Ω *be a belief state,* o *be an* `one-of` *clause, and* δ *be an interpretation of* o. *If* α *is a plan achieving* δ *from* Ω, *then* $\Omega \overset{\alpha}{\rightsquigarrow} \widehat{succ}^*(\alpha, \Omega)[o]$.

Although simple, the above proposition shows that an abstract point can be characterized by an interpretation of the `one-of` clause in consideration. I.e., if the intention is to reduce the uncertainty caused by an `one-of` clause o in the belief state Ω, then it is reasonable to focus on the set of interpretations of o that could be reached from Ω, i.e., on $\Omega|_o$ and the interpretations reachable from $\Omega|_o$. Thus, we can reduce the original domain to a domain related to o and analyze this domain to look for an abstract point.

Given a planning domain (F, O), the reduced domain of (F, O) by the abstraction of o, denoted by $Ab(F, O, o)$, is the planning domain (F', O') obtained from (F, O) where

- $a{:}\phi|_o {\rightarrow} \psi|_o {\in} O'$ iff $a{:}\phi {\rightarrow} \psi {\in} O$ and $pre(a) {\cup} \phi$ satisfies o;
- for each a in O', $pre(a)$ is changed to $pre(a)|_o$;

- F' contains l iff $l \in F$ and l or $\neg l$ occurs in some conditional effect in O'.

Let $Ab(F, O, o)$ be the reduced domain of (F, O) by the abstraction of o. The transition graph of $Ab(F, O, o)$, denoted by $G(o)$, is defined as a labeled graph (V, E) where each $\delta \in V$ is an interpretation of o and $(\delta, a, \delta') \in E$ iff there exits a in $Ab(F, O, o)$ such that $succ(a, \delta)$ satisfies δ'. The reduced graph of a belief state Ω, denoted by $G(\Omega, o)$, is obtained by removing from $G(o)$ every node δ that is not reachable from a node in $\Omega|_o$ and the edges coming in or out from these nodes. Fig. 1 shows the transition graph of $Ab(F, O, o)$ for Exp. 2-3. The label in a node specifies what is true in the interpretation. Labeled links between nodes represent actions. The dotted oval contains the interpretations satisfying the `one-of` clause.

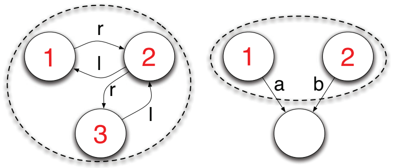

Figure 1: Transition Graph for Reduced Domain

Observe that Defs 4-5 imply that an interpretation δ of o that is reachable by every other interpretations in $\Omega|_o$ is a candidate for being an abstract point. So, computing a candidate convergent point of $\Omega|_o$ can be done by (*i*) creating $G(\Omega, o)$; and (*ii*) searching for a node that is reachable from every interpretation satisfying o. As seen in Figure 1 (left), there could be several candidates. Furthermore, checking reachability between the nodes is, theoretically, a computationally expensive task since the $G(\Omega, o)$ might have a number of nodes exponential in the size of the `one-of` clause.

To this end, we develop a greedy method for predicting of an abstract point as follows. For an `one-of` clause o and a belief state Ω, we say that o is *probably closed-convergent* in Ω if for every action a in $Ab(F, O, o)$, a maintains o, i.e.., for every interpretation δ satisfies o, $succ(a, \delta)$ satisfies o. Otherwise, o is *probably open-convergent* in Ω. We then identify an interpretation δ in $G(\Omega, o)$ as follows.

- *o is probably closed-convergent*: selects δ which satisfies o and has the minimal number of outgoing links; and
- *o is probably open-convergent*: selects δ which does not satisfy o and has the maximal number of incoming links.

The rationale for the above prediction is as follows. If o is probably closed-convergent (Fig. 1, left) then the node with lowest number of outgoing edges in $G(\Omega, o)$ is likely the most difficult node to reach the goal. Since a conformant plan is required to achieve the goal from any node, it is reasonable to make the plan going through this node. If o is probably open-convergent (Fig. 1, right) then the node with highest number of incoming edges in $G(\Omega, o)$ that does not satisfy o is likely the door to the goal of the problem. This is summarized in the following algorithm.

It is easy to see that the complexity of the above algorithm depends on the construction of (*i*) $Ab(F, O, o)$; and (*ii*) $G(\Omega, o)$. The first item can be done in linear time in the size of the domain (in term of the size of $|O| + |F|$). The second item can be done in $O(|o| \times (|O| + |F|))$ because

Algorithm 1 *Predict_Abstract_Point*(o, Ω, P)

```
1: Input: $P=\langle F, O, I, G\rangle$–conformant planning problem
          $\Omega$–a belief state
          $o$–an one-of-clause in $P$
2: Output: A possible abstract point of $\Omega$ for $o$
3: Compute $(F', O')=Ab(F, O, o)$ and $G(\Omega, o)=(V, E)$
4: if $G(\Omega, o)$ contains only nodes satisfying $o$ then
5:     return $\delta$ with lowest number of outgoing edges
6: else
7:     return $\delta$ with highest number of incoming edges
8: end if
9: return Unknown
```

we only need to scan through the set of interpretations satisfying o and create additional nodes, i.e., Alg. 1 is relatively efficient to implement. Before we discuss how this can be used in a planner, we will turn out attention to the case of multiple `one-of` clauses in a planning problem.

Stratification Between `one-of`-Clauses

A conformant problem $P = \langle F, O, I, G\rangle$ usually has several `one-of` clauses, say $o_1, \ldots, o_n$, which may interact with each other. Assume that Algo. 1 can be used to identify possible abstract points $\delta_1, \ldots, \delta_n$ of these clauses. If we were to use these possible abstract points as intermediate goals in solving the problem, the interaction among $o_1, \ldots, o_n$ might dictate an order among these points, in the same way as a reasonable order among landmarks affects the search for a solution in a classical planner (see, e.g., (Tran et al. 2009)).

To address the above problem, we introduce the notion of a stratification between `one-of` clauses. This notion is inspired by the notion of stratification in classical planning (Chen and Yao 2009). To this end, we define the notion of dependency between `one-of`-clauses as follows.

Definition 6. *Let o_1 and o_2 be two different* `one-of` *clauses in a conformant problem P. We say o_1 depends on o_2, denoted by $o_1 \prec o_2$, if there exists action a and an effect $\underline{a : \phi \rightarrow l}$ in O such that $pre(a)$ satisfies o_1 and $l \in lit(o_2) \cup \overline{lit(o_2)}$.*

The notions of a level-mapping and a stratification of a set of `one-of` clauses are defined next.

Definition 7. *Let P be a planning problem. A* level-mapping *function of P is a mapping lv from the set of* `one-of` *clauses in P into the set of non-negative integers.*

A level-mapping lv is a stratification *of the* `one-of` *clauses in P if lv satisfies the following properties:*

- *$lv(o)=0$ if o does not depend on any* `one-of` *clause;*
- *$lv(o) > \max\{lv(o') \mid o$ depends on $o'\}$.*

A stratification of the `one-of` clauses in P can be computed by (*i*) creating a graph whose nodes are the `one-of` clauses in P and whose links encode the dependencies between them; and (*ii*) using a labeling algorithm to assign the level to the nodes. This can be done in linear time in the size of the domain and the number of `one-of` clauses in P.

Experimental Evaluation

Let us describe how the proposed technique can be used in improving state-of-the-art planners. Let P_{cl} and P_{cf}

denote a classical and a conformant planner, respectively. CPLS($P_{\rm cl}, P_{\rm cf}$) denotes a conformant planner obtained by integrating the new technique—described in the previous sections—into $P_{\rm cf}$ and $P_{\rm cl}$. The overall algorithm implemented in CPLS($P_{\rm cl}, P_{\rm cf}$) is given in Fig. 2. CPCL is a conformant planner[1] used for the verification of abstract points.

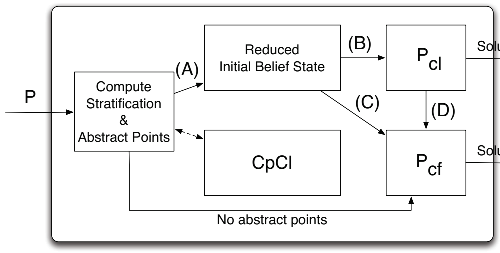
P
Compute Stratification & Abstract Points
(A)
Reduced Initial Belief State
(B)
(C)
P_{cl}
(D)
P_{cf}
Solution
Solution
CpCl
No abstract points

Figure 2: Overall Structure for CPLS($P_{\rm cl}, P_{\rm cf}$)

Given a problem P, CPLS($P_{\rm cl}, P_{\rm cf}$) starts by computing a stratification of the `one-of` clauses in P. If no stratification exists, the conformant planner $P_{\rm cf}$ is used to find a solution of P. Otherwise, CPLS($P_{\rm cl}, P_{\rm cf}$) identifies, for each `one-of` clause—from the lowest to the highest level of the stratification—a possible abstract point (via Algorithm 1) and verifies that it is an abstract point by calling CPCL to find an abstract path (represented by the dotted line). If the process finishes successfully (Fig. 2, (A)), its result will be a belief state with a smaller degree of uncertainty than that of the initial belief state. If this belief state is a singleton (Fig. 2, (B)), then the classical planner, $P_{\rm cl}$, is used to solve the problem. Otherwise (Fig. 2, (C)), $P_{\rm cf}$ is used. If $P_{\rm cl}$ fails to find a solution (Fig. 2, (D)), then CPLS($P_{\rm cl}, P_{\rm cf}$) uses $P_{\rm cf}$ to find a solution.

Observe that if $P_{\rm cl}$ is invoked and succeeds, the performance of CPLS($P_{\rm cl}, P_{\rm cf}$) does not depend on the performance of $P_{\rm cf}$. Otherwise, the overall overhead of CPLS($P_{\rm cl}, P_{\rm cf}$), comparing to $P_{\rm cf}$, in solving a problem P includes the following costs: computing a stratification, determining the possible abstract points, verifying the abstract points, and finding a solution using $P_{\rm cl}$. As we discussed earlier, the extra cost of the first two components is rather small. Theoretically, the cost of the last two components could be high. Our experiments show that, for the majority of the benchmarks, the total overhead is reasonable.

We instantiate CPLS($P_{\rm cl}, P_{\rm cf}$) using the following planners: $P_{\rm cl}$ is the planner LAMA (Richter and Westphal 2010) and $P_{\rm cf}$ is one of the four conformant planners— CPA (Tran et al. 2009), DNF (To, Pontelli, and Son 2009), `t0` (Palacios and Geffner 2009), and CPCL (Nguyen et al. 2011). These planners have been shown to outperform all other planners in the literature. The experiments have been performed on a Intel Core2 Quad CPU Q9400 2.66GHz machine, with 4Gb memory, a run-time cutoff of 30 minutes.

The benchmark set contains 1050 instances of 23 domains from the recent IPCs (2006 and 2008) and from the distribution of CFF, `t0`, and CNF. Due to lack of space, we omit the precise description of each domain.

Applicability of Reducing Uncertainty: Table 1[2] shows the effectiveness of the module for reducing the uncertainty of CPLS(LAMA,$P_{\rm cf}$) in some representative instances. In summary, out of 1050 instances of 23 domains, CPLS(LAMA,$P_{\rm cf}$) can predict and successfully reduce the uncertainty to a single state in 879 instances of 18 domains. The reasons for the inapplicability of the technique are: (*i*) the problem has disjunctive goals, which might require a set of convergent points instead of a single one (e.g., `adder`, IPC 2006); (*ii*) the greedy algorithm for predicting abstract points fails to yield a "correct" abstract point (e.g., `block`).

Instance	(A)	(B)	(C)	(D)	Instance	(A)	(B)	(C)	(D)
coins-20	8	2	$8^3 * 10^5$	1	coins-30	25	2	10^{25}	1
bomb-50-5	50	1	2^{50}	1	bomb-100-100	100	1	2^{100}	1
cube-39-19	3	1	39^{19}	1	cube-119-59	3	1	59^{119}	1
ds-8-2	2	1	2^{12}	1	ds-8-3	3	1	2^{24}	1
or-ds-8-2	2	1	2^{4096}	1	or-ds-8-3	3	1	$2^{2^{24}}$	1

Table 1: Statistics of the technique on some domains

Instance	CPA(H)	`t0`	DNF	CPCL	CPLS(.,.)	Ovh
blw-03	20.4/205	48.51/80	307/325	**1.3**/266	*	1.1
blw-04	AB	AB	AB	**29.5**/1384	*	1.2
coins-20	0.74/195	0.15/108	0.97/99	**0.1**/163	**0.1**/187	0.092
coins-30	AB	AB	AB	1.0/1107	**8.6**/1302	0.57
comm-15	2.29//95	**0.092**/110	3.43/125	0.1/97	*	0.08
comm-25	1222/389	1.55/453	1797/501	**0.8**/294	*	0.69
sortnet-5	**0.02**/13	0.18/15	0.03/15	0.05/15	*	0.06
sortnet-15	240/74	AB	**35**/118	63.9/120	*	129
sortnum-5	AB	1.9/10	1.67/10	0.81/10	**0.01**/10	0.04
sortnum-10	AB	AB	AB	**1.31**/45	57/45	0.46
uts-10	14.3/89	0.88/59	2.66/66	0.26/58	**0.24**/420	0.16
uts-20	8.94/156	0.56/85	1.65/109	**0.097**/58	0.12/555	0.06
uts-30	4.9/74	0.79/67	1.39/73	**0.17**/64	0.209/478	0.1
uts-cycle-03	**0.01**/3	0.14/3	0.01/3	0.04/3	*	0.035
uts-cycle-15	AB	AB	AB	**1314**/272	*	886.49
raos-keys-02	0.26/32	**0.02**/21	0.09/39	0.05/38	*	0.02
raos-keys-04	AB	AB	AB	**16.78**/163	*	12.88
forest-03	AB	0.62/45	TO	**0.46**/167	*	0.24
forest-09	AB	AB	AB	**183.8**/963	*	0.69
b-10-1	0.15/19	**0.01**/20	0.04/19	0.02/19	0.03/19	0.01
b-50-5	2.64/95	**0.11**/100	1.49/95	0.12/95	0.12/95	0.07
b-100-100	AB	6.26/200	TO	4.27/100	**2.85**/200	1.98
cube-39-19	TO	3.6/171	198/1023	**0.23**/171	0.29/861	0.16
cube-119-59	AB	AB	AB	**1.9**/531	5.8/10440	0.92
sqr-c-32-16	14.8/928	0.95/93	7.3/340	**0.12**/72	0.14/512	0.09
sqr-c-52-26	208/2374	4.4/153	51/659	**0.23**/154	0.36/1352	0.16
sqr-c-120-60	AB	AB	AB	**1.03**/358	3.02/6873	0.56
ring-20	AB	1.78/95	AB	**0.12**/72	*	0.08
ring-100	AB	AB	AB	**8.71**/1640	*	4.16
safe-10	0.04/10	0.02/10	0.029/10	**0.027**/10	0.03/10	0.20
safe-100	339/100	1.26/100	4.06/100	0.93/100	**0.11**/100	0.05

Table 2: Results for IPC and CFF Domains (Time in *seconds*)

Performance Evaluation: The experimental results are reported in Tables 2–4. AB and TO denote an execution aborted by the planner due to "out of memory" and "timeout" respectively. Each column CPA, `t0`, DNF, or CPCL reports the time and solution length from the corresponding planner. Column Ovh reports the overhead incurred by

[1]CPCL (Nguyen et al. 2011) implements a new algorithm for conformant planning using a classical planner. It outperforms other planners in several benchmarks but is unable to solve the problems in the CNF distribution.

[2](A): Number of Uncertainty Clauses; (B) Number of Different Levels in Stratification; (C) and (D): size before/after reduction.

CPLS(LAMA, P_{cf}), i.e., the time spent to predict and verify abstract points and to compute the reduced belief state before LAMA or P_{cf} is called to compute a solution.

Column CPLS(.,.) reports the performance of CPLS(LAMA, P_{cf}). It is either a star ('*') or of the form $time/length$ as in other columns. A star indicates that the process of computing an abstract point is unsuccessful or the reduced belief state contains more than one state. In this case, the performance of CPLS(LAMA, P_{cf}) equals the performance of P_{cf} plus Ovh, e.g., in `blw-03` (Table 2), CPLS(LAMA, CPA) will take 20.4+1.1=21.5 second to find the first solution which has 205 actions. Otherwise, the process of computing an abstract point is successful and the reduced belief state contains exactly one state. In this case, the performance of CPLS(LAMA, P_{cf}) is independent from P_{cf} and is the same for all P_{cf}; thus, we only need to report it once. E.g., in `coins-30`, CPLS(LAMA, P_{cf}) takes 8.6+0.57=9.17 second to find a solution of 1302 actions.

Observe that, except for `uts-cycle-15` and `raos-keys-04`, the overhead incurred by the reduction technique is rather small. On the other hand, the benefits are quite clear: the scalability of CPLS(LAMA,P_{cf}) cannot be matched by any planner.

Instance	CPA(H)	t0	DNF	CPCL	CPLS(.,.)	Ovh
ds-8-2	96/1480	13/ 639	54.4/302	1.36/834	**0.33**/392	0.17
ds-8-3	225/2227	134/761	34.4/629	1.57/962	**0.55**/561	0.26
ds-12-7	AB	AB	AB	15/4036	**12.46**/2459	0.57
ds-12-9	AB	AB	AB	20/4612	**18.85**/3103	0.94
push-8-2	213/1099	AB	68/903	2.52/880	**6.83**/3038	0.28
push-12-9	AB	AB	AB	**35.1**/2761	84.9/2754	6.16
lng-8-2-1	104/314	28.8/48	48.41/73	2.46/438	**1.16**/44	0.99
lng-12-1-4	AB	AB	AB	95.2/2010	**12.22**/46	11.09
lng-12-5-4	AB	AB	AB	265/2010	**179.27**/46	167.47
1-ds-8-7-dis	AB	AB	AB	3.21/1026	**1.39**/454	0.86
1-ds-12-9-dis	AB	AB	AB	83.2/3462	**10.40**/1080	5.2

Table 3: Results for Challenging Domains (t0 Distribution)

Table 3 shows the performance of CPLS(LAMA,P_{cf}) in the challenging domains in the `t0` distribution. The reduction technique is even more impressive in this test suite. CPLS(LAMA,P_{cf}) can solve all problems and is slower than CPCL only in a few instances of the `push` domains.

Instance	t0	CNF	CPLS(.,.)	Ovh
or-coins-20	0.137/107	1.1/114	**0.10**/163	0.092
or-coins-30	AB	AB	**8.6**/1302	0.57
or-ds-8-2	22.362/639	0.64/264	**0.33**/392	0.17
or-ds-8-3	275.802/761	27.85/392	**0.55**/561	0.26
or-push-8-3	AB	**2.5**/303	11.4/3657	0.4
or-push-12-7	AB	AB	**29.30**/2595	4.34
or-push-12-9	AB	AB	**35.10**/2761	6.16
or-lng-8-2-1	AB	AB	**1.16**/44	0.99
or-lng-12-1-4	AB	AB	**12.22**/46	11.09
or-lng-12-5-4	AB	AB	**179.27**/46	167.4
or-1-ds-8-5-dis	AB	AB	**0.34**/454	0.54
or-1-ds-8-7-dis	AB	AB	**0.45**/454	0.86
or-1-ds-8-9-dis	AB	AB	**1.88**/454	1.24
or-1-ds-12-7-dis	AB	AB	**7.5**/1080	0.57
or-1-ds-12-9-dis	AB	AB	**10.40**/1080	0.94

Table 4: Results for Or Challenging Domains (CNF Distribution)

Table 4 shows that the reduction technique is very useful in the domains from the CNF distribution—which are very difficult for other planners. CPA, DNF, CPCL fail to solve the large instances reported and are omitted from the table.

Conclusion

We proposed a notion called viable landmarks for conformant planning and developed a technique for reducing uncertainty based on this notion. We developed an algorithm for approximating viable landmarks and proposed a scheme for the integration of this algorithm within current conformant planners and a classical planner. We experimented with this idea using the best conformant and classical planners and evaluated these implementations on a large pool of benchmarks from different sources. The results show that the technique of reducing uncertainty helps to improve the performance and scalability of these planners in many problems. These results show that many techniques developed for classical planning (e.g., landmark, abstraction, stratification) can be extended effectively to conformant planning.

References

Baral, C.; Kreinovich, V.; and Trejo, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *AIJ* 122:241–267.

Brafman, R., and Hoffmann, J. 2004. Conformant planning via heuristic forward search: A new approach. In *ICAPS*.

Bryce, D.; Kambhampati, S.; and Smith, D. 2006. Planning Graph Heuristics for Belief Space Search. *JAIR* 26:35–99.

Chen, Y., and Yao, G. 2009. Stratified Planning. In *IJCAI*.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *JAIR* 22:215–278.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *IJCAI*, 1728–1733.

Nguyen, K.; Tran, V.; Son T. C.; and Pontelli, E. 2011. A New Approach To Conformant Planning. NMSU-CS-TR01.

Palacios, H., and Geffner, H. 2009. Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width. *JAIR* 35:623–675.

Richter, S., and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *JAIR* 39:127–177.

Smith, D., and Weld, D. 1998. Conformant Graphplan. In *AAAI*, 889–896.

To, S. T.; Pontelli, E.; and Son, T. C. 2009. A conformant planner with explicit disjunctive representation of belief states. In *ICAPS*.

To, S. T.; Son, T. C.; and Pontelli, E. 2010. A New Approach to Conformant Planning using CNF. 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*, 5418 of *LNCS*, 239–253. Springer.

Vidal, V., and Geffner, H. 2006. Branching and Prunning: An Optimal Temporal POCL Planner based on Constraint Programming. *AIJ* (3):298–335.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. Ff-replan: A baseline for probabilistic planning. In *ICAPS*, 352–259.

Zhu, L., and Givan, R. 2004. Heuristic planning via roadmap deduction. In *IPC-4*. 64–66.