Effective Heuristics and Belief Tracking for Planning with Incomplete Information

Alexandre Albore
Universitat Pompeu Fabra
Barcelona, SPAIN
alexandre.albore@upf.edu

Miquel Ramírez
Universitat Pompeu Fabra
Barcelona, SPAIN
miquel.ramirez@upf.edu

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, SPAIN
hector.geffner@upf.edu

Abstract

Conformant planning can be formulated as a path-finding problem in belief space where the two main challenges are the heuristics to guide the search, and the representation and update of beliefs. In the translation-based approach recently introduced by Palacios and Geffner, the two aspects are handled together by translating conformant problems into classical ones that are solved with classical planners. While competitive with state-of-the-art methods, the translation-based approach runs however into three difficulties. First, complete translations are expensive for problems with high width; second, incomplete translations can generate infinite heuristic values for problems that are solvable; and third, aspects that are specific to the conformant setting, such as the cardinality of beliefs, are not accounted for.

In this work, we build on the translation-based approach but not for solving conformant problems with a classical planner but for deriving heuristics and computing beliefs in the context of a standard belief-space planner. For this, a novel translation K_S^i is introduced that is always complete, but which is sound for problems with width bounded by i. A new conformant planner, called T1, builds then on this translation for $i = 1$, extending the heuristic that results with a second heuristic obtained from invariant 'oneof expressions'. A number of experiments is performed to compare T1 with state-of-the-art conformant planners.

Introduction

Conformant planning with deterministic actions is one the simplest form of planning with uncertainty. A deterministic conformant problem is like a classical problem but with many possible initial states instead of one, and a plan is conformant when it works for each one of them. In spite of its simplicity, the conformant planning problem is harder than classical planning (Haslum and Jonsson 1999; Turner 2002) and lies at the heart of recent methods for computing contingent plans and deriving finite-state controllers (Hoffmann and Brafman 2005; Bonet, Palacios, and Geffner 2009).

Conformant planning can be formulated as a path-finding problem in belief space, where the computational challenges are the heuristics to guide the search, and the belief representation and update (Bonet and Geffner 2000). This formulation is the basis of the most recent conformant planners such as Conformant-FF, MBP, POND, and CNF (Brafman and Hoffmann 2004; Bertoli et al. 2006; Bryce, Kambhampati, and Smith 2006; To, Son, and Pontelli 2010). The exception is the planner T0 which is based on a translation of conformant problems P into classical problems $K(P)$ that are solved by off-the-shelf classical planners (Palacios and Geffner 2009).

The translation-based approach is competitive with belief search approaches, and is in fact an instance of them, with beliefs over the conformant problem P encoded as states over the translation $K(P)$, and heuristics over beliefs reduced to classical heuristics over states. In spite of its performance, however, the translation-based approach runs into three problems. The first is that complete translations, i.e., those in which every conformant plan for P appears as a classical plan for $K(P)$, are size-exponential in the width of P. The second, no less important, is that incomplete translations $K(P)$ cannot be used for heuristic guidance, as the heuristic values that they generate can be infinite even when the problem P is solvable. Last, aspects that are specific to the conformant setting and that proved useful then, like the cardinality of beliefs, seem to get lost in the translation.

In this work, we build on the translation-based approach but not for solving conformant problems with a classical planner but for deriving heuristics and computing beliefs inside a standard belief-space planner. The basis of the new planner called T1 is a new translation K_S^i, that unlike the translation K_i considered by Palacios and Geffner, is always *tractable* and *complete*, but not always sound. The translation K_S^i is *sound* however for problems P with conformant width no greater than i. Palacios' and Geffner's K_i translation, on the other hand, is *tractable* and *sound*, but *complete* for problems with width bounded by i. Thus, while K_i gives up completeness for problems with width beyond i, K_S^i gives up soundness then. For conformant planning this means two things. First, that heuristics based on the new translation K_S^i will produce infinite values only when the problem is unsolvable. Second, that the belief literals resulting from this translation must be checked for validity in certain cases. The conformant planner T1 uses the translation K_S^i for $i = 1$ to generate *heuristic* and *candidate belief literals*. The beliefs are then verified with a SAT engine in the way it is done by conformant-FF (Brafman and Hoffmann

2004). Moreover, the heuristic resulting from the translation K^1_S is extended with a second heuristic that is obtained from *invariant 'oneof expressions'* derived from the problem. As we will see, this second heuristic is related to both *cardinality heuristics* (Bertoli and Cimatti 2002), and *landmark heuristics* (Richter, Helmert, and Westphal 2008).

The paper is organized as follows. We start with a review of conformant planning and the translation-based approach. We then introduce the new translation and the conformant planner that uses it. We finally present experimental results and a brief discussion.

Conformant Planning

We consider deterministic conformant planning problems P given by tuples of the form $P = \langle F, O, I, G\rangle$ where F stands for the fluent symbols, O for a set of actions a, I for the set of clauses over F defining the initial situation, and G for the set of goal literals over F defining the goal. In addition, every action a has a precondition given by a set of fluent literals, and a set of conditional effects $C \rightarrow L$ where C is a set of fluent literals and L is a literal. We refer to the conditional effects $C \rightarrow L$ of an action a as the *rules* associated with a, and sometimes write them as $a : C \rightarrow L$, where C may be empty. When L is a literal, we take $\neg L$ to denote the complement of L.

A state s is a truth assignment over the fluents F and a possible initial state s of P is a state that satisfies the clauses in I. For a state s, we write $I(s)$ to refer to the set of atoms (positive literals) that are true in s, and write P/s to refer to the *classical planning problem* $P/s = \langle F, I(s), O, G\rangle$ which is like the conformant problem P except for the initial state that is fixed to s.

An action sequence $\pi = \{a_0, a_1, \ldots, a_n\}$ is a *classical plan* for P/s if the action sequence π is executable in the state s and results in a goal state. Likewise, an action sequence π is a *conformant plan* for P iff π is a classical plan for P/s for every possible initial state s of P.

The most common approach to conformant planning is based on the belief state formulation (Bonet and Geffner 2000). A belief state b is the non-empty set of states that are deemed possible in a given situation, and every action a executable in b, maps b into a belief state b_a. The conformant planning task becomes a path-finding problem in a graph where the nodes are belief states b, the source node b_0 is the belief state corresponding to the initial situation, and the target belief states b_G are those where the goals are true.

Translation into Classical Planning

The translation-based approach to conformant planning maps deterministic conformant problems P into classical problems $K(P)$ that can be solved by off-the-shelf planners (Palacios and Geffner 2009). The approach is also closely related to the notion *planning at the knowledge level* formulated in (Petrick and Bacchus 2002), except that the epistemic encoding is derived automatically.

The simplest translation, called K_0, replaces the literals L in P by literals KL and $K\neg L$ that aim at capturing whether L is 'known to be true' and 'known to be false' respectively.

Definition 1 *For a* deterministic *conformant problem* $P = \langle F, I, O, G\rangle$*, the translation* $K_0(P) = \langle F', I', O', G'\rangle$ *is a classical planning problem where*

- $F' = \{KL, K\neg L \mid L \in F\}$,
- $I' = \{KL \mid L$ *is a unit clause in* $I\}$,
- $G' = \{KL \mid L \in G\}$, *and*
- O' = O *with each precondition* L *for* $a \in O$ *replaced by* KL*, and each conditional effect* $C \rightarrow L$ *replaced by* $KC \rightarrow KL$ *and* $\neg K\neg C \rightarrow \neg K\neg L$.

The expressions KC and $\neg K\neg C$ for $C = L_1, L_2 \ldots$ are abbreviations for $KL_1, KL_2, \ldots$ and $\neg K\neg L_1, \neg K\neg L_2, \ldots$ respectively. The intuition behind the translation is simple: first, the literal KL is true in I' if L is known to be true in I; otherwise it is false. This removes all uncertainty from $K_0(P)$, making it a classical planning problem. In addition, to preserve soundness, each rule $a : C \rightarrow L$ in P is mapped into *two* rules: a **support rule** $a : KC \rightarrow KL$, that ensures that L is known to be true when the condition C is known to be true, and a **cancellation rule** $a : \neg K\neg C \rightarrow \neg K\neg L$ that guarantees that $K\neg L$ is deleted (prevented to persist) when action a is applied and C is not known to be false.

The translation $K_0(P)$ is sound as every classical plan that solves $K_0(P)$ is a conformant plan for P, but incomplete, as not all conformant plans for P are classical plans for $K_0(P)$.

The more general translation scheme $K_{T,M}$ builds on K_0 using two parameters: a set T of *tags* t and a set M of merges m. The tags and the merges are used to account for conformant plans that reason by cases; indeed, the tags are used to introduce assumptions about the initial situation that are eliminated via the merges. The new literals KL/t in the translation express that L is known to be true if t is true in the initial situation. A merge m is a non-empty collection of tags t in T that stands for the Disjunctive Normal Form (DNF) formula $\bigvee_{t\in m} t$. A merge m is *valid* when one of the tags $t \in m$ must be true in I. The translation $K_{T,M}$ is sound as long as the merges in M are valid. A merge m for a literal L in P translates into the 'merge action' with single effect

$$\bigwedge_{t\in m} KL/t \;\rightarrow\; KL \quad .$$

The set of 'merge actions' associated with the set of merges M is referred to as O_M. The translation $K_{T,M}(P)$ is the basic translation $K_0(P)$ 'conditioned' with the tags t in T and extended with the set O_M of actions. The literals KL are assumed to stand for the literals KL/t where t is the 'empty tag'. The empty tag expresses no assumption about the initial situation and is assumed implicitly in every set T.

Definition 2 *Let* $P = \langle F, I, O, G\rangle$ *be a conformant problem, then* $K_{T,M}(P) = \langle F', I', O', G'\rangle$ *is a* classical planning problem *where*

- $F' = \{KL/t, K\neg L/t \mid L \in F$ *and* $t \in T\}$,
- $I' = \{KL/t \mid I, t \models L\}$,
- $G' = \{KL \mid L \in G\}$, *and*
- O' = $O_M \cup O$ *with each precondition* L *for* $a \in O$ *replaced by* KL*, and each conditional effect* $C \rightarrow L$ *replaced by* $KC/t \rightarrow KL/t$ *and* $\neg K\neg C/t \rightarrow \neg K\neg L/t$.

The translation $K_{T,M}$ reduces to the basic translation K_0 when M is empty and T contains only the empty tag. For suitable choices of T and M, the translation $K_{T,M}$ is both *sound* and *complete*: sound, meaning that the classical plans for $K_{T,M}(P)$ are all conformant plans for P once merge actions are removed, and complete, meaning that all conformant plans for P yield classical plans for $K_{T,M}(P)$ once merge actions are added.

One way to get a complete translation is by associating the tags in T with the set S_0 of possible initial states of P (in addition to the empty tag). This translation is called K_{S_0}, which is the instance of the general translation $K_{T,M}$ that results from setting T to S_0 and having one merge in M equal to S_0 for each precondition and goal literal. The translation K_{S_0} is complete but exponential in the worst case. On the other hand, the translation K_i, for an integer $i \geq 0$, is exponential only in i, and complete for problems P with *width* bounded by i. The width of a conformant problem is related to the maximum number of variables whose values are initially unknown, such that all of them are relevant to a certain precondition or goal (Palacios and Geffner 2009).

A New Translation

The main limitation of the translation-based approach is that complete translations are not feasible for problems with large width, while incomplete translations may end up mapping a solvable problem into an unsolvable one. In order to benefit from tractable translations while avoiding this limitation, we take a different approach: we formulate *a new translation that is tractable and complete, but which is not always sound.* The new translation can't be used to solve conformant problems with a classical planner, but is effective for deriving heuristics and computing beliefs inside a belief-space planner. The key idea is to sample and use a polynomial number of initial states, even when their total number is exponential.

We start with the notion of *basis* (Palacios and Geffner 2009). For a conformant problem P and a set of states $S \subseteq S_0$, where S_0 is the set of possible initial states of P, let $P[S]$ stand for the conformant problem that is like P but with the set of initial states restricted to S. $P[S]$ is thus equal to P when $S = S_0$.

Definition 3 *S is a* basis *for P iff any conformant plan for $P[S]$ is a conformant plan for P.*

If S is a basis for P, all states in $S_0 \setminus S$ can be ignored, as the plans for $P[S]$ are plans for P, while the plans for P are always plans for $P[S]$. Bases S are computationally useful when $|S|$ is exponentially smaller than $|S_0|$. Palacios and Geffner show that

Theorem 4 *Conformant problems P with width bounded by i have bases of size exponential in i even if the size of S_0 is exponential in the number of fluents.*

As an example, if the only uncertainty in P is due to a clause $x_1 \vee \cdots \vee x_n$ in I, and the x_k's are the only uncertain literals in I that are *relevant* to a precondition or goal literal L (i.e., their negations $\neg x_k$ are not relevant to a precondition or goal), then S_0 will have 2^n possible initial states, but the set of n states S that make just one of the x_k literals true forms a basis for P. The proof of the theorem relies on the following result, where the expression $rel(s, L)$ is used to denote the set of literals in the state s that are *relevant* to a literal L:

Theorem 5 *$S \subseteq S_0$ is a basis for P if for each precondition or goal literal L, and each state s in S_0, S contains a state s' such that $rel(s', L) \subseteq rel(s, L)$.*

In the example, the set S_0 contains states where multiple x_k's are true, yet S above is a basis, as for any state s in S_0, there is state s' in S where just one of those literals x_k is true.

The notion of relevance defined by Palacios and Geffner depends on the conditional effects, and not on the action preconditions that must be known for certainty and hence do not 'propagate' uncertainty. The proof of Theorem 5 that relates relevance and bases, relies on the fact that an (applicable) action sequence achieves a literal L from a state s if it achieves L from a state s' with less relevant fluents; namely such that $rel(s', L) \subseteq rel(s, L)$.

We will extend these results to define a novel translation that is the old K_{S_0} translation but applied not to P but to $P[S]$ for a suitable subset S of S_0. Let's define first the translation $K_S(P)$ where $S \subseteq S_0$:

Definition 6 *K_S is K_{S_0} but applied to $P[S]$ rather than to P, i.e. $K_S(P)$ is $K_{S_0}(P[S])$.*

From the above definitions and the properties of K_0, it's easy to see that

Theorem 7 *The translation $K_S(P)$ is* complete.

On the other hand, the soundness of the translation depends on whether S is a basis for P:

Theorem 8 *$K_S(P)$ is* sound *if S is a* basis *for P.*

It follows from this that if we can define S to be a basis for P when the width of P is bounded by i, we will guarantee that the K_S translation will be sound for problems with width no greater than i, and complete for all problems. We will denote the set of samples needed for this as S^i, and we will define this set in terms of the samples $S^i(L)$ needed for each precondition and goal literal L.

For a fluent formula C consistent with I, let $s_L(C)$ stand for any state that satisfies I, C, and *a minimum number of literals relevant to L*. If we refer to $|rel(s, L)|$ as the L-rank of s or simply the rank of s, then $s_L(C)$ stands for any lowest-ranked state that satisfies I and C. There is always one such state, although it is not necessarily unique.

For the example above, if C is x_k, the states s that satisfy x_k and no other x_j literal, will have rank 1, while the states that satisfy x_k and all other x_j literals will have rank n. The expression $s_L(C)$ will thus denote any of the former states.

Let now $T^i(L)$ stand for the set of tuples t of at most i literals L' in P that are relevant to L, are initially unknown in P, and such that t is jointly consistent with I. These tuples play the role of the tags in the $K_{T,M}$ translation, representing consistent sets of assumptions about the initial situation that are relevant to L. For $i = 0$, the only tuple in $T^i(L)$ is

$t = true$. The set of samples S^i for any conformant problem P and any $i \geq 0$ is then defined as:

$$S^i \stackrel{\text{def}}{=} \bigcup_L S^i(L)$$

where L ranges over the precondition and goal literals in P, and $S^i(L)$ is

$$S^i(L) \stackrel{\text{def}}{=} \{\, s_L(t) \mid t \in T^i(L) \}\quad .$$

In words, S^i is defined to contain, for each precondition and goal literal L in P and each tuple of assumptions (tags) $t \in T^i(L)$, a sample s that satisfies t and a *minimum number* of literals relevant to L. Since these states are not necessarily unique, the set S^i is not unique either. This definition yields the samples from S_0 that we want:

Theorem 9 *S^i is a basis for P if the width of P is bounded by i.*

The proof of this involves an alternative characterization of the notion of width. For a tuple (tag) t in $T^i(L)$, let t^* stand for the tuple that extends t with all the literals deducible from t and I, and let $m^i(L)$ stand for the subset of tuples t in $T^i(L)$ that admit states that do not make true additional literals relevant to L; i.e.[1]

$$m^i(L) \stackrel{\text{def}}{=} \{\, t \mid t \in T^i(L) \text{ such that } rel\,(s_L(t), L) \subseteq t^* \}\,.$$

If we take $m^i(L)$ to stand for the 'merge' or DNF formula

$$\bigvee_{t \in m^i(L)} \bigwedge_{L' \in t} L'\ ,$$

it turns that the *width* of a literal L is just the minimum value of i that renders the merge valid in I:

Theorem 10 *The width $w(L)$ of literal L is the min i value for which the DNF formula $m^i(L)$ is entailed by I.*

Recall that the width of P, $w(P)$, is just $\max_L w(L)$ where L ranges over the precondition and goal literals in P. Theorem 10 thus imply that $w(P)$ is the min value of i for which $m^i(L)$ is a valid merge for any precondition or goal L. Palacios' and Geffner's K_i translation is an instance of the $K_{T,M}$ translation that uses these merges along with the tags in them. The translation K^i_S below is a variation of the K_{S0} translation that uses min-ranked states $s_L(t)$ associated with the tags instead.

For proving Theorem 9: if $m^i(L)$ is valid in I, every state s that satisfies I must also satisfy a tag t in $m^i(L)$, but then from the definition of $m^i(L)$, for any state $s' = s_L(t)$, we will have that $rel(s', L) \subseteq rel(s, L)$, and from Theorem 4, that the set $S^i(L)$ of states $s_L(t)$ for $t \in m^i(L)$ forms a basis for P, provided that L is the only precondition or goal. Else, the union of all such sets $S^i(L)$ is required as expressed in the definition of S^i.

Defining finally the translation $K^i_S(P)$ as $K_S(P)$ for $S = S^i$, it's direct to prove that

[1]Note that while the state denoted by $s_L(t)$ is not necessarily unique as it may denote any initial state s that satisfies t and minimizes $|rel(s, L)|$, if the relation $rel(s, L) \subseteq t^*$ is true for one of them, it will be true for any other.

Theorem 11 *The translation $K^i_S(P)$ has a size that is exponential in i, is complete, and is sound for problems P with width bounded by i.*

For the example above, the $K^i_S(P)$ translation for $i = 1$ is like the $K_{S_0}(P)$ translation, except that the possible initial states that make more than one x_k literal true are discarded. Thus, in this case, the effect of the translation is to replace the inclusive disjunction $x_1 \vee \cdots \vee x_n$ by the exclusive disjunction $oneof(x_1, \ldots, x_n)$. The result would be different, however, if some negative literals $\neg x_k$ were relevant to a precondition or goal.

While the T0 planner uses the $K_1(P)$ translation that is complete for problems with width bounded by 1 and is always sound, the new planner T1 uses the $K^1_S(P)$ translation that is sound for problems with width bounded by 1 and is always complete. The set S^1 of samples is obtained from its definition by computing the lowest ranked states $s_L(L')$ that satisfy $I \cup \{L'\}$ for various literals L'. These states are computed by compiling I into d-DNNF (Darwiche 2002), a logical form that allows the computation of a number of otherwise intractable operations, such as consistency, model counting, and lowest ranked models, in time that is linear in the size of the compilation (which is at worst exponential in the number of variables, but not necessarily so).

The d-DNNF compilation is also used for selecting the min-ranked states $s_L(t)$ when they are not unique so that the sets $S^i(L)$ overlap as much as possible, and hence, the size of their union S^i is minimized. For example, if L and L' are the only preconditions or goals in P, I is given by the disjunctions $x_1 \vee \cdots \vee x_n$ and $y_1 \vee \cdots \vee y_n$, and the only uncertain literals relevant to L and L' are the x_k's and y_k's respectively, then arbitrary choices of the sets $s_L(x_k)$ and $s_{L'}(y_k)$ may result in a sample set S^1 with $2n$ states. On the other hand, it is possible to choose the states $s_L(x_k)$ to be equal to the states $s_L(y_k)$, e.g. by making all other x's and y's false, so that the size of S^1 is just n.

These optimizations are implemented by exploiting the capabilities of the d-DNNF compilation. Other optimizations, like removing states $s_L(t)$ from $S^i(L)$, when the tag t can be removed from the merge $m^i(L)$ without affecting its validity, are accommodated as well. These optimizations are aimed at minimizing the size of the sample set S^i while retaining its properties.

The T1 Planner

The conformant planner T1 uses the sample set S^1 of the $K^1_S(P)$ translation for deriving heuristics and tentative belief literals that are verified and used in the context of a belief-space search algorithm. No verification however is needed for problems with width 0 or 1. We present in order the search graph, the heuristics, and the search algorithm.

Nodes and Beliefs

A node in the search graph represents a belief state and corresponds to a tuple $n = \langle \pi, S, R \rangle$, where π is the plan prefix used to reach n from the root node of the search, S is the sample set S^1 progressed through π, and R is a set of lit-

erals.[2] The belief state b_n represented by the node n corresponds to the whole set S_0 of initial states of P progressed through π. This belief state is not computed explicitly but is represented implicitly in the plan prefix π. The set of literals in R are sound with respect to b_n but are not necessarily complete; i.e. R contains literals all of which are known to be true in b_n, but it does not contain in general all such literals. Likewise, S is complete with respect to b_n but not necessarily sound; i.e., all literals true in b_n are true in all the states in S, but not the other way around; i.e.

$$\text{Literals in } R \subseteq \text{ Literals true in } b_n \subseteq \text{ Literals true in } S$$

Still, if the problem has width bounded by 1, the set of true precondition and goal literals coincide in b_n and S, and such literals are added to R.

The root node in the search is $n_0 = \langle \pi_0, S^1, R_0 \rangle$, where π_0 is the empty plan, S^1 is the approximation of the initial belief set in the $K^1_S(P)$ translation, and R_0 is the set of literals known to be true in the initial situation.

The goal nodes are the nodes $n = \langle \pi, S, R \rangle$ where R contains the goals of the problem.

The edges in the graph correspond to the applicable actions. An action is applicable in $n = \langle \pi, S, R \rangle$ when the action preconditions are all in R. The node that results from applying the action a is given by $n' = \langle \pi', S', R' \rangle$ where π' is π with the action a appended, S' is the result of progressing each of the states $s \in S$ through a, and R' is the set of literals obtained by progressing the set of literals R in n through the action a as follows: $L \in R'$ iff there is an effect $a : C \rightarrow L$ such that $C \subseteq R$ (L is added) or $L \in R$ and for every effect $a : C' \rightarrow \neg L$, R contains the complement of a literal in C' (L persists). The computation of R' from R corresponds to the application of the support and cancellation rules in the translation $K_0(P)$ that involves no tags, which corresponds in turn with the 0-approximation semantics (Baral and Son 1997).

While there is no need for the set of literals R in $n = \langle \pi, S, R \rangle$ to include all literals true in the belief state b_n for the planner to be complete, R must be complete with respect to action preconditions and goals. That is, if a goal is true in b_n or an action is applicable in b_n, then the goal and the action preconditions must be in R. For problems with width no larger than 1, this is guaranteed as the sample set S is not only complete then but sound, and hence the literals true in S can be added safely to R. For problems with higher width, on the other hand, a verification operation is carried out after the node has been generated by looking at the samples in S, and if necessary, by using a SAT solver as in Conformant-FF (Brafman and Hoffmann 2004). Basically, a precondition or goal literal L is added to R when L is true in S, and this belief is certified by calling a SAT solver over a suitable CNF formula. The formula corresponds to the semantics of the action sequence π, executed from the initial situation I: I is encoded at time 0, the conditional effects of the i-th action in the plan prefix π are encoded at each time slice $[i, i+1]$, and the *negation* of the literal L to be tested is encoded at time $|\pi|$. The literal L is true in b_n iff the resulting CNF formula is unsatisfiable.

A last operation is performed in T1 to test whether two nodes n and n' represent the same belief state $b_n = b_{n'}$. This test is not needed for completeness, but for saving duplicate work. When the problem width is 0 or 1, the two nodes $n = \langle \pi, S, R \rangle$ and $n' = \langle \pi', S', R' \rangle$ can be safely collapsed when $S = S'$ and $R = R'$. For problems with higher width, however, an additional test is needed. Basically, n and n' are collapsed if in addition for each possible initial state s of the problem, the plans π and π' result in the same state s'. This test which is sound is performed through a single SAT call.[3] Conformant-FF performs a similar test for the same purpose.

Classical Heuristic

We have defined the nodes of the search graph, the initial and goal nodes, and the directed edges. It is easy to show that the paths that connect the initial and goal nodes in the graph, encode the conformant plans for the problem. We now focus on the heuristics for efficiently finding one such path, not necessarily optimal. The planner T1 uses two heuristics in combination, that we call, the *classical heuristic* and the *certainty heuristic* for reasons that will be clear below.

The classical heuristic $h_C(n)$ for a node $n = \langle \pi, S, R \rangle$ is defined as the estimated cost of the *classical planning problem* $K_S(P)$. For the root node, this is the estimated cost of the translation $K^1_S(P)$, as S is then S^1.

The estimated cost of the classical problem $K_S(P)$ is obtained from the combination of the additive and relaxed plan heuristics (Bonet and Geffner 2001; Hoffmann and Nebel 2001), as formulated in (Keyder and Geffner 2008), where a relaxed plan is constructed backward from the goal, by collecting the best supporters of the atoms in the goal, and recursively, the best supporters of the preconditions of those supporters. The estimated cost is the number of actions in the relaxed plan, while the *helpful actions* (to be used in the search), are as in FF, the applicable actions that add a precondition or goal in the relaxed plan.

Certainty Heuristic

The *certainty heuristic* $h_K(n)$ is defined in terms of a set of 'oneof invariants' given explicitly in the problem or derived from it. A given oneof invariant is an exclusive disjunction $oneof(x_1, \ldots, x_n)$ in the initial situation I that is maintained by the actions in the problem. An action maintains a formula, if the formula is true after the action if it was true when the action was applied. More precisely, a

[2] In the implementation, the set S of progressed samples is not stored in the nodes; rather the progression is done when required.

[3] The SAT call is over two formulas like the one above for verifying if L follows from a plan prefix π from I, except that now two disjoint formulas are built, encoding the plan prefixes π and π', the first involving fluent variables x_i for atoms x in the problem, $0 \leq i \leq |\pi|$, and the second with primed variables x'_k for atoms x in the problem, $0 \leq k \leq |\pi'|$. Then the two formulas are joined along with two other formulas built as follows. The first, establishing the equivalence between the x variables at the beginning of the plans, $\wedge_x(x_0 \equiv x'_0)$, the second postulating a non-equivalence at the end of the plans, $\vee_x \neg(x_{|\pi|} \equiv x'_{|\pi'|})$, where x ranges in both cases over the fluents of the problem.

$oneof(x_1, \ldots, x_m)$ expression is invariant if all pairs x_i, x_k in it are mutex for $i \neq k$, and every action a with an effect $C \rightarrow \neg x_i$, has an effect $C \rightarrow x_k$, for $1 \leq i, k \leq m$. When a $oneof(x_1, \ldots, x_m)$ expression in I is not invariant, an attempt is made to find a set of literals $y_1, \ldots, y_l$ such that the expression $oneof(x_1, \ldots, x_n, y_1, \ldots, y_l)$ is true in I (i.e., that each y_i is mutex with the other literals) *and* invariant. For example, a problem with an object o in an unknown grid cell that can be picked up by a robot and placed in a box, can be encoded with a $oneof(at(o, c_1), \ldots, at(o, c_n))$ expression in I that is not invariant, but which can be completed into the invariant $oneof(at(o, c_1), \ldots, at(o, c_n), hold(o), at(o, box))$. The computation of these completions is simple and follows the computation of similar invariants in classical planning (Helmert 2009).

The *certainty heuristic* $h_K(n)$ for a node $n = \langle \pi, S, R \rangle$ is then defined as the number of literals x_i in oneof invariants with a literal in the goal, such that $\neg x_i$ is not in R. In other words, if we take the literals x_i in the invariants as representing the values $X = X_i$ of some *multivalued variable* X that is mentioned in the goal, $h_K(n)$ simply counts the possible values of such variables that have not yet been knocked out of the current belief. Indeed if x_i is in the goal, then at the end of any conformant plan x_i must be known, and so must the negation of the literals x_k that are mutex with x_i.

For example, in a 1×10 corridor where a robot can move either left or right, and 1 and 10 are the left and rightmost positions, the $oneof(x_1, \ldots, x_{10})$ expression is an invariant encoding the different positions of the robot. If the initial position is either 1 or 2 on the left, and the goal is to reach position 5, then $h_K(n_0)$ for the root node will be 2. The best action according to the certainty heuristic would be to move left, away from the goal, reducing the heuristic and the uncertainty by 1. On the other hand, the best action according to the classical heuristic would be to move right, towards the goal, which however cannot be achieved if the uncertainty is not first removed.

The certainty heuristic has relation to two heuristics used in planning that appeared to be completely orthogonal up to now. One is the *cardinality heuristic* that simply counts the number of states in a belief state (Bertoli and Cimatti 2002). The other is the *landmark heuristic*, popularized by the classical planner LAMA (Richter, Helmert, and Westphal 2008), that counts the number of unachieved landmarks, where a landmark is an atom that is made true by all plans (Hoffmann, Porteous, and Sebastia 2004). The relation to the cardinality heuristic is very direct in problems where all the uncertainty comes from a set of multi-valued variables that appear in the goal and whose initial value is unknown. In such cases, the target belief states have cardinality one, and hence it makes sense to establish a preference for belief states with lower cardinality. However, while the cardinality heuristic takes the *product* of the cardinalities of the beliefs over the variables, the certainty heuristic takes the *sum*. This sum corresponds to the number of literals that must be achieved in all the plans; namely, the goal literals x_i, along with the negation of all literals x_k that are mutex with x_i. These literals are like *epistemic landmarks*, meaning that any conformant plan must achieve all these literals with *certainty*.[4]

Search Algorithm

The two heuristics $h_C(n)$ and $h_K(n)$ are used in the context of a multi-queue best first search algorithm following the classical planners FD and LAMA (Helmert 2006; Richter, Helmert, and Westphal 2008). In T1, this is a best first search with three open lists Q1, Q2, and Q3. Children nodes obtained using helpful actions or actions that decrease the value of the certainty heuristic are placed in the first two queues; while those obtained with other actions are placed in the last queue. The expansions of the first two queues alternate, and every tenth iteration, the last queue is expanded. Expanding a queue means picking up the best node in the queue, checking if it is a goal node, and if not, producing all of its children. Nodes in Q1 and Q3 are ordered with h_C while nodes in Q2 are ordered with h_K. Ties are broken in each of the queues using first the other heuristic, and then the accumulated cost to the node.

Experimental Results

The conformant planner T1 involves five parts: parsing, sampling (computing S^1), search, computation of the heuristics, and verification of beliefs (for widths greater than 1). The parser is implemented on top of CONFORMANT–FF sources, the computation of the samples uses the CNF to d–DNNF compiler `c2d` (Darwiche 2004), while the verifications are done with MINISAT-2.2. T1 is written in `C++`.

We compared T1 with DNF (To, Pontelli, and Son 2009) and T0 (Palacios and Geffner 2009). Conformant-FF, POND (Bryce, Kambhampati, and Smith 2006), and MBP (Bertoli et al. 2006) perform well on several domains but have smaller coverage. We wanted to test also the recent CNF planner (To, Son, and Pontelli 2010), but it was not available yet from the authors. For DNF preprocessing we used SWI PROLOG rather than the proprietary SICSTUS PROLOG used by the authors. T0 has been used with FF (Hoffmann and Nebel 2001).

The results of the comparison are shown in Table 1. The problems are from past IPC competitions and various planner's distributions. For each of the 17 domains considered, three instances are shown, from easy to hard, when at least one planner solved them. The experiments have been executed on a cluster of multi–core, multi–CPU machines with a clock speed of 2.33GHz running Linux, with 2h and 2GB for time and memory outs.

T1 solves 42 problems out of 47 (89%), while T0 solves 38, and DNF 35. In general, however, T0 is fastest, solving most of the (translated problems) with the effective EHC search in FF (such instances appear with a number of expanded nodes $x + 0$, meaning that x nodes where expanded by EHC and 0 by the greedy best-first search). DNF does comparatively best on the Corner versions of the Square and Cube domains (Corners-Cube and Corners-Square), where

[4] The certainty heuristic as defined overestimates the true count by the number of one-of invariants with a literal in the goal, yet this difference is a constant that has no effect in the search.

	T0			DNF			T1			
Problem	T	L	E	T	L	E	T	L	E	$\lvert S_0^1\rvert$ / $\lvert b_0\rvert$
1-Dispose-8-1	**79**	1316	119+0	OM	–	–	150	**560**	3.6k	64/64
Blocks-1*	U	–	7+872	0.1	7	5	**0.05**	**6**	23	2/2
Blocks-2*	0.2	23	86+0	**0.1**	38	20	0.3	**20**	72	5/5
Blocks-3*	71	**80**	5.3k+0	**4.9**	307	125	TO	–	6.3k	12/125
Bomb-b100-t10	7.8	290	4.4k+0	**2.6**	**190**	17k	4.8	**190**	190	$101/2^{100}$
Bomb-b100-t60	**3.5**	**140**	1.1k+0	23	**140**	175k	21	**140**	140	$101/2^{100}$
Bomb-b100-t100	**7.0**	**100**	201+0	50	**100**	348k	OM	–	42	$101/2^{100}$
Coins-18	**0.1**	**97**	679+0	1.0	100	2.4k	2.8	191	528	$38/2 \cdot 10^6$
Coins-17	**0.1**	**96**	382+0	1.0	110	3.1k	13	313	1k	$43/2 \cdot 10^6$
Coins-21	OM	–	–	OM	–	–	OM	–	–	$185/1 \cdot 10^{16}$
Comm-20	**0.3**	**278**	1.1k+0	171.6	296	1.5k	3.0	295	616	$2\,/\,2^{20}$
Comm - 25	**1.2**	**453**	1.8k+0	1.8k	501	2.5k	13.1	478	1k	$2/2^{25}$
Comm-30	**2**	**593**	2.3k+0	OM	–	–	30	629	1.3k	$2/2^{30}$
Cube-67	33	**294**	7.6k+0	1895	2019	71k	**11**	303	308	$67/67^3$
Cube-91	OM	–	–	2200	2271	73k	**33**	**408**	408	$91/91^3$
Cube-139	OM	–	–	TO	–	–	**156**	**622**	623	$139/139^3$
Corners-Cube-15	**0.6**	147	5.8k+10k	0.7	**117**	1.4k	2.7	159	5.7k	2/8
Corners-Cube-20	2.2	258	13k+29k	**1.8**	**217**	2511	8.7	248	12k	2/8
Corners-Cube-55	OM	–	–	**18.6**	**806**	10k	768	1644	182k	2/8
Corners-Sqre-36	1.0	412	2.6k+16k	**0.7**	**138**	451	12	412	8.7k	2/4
Corners-Sqre-72	20	1474	10k+141k	**12**	**615**	3.3k	242	1474	44k	2/4
Corners-Sqre-120	190	3898	29k+681k	**75**	**1870**	10k	2754	4014	136k	2/4
Dispose-12-1	**55**	1274	267k+0	5590	**330**	13k	496	786	2.5k	144/144
Dispose-12-2	2037	1437	3922k+0	5810	**567**	18k	**4800**	1195	4.2k	$230/144^2$
Dispose-12-3	OM	–	–	**6305**	**1131**	34k	TO	–	3.6k	$316/144^3$
Logistics-4-3-3	**0.01**	**24**	53+0	7.48	160	16k	0.1	44	143	$10/4^3$
Logistics-2-10-10	**0.4**	**84**	414+0	OM	–	–	52	86	670	$11/2^{10}$
Logistics-4-10-10	**0.7**	**125**	774+0	OM	–	–	162	150	1.3k	$24/4^{10}$
Look-Grab-4-2-1*	U	–	3+16	OM	–	–	**2**	**42**	138	16/256
Look-Grab-8-1-1	59	242	6.4k+44	OM	–	–	**15**	**106**	377	64/64
Look-Grab-8-3-2*	OM	–	–	OM	–	–	**3749**	**56**	3.5k	$20/64^3$
Push-To-8-1	**83**	464	74k+0	134	**163**	3.9k	262	538	6.5k	64/64
Push-To-8-2	817	423	131k+0	195	**162**	124k	**91**	329	1.3k	$101/64^2$
Push-To-8-3	1213	597	132k+0	OM	–	–	**141**	**335**	1.3k	$139/64^3$
Raos-Keys-2*	**0.02**	**16**	22+0	0.5	22	70	0.7	21	130	2/4
Ring-10	**0.1**	55	530+0	1546	39	107	0.7	**41**	484	10/590k
Ring-20	**2.0**	95	2.3k+0	OM	–	–	28	**91**	3.2k	$20/6 \cdot 10^{10}$
Ring-30	**24**	**121**	8.1k+0	OM	–	–	385	133	119k	$30/6 \cdot 10^{15}$
Safe-30	**0.06**	**30**	30+0	0.2	**30**	465	0.15	**30**	30	30/30
Safe-70	**0.5**	**70**	70+0	2	**70**	2.5k	3	**70**	70	70/70
Safe-100	**1.0**	**100**	100+0	5	**100**	5k	12	**100**	100	100/100
Square-24	0.4	**69**	172+0	3	351	2.6k	**0.2**	70	70	$24/24^2$
Square-92	36	**273**	1413+0	1236	2444	36k	**15**	274	274	$92/92^2$
Square-120	OM	–	–	2376	2813	45k	**38**	**358**	358	$120/120^2$
Uts-k 30	**4.0**	**89**	92+0	8.5	101	13k	10	112	690	$30/2^{30}$
Uts-k 40	**13.4**	**119**	122+0	26.5	136	31k	46	143	1.1k	$40/2^{40}$
Uts-k 50	**33.8**	**149**	152+0	63.9	171	61k	OM	–	–	$50/2^{50}$
#Solved/#Total	**38/47**			**35/47**			**42/47**			

Table 1: Performance of T0, DNF and T1 over 47 conformant problems: T is total time, L is plan length, and E is number of expanded nodes (for T0, number of expanded by EHC + number expanded by GBFS). $|S_0^1|$ is number of initial state samples, and $|b_0|$ is total number of initial states. Numbers followed by k denote thousands. TO, OM, and U denote time out, out of memory, and reported unsolvable (incorrectly). Problems with width > 1 marked with asterisk. Highlighted entries show fastest executions and shortest plans. Last row shows overall coverage.

neither of the two heuristics used in T1 appears to be useful (this can be seen in the number of expanded nodes). T0 doesn't well either in this domain, as the heuristic used by FF over the K_1 translation is similar to one of the heuristics used in T1 (the classical heuristic h_C). DNF solves these instances with much less expansions, meaning that the heuristic it's using is the most informative for these domains. In most other domains, however, both T1 and T0 expand much less nodes than DNF, which nonetheless, expands nodes very fast. For example, in the second Bomb problem, DNF expands 175k nodes in 23 secs, while T0 and T1 expand 1.1k nodes in 3.5 secs, and 140 nodes in 21 secs, respectively. This means, that in this instance, DNF,

		h_C				h_K				T1			
Domain	I	S	T	E	L	S	T	E	L	S	T	E	L
Bomb	9	7	71	4k	101	7	11	773	101	**8**	2	100	101
Coins	9	**8**	1	888	78	**8**	8	7k	74	**8**	3	425	166
Comm	9	**9**	2	1k	176	**9**	21	30k	175	**9**	5	440	214
Square(Ctr)	10	4	18	5k	186	**10**	0.2	224	44	**10**	0.1	43	44
Square(Cor)	11	10	604	212k	659	**11**	51	81k	119	**11**	100	18k	661
Cube(Ctr)	12	6	84	32k	188	10	1	890	61	**12**	0.1	61	58
Cube(Cor)	11	8	92	219k	271	10	4	26k	88	**11**	12	15k	269
Dispose	11	7	664	8k	349	**9**	57	2k	190	8	134	1k	491
Logistics	4	2	0.2	546	30	2	544	1613k	30	**4**	0.1	554	78
Look-Grab	6	**6**	0.1	96	10	3	41	92k	7	**6**	0.1	20	11
Push-To	8	**6**	657	21k	247	**6**	238	12k	116	**6**	83	1k	237
Ring	7	6	1	1k	17	5	571	58k	17	**8**	0.2	214	31
Safe	5	**5**	0.05	40	10	1	5	10k	10	**5**	0.04	9	10
UTS-k	15	**15**	0.06	26	7	2	0.05	154	7	13	0.04	10	9
Coverage	127	99				93				119			

Table 2: Comparison of T1 with one heuristic (either h_C or h_K) and the two heuristics combined. I is number of instances, S is number of instances solved; T, E, and L are avg. time, avg. number of expanded nodes, and avg plan length, respectively Averages taken over instances solved by the three configurations.

T0, and T1 expand roughly 7k nodes per second, 300 nodes per second, and 70 nodes per second respectively. In general, T1 expands fewer nodes than T0, but not necessarily faster. Part of the explanation for the slow node expansion rate of T1 vs. T0 is what expansion means in each of the two planners. In T0, while FF is running in EHC mode, an expansion is the application of the helpful actions only. This search is incomplete, but as it can be seen in the table, it's often quite effective. In T1, on the other hand, an expansion is a full expansion: all the children nodes are generated, and if 'helpful' they are placed in the right queue. This, however, is a problem in instances with high branching factors where the number of nodes *generated* can be much larger than the number of nodes *expanded.* This is the reason that T1 runs out of memory in domains such as Bomb and Logistics, where it expands few nodes that lead however to the generation of hundred of thousands of nodes. This problem could avoided by delaying the generation and evaluation of such nodes. Such techniques are used in recent classical planners such as FD and LAMA.

Table 2 compares the performance of T1 when restricted to use just one heuristic, either the classical heuristic h_C or the certainty heuristic h_K, as opposed to the two heuristics combined. When running T1 with a single heuristic, the 'helpful' queue corresponding to the other heuristic is removed. As it can be seen, there are some domains where the classical heuristic is better than the certainty heuristic, and other domains, where the opposite is true. At the same time, the combination in T1 seems to get the best of both, in certain cases solving instances that cannot be solved with one of the heuristics alone. This doesn't mean, however, that the synergies between the two heuristics cannot be exploited further.

Conclusions

The translation-based approach to conformant planning is elegant and exhibits good performance in relation to approaches that explicitly search in belief space. The challenge in the latter is the belief representation and heuristic; the limitation in the former is that complete translations may

require exponential time and space, while incomplete translations may result in unsolvable problems. In this work, we have tried to combine the benefits of the two approaches: the flexibility of planners that search explicitly in belief space, with the heuristics and beliefs that arise from translations. For this, we have formulated a novel translation K_S^i that is always tractable and complete, and sound for problems with width bounded by i. The T1 planner uses the set of samples that results from the K_S^1 translation in the context of a belief search planner. The flexibility of the belief search planner is exploited in two manners: incorporating a second heuristic derived from 'oneof invariants' in the problem that is related to cardinality and landmark heuristics, and using a multi-queue best first search algorithm patterned after the classical planners FD and LAMA. The experimental results show that T1 is competitive with state-of-the-art conformant planners in number of problems solved and quality of solutions. Additional progress on scalability seems feasible by exploiting further the synergies between the two heuristics, and by dealing in a more efficient manner with the huge number of nodes that are generated in problems with large branching factors. The individual width of precondition and goal literals, as opposed to the width of the problem given by the max such width, could also be used to avoid some verifications and, in principle, to simplify the detection of duplicate nodes. Conformant problems by themselves are not too interesting, but as shown in recent years, they form the basis of state-of-the-art methods for planning with sensing, and for deriving finite-state controllers, and hence, their scalability is critical.

Acknowledgements

We thank the anonymous reviewers for useful comments. This work is partially supported by grants TIN2009-10232, MICINN, Spain, and FP7 / 2007-2013 under grant agreement no. 270019.

References

Baral, C., and Son, T. C. 1997. Approximate reasoning about actions in presence of sensing and incomplete information. In *Proc. of ILPS 1997*, 387–401. MIT Press.

Bertoli, P., and Cimatti, A. 2002. Improving heuristics for planning as search in belief space. In *Proc. AIPS'02*, 143–152. AAAI Press.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2006. Strong planning under partial observability. *Artificial Intelligence* 170(4-5):337–384.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS'00*, 52–61. AAAI Press.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Bonet, B.; Palacios, H.; and Geffner, H. 2009. Automatic derivation of memoryless policies and finite-state controllers using classical planners. In *Proc. of ICAPS-09*, 34–41. AAAI Press.

Brafman, R., and Hoffmann, J. 2004. Conformant planning via heuristic forward search: A new approach. In *Proc. of ICAPS-04*, 355–364. AAAI Press.

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

Darwiche, A. 2002. On the tractable counting of theory models and its applications to belief revision and truth maintenance. *JANCL* 11(1–2):11–34.

Darwiche, A. 2004. New advances in compiling cnf into decomposable negation normal form. In *Proc. of ECAI-04*, 328–332.

Haslum, P., and Jonsson, P. 1999. Some results on the complexity of planning with incomplete information. In *Proc. ECP-99, Lect. Notes in AI Vol 1809*, 308–318. Springer.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence* 173(5-6):503–535.

Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In *Proc. of ICAPS-05*, 71–80. AAAI Press.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

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

Keyder, E., and Geffner, H. 2008. Heuristics for planning with action costs revisited. In *Proc. of ECAI-08*, 588–592.

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

Petrick, R., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. AIPS'02*, 212–221.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proc. AAAI*, 975–982.

To, S.; Pontelli, E.; and Son, T. 2009. A conformant planner with explicit disjunctive representation of belief states. In *Proc. of ICAPS-09*, 305–312.

To, S.; Son, T.; and Pontelli, E. 2010. A New Approach to Conformant Planning using CNF. *Proc. of ICAPS-10* 169–176.

Turner, H. 2002. Polynomial-length planning spans the polynomial hierarchy. In *JELIA '02: Proc. of the European Conference on Logics in AI*, 111–124. Springer-Verlag.