Compiling Contingent Planning into Classical Planning: New Translations and Results

Héctor Palacios
Universitat Pompeu Fabra
Barcelona, Spain
hector.palacios@upf.edu

Alexandre Albore
ONERA and INRA
Toulouse, France
alexandre.albore@onera.fr

Hector Geffner
ICREA and Universitat Pompeu Fabra
Barcelona, Spain
hector.geffner@upf.edu

Abstract

Recently, Brafman and Shani have introduced a mapping for transforming deterministic contingent planning problems into classical ones. Their translation is interesting and elegant, but exponential in the number of possible initial states. They use it for action selection in an *on-line* contingent planner by calling a classical planner on an *approximate translation*, where the set of possible initial states is replaced by a sample of few states.

In this work, we introduce two alternative translations of contingent into classical problems, that are both polynomial, and we test them by solving contingent problems *off-line* using classical planners. While the results are not at the level of the last generation of off-line contingent planners like CNF, DNF, or CLG, they are meaningful enough and are on par with other recent contingent planners such as Contingent-FF, POND, and MBP. Moreover, the limitations in performance are not always a result of the size of the translations, but more a consequence of the limitations of current classical planning algorithms. These new translations thus enlarge the scope of problems that can be effectively solved by classical planners, while at the same time they present classical planners with new challenges.

Introduction

In the last few years, it has been shown that translations that map planning problems with incomplete information into planning problems with complete information can be computational effective and useful. This approach includes the translation of conformant problems into classical problems that underlies the conformant planner T0 (Palacios and Geffner 2009), the translation of contingent problems into fully observable non-deterministic problems used in the off-line and on-line contingent planner CLG (Albore, Palacios, and Geffner 2009), and the translation of contingent problems into classical problems used by the on-line contingent planner MSPR (Brafman and Shani 2012b). The first two translations are based on considerations of *width*, and are polynomial in the problem size. The translations used by these planners are a special case of more general complete translations that in the worst case are exponential in the number of problem variables. The translation underlying MSPR, on the other hand, is in the worst case doubly exponential in the problem size, as is exponential in the number of possible initial states. Brafman and Shani use their translation heuristically, for selecting actions in an *on-line* contingent planner by calling a classical planner on the *approximate translation* that results when the set of possible initial states is replaced by a smaller set of 4–8 samples.

In this work, we build on Brafman's and Shani's ideas to introduce two alternative translations for mapping contingent planning problems to classical problems. Unlike Brafman's and Shani's translations, however, our translations are polynomial in the number of possible initial states. Furthermore, we do not test these translations heuristically for selecting actions but for *computing full solutions to contingent problems using classical planners.* We show that, while the results from our translations are not at the level of the last generation off-line contingent planners like CNF, DNF (To, Pontelli, and Son 2011) or CLG, they are meaningful enough and are on par with other recent contingent planners such as Contingent-FF (Hoffmann and Brafman 2005), POND (Bryce, Kambhampati, and Smith 2006), and MBP (Bertoli et al. 2006). Moreover, the limitations in performance are not always a result of the size of the translations, but of limitations of current classical planning algorithms. The new translations thus enlarge the scope of problems that can be effectively solved by classical planners while at the same they present classical planners with new challenges. Furthermore, we expect these ideas to be relevant to other types of planning problems as well. For example, Brafman and Shani, along with Zilberstein, have used the ideas underlying their contingent translation to provide an approach for solving Q-Dec-POMDPs using classical planners (Brafman, Shani, and Zilberstein 2013). Q-Dec-POMDPs are collaborative multiagent planning models that are similar to Dec-POMDPs (Bernstein, Zilberstein, and Immerman 2000), except that uncertainty is represented by sets of states rather than probability distributions. In principle, our translations could be used for the same purpose but with potentially better results.

The rest of the paper is organized as follows. We consider the model and representation of (deterministic) contingent planning problems, then present each of two new translations along with their formal properties, and finally, the experimental results.

Contingent Planning

We review the model and representation of contingent planning problems.

Model

The model underlying deterministic contingent planning can be characterized as a tuple $\mathcal{S} = \langle S, S_0, S_G, A, O, f, o\rangle$ where S is a finite set of states, $S_0 \subseteq S$ is the set of possible initial states, S_G is the set of goal states, A is a set of actions with $A(s)$ denoting the actions in A that are applicable in the state s, and O is a set of observation tokens. An action a applicable in a state s changes the state to $s' = f(a, s)$ and results in the observation token $o(s', a) \in O$. Executions are sequences of action-observation pairs $a_0, o_0, a_1, o_1, \ldots$ and beliefs represent the sets of states that are possible. The initial belief state is $b_0 = S_0$, and if b is the belief before the action a is applied, the belief right after a is $b_a = \{s' \mid s' = f(a, s) \text{ and } s \in b\}$, while $b_a^o = \{s \mid s \in b_a \text{ and } o = o(s, a)\}$ is the belief after getting then the observation token o.

An execution $a_0, o_0, a_1, o_1, \ldots$ is *possible* in the model $\mathcal{S}$ if, starting from the initial belief b_0, each action a_i is applicable in the belief b_i resulting from the execution up to a_i, i.e. $a_i \in A(s)$ for all $s \in b_i$, and b_{i+1} is non-empty, where $b_{i+1} = b_a^o$ for $b = b_i$, $a = a_i$, and $o = o_i$. A *belief policy* π is a function mapping belief states into actions, while a *tree policy* π is a function mapping executions into actions. The executions $a_0, o_0, a_1, o_1, \ldots$ induced by a belief or tree policy π are the possible executions in which a_i is the action dictated by the policy given the belief b_i and the execution up to a_i respectively. A policy solves the model if all such executions reach a *goal* belief state, i.e., a belief state $b \subseteq S_G$. *Off-line methods* focus on the computation of such policies; *on-line methods* focus on the computation of the action for the current belief or execution. The difference between belief and tree policies is that the former treat executions leading to the same belief state as equivalent. Thus, while belief policies are represented by graphs, tree policies are represented by trees (Geffner and Bonet 2013).

Representation

We assume that contingent models are represented in compact form through tuples $P = \langle F, I, A_F, A_N, G\rangle$ where F stands for a set of atoms, I is a set of clauses over F representing the initial situation, and G is a set (conjunction) of literals over F representing the goals. We assume that A_F represents a set of physical actions, and A_N a set of sensing actions. Both physical and sensing actions a have a precondition $Pre(a)$ that is a conjunction of literals. Physical actions a are characterized by a set of conditional effects $a : C \rightarrow E$ on the world, where C and E are sets (conjunction) of literals, while sensing actions $a(q)$ reveal the truth value of the atom q in F.

The planning problem $P = \langle F, I, A_F, A_N, G\rangle$ defines the state model $\mathcal{S}(P) = \langle S, S_0, S_G, A, O, f, o\rangle$, where S is the set of valuations over the atoms in F, S_0 and S_G are the sets of valuations that satisfy I and G respectively, $A(s)$ is the set of actions in $A_F \cup A_N$ whose preconditions are true in s, $f(a, s)$ is the state-transition function determined by the conditional effects associated with physical actions $a \in A_F$, and $f(a, s) = s$ for sensing actions in $a \in A_N$. The set O contains the tokens $\top$ and $\bot$ such that $o(s, a)$ is $\top$ for actions a in A_F, and $o(s, a)$ is $\top$ or $\bot$ according to the truth value of atom q in s when a is the sensing action $a(q)$ in A_N.

The distinction between purely physical actions and purely sensing actions is a convenient simplification; the generalization to actions that involve both physical and sensing effects requires extra notation, but is not a computational challenge.

First Translation

The first translation of contingent problems P into classical problems $C_i(P)$ that we consider here, follows Brafman's and Shani's closely, but while they map each action a in P to $2^{|S_0|}$ actions $a(S')$, where S' is a subset of S_0, our translation is polynomial in $|S_0|$ and introduces a linear number of actions.[1]

For convenience, we will assume that the language of the classical translations $C_i(P)$ supports *axioms*. Axioms allow the definition of new, derived atoms in terms of primitive ones, called then the primitive fluents. The derived fluents can be used in action preconditions, goals, and in the body of conditional effects. Axioms are part of the classical PDDL standard and many classical planners support axioms. While it is possible to compile axioms away, there are also benefits for dealing with them directly in the computation of the heuristics and in the progression of the state (Thiébaux, Hoffmann, and Nebel 2005). In our implementation, we compile axioms away so that the translations can be fed into almost any existing classical planner supporting STRIPS, negation, and conditional effects.

The first translation $C_1(P)$ comprises fluents L/s for the literals L in P and the possible initial states $s \in b_I$ where $b_I = S_0$. The literals in P are those of the form p and $\neg p$ for $p \in F$. The literals L/s represent that L is true under the assumption that s is the true hidden initial state. The other primitive fluents in the translation $D(s, s')$ represent that the execution so far contains enough information to distinguish one hidden initial state s from another s' (Brafman and Shani 2012a). The expression C/s, when C is a conjunction of literals L, stands for the conjunction of the literals L/s.

Definition 1. *Let* $P = \langle F, I, A_F, A_N, G\rangle$ *be a* deterministic contingent problem. *The translation* $C_1(P)$ *of* P *is the* classical planning problem with axioms *such that* $C_1(P) = \langle F', I', A', G', X'\rangle$, *where*

[1] There is actually a paragraph in (Brafman and Shani 2012b) that suggests that while translations involving a number of actions linear in $|S_0|$ appears to be feasible, the authors didn't consider them because such actions would have "many conditional effects" and hence would be "challenging for current classical planners". This is a puzzling comment though, as an exponential number of actions is certainly much worse than a quadratic number of effects. For example, for a single action and $|S_0| = 30$, the contrast is between 1 billion actions with 20 conditional effects each vs. 30 actions with 900 effects.

- $F' = \{L/s \,:\, L \in P, s \in b_I\} \cup \{D(s,s') \,:\, s, s' \in b_I\}$,
- $I' = \{L/s \,:\, L \in P, s \in b_I, s \models L\}$,
- $G' = G$,
- $A' = \{a(s) \,:\, a \in A_F \cup A_N, s \in b_I\}$ *such that preconditions* L *in* $Pre(a)$ *are replaced by preconditions* XL/s *in* $Pre(a(s))$*, and effects*

  $C/s', \neg D(s,s') \rightarrow E/s'$

  for each $s' \in b_I$ *in place of the effect* $C \rightarrow E$ *for* $a \in A_F$,

  $\neg D(s,s'), \neg D(s,s''), p/s', \neg p/s'' \rightarrow D(s',s''), D(s'',s')$

  for each pair of states s', s'' *in* b_I *for* $a = a(p) \in A_N$,
- X' *is a set of axioms:*
  - *one for each* derived fluent XL/s *such that* L *is a literal precondition in* P *and* $s \in b_I$*, with definition* $\wedge_{s' \in b_I}[L/s' \vee D(s,s')]$,
  - *one for each literal* L *in* G*, with definition* $\wedge_{s \in b_I} L/s$.

In words, the primitive fluents in $C_1(P)$ represent the truth of the literals L in P conditioned on each possible hidden initial state s, and the (in)accessibility relation $D(s,s')$ among the "worlds" s and s'. Initially, these worlds are all accessible from each other and $D(s,s')$ is false for all such pairs. On the other hand, L/s is true initially if L is true in s. The goal G' of $C_1(P)$ is the same as the (conjunctive) goal G of P, and the truth of each goal literal L follows from the truth of the primitive fluent literals L/s in the translation via an axiom. For each action a in P, there is an action $a(s)$ in $C_1(P)$ for $s \in b_I$, with preconditions XL/s replacing the preconditions L of a in P. *The intuitive meaning of the derived literal* XL/s *is that* L *is known to be true in the execution that is associated with the hidden state* s. We don't use the notation KL/s from (Palacios and Geffner 2009), which means something different; namely, that L is true given s. Using the notation from modal logics, XL/s stands for $s \supset KL$, while KL/s stands for $K(s \supset L)$. Last, we write L/t rather than KL/t as in (Palacios and Geffner 2009), because when "tags" t represent the complete initial states s, it is not possible for KL/t and $K\neg L/t$ to be both false. Since if one is false, the other must be true, there is also no need for "cancellation axioms" (Palacios and Geffner 2009). The translation appears closest to the one in (Brafman and Shani 2012a), except for the use of action preconditions XL/s rather than L, which otherwise would not preserve completeness. Indeed, for an action $a(s)$ to be applicable in the classical problem $C_1(P)$, it is *not sufficient* for the literal L/s to be true for a precondition L of a in P; that is too weak. But it is *not necessary* for the derived literal L in $C_1(P)$ to be true either; that is too strong. Rather XL/s must be true, which according to its axiom will be true when L/s' is true for all the states s' that cannot be distinguished from s.

The number of actions in the translation is $O(A \cdot |b_I|)$, the number of fluents is $O(|b_I|^2)$, and the maximum number of conditional effects per (sensing) action is $O(|b_I|^2)$. In Brafman's and Shani's translation the numbers are $O(A \cdot 2^{|b_I|})$, $O(|b_I|^2)$, and $O(|b_I|)$ respectively. Thus, the number of conditional effects is reduced from quadratic to linear, but the number of actions explodes from linear to exponential.

The soundness and completeness of the translation $C_1(P)$ can be expressed in terms of the *tree policies* that solve P.

Theorem 1 (Completeness $C_1(P)$)**.** *Let* $n_0, \ldots, n_k$ *be a topological enumeration of the internal nodes in a policy tree that solves* P *where no node precedes its parent, and let* $\pi(n_i)$ *represent the action performed by the policy in the node* n_i *and* $D(n_i)$ *represent the hidden states in* b_I *that are compatible with the execution up to* n_i*. Then, any action sequence* $\pi'(n_0), \ldots, \pi'(n_k)$ *where* $\pi'(n_i) = a(s)$*,* $\pi(n_i) = a$*, and* $s \in D(n_i)$*, is a classical plan for* $C_1(P)$*.*

Theorem 2 (Soundness $C_1(P)$)**.** *Let* $b_0, \ldots, b_k$ *be a classical plan for* $C_1(P)$ *such that* $D_i(s,s')$ *represents the status of the* $D(s,s')$ *fluents when the action* b_i *is applied. Let* $n_0, \ldots, n_k$ *be a set of nodes such that node* n_i *is the parent of node* n_j *if A)* $i < j$*,* $b_i = a(s)$*,* $b_j = a'(s')$*,* $D_i(s,s')$ *is true, and B) condition A is not true for any* k*,* $i < k < j$*. In such a case, the edge from* n_i *to* n_j *is labeled* $\top$ *either if* a *is a physical action, or if* a *is a sensing action and* $D_j(s,s')$ *is true. Else the edge is labeled* $\bot$*. Then the policy* $\pi(n_i) = a$ *over the resulting labeled tree is a tree policy that solves* P*.*

There are a number of optimizations that can be accommodated in the translation. In particular, all the actions $a(s)$ for the same action a from P are equivalent in states s' where $D(s,s')$ holds, making the branching factor of the classical problem $C_1(P)$ unnecessarily large. A simple optimization in this case is to make just one of those actions applicable. This can be achieved by introducing a static ordering "$<$" among states, and by adding a derived fluent $first(s)$ as precondition of $a(s)$, such that $first(s)$ is defined by means of the axiom $\wedge_{s' < s} D(s',s)$. This means that s represents the first state in the group of all states that are indistinguishable from s. Notice that the predicate D is symmetric, and hence $D(s',s)$ is true iff $D(s,s')$ is true. This property can also be used to reduce the quadratic number of fluents and conditional effects by half.

Second Translation

The translation above converts the sequence of actions in *any* topological traversal of the policy tree into a classical plan. This however is not needed for the translation to be complete. For this, it is enough to account for the actions in *one specific traversal* of the policy tree; for example the unique depth-first search traversal where the child following a sensing actions $a(p)$ where p is true is considered first. In order to achieve this, we accommodate a *stack* in the translation where the states that predict p to be false are pushed, in order to be dealt with later. A parameter of this translation is the *stack size*. A stack size of k will suffice to obtain contingent plans with branches that accommodate up to k observations. The translation will be polynomial in this stack parameter, yet the resulting classical plans may be exponential in it. This however is not a characteristic of the translation but of the nature of full contingent plans represented by trees: their size is exponential in the maximum number of observations gathered along a branch of the tree. For this reason, the stack parameter is small in the benchmarks, as otherwise off-line contingent plans would not be able to solve them.

Of course, on-line contingent planners do not compute full solutions and do not have this limitation.

The second translation $C_2(P)$ preserves the fluents L/s from the first translation but removes the $D(s, s')$ fluents, that are quadratic in number and require a quadratic number of conditional effects. Instead, the new translation uses fluents $m(s)$ for keeping track of the set of possible initial states s that are possible given the execution so far. In addition, preconditions XL/s in the first translation are replaced by preconditions XL where the context given by the hidden state s is implicit. The stack is represented by fluents $lev(l)$ to indicate the top of the stack, $stack(s, l)$ to indicate that the hidden state s has been pushed onto the stack at level l, and static fluents $next(l, l + 1)$ that represent that one level follows the other. For simplicity, we omit these static fluents which are true for each $l \in [0, M]$ where $M + 1$ is the max stack level. For convenience, we also assume that the goal G is a single atom; if it is not the case, problems can be brought into this form in the standard way by adding a dummy goal and a final action.

Definition 2. *Let $P = \langle F, I, A_F, A_N, G\rangle$ be a* deterministic contingent problem*. Then the translation $C_2(P)$ of P for a given stack parameter $M > 0$ is the* classical planning problem with axioms $C_2(P) = \langle F', I', A', G', X'\rangle$ *where*

- $F' = \{L/s, m(s), lev(l), stack(s, l) : L \in P, s \in b_I, l \in [0, M]\}$
- $I' = \{L/s, m(s), lev(0) : L \in P, s \in b_I, s \models L\}$,
- $G' = G$,
- $A' = A_F \cup \{a(q, l), pop(l+1) : a(q) \in A_N, l \in [0, M]\}$ *such that*
  - ***Physical actions:*** *preconditions L of $a \in A_F$ replaced by XL and $\neg XG$; effects $a : C \to E$ replaced by $a : C/s, m(s) \to E/s$ for each $s \in b_I$,*
  - ***Sensing actions:*** *preconditions L of $a(q)$ in A_N become preconditions XL for $a(q, l)$ in addition to $lev(l)$, $\neg XG$, $\neg Xq$, $\neg X\neg q$; effects of $a(q, l)$ are $\neg lev(l)$, $lev(l + 1)$, and conditional effect $m(s), \neg q/s \to stack(s, l + 1), \neg m(s)$,*
  - ***Pop actions:*** *preconditions of $pop(l)$ are $lev(l)$ and XG; effects are $\neg lev(l)$, $lev(l - 1)$ and conditional effects $m(s) \to \neg m(s)$ and $stack(s, l) \to m(s), \neg stack(s, l)$ for each $s \in b_I$.*
- *X' is a set of axioms:*
  - *one for each* derived fluent *XL such that L is literal precondition or goal in P with definition $\wedge_{s\in b_I}[L/s \vee \neg m(s)]$,*
  - *one for each literal L in G with definition $\wedge_{s\in b_I} L/s$.*

The number of actions in the translation is $O(|A_F| + M \cdot |A_N|)$ where $M + 1$ is the stack size, the number of fluents is $O((|F| + M) \cdot |b_I|)$, while the maximum number of conditional effects per action is $O(|S| \cdot |F|)$. None of these numbers grows with the square number of states in b_I as in the translation $C_1(P)$, or exponentially in $|b_I|$ as in the translation of Brafman and Shani. The emulation of a stack in the translation is reminiscent of Domshlak's compilation scheme for mapping a class of fault tolerant planning problems into classical problems (Domshlak 2013).

In relation to $C_1(P)$, the new translation replaces the $D(s, s')$ literals encoding the accessibility relations among worlds by the $m(s)$ literals that represent the set of hidden initial states that are possible given the current execution. The new translation uses also a stack where the hidden states s that predict $\neg q$ after the execution of the sensing action $a(q, l)$ are stored. The translation makes also sure that q is not known either true or false by adding the literals $\neg Xq$ and $\neg X\neg q$ in the action precondition. Finally, the translation adds the $pop(l+1)$ actions: they are triggered when an execution reaches the goal in the form of the XG literal, so that the executions associated with hidden states that have been pushed onto the stack become current and can be extended to reach the goal as well.

The soundness and the completeness of the translation $C_2(P)$ can be expressed as follows. In a policy tree for P, we refer by the level $lev(n_i)$ of a node n_i to the number of "right turns" in the path from the root of the tree to the node n_i, where a right turn corresponds to the observation that an atom q is false following a sensing action $a(q)$.

Theorem 3 (Completeness $C_2(P)$)**.** *Let $n_0, \ldots, n_k$ be a DFS enumeration of* all *the nodes in a policy tree that solves P where no node precedes its parent and where children associated with positive observations come first. Let $\pi(n_i)$ represent the action performed by the policy in node n_i, let $D(n_i)$ represent the set of hidden states in b_I which are compatible with the execution up to n_i, and let $lev(n_i)$ be the level of the node n_i in the tree. If these levels are not greater than $M + 1$, then the action sequence $\pi'(n_0), \ldots, \pi'(n_{k-1})$ is a classical plan for $C_2(P)$, where $\pi'(n_i) = \pi(n_i)$ if $\pi(n_i)$ is a physical action, $\pi'(n_i) = a(q, l)$ if $\pi(n_i)$ is the sensing action $a(q)$ and $l = lev(n_i)$, and $\pi'(n_i) = pop(l)$ if n_i is leaf node of the tree and $l = lev(n_i)$.*

Theorem 4 (Soundness $C_2(P)$)**.** *Let $b_0, \ldots, b_k$ be a classical plan for $C_2(P)$ such that $m_i(s)$ represents the status of the $m(s)$ fluents when the action b_i is applied. Let $n_0, \ldots, n_k$ be a set of nodes such that node n_i is the parent of node n_j if $i < j$ and there is a state s such that both $m_i(s)$ and $m_j(s)$ are true and there is no k, $i < k < j$, such that $m_k(s)$ is true as well. In such a case, the edge from n_i to n_j is labeled $\top$ either if a is a physical action, or if a is a sensing action and $j = i+1$. Else the edge is labeled $\bot$. Then the policy $\pi(n_i)$ over the internal nodes n_i of the tree such that $\pi(n_i) = b_i$ if b_i is a physical action, and $\pi(n_i) = a(q)$ if b_i is a sensing action $a(q, l)$, represents a policy that solves P.*

Empirical evaluation

We will turn now to evaluate the translations empirically over benchmarks collected from the distributions of contingent-FF, Pond, and CLG, and DNFct. The experiments were run on a cluster of Linux boxes AMD Opteron Abu Dhabi 6378 processors at 2.4 GHz. Each experiment had a cutoff of 2h or 4GB of memory. For the translation C_2 we used an stack size $M = 6$.

Table 1 compares existing contingent planners with the classical planner Fast Downward LAMA 2011, that we will

	$C_1(P)$		$C_2(P)$		Pond		MBP		CFF		CLG		DNFct	
Problem	time	size	time	size	time	size	time	size	time	size	time	size	time	size
blocks 3	0.0	7	0.2	6	0.01	5	0.33	7	0.02	6	0.08	6	0.58	5
blocks 7	45.8	63	116.9	63	OM		TO		0.46	49	4.58	55	11.6	69
blocks 11	490	115	TO		OM		TO		TO		35.68	115/18	OM	
blocks 15	463.9	119	TO		OM		TO		TO		144.2	157/22	OM	
ebtcs 50	1805	100	388.7	100	6.0	99	TO		11.96	99	0.02	21	0.13	101
ebtcs 70	MT		2216	140	29.8	139	TO		69.66	139	24.79	209	1.49	276
ebtcs 90	MT		TO		TO		TO		255	179	69.99	269	3.19	356
ebtcs 150	MT		TO		TO		TO		MC		603.3	449	6.25	596
grid 3	92.7	26	153.7	119	105	148	TO		943	58	1180	111	1.97	313
grid 4	1809	45	TO		OM		TO		TO		1558	884	2.9	982
grid 5	1260	70	134.5	97	OM		TO		TO		657	208	12.59	133
medpks 50	2495	101	6084	101	192.5	100	TO		164.9	100	2.32	101	1.45	201
medpks 70	MT		OM		TO		TO		968.6	140	7.51	141	1.49	141
medpks 90	MT		OM		TO		TO		TO		24.35	199	2.69	199
unix 1	0.1	19	0.2	17	5.1	26	11.58	21	0	7	0.01	21	0.01	17
unix 2	12.4	72	14.3	48	1.71	48	TO		0.13	48	0.35	50	0.78	48
unix 3	1521	197	TO		OM		TO		3.84	111	4.93	113	2.02	111
unix 4	MT		OM		OM		TO		143	238	78.9	240	16.26	238
doors 3	0.0	13	0.0	13	-	-	-	-	E		0	13	0.01	17
doors 5	429.7	164	TO		-	-	-	-	E		0.4	144	0.04	146
doors 7	MT		TO		-	-	-	-	E		13.4	2153	4.28	2193
localize 3	5.7	31	0.7	24	-	-	-	-	42	53	0.02	33	0.01	19
localize 5	301	81	8.0	76	-	-	-	-	MC		1.86	112	0.57	49
localize 7	5551	171	161.2	177	-	-	-	-	MC		6.89	231	0.72	80
localize 11	TO		TO		-	-	-	-	MC		63.72	577	1.2	144

Table 1: Comparison of Contingent planners. Translations solved by classical planner LAMA. TO stands for time out, OM for out of memory, MC and E that the problem is too big or a faulty execution respectively for CFF, MT that translator went memory out, '-' means no information on performance of planner over instance.

refer simply as LAMA (Richter and Westphal 2010; Helmert 2006), ran over the two translations. As it can be seen from the table, CLG and DNFct do best, but LAMA over $C_1(P)$ and $C_2(P)$ does not trail behind POND, Contingent-FF, and MBP in terms of coverage and quality.

Table 2 shows performance of three classical planners over the translations C_1 and C_2. The planners LAMA, FF (Hoffmann and Nebel 2001), and a version of the SIW planner that uses the h_{add} heuristic (Lipovetzky and Geffner 2012).

In almost all the instances, SIW is slower in generating the nodes than the other planners, but this is compensated by a better ratio between the number of expanded nodes and the length of plans, meaning that the search is informed enough to go to the goal while expanding few nodes. LAMA has the opposite behavior, generating nodes fast but expanding many more nodes per step in the plan. The relaxed heuristic of FF is not very informed except for the Medpks domain. LAMA uses a heuristic derived from landmarks in combination with the well-known FF heuristic; the initial values of these two heuristics are reported in 'heur' column.

There is a correlation between the size of the PDDL file and the overhead that affects the scalability of the classical planners. The size of the PDDLs is directly proportional to the number of fluents, actions and conditional effects in the translation. For creating the actual PDDLs, axioms are compiled away in ramifications and actions interleaved during the execution. These dummy actions represent many of the actions appearing in the resulting classical plans. Due to the treatment of axioms, the PDDLs are actually larger than the sizes that follow from the analysis in Sections 3 and 4. As a reference, the translation K_{s0} that maps conformant into classical planning problems (Palacios and Geffner 2009) and is linear in the number of initial states, dealing with a conformant problem equivalent to Blocks7, produces a classical problem with 445/1k/13k actions/atoms/conditional-effects. In contrast C_1 produces 3k/1k/177k and C_2 produces 777/1k/184k. K_{s0} and C_2 have a similar number of actions and atomos, while C_1 has more actions, as they grow linear with the number of initial states. In contrast, both C_1 and C_2 have many more conditional effects. Table 3 shows figures concerning the translation of selected instances, in comparison with the figures that correspond to the translation used by the CLG planner.

We have also tested our translations with many other classical planners and configurations that we are not reporting in the tables. We tried for example a version of the Fast Downward planner with the causal graph and the landmark heuristics. These are heuristics that are not as sensible to the approximations made by the delete-relaxation. We also tried other variants of the width-based algorithms in the SIW and BFS(f) family reported in (Lipovetzky and Geffner 2012). And finally, we tried the SAT-based planner Mp. In all cases, we didn't obtain better results, and for Mp the results were definitely inferior, reporting solutions to 8 of the 60 instances we tried. The coverage of the FD planner with the

C_1 **Translation**		SIW			LAMA				FF		
problem	pddl	size	#exp/#act	node/sec	size	#exp/#act	heur	node/sec	size	heur	node/sec
dispose 3 1	103	213	11.77	839	174	24.32	13/6	21018	OM	6	
dispose 4 1	522	576	33.35	40	375	1310	20/6	7688	OM	6	
doors 3	11.1	39	1.18	N/A	39	141.5	10/6	58417	39	6	N/A
doors 5	2115	930	22.74	16	492	67.63	35/8	1031	OM	8	
ebtcs 30	913.4	180	4.67	6	180	32.37	35/4	24459	OM	4	
ebtcs 50	4043	300	4.56	1	300	169.5	55/4	2338	OM	4	
elog 5	1081	450	25.97	3	330	2085	27/18	15812	OM	18	
elog 7	3621	TO			TO		39/23		OM	23	
grid 2	891	111	2.22	39	27	24081	40/9	1603	27	9	N/A
grid 4	7367	TO			135	212.01	106/18	24	111	20	49
localize 3	235	90	3.66	195	93	3383	11/7	8763	72	17	2597
localize 5	3431	558	35.68	2	243	90.17	22/6	948	OM	24	
medpks 30	1184	180	2.81	2	180	2.09	92/62	27755	180	63	342
medpks 50	5542	TO			303	307	152/102	17574	303	103	16
push 3 1	562.4	177	8.19	65	165	13.87	12/7	4926	OM	8	
unix 2	3169	453	35.62	4	216	1.45	15/6	1116	OM	6	
unix 3	81058	OM			591	3.26	31/6	35	OM		
C_2 **Translation**		SIW			LAMA				FF		
problem	pddl	size	#exp/#act	node/sec	size	#exp/#act	heur	node/sec	size	heur	node/sec
dispose 4 1	450	395	4.38	89	259	16	17/7	122	OM	6	
dispose 5 1	1303	667	9.11	25	411	4.72	26/7	18	OM	6	
doors 3	34.1	29	1.69		29	28.07	3/5	11450	29	5	8600
doors 5	850	355	3.24	43	TO		25/9		335	9	683
ebtcs 30	1317	243	2.61	28	177	4.2	31/5	83		4	
ebtcs 50	5405	363	6.42	5	297	4.4	51/5	27	OM	4	
elog 5	362	301	5.99	52	287	148.4	24/21	1758		19	
elog 7	553	481	7.23	42	401	554.5	36/21	778	OM	17	
grid 3	535	407	6.72	44	257	62.77	36/12	187	89	12	732
grid 5	1367	IW>2			97	163.15	136/16	39	OM	16	
localize 3	82.4	75	4.05	527	59	16469	8/5	2472	65	10	6564
localize 7	778	791	7.35	13	417	20.16	34/5	146	OM	22	
medpks 50	10609	299	5.87	1	299	1.49	101/53	21	299	53	108
medpks 99	86763	OM			TO				593	102	9
push 3 1	208	147	2.23	163	121	6.96	9/7	994		7	
push 3 2	4342	OM			1467	10.51	162/10	7	OM	9	
unix 2	718	137	8.39	36	117	4.21	12/9	147	OM	5	
unix 3	5918	OM			TO		28/9		OM	5	

Table 2: Performance of classical planners over the translations. 'N/A' indicates that the resolution time is too small to be reported. 'pddl' is the size in KB of the input files, '#exp/#act' the ratio between expanded nodes and actions in the plan, 'heur' the initial heuristic value, 'node/sec' generated nodes per second. TO stands for time out, OM for out of memory. IW>2 means SIW failed.

indicated heuristics was 26/30 for C_1/C_2, and for FD using the landmark heuristic was 15/25. FF coverage was 11/14. SIW, as presented by the authors, coverage was 16/25.

It is interesting to note that none of the reported classical planner dominated the others, and moreover, the planners exhibit very different behaviors over the translations, with some planners solving some of the instances rather easily while others produce memory or time outs in the same instances. Also, some encoding details are important. For example, we found that copying the preconditions into conditions[2] before the translation made a big difference in the performance of the FF planner that jumped from solving only 14 instances of C_2, to solving 27.

The size of the translations does not coincide with the sizes predicted by the analysis in Sections 3 and 4. The reason is the compilation of the axioms. We compiled the axioms away because many of the planners do not handle axioms properly. This, however, doesn't seem to be a good idea, because the elimination of the axioms produces many additional fluents and actions, that may be affecting performance drastically. One of the few modern planners that provides native support for axioms is FF-X (Thiébaux, Hoffmann, and Nebel 2005), but the overhead is then considerable as the class of axioms handled by FF-X is much more expressive than what is needed by our translations. FF-X supports indeed recursive, stratified definitions, while our definitions are simple and acyclic.

One open question is related with the low performance of the SAT-based planner Mp, which solves seven instances over the two translations C_1 and C_2. A translation to be ef-

[2]Such operation is sound and helps the delete-relaxation heuristics.

Problem	P	$C_1(P)$		$C_2(P)$		CLG	CLG-S_0
	Act / Atom / CE	Time	Act / Atom / CE	Time	Act / Atom / CE	Act / Atom / CE	Act / Atom / CE
blocks7	455 / 72 / 1k	1.9	3k / 1k / 177k	5.4	777 / 1k / 184k	6k / 4k / 263k	1k / 2k / 56k
blocks11	1k / 152 / 5k	6.4	12k / 2k / 570k	31.4	2k / 2k / 488k	15k / 10k / 736k	4k / 4k / 176k
dispose-2 2	62 / 54 / 96	1.4	322 / 1k / 19k	1.1	73 / 1k / 15k	171 / 596 / 2k	716 / 1k / 24k
dispose-2 3	71 / 60 / 110	125.7	1k / 8k / 1m	7.3	102 / 7k / 108k	239 / 716 / 3k	6k / 8k / 818k
doors 3	48 / 50 / 72	0	71 / 125 / 385	0.1	91 / 155 / 1k	139 / 497 / 1k	149 / 354 / 1k
doors 5	160 / 138 / 240	15.2	1k / 3k / 188k	5.7	303 / 2k / 73k	867 / 2k / 16k	2k / 3k / 115k
ebtcs-30	61 / 40 / 91	9.2	962 / 3k / 54k	3.4	218 / 3k / 189k	2k / 3k / 108k	2k / 4k / 197k
ebtcs-50	101 / 60 / 151	70.9	2k / 8k / 251k	16.4	358 / 8k / 823k	5k / 8k / 450k	7k / 11k / 845k
elog5	93 / 56 / 159	0.6	746 / 712 / 31k	0.5	235 / 865 / 27k	246 / 602 / 2k	672 / 1k / 12k
elog7	93 / 56 / 159	1.7	1k / 1k / 100k	0.8	235 / 1k / 44k	266 / 628 / 2k	992 / 1k / 23k
grid-4	232 / 134 / 436	26.7	6k / 6k / 649k	8.3	359 / 6k / 101k	2k / 3k / 49k	6k / 8k / 308k
grid-5	196 / 122 / 364	23.2	5k / 5k / 581k	7.2	311 / 5k / 90k	2k / 2k / 41k	5k / 7k / 278k
localize 11	9 / 90 / 633	229.7	686 / 15k / 7m	8.5	36 / 25k / 200k	12k / 13k / 1m	18k / 19k / 2m
localize 13	9 / 117 / 853	672.7	929 / 28k / 18m	19.5	36 / 45k / 350k	22k / 23k / 3m	32k / 35k / 7m
medpks-050	102 / 112 / 151	89.7	2k / 14k / 259k	46.5	358 / 21k / 1m	7k / 8k / 573k	13k / 16k / 1m
medpks-150	300 / 310 / 448	-	MT	2862	1k / 181k / 41m	67k / 70k / 13m	112k / 138k / 27m
push 3 1	59 / 52 / 109	0.4	533 / 472 / 20k	0.2	111 / 616 / 17k	266 / 683 / 5k	349 / 644 / 8k
push 3 2	94 / 62 / 170	-	MT	5.4	191 / 8k / 446k	502 / 1k / 9k	9k / 12k / 1m
unix 2	253 / 38 / 491	1.7	3k / 946 / 134k	0.7	335 / 1k / 62k	649 / 951 / 21k	795 / 1k / 30k
unix 3	1k / 70 / 2k	49	28k / 4k / 3m	9.4	1k / 5k / 608k	2k / 3k / 184k	3k / 4k / 280k

Table 3: Translation data for selected instances: Act, Atom, and CE stand for the number of actions, fluents, and conditional effects. Time is the translation time in seconds. P stands for the original problem, MT means that the translator went memory out.

fective should usually take into account the kind of problems where state-of-the-art planners perform the best. Which translation should work better for Mp? The translation C_2 is indeed smaller than C_1, but imposes an ordering on the traversals of the policy trees.

The implementations of the translations could be improved in a number of ways. One of the first to be explored is the value of compiling the axioms vs. dealing with them directly in the state progression and in the computation of the heuristics. Other details in the implementation could narrow the gap in performance with the best off-line contingent planners further. More powerful classical planning algorithms would help as well, indicating that those translated problems are a challenging test bench for classical planners. The huge differences in performance that results from the use of the different planners over the translations, suggest that there is also a lot of room for improvement, and for adjusting the details of the translations, to make the most of the techniques captured by these planners.

Summary

We have introduced two translations for mapping deterministic contingent problems into classical problems that, unlike Brafman's and Shani's translation, are polynomial and can be used for solving contingent problems off-line. From the empirical results obtained so far, the approach is not yet at the level of the last generation of off-line contingent planners, but it is certainly on par with other recent contingent planners such as Contingent-FF, POND, and MBP. Moreover, the limitations in performance are not always a result of the size of the translations, so further advances in classical planning algorithms will have an impact on the class of contingent problems that can be solved in this manner. A limitation of off-line contingent planners regardless of the approach is that the size of plans is often exponential in the number of observations, a limitation that is not shared by online planners. We expect these ideas to be relevant to other types of planning problems as well, like Q-Dec-POMDPs (Brafman, Shani, and Zilberstein 2013), and other types of partially observable multiagent planning problems.

References

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proc. Int. Joint Conf. of Artificial Intelligence (IJCAI-09)*, 1623–1628.

Bernstein, D.; Zilberstein, S.; and Immerman, N. 2000. The complexity of decentralized control of Markov decision processes. In *Proc. of the 16th Conf. on Uncertainty in Artificial Intelligence*, 32–37.

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

Brafman, R., and Shani, G. 2012a. Replanning in domains with partial information and sensing actions. *Journal of Artificial Intelligence Research* 45(1):565–600.

Brafman, R. I., and Shani, G. 2012b. A multi-path compilation approach to contingent planning. In *Proc. of AAAI*.

Brafman, R. I.; Shani, G.; and Zilberstein, S. 2013. Qualitative planning under partial observability in multi-agent domains. In *Proc. of AAAI*.

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

Domshlak, C. 2013. Fault tolerant planning: Complexity and compilation. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS 13)*.

Geffner, H., and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In *Proc. 15th Int. Conf. on Automated Planning and Scheduling (ICAPS 2005)*, 71–80.

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

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In *Proc. of European Conf. of Artificial Intelligence (ECAI 12)*, 540–545.

Palacios, H., and Geffner, H. 2009. Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width. *Journal of Artificial Intelligence Research* 35:623–675.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39(1):127–177.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In defense of PDDL axioms. *Artificial Intelligence* 168(1–2):38–69.

To, S. T.; Pontelli, E.; and Son, T. C. 2011. On the effectiveness of CNF and DNF representations in contingent planning. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI-11)*, 2033–2038.