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 satisfying 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 this technique is effective 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 $(F, O, I, G)$, 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 $\neg p$. $\ell$ denotes the complement of the literal $\ell$, and it is defined as $\ell = -\ell$, where $\neg \neg p = p$ for $p \in F$. For a set of literals $L$, $L = \{\ell \mid \ell \in L\}$, and $L$ is often used to represent $\wedge_{\ell \in L} \ell$. 

\[ \text{Partially supported by the NSF grant IIS-0812267} \]

Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
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 $\ell$, denoted by $X \models \ell$, if $\ell$ is a member of $X$. For a belief state $S$, $S \models \ell$ if and only if for every $u \in S$, $u \models \ell$.

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

The initial state $I$ is a set of literals, one-of clauses (each of the form one-of($\psi_1, \ldots, \psi_n$)), and $\Omega$ clauses (each of the form $\Omega \models \psi(\psi_1, \ldots, \psi_m)$), where each $\psi_i$ is a set of literals.

A set of literals $X$ satisfies the one-of clause 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 $j \neq i$, $1 \leq j \leq n$, $\psi_j \cap X \neq \emptyset$. $X$ satisfies the $\Omega$ clause $\Omega \models \psi(\psi_1, \ldots, \psi_m)$ if there exists some $1 \leq i \leq m$ such that $\psi_i \subseteq X$.

By $\text{ext}(I)$ we denote the set of all states satisfying every literal in $I$, every one-of clause in $I$, and every $\Omega$ clause in $I$. E.g., if $F = \{a, f, h\}$ and $I = (\Omega \models \psi(\psi_1, \psi_2, \psi_3))$, then $\text{ext}(I) = \{ \{g, h, \neg \gamma\}, \{g, \neg h, f\}, \{-g, h, \neg f\}\}$.

A goal $G$ is a collection of literals and $\Omega$ clauses.

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

**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 $P = \langle \{f, g, h\}, O, I, \{\ell\} \rangle$ where $I = \{\text{one-of}(f,g), \text{one-of}(h,\neg h)\}$ and $O$ contains $a : f, \neg g \rightarrow f, \neg g$; $b : f, g \rightarrow h$; $c : f \rightarrow \neg f$; and $c : g \rightarrow \neg g$.

It is easy to see that $[a, b]$ is a solution of $P$. Yet, there exists no literal $\ell$ different from $h$ that is true during the execution of $[a, b]$ in $P$.

**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 $s \in \text{succ}^*(c, \text{ext}(I))$, $\neg f \in s$, and $\neg g \in s$. In other words, the uncertainty with regards to the clause one-of$(f, g)$ has been removed from $\text{succ}^*(c, \text{ext}(I))$. Moreover, $\text{succ}^*(c, \text{ext}(I))$ has fewer elements than $\text{succ}^*(a, \text{ext}(I))$. 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, $\text{succ}^*(c, \text{ext}(I))$ 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 $\Omega$-clauses.

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

In Example 1, $\text{succ}^*(c, \text{ext}(I))$ is a convergent point of $\text{ext}(I)$ for one-of$(f, g)$. A convergent point is only useful for the search of a solution if there exists a action sequence $\alpha$ 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, $\Omega$ be a belief state, $\alpha$ be an action sequence, and $o$ be an one-of clause. We say that $S$ is an abstract point of $\Omega$ for $o$ w.r.t. $\alpha$, denoted by $\Omega \prec S[\alpha]$, if $\Omega \prec S[o]$ and $\text{succ}^*(\alpha, \Omega) \subseteq S$. We call $\alpha$ an abstract path for $o$ from $S$ to $\Omega$. $S$ is an abstract point of $\Omega$ for $o$ if there exists an action sequence $\alpha$ such that $\Omega \prec S[o]$.

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

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

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

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

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

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

999
\( \beta \) 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 \( P = (F, O, I, G) \) by (i) finding a viable landmark \( S \) of \( P \); and (ii) solving the problem \( P' = (F, O, S, G) \). Prop. 1 implies that instead of \( S \), we can find an action sequence \( \alpha \) and solve the problem \( P'' = (F, O, \text{succ}'(\alpha, \text{ext}(I)), G) \). By definition of a viable landmark, we know that there exists an one-of clause \( o \in I \) such that every literal \( \ell \) in \( o \) is known to be either true or false in \( S \) (or in \( \text{succ}'(\alpha, \text{ext}(I)) \)). 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 \( \text{succ}'(\alpha, \text{ext}(I)) \)) will be easier than the search from the initial belief state \( \text{ext}(I) \). However, if \( o \) 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 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 \( P = (F, O, I, G) \). 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., \( \Sigma^P \) (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 \( \Omega \) 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 \( P = \{(1, 2, 3), O, \{\text{one-of}(1, 2, 3)\}, \{2\}\} \) where \( O \) contains \( l : i \rightarrow i + 1 \) for \( i = 2, 3 \) and \( r : i \rightarrow i + 1 \) for \( i = 1, 2 \); and \( \text{pre}(l) = \text{pre}(r) = \emptyset \). \( i \) indicates the location, and \( l \) and \( r \) stand for \( \text{left} \) and \( \text{right} \). The initial belief state \( \Omega = \text{ext}(I) \) consists of 3 states, \( s_1 = \{1, 2, 3\} \), \( s_2 = \{1, 2, 3\} \), and \( s_3 = \{1, 2, 3\} \).

We can check that each \( s_i \) is an abstract point of the \( \Omega \) for one-of \((1, 2, 3)\). Consider \( s_3 \). We observe that for \( \alpha_1 = [r, r] \) \( \alpha_2 = [r, r] \) and \( \alpha_3 = [], \) it holds that \( \text{succ}'(\alpha_1, s_i) = s_3 \). Furthermore, \( \text{succ}'(\alpha_2, s_i) = s_3 \). It is easy to see that \( \text{succ}'(\alpha_3, s_i) = s_3 \). We can reduce the original domain to a domain related to \( \text{succ}'(\alpha, s_i) \).

**Example 3.** Let us consider a variant of the cube problem encoded by \( P = \{(1, 2, 3), O, \{\text{one-of}(1, 2, 3), \text{one-of}(1, 2, 3)\}, \{4\}\} \) where \( O \) contains \( a : 1 \rightarrow 3; b : 2 \rightarrow 3, -2; c : 3 \rightarrow 4, \) and \( \text{pre}(a) = \text{pre}(b) = \text{pre}(c) = \emptyset \). The initial belief state \( \Omega = \text{ext}(I) \) consists of 2 states, \( s_1 = \{-2, 3, -4\} \) and \( s_2 = \{-2, 3, -4\} \).

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

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 \( \Omega \) if there exist a belief state \( S \) and an action sequence \( \alpha \) such that \( \text{succ}'(\alpha, s_i) \in S \) for each \( s_i \) such that \( 1 \leq j \leq n \).

Clearly, if \( o \) is open-convergent in \( \Omega \) then we know that there exists some \( S \) and \( \alpha \) such that \( \Omega \overset{s}{\sim} S[o] \).

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 \( \delta \) is an interpretation of \( o \) if (i) for every literal \( l \in \delta \), \( l \in \text{lit}(o) \cup \overline{\text{lit}(o)} \); 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 \( s|o = s \cap (\text{lit}(o) \cup \overline{\text{lit}(o)}) \). For a belief state \( \Omega \), let \( \Omega|o = \{s|o | s \in \Omega \} \).

**Proposition 3.** Let \( \Omega \) be a belief state, \( o \) be an one-of clause, and \( \delta \) be an interpretation of \( o \). If \( \alpha \) is a plan achieving \( \delta \) from \( \Omega \), then \( \Omega \overset{s}{\sim} \text{succ}'(\alpha, \Omega)[o] \).

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 \( \Omega \), then it is reasonable to focus on the set of interpretations of \( o \) that could be reached from \( \Omega \), 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:
- \( \forall \phi |_{\Omega} \rightarrow \psi_{|\Omega} \in O' \) if \( \forall \phi |_{\Omega} \rightarrow \psi_{|\Omega} \in O \) and \( \text{pre}(a)|_{\Omega} \) satisfies \( o \);
- for each \( a \in O' \), \( \text{pre}(a) \) is changed to \( \text{pre}(a)|_{\Omega} \).

1000
• $F'$ 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 $(\delta, a, \delta') \in E$ if there exist $a \in Ab(F, O, o)$ such that $\text{succ}(a, \delta)$ satisfies $\delta'$. The reduced graph of a belief state $\Omega$, denoted by $G(\Omega, o)$, is obtained by removing from $G(o)$ every node $\delta$ that is not reachable from a node in $\Omega$, 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 $\delta$ 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 $\Omega$, we say that $o$ is probably closed-convergent in $\Omega$ if for every action $a$ in $Ab(F, O, o)$, $a$ maintains $o$, i.e., for every interpretation $\delta$ of $o$, $\text{succ}(a, \delta)$ satisfies $o$. Otherwise, $o$ is probably open-convergent in $\Omega$. We then identify an interpretation $\delta$ in $G(\Omega, o)$ as follows.

• $o$ is probably closed-convergent: selects $\delta$ which satisfies $o$ and has the minimal number of outgoing links; and

• $o$ is probably open-convergent: selects $\delta$ 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 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 $a' : o_1 \rightarrow o_2$ in $O$ such that $\text{pre}(a)$ satisfies $o_1$ and $l \in \text{lit}(o_2) \cup \text{lit}(o_1)$.

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 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;

• $lv(o) > \max\{lv(o') : o'$ 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 $P_{cl}$
denote a classical and a conformant planner, respectively.  
\( \text{CPLS}(P_{cl}, P_{cf}) \) denotes a conformant planner obtained by 
integrating the new technique—described in the previous 
sections—into \( P_{cf} \) and \( P_{cl} \). The overall algorithm implemented 
in \( \text{CPLS}(P_{cl}, P_{cf}) \) is given in Fig. 2. CPLCL is a con-
formant planner\(^1\) used for the verification of abstract points.

![Figure 2: Overall Structure for CPLS\((P_{cl}, P_{cf})\)](image)

Given a problem \( P \), \( \text{CPLS}(P_{cl}, P_{cf}) \) starts by computing 
a stratification of the one-of clauses in \( P \). If no stratification 
exists, the conformant planner \( P_{cf} \) is used to find a solution 
of \( P \). Otherwise, \( \text{CPLS}(P_{cl}, P_{cf}) \) 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 \( \text{CPLC} \) 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, \( P_{cl} \), is used to solve 
the problem. Otherwise (Fig. 2, (C)), \( P_{cf} \) is used. If \( P_{cl} \) fails 
to find a solution (Fig. 2, (D)), then \( \text{CPLS}(P_{cl}, P_{cf}) \) uses \( P_{cf} \) 
to find a solution.

Observe that if \( P_{cl} \) is invoked and succeeds, the 
performance of \( \text{CPLS}(P_{cl}, P_{cf}) \) does not depend on the 
performance of \( P_{cf} \). Otherwise, the overall overhead of 
\( \text{CPLS}(P_{cl}, P_{cf}) \), comparing to \( P_{cf} \), 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 \( P_{cl} \). 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 \( \text{CPLS}(P_{cl}, P_{cf}) \) using the following 
planners: \( P_{cl} \) is the planner \( \text{LAMA} \) (Richter and Westphal 2010) 
and \( P_{cf} \) is one of the four conformant planners—\( \text{CPA} \) (Tran 
et al. 2009), \( \text{DNF} \) (To, Pontelli, and Son 2009), \( \top 0 \) (Palacios and 
Geffner 2009), and \( \text{CPLC} \) (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 distribu-
tion of CFF, \( \top 0 \), and CNF. Due to lack of space, we omit 
the precise description of each domain.

**Applicability of Reducing Uncertainty**  
Table 2 shows the effectiveness of the module for reducing the 
uncertainty of \( \text{CPLS}(\text{LAMA}, P_{cf}) \) in some representative 
instances. In summary, out of 1050 instances of 23 domains, 
\( \text{CPLS}(\text{LAMA}, P_{cf}) \) 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., \text{adder,} 
IPC 2006); (ii) the greedy algorithm for predicting abstract 
points fails to yield a “correct” abstract point (e.g., block).

![Table 1: Statistics of the technique on some domains](image)

**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, \( \top 0 \), DNF, or \( \text{CPLC} \) 
reports the time and solution length from the corresponding 
planner. Column \( \text{Ovh} \) reports the overhead incurred by

\(^1\)CPLC (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.

<table>
<thead>
<tr>
<th>Instance</th>
<th>CPA(H)</th>
<th>t0</th>
<th>DNF</th>
<th>CPLS</th>
<th>CPLS(...)</th>
<th>Ovh</th>
</tr>
</thead>
<tbody>
<tr>
<td>ds-8-2</td>
<td>360/240</td>
<td>134.639</td>
<td>54.4302</td>
<td>1.68585</td>
<td>0.35/122</td>
<td>0.17</td>
</tr>
<tr>
<td>ds-8-3</td>
<td>225/2227</td>
<td>134.761</td>
<td>54.4829</td>
<td>1.59/262</td>
<td>0.35/267</td>
<td>0.26</td>
</tr>
<tr>
<td>ds-12-7</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>13/2836</td>
<td>12.06/2549</td>
<td>0.37</td>
</tr>
<tr>
<td>ds-12-9</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>20/6712</td>
<td>18.83/2103</td>
<td>0.21</td>
</tr>
<tr>
<td>push-8-2</td>
<td>213/1099</td>
<td>AB</td>
<td>68/903</td>
<td>2.52/880</td>
<td>6.68/3038</td>
<td>0.28</td>
</tr>
<tr>
<td>push-12-9</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>35.4/761</td>
<td>84.9/2754</td>
<td>0.57</td>
</tr>
<tr>
<td>ling-8-2-1</td>
<td>104/534</td>
<td>28.83/8</td>
<td>48.41/73</td>
<td>2.46/358</td>
<td>1.16/44</td>
<td>0.99</td>
</tr>
<tr>
<td>ling-12-1-4</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>95/2/109</td>
<td>12.22/46</td>
<td>11.09</td>
</tr>
<tr>
<td>ling-12-5-4</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>265/109</td>
<td>179.2/2746</td>
<td>167.47</td>
</tr>
<tr>
<td>1-de-8-7-dis</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>3.21/1026</td>
<td>1.39/1454</td>
<td>0.86</td>
</tr>
<tr>
<td>1-de-12-9-dis</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>83.2/346</td>
<td>10.40/1080</td>
<td>3.2</td>
</tr>
</tbody>
</table>

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 CPL only in a few instances of the push domains.

<table>
<thead>
<tr>
<th>Instance</th>
<th>CPA(H)</th>
<th>CPLS</th>
<th>Ovh</th>
</tr>
</thead>
<tbody>
<tr>
<td>or-coms-20</td>
<td>0.137/1027</td>
<td>1.10/1114</td>
<td>0.10/163</td>
</tr>
<tr>
<td>or-coms-30</td>
<td>0.611/302</td>
<td>1.10/1114</td>
<td>0.10/163</td>
</tr>
<tr>
<td>or-ds-8-2</td>
<td>22.36/2639</td>
<td>0.64/264</td>
<td>0.25/262</td>
</tr>
<tr>
<td>or-ds-8-3</td>
<td>275.302/761</td>
<td>27.85/922</td>
<td>0.25/761</td>
</tr>
<tr>
<td>or-push-8-3</td>
<td>AB</td>
<td>2.53/903</td>
<td>11.5/365</td>
</tr>
<tr>
<td>or-push-12-9</td>
<td>AB</td>
<td>25.07/267</td>
<td>8.34</td>
</tr>
<tr>
<td>or-push-12-7</td>
<td>AB</td>
<td>35.7/267</td>
<td>8.34</td>
</tr>
<tr>
<td>or-ling-8-2-1</td>
<td>AB</td>
<td>35.7/267</td>
<td>8.34</td>
</tr>
<tr>
<td>or-ling-12-1-4</td>
<td>AB</td>
<td>179.2/2746</td>
<td>167.47</td>
</tr>
<tr>
<td>or-ling-12-5-4</td>
<td>AB</td>
<td>179.2/2746</td>
<td>167.47</td>
</tr>
<tr>
<td>or-1-de-8-5-dis</td>
<td>AB</td>
<td>0.45/1454</td>
<td>0.86</td>
</tr>
<tr>
<td>or-1-de-8-9-dis</td>
<td>AB</td>
<td>1.38/154</td>
<td>1.25</td>
</tr>
<tr>
<td>or-1-de-12-7-dis</td>
<td>AB</td>
<td>7.57/1080</td>
<td>0.57</td>
</tr>
<tr>
<td>or-1-de-12-9-dis</td>
<td>AB</td>
<td>10.40/1080</td>
<td>3.2</td>
</tr>
</tbody>
</table>

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

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.