Tractable Monotone Temporal Planning

Martin C. Cooper* **Frédéric Maris*** **Pierre Régnier***

IRIT, University of Toulouse
Toulouse, France
{cooper, maris, regnier}@irit.fr

Abstract

This paper describes a polynomially-solvable sub-problem of temporal planning. Polynomiality follows from two assumptions. Firstly, by supposing that each sub-goal fluent can be established by at most one action, we can quickly determine which actions are necessary in any plan. Secondly, the monotonicity of sub-goal fluents allows us to express planning as an instance of $STP^{\neq}$ (Simple Temporal Problem, difference constraints). Our class includes temporally-expressive problems, which we illustrate with an example of chemical process planning.

Introduction

Planning is a field of AI which is intractable in the general case (Erol, Nau, Subrahmanian, 1995). In particular, propositional planning is PSPACE-Complete (Bylander, 1994).

Nevertheless, a lot of work has been done on the computational complexity of non-optimal and optimal planning for classical benchmark domains. (Helmert, 2003, 2006), (Slaney, Thiébaux, 2001) proved that most of them can be solved by simple procedures running in low-order polynomial time. Moreover, the planners FF (Hoffmann, 2005) and eCPT (Vidal, Geffner, 2005) empirically proved that the number of benchmarks that can be solved without search may be even larger.

Since the work of (Bäckström, Klein, 1991) on the SAS formulation of planning, several studies have also been performed to define tractable classes of planning problems. Many of these tractability results are based on syntactic restrictions on the set of operators (Bylander, 1994), (Bäckström, Nebel, 1995), (Erol, Nau, Subrahmanian, 1995), (Jonsson, Bäckström, 1998) and another important body of work focused on the underlying structure of planning problems which can be highlighted using the causal graph (Knoblock, 1994). With restrictions on the causal graph structure, tractable classes can be exhibited (Jonsson, Bäckström, 1994, 1995, 1998), (Williams, Nayak, 1997), (Domshlak, Dinitz, 2001), (Brafman, Domshlak, 2003, 2006), (Helmert, 2003, 2006), (Jonsson, 2007), (Haslum 2007, 2008), (Giménez, Jonsson, 2008), (Katz, Domshlak, 2008). A unified framework to classify the complexity of planning under causal graph restrictions is given in (Chen, Giménez, 2008).

However, in real application domains, the assumptions of classical planning are too restrictive: a temporal planning framework must be used to formalize temporal relations between actions as temporal constraints. In the PDDL 2.1 temporal framework (McDermott, 1998), (Fox, Long, 2003), the PSPACE-complete complexity of classical planning can be preserved only when different instances of the same action cannot overlap. If they do overlap, testing the existence of a valid plan becomes an EXPSPACE-complete problem (Rintanen, 2007). In this paper we present a polynomially-solvable sub-problem of temporal planning. To our knowledge no previous work has specifically addressed this issue.

The article is organized as follows: Section 2 presents our temporal framework. Section 3 introduces the notion of monotonicity of fluents. Section 4 studies how to determine whether fluents are monotone. Section 5 gives an example of a temporal planning problem that can be solved in polynomial time: temporal chemical process planning. All solutions to this example require concurrent actions. Sections 6 and 7 conclude and give an outlook on future research.

* supported by ANR Project ANR-10-BLAN-0210.

Temporal Planning

We study temporal propositional planning in a language based on the temporal aspects of PDDL2.1. A *fluent* is a positive or negative atomic proposition. As in PDDL2.1, we consider that changes of the values of fluents are instantaneous but that conditions on the value of fluents may be imposed over an interval. An *action a* is a quadruple <Cond(*a*), Add(*a*), Del(*a*), Constr(*a*)>, where the set of conditions Cond(*a*) is the set of fluents which are required to be true for *a* to be executed, the set of additions Add(*a*) is the set of fluents which are established by *a*, the set of deletions Del(*a*) is the set of fluents which are destroyed by *a*, and the set of constraints Constr(*a*) is a set of constraints between the relative times of events which occur during the execution of *a*. An event corresponds to one of four possibilities: the establishment or destruction of a fluent by an action *a*, or the beginning or end of an interval over which a fluent is required by an action *a*. In PDDL2.1, events can only occur at the beginning or end of actions, but we relax this assumption so that events can occur at any time provided the constraints Constr(*a*) are satisfied.

We use the notation $a \rightarrow f$ to denote the event that action *a* establishes fluent *f*, $a \rightarrow \neg f$ to denote the event that *a* destroys *f*, and $f \mid\!\rightarrow a$ and $f \rightarrow\!\mid a$, respectively, to denote the beginning and end of the interval over which *a* requires the condition *f*. If *f* is already true (respectively, false) when the event $a \rightarrow f$ ($a \rightarrow \neg f$) occurs, we still consider that *a* establishes (destroys) *f*. A temporal plan may contain several instances of the same action, but since most of the temporal plans studied in this paper contain at most one instance of each action, for notational simplicity, we only make the distinction between actions and instances of actions if this is absolutely necessary. We use the notation $\tau(E)$ to represent the time in a plan at which an event E occurs.

For a given action *a*, let Events(*a*) represent the different events which constitute its definition, namely ($a \rightarrow f$) for all *f* in Add(*a*), ($a \rightarrow \neg f$) for all *f* in Del(*a*), ($f \mid\!\rightarrow a$) and ($f \rightarrow\!\mid a$) for all *f* in Cond(*a*). The definition of an action *a* includes constraints Constr(*a*) on the relative times of events in Events(*a*). As in PDDL2.1, we consider that the length of time between events in Events(*a*) is not necessarily fixed and that Constr(*a*) corresponds to interval constraints on pairs of events, such as $\tau(f \mid\!\rightarrow a) - \tau(f \rightarrow\!\mid a) \in [\alpha, \beta]$ for some constants α,β. We use $[\alpha_a(E_1,E_2), \beta_a(E_1,E_2)]$ to denote the interval of possible values for the relative distance between events E_1,E_2 in action *a*. A fixed length of time between events $E_1,E_2 \in$ Events(*a*) can, of course, be modelled by setting $\alpha_a(E_1,E_2) = \beta_a(E_1,E_2)$. We now introduce two basic constraints that all temporal plans must satisfy.

inherent constraints on the set of actions A: for all $a \in$ A, *a* satisfies Constr(*a*), i.e. for all pairs of events $E_1,E_2 \in$ Events(a), $\tau(E_1) - \tau(E_2) \in [\alpha_a(E_1,E_2), \beta_a(E_1,E_2)]$.

contradictory-effects constraints on the set of actions A: for all $a_i,a_j \in$ A, for all positive fluents $f \in$ Del(a_i) $\cap$ Add(a_j), $\tau(a_i \rightarrow \neg f) \neq \tau(a_j \rightarrow f)$.

Definition 1. A *temporal planning problem* <I,A,G> consists of a set of actions A, an initial state I and a goal G, where I and G are sets of fluents.

Notation: If A is a set of action-instances, then Events(A) is the union of the sets Events(*a*) (for all action-instances $a \in$ A).

Definition 2. P = <A,τ>, where A is a set of action-instances $\{a_1,...,a_n\}$ and τ is a real-valued function on Events(A), is a *temporal plan* for the problem <I,A′,G> if
(1) A ⊆ A′, and
(2) P satisfies the inherent and contradictory-effect constraints on A;

and when P is executed (i.e. fluents are established or destroyed at the times given by τ) starting from the initial state I:
(3) for all $a_i \in$ A, each $f \in$ Cond(a_i) is true when it is required, and
(4) all goal fluents $g \in$ G are true at the end of the execution of P.

We now look in more detail in the type of constraints that we impose on the relative times of events within an action.

Definition 3. An *interval constraint* C(x,y) on real-valued variables x,y is a binary constraint of the form x-y ∈ [a,b] where a,b are real constants.

Definition 4. (Jeavons and Cooper 1995) A binary constraint C(x,y) is *min-closed* if for all pairs of values (x_1,y_1), (x_2,y_2) which satisfy C, $(\min(x_1,x_2),\min(y_1,y_2))$ also satisfies C.

Lemma 1. Let A=$\{a_1,...,a_n\}$ be a set of actions and A′ a set of action-instances in which each action a_i (i=1,...,n) occurs $t_i \geq 1$ times. Let τ be a real-valued function on the set of events in A′. For each E ∈ Events(a_i), let E[j] (j=1,...,t_i) represent the occurrence of event E within the jth instance of a_i. For $i \in$ {1,...,n}, define the real-valued function τ_{min} on the set of events in the set of actions A by $\tau_{min}(E) =$

$\min\{\tau(E[j]) \mid j=1,...,t_i\}$. If τ satisfies the inherent constraints on A′, then τ_{min} satisfies the inherent constraints on A.

Proof: All interval constraints are min-closed (Jeavons and Cooper 1995). By applying the definition of min-closedness t_i-1 times, for each action a_i, we can deduce that if τ satisfies an interval constraint on each of the t_i instances of a_i, then τ_{min} satisfies this constraint on the action a_i. In other words, for all pairs of events E_1,E_2 in Events(a_i), if $\tau(E_1[j]) - \tau(E_2[j]) \in [\alpha_a(E_1,E_2), \beta_a(E_1,E_2)]$ for $j=1,...,t_i$, then $\tau_{min}(E_1) - \tau_{min}(E_2) \in [\alpha_a(E_1,E_2), \beta_a(E_1,E_2)]$. Hence if τ satisfies the inherent constraints on A′, then τ_{min} satisfies the inherent constraints on A.

Definition 5. A temporal planning problem <I,A,G> is *positive* if there are no negative fluents in the conditions of actions nor in the goal G.

In this paper, we will only consider positive temporal planning problems <I,A,G>. It is well known that any planning problem can be transformed into an equivalent positive problem in linear time by the introduction of a new fluent *notf* for each positive fluent *f* (Ghallab, Nau, Traverso 2004). By this assumption, G and Cond(*a*) (for any action *a*) are composed of positive fluents. By convention, Add(*a*) and Del(*a*) are also composed exclusively of positive fluents. The initial state I, however, may contain negative fluents.

We will need the following notion of *establisher-uniqueness* in order to define our tractable class of temporal planning problems. This is equivalent to post-uniqueness in SAS^+ planning (Jonsson, Bäckström, 1998) restricted to Boolean variables but generalised so that it applies to a specific subset of the positive fluents. In the next section, we apply it to the set of subgoals S, those fluents which are essential to the realisation of the goal.

Definition 6. A set of actions $A=\{a_1,...,a_n\}$ is *establisher-unique* relative to a set of positive fluents S if for all $i \neq j$, $Add(a_i) \cap Add(a_j) \cap S = \varnothing$, i.e. no fluent of S can be established by two distinct actions of A.

If a set of actions is establisher-unique relative to the set of subgoals of a problem, then we can determine in polynomial time the set of actions which are necessarily present in a temporal plan. There remains the problem of determining how many times each action must occur and then scheduling these action-instances in order to produce a valid temporal plan.

Monotone Planning

In this section, we introduce the notion of monotonicity of fluents. Together with establisher-uniqueness, the monotonicity of fluents is sufficient for a polynomial-time algorithm to exist for temporal planning.

Definition 7. A fluent *f* is *–monotone* (relative to a positive temporal planning problem <I,A,G>) if, after being destroyed *f* is never re-established in any temporal plan for <I,A,G>. A fluent *f* is *+monotone* (relative to <I,A,G>) if, after having been established *f* is never destroyed in any temporal plan for <I,A,G>. A fluent is *monotone* (relative to <I,A,G>) if it is either + or –monotone (relative to <I,A,G>).

Examples: In fairly obvious contexts, the following fluents are –monotone: alive (of a person), never-used, live (of a match), ready-to-eat (of a meal). Similarly, the following fluents are all +monotone: not-alive, burnt, dissolved, burst (of a balloon), eaten, cooked, graduated, born and extinct. The detection of the monotonicity of fluents is discussed in Section 4.

Notation: If A is a set of actions, we use the notation Del(A) to represent the union of the sets Del(*a*) (for all actions $a \in A$). Add(A), Cond(A), Constr(A) are defined similarly.

The following lemma follows trivially from Definition 7.

Lemma 2. If $f \notin Add(A) \cap Del(A)$, then *f* is both –monotone and +monotone relative to the positive temporal planning problem <I,A,G>.

We now introduce three other sets of constraints which will only be applied to monotone fluents:

–authorisation constraints on the positive fluent *f* and the set of actions A: for all $a_i,a_j \in A$, if $f \in Del(a_j) \cap Cond(a_i)$, then $\tau(f \rightarrow\mid a_i) < \tau(a_j \rightarrow \neg f)$.

+authorisation constraints on the positive fluent *f* and the set of actions A: for all $a_i,a_j \in A$, if $f \in Del(a_j) \cap Add(a_i) \cap (Cond(A) \cup G)$, then $\tau(a_j \rightarrow \neg f) < \tau(a_i \rightarrow f)$.

causality constraints on the positive fluent *f* and the set of actions A: for all $a_i,a_j \in A$, if $f \in (Cond(a_j) \cap Add(a_i))\backslash I$ then $\tau(a_i \rightarrow f) < \tau(f \mid\rightarrow a_j)$.

Definition 8. A temporal plan for a positive temporal planning problem <I,A,G> is *monotone* if each pair of actions (in A) satisfies the +authorisation constraints for all

+monotone fluents and the –authorisation constraints for all –monotone fluents.

The following lemma follows directly from the definition of monotonicity along with the fact that a fluent cannot be simultaneously established and destroyed in a temporal plan.

Lemma 3. Suppose that the positive fluent *f* is monotone relative to a positive temporal planning problem <I,A,G> and that actions $a_i,a_j \in$ A are such that $f \in$ Add(a_i) $\cap$ Del(a_j). If *f* is +monotone relative to this problem, then in all temporal plans including both a_i and a_j, $\tau(a_j \rightarrow \neg f) < \tau(a_i \rightarrow f)$. If *f* is –monotone relative to this problem, then in all temporal plans including both a_i and a_j, $\tau(a_i \rightarrow f) < \tau(a_j \rightarrow \neg f)$.

Proposition 1. If each fluent in Cond(A) is monotone relative to a positive temporal planning problem <I,A,G>, then all temporal plans for <I,A,G> are monotone.

Proof: Let P be a temporal plan. Consider firstly a positive –monotone fluent *f*. We have to show that the –authorisation constraints are satisfied for *f* in P i.e. that *f* is not destroyed before (or at the same time as) it is required in P. But this must be the case since *f* cannot be re-established once it is destroyed. Consider secondly a positive +monotone fluent *f*. By Lemma 3, the +authorisation constraint is satisfied for *f* in P.

Definition 9. Given a temporal planning problem <I,A,G>, the *set of sub-goals* is the minimum subset SG of Cond(A) $\cup$ G satisfying

1. G $\subseteq$ SG
2. for all $a \in$ A, if Add(a) $\cap$ (SG\I) $\neq \emptyset$, then Cond(a) $\subseteq$ SG.

The *reduced set of actions* is $A^r = \{a \in A \mid \text{Add}(a) \cap (SG\backslash I) \neq \emptyset\}$.

We can clearly determine SG and then A^r in polynomial time and the result is unique. If each fluent in Cond(A^r) $\cup$ G is monotone, we say that a plan P for the temporal planning problem <I,A,G> *satisfies the authorisation constraints* if each –monotone fluent satisfies the –authorisation constraints and each +monotone fluent satisfies the +authorisation constraints (it is assumed that we know, for each fluent $f \in$ Cond(A^r) $\cup$ G, whether *f* is + or –monotone.)

Theorem 1. Given a positive temporal planning problem <I,A,G>, where A is a set of actions such that Constr(A) are interval constraints, let SG and A^r be, respectively, the set of sub-goals and reduced set of actions. If the set of actions A^r is establisher-unique relative to SG, each fluent in Cond(A^r) $\cup$ G is monotone relative to <I,A^r,G> and each fluent in I $\cap$ (Cond(A^r) $\cup$ G) is –monotone relative to <I,A^r,G>, then <I,A,G> has a temporal plan P if and only if
(1) G $\subseteq$ (I\Del(A^r)) $\cup$ Add(A^r)
(2) Cond(A^r) $\subseteq$ I $\cup$ Add(A^r)
(3) all fluents $g \in$ G $\cap$ Del(A^r) $\cap$ Add(A^r) are +monotone relative to <I,A^r,G>
(4) the set of authorisation, inherent, contradictory-effects and causality constraints has a solution over the set of actions A^r.

Proof: ($\Rightarrow$) A^r is the set of actions which establish sub-goals f in SG\I. SG = Cond(A^r) $\cup$ G. Since A^r is establisher-unique relative to SG, each sub-goal $f \in$ SG\I has a unique action which establishes it. Hence each action in A^r must occur in the plan P. Furthermore, (Add(A)\Add(A^r)) $\cap$ (Cond(A^r)\I) = $\emptyset$ by Definition 9. It follows that (2) is a necessary condition for a temporal plan P to exist.

Let P′ be a version of P in which we only keep actions in A^r. P′ is also a valid temporal plan since no conditions of actions in P′ and no goals in G are established by any of the actions eliminated from P, except possibly if they are also in I. Such fluents $f \in$ I $\cap$ (Cond(A^r) $\cup$ G) are –monotone, by hypothesis, and hence cannot be established in P after being destroyed. It follows that any such *f* was already true when established in P by any action in A\ A^r.

(1) is clearly a necessary condition for P′ to be a valid temporal plan. Consider $g \in$ G $\cap$ Del(A^r) $\cap$ Add(A^r). From Lemma 3, we can deduce that *g* cannot be –monotone, since *g* is true at the end of the execution of P′. Thus (3) holds. Let P_{min}=<A^r,τ_{min}> be the version of the temporal plan P′=<A^r,τ> in which we only keep one instance of each action $a_i \in A^r$ (and no instances of the actions in A\A^r) and τ_{min} is defined from τ by taking the first instance of each event in Events(a_i), for each action $a_i \in A^r$, as described in the statement of Lemma 1. We will show that P_{min} satisfies the authorisation, inherent, contradictory-effects and causality constraints.

By Proposition 1, the temporal plan P′ must be monotone. Since P′ is monotone and by the definition of a temporal plan, the authorisation constraints are all satisfied. P′ must also, by definition of a temporal plan, satisfy the inherent and contradictory-effects constraints. It follows from Lemma 1 that P_{min} also satisfies the inherent constraints. Since the events in P_{min} are simply a subset of the events in P′, P_{min} necessarily satisfies both the authorisation constraints and the contradictory-effects constraints.

Consider a positive fluent $f \in$ (Cond(a_j) $\cap$ Add(a_i))\I, where a_i, $a_j \in A^r$. Since $a_j \in A^r$, we know that Add(a_j) $\cap$ (SG\I) $\neq \emptyset$ and hence that Cond(a_j) $\subseteq$ SG, by the definition of the set of sub-goals SG. Since $f \in$ Cond(a_j) we can deduce that $f \in$ SG. In fact, $f \in$ SG\I since we assume that

$f \notin$ I. It follows that if $f \in$ Add(a) for some $a \in$ A, then $a \in A^r$. But we know that A^r is establisher-unique (relative to SG). Hence, since $f \in$ Cond(a_j) $\subseteq$ Cond(A^r) and $f \in$ Add(a_i), f can be established by the single action $a=a_i$ in A. Since $f \notin$ I, the first establishment of f by an instance of a_i must occur in P′ before f is first required by any instance of a_j. It follows that the causality constraint must be satisfied by f in P_{min}.

($\Leftarrow$) Suppose that (1) and (2) are satisfied by A^r. Let P be a solution to the set of authorisation, inherent, contradictory-effects and causality constraints over A^r. A solution to these constraints uses each action in A^r (in fact, it uses each action exactly once since it assigns one start time to each action in A^r). Consider any $g \in$ G. By (1), $g \in$ (I\Del(A^r)) $\cup$ Add(A^r). If $g \notin$ Del(A^r), then g is necessarily true at the end of the execution of P. On the other hand, if $g \in$ Del(a_j) for some action $a_j \in A^r$, then by (1) there is necessarily some action $a_i \in A^r$ which establishes g. Then, by (3) g is +monotone. Since P satisfies the +authorisation constraint for g, a_i establishes g after all deletions of g. It follows that g is true at the end of the execution of P.

Consider some –monotone $f \in$ Cond(a_j) where $a_j \in A^r$. Since the –authorisation constraint is satisfied for f in P, f can only be deleted in P after it is required by a_j. Therefore, it only remains to show that f was either true in the initial state I or it was established some time before it is required by a_j. By (2), $f \in$ I $\cup$ Add(A^r), so we only need to consider the case in which $f \notin$ I but $f \in$ Add(a_i) for some action $a_i \in A^r$. Since P satisfies the causality constraint, $\tau(a_i \rightarrow f) < \tau(f \mid\rightarrow a_j)$ and hence, during the execution of P, f is true when it is required by action a_j.

Consider some $f \in$ Cond(a_j), where $a_j \in A^r$, which is not –monotone. By the assumptions of the theorem, f is necessarily +monotone and $f \notin$ I. First, consider the case $f \notin$ Del(A^r) $\cap$ Add(A^r). By Lemma 2, f is –monotone which contradicts our assumption. Therefore $f \in$ Del(a_k) $\cap$ Add(a_i), for some a_i, $a_k \in A^r$, and recall that $f \notin$ I. Since the +authorisation constraint is satisfied for f in P, any destruction of f occurs before f is established by a_i. It then follows from the causality constraint that the condition f will be true when required by a_j during the execution of P.

Theorem 2. Let Π^{U+M} be the class of positive temporal planning problems <I,A,G> in which A is establisher-unique relative to Cond(A) $\cup$ G, all fluents in Cond(A) $\cup$ G are monotone relative to <I,A,G> and all fluents in I $\cap$ (Cond(A) $\cup$ G) are –monotone relative to <I,A,G>. Then Π^{U+M} can be solved in time $O(n^3)$ and space $O(n^2)$, where n is the total number of events in the actions in A. Indeed, we can even find a temporal plan with the minimum number of action-instances in the same complexity.

Proof: This follows almost directly from Theorem 1 and the fact that the set of authorisation, inherent, contradictory-effects and causality constraints are $STP^{\neq}$ (Koubarakis, 1992). An instance of $STP^{\neq}$ can be solved in $O(n^3+k)$ time and $O(n^2+k)$ space (Gerevini, Cristani, 1997), where n is the number of variables and k the number of inequations (i.e. constraints of the form $x_j - x_i \neq d$). Here, the only inequations are the contradictory-effects constraints of which there are at most n^2, so $k=O(n^2)$. Furthermore, the calculation of SG and A^r is clearly $O(n^2)$.

Establisher-uniqueness tells us exactly which actions must belong to the temporal plan. Then, as shown in the proof of Theorem 1, the monotonicity assumptions imply that we only need one instance of each of these actions. It then trivially follows that we solve the optimal version of the temporal planning problem, in which the aim is to find a temporal plan with the minimum number of action-instances, by solving the set of authorisation, inherent, contradictory-effects and causality constraints.

Detecting Monotonicity of Fluents

A class Π of instances of an NP-hard problem is generally considered tractable if it satisfies two conditions: (1) there is a polynomial-time algorithm to solve Π, and (2) there is a polynomial-time algorithm to recognise Π. It is clearly polynomial-time to detect whether all actions are establisher-unique. On the other hand, our very general definition of monotonicity of fluents implies that this is not the case for determining whether fluents are monotone.

Theorem 3. Determining whether a fluent of a temporal planning problem <I,A,G> is monotone is PSPACE-hard if overlapping instances of the same action are not allowed in plans and EXPSPACE-complete if overlapping instances of the same action are allowed.

Proof: Notice that if <I,A,G> has no solution, then all fluents are trivially monotone by Definition 7, since they are neither established nor destroyed in any plans. It is sufficient to add two new goal fluents f_1,f_2 and two new instantaneous actions to A, a_1 which simply adds f_1 and a_2 which has f_1 as a condition, adds f_2 and deletes f_1 (a_1 and a_2 being independent of all other fluents) to any problem <I,A,G>: f_1 is monotone if and only if the resulting problem has no temporal plan. The theorem then follows from the fact that testing the existence of a temporal plan for a temporal planning problem <I,A,G> is PSPACE-hard if overlapping instances of the same action are not allowed in plans and EXPSPACE-complete if overlapping instances of the same action are allowed (Rintanen, 2007).

We can nevertheless detect the monotonicity of certain fluents in polynomial time. There are several ways in which we might demonstrate that a fluent is monotone. In this section we give some examples which give rise to low-order polynomial-time algorithms. Given Theorem 3, we clearly do not claim to be able to detect all monotone fluents with these rules. The set of temporal planning problems whose fluents can be proved +monotone or –monotone by the rules given in this section, as required by the conditions of Theorem 2, represents a tractable class, since it can be both recognised and solved in polynomial time.

Our first rule simply restates Lemma 2 from the previous section.

Rule 1: If there are no actions $a, a' \in A^r$ such that $f \in Del(a) \cap Add(a')$ then f is both +monotone and –monotone.

It may also be possible to show that a fluent is monotone because of its links with other fluents which are already known to be monotone. Consider a simple example of a planning problem with the two fluents *item_in_shop*, *item_for_sale* and only two actions which have these fluents as conditions or effects: the action DISPLAY_ITEM which has a condition *item_in_shop* and establishes *item_for_sale*, and the action SELL_ITEM which has a condition *item_for_sale* and destroys both *item_for_sale* and *item_in_shop*. For simplicity, suppose that both actions are instantaneous. The fluent *item_in_shop* is –monotone since it can never be established. (It may or may not be present in the initial state.) We cannot apply Rule 1 to deduce the monotonicity of *item_for_sale*, since it can be both established and destroyed. However, since *item_in_shop* is –monotone, it is clear that DISPLAY_ITEM cannot be executed after SELL_ITEM. It then follows that *item_for_sale* is –monotone. We formalise this basic idea in the following two rules.

Rule 2: Suppose that the reduced set of actions A^r is establisher-unique relative to the set of sub-goals SG (as defined by Definition 9) and let a_f denote the unique action that establishes fluent $f \in SG$. If for all $a \in A^r$ such that $f \in Del(a)$,
either $\exists$ a –monotone fluent $p \in Del(a) \cap Cond(a_f)$ such that $\tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f) \leq \tau(p \rightarrow\mid a_f) - \tau(a_f \rightarrow f)$,
or $\exists$ a –monotone fluent $p \in Del(a) \cap Add(a_f)$ such that
$\tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f) \leq \tau(a_f \rightarrow p) - \tau(a_f \rightarrow f)$,
or $\exists$ a +monotone fluent $p \in Add(a) \cap Del(a_f)$ such that
$\tau(a \rightarrow p) - \tau(a \rightarrow \neg f) \leq \tau(a_f \rightarrow \neg p) - \tau(a_f \rightarrow f)$,
or $\exists$ a +monotone fluent $p \in (Cond(a) \cap Del(a_f))\backslash I$ such that
$\tau(p \rightarrow\mid a) - \tau(a \rightarrow \neg f) \leq \tau(a_f \rightarrow \neg p) - \tau(a_f \rightarrow f)$,
then f is –monotone.

Rule 3: Suppose that set of actions A^r is establisher-unique relative to the set of sub-goals SG and let a_f denote the unique action that establishes fluent $f \in SG$. If for all $a \in A^r$ such that $f \in Del(a)$,
either $\exists$ a –monotone fluent $p \in Cond(a) \cap Del(a_f)$ such that $\tau(a_f \rightarrow \neg p) - \tau(a_f \rightarrow f) \leq \tau(p \rightarrow\mid a) - \tau(a \rightarrow \neg f)$,
or $\exists$ a –monotone fluent $p \in Add(a) \cap Del(a_f)$ such that
$\tau(a_f \rightarrow \neg p) - \tau(a_f \rightarrow f) \leq \tau(a \rightarrow p) - \tau(a \rightarrow \neg f)$,
or $\exists$ a +monotone fluent $p \in Del(a) \cap Add(a_f)$ such that
$\tau(a_f \rightarrow p) - \tau(a_f \rightarrow f) \leq \tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f)$,
or $\exists$ a +monotone fluent $p \in (Del(a) \cap Cond(a_f))\backslash I$ such that
$\tau(p \rightarrow\mid a_f) - \tau(a_f \rightarrow f) \leq \tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f)$,
then f is +monotone.

Proposition 2. Rules 2 and 3 are valid.

Proof: We only give the proof of the validity of Rule 2, since the proof of Rule 3 is entirely similar. Suppose that the premises of Rule 2 hold. Consider an arbitrary fluent $f \in SG$ and let a_f denote the unique action that establishes fluent $f \in SG$. Suppose that $f \in Del(a)$ and there is a –monotone fluent $p \in Del(a) \cap Cond(a_f)$ such that $\tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f) \leq \tau(p \rightarrow\mid a_f) - \tau(a_f \rightarrow f)$. But, since p is –monotone, we know that $\tau(p \rightarrow\mid a_f) < \tau(a \rightarrow \neg p)$. It follows directly that $\tau(a_f \rightarrow f) < \tau(a \rightarrow \neg f)$. By a similar argument for the other three cases listed in Rule 2, we can deduce that for all actions $a \in A^r$ such that $f \in Del(a)$, $\tau(a_f \rightarrow f) < \tau(a \rightarrow \neg f)$. Since a_f is the unique action that establishes f, we can deduce that f can never be established after it is destroyed and hence is –monotone.

The following theorem now follows from the fact that Rules 1,2 and 3 can clearly be applied until convergence in polynomial time.

Theorem 4. Let Π be the class of positive temporal planning problems <I,A,G> in which A is establisher-unique relative to $Cond(A) \cup G$, all fluents in $Cond(A) \cup G$ are monotone and all fluents in $I \cap (Cond(A) \cup G)$ are –monotone, where monotonicity of all fluents can be detected by applying Rules 1, 2 and 3 until convergence. Then Π is tractable.

An Example of Chemical Process Planning

The Temporal Chemical Process domain involves different kinds of operations on chemicals that are performed in the industrial production of compounds. For example, there is an operator that can *activate* a source of raw material.

Then, this raw material can be *catalysed* in two ways to *synthesize* two different products. These products can be *mixed* and *reacted* using the raw material once again to produce the desired compound. This process is illustrated by the temporal plan given in the figure. We represent non-instantaneous actions by a rectangle. The duration of an action is given in square brackets after the name of the action. Conditions are written above an action, and effects below. The initial state and the goal of corresponding planning problem are:

`I={Available(water),Available(s),Available(c₁), Available(c₂)}`

`G={Reacted(p₁,p₂)}`

Given the temporal planning problem <I,A,G>, where A is the set of all actions from the Temporal Chemical Process domain, the set of sub-goals SG and the reduced set of actions A^r are:

`SG={Reacted(p₁,p₂), Reacting(s), Mixed(p₁,p₂), Available(water), Available(s), End-Catalyze(p₁), End-Catalyze(p₁), Synthesized(p₂), Synthesized(p₂), Available(c₁), Available(c₂), Catalyzing(p₁,c₁), Catalyzing(p₂,c₂)}`

A^r`={REACT(p₁,p₂,s), ACTIVATE(s), MIX(p₁,p₂), CATALYZE(p₁,s,c₁), SYNTHESIZE(p₁,c₁), CATALYZE(p₂,s,c₂), SYNTHESIZE(p₂,c₂)}`

For all $i \neq j$ such that $\{a_i, a_j\} \subset A^r$ we have $\text{Add}(a_i) \cap \text{Add}(a_j) \cap SG = \emptyset$. Hence, the set of actions A^r is establisher-unique relative to SG. We can immediately remark that fluent `Available(water)` is never added or destroyed, fluents `Reacted(p₁,p₂)`, `Mixed(p₁,p₂)`, `End-Catalyze(p₁)`, `Synthesized(p₁)`, `End-Catalyze(p₂)`, `Synthesized(p₂)` are only added, and fluents `Available(s)`, `Available(c₁)`, `Available(c₂)` are only destroyed. Thus, none of these fluents are in $\text{Add}(A^r) \cap \text{Del}(A^r)$, and by Rule 1, they are –monotone. Using Rule 2, we can then prove that `Reacting(s)` is –monotone. a_f = `ACTIVATE(s)` is the unique action that establishes fluent f = `Reacting(s)` $\in$ SG. a = `ACTIVATE(s)` is also the unique action that destroys f = `Reacting(s)` and for the –monotone fluent p = `Available(s)` $\in \text{Del}(a) \cap \text{Cond}(a_f)$, we have $\tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f) \leq \tau(p \rightarrow\mid a_f) - \tau(a_f \rightarrow f)$. Therefore, by Rule 2, `Reacting(s)` is –monotone. By a similar argument, again using Rule 2, we can prove that `Catalyzing(p₁,c₁)` and `Catalyzing(p₂,c₂)` are –monotone.

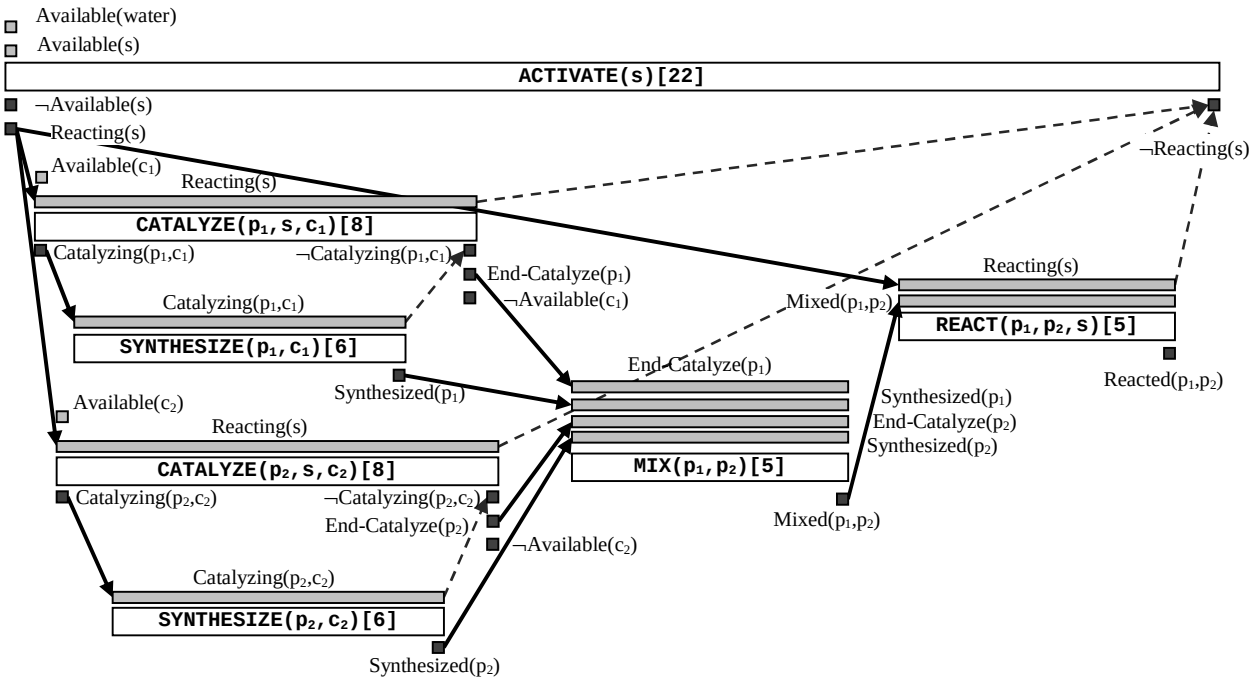
The set of actions A^r is establisher-unique relative to SG, each fluent in $\text{Cond}(A^r) \cup G$ is monotone and each fluent in $I \cap (\text{Cond}(A^r) \cup G)$ is –monotone relative to <I,A^r,G>. Moreover,

(1) $G \subseteq (I \backslash \text{Del}(A^r)) \cup \text{Add}(A^r)$

(2) $\text{Cond}(A^r) \subseteq I \cup \text{Add}(A^r)$

(3) all fluents $g \in G \cap \text{Del}(A^r) \cap \text{Add}(A^r)$ are +monotone relative to <I,A^r,G> (trivially, since this set is empty)

(4) There are no contradictory effects nor +authorisation constraints. The remaining set of constraints has a solution over the set of actions A^r.



Thus, by Theorem 1, the problem <I,A,G> has a solution-plan, shown in the figure (causality constraints are represented by bold arrows, and –authorisation constraints by dotted arrows), and, by Theorem 2, this solution can be found in polynomial time.

Many other temporal planning problems from the chemical industry can also be solved in polynomial time in a similar manner. For example, acetylene is a raw material derived from calcium carbide using water. Then, a vinyl chloride monomer is produced from acetylene and hydrogen chloride using mercuric chloride as a catalyst. PVC is then produced by polymerization. Other examples occur in the pharmaceutical industry in the production of drugs (such as paracetamol or ibuprofen) and, in general, in many processes requiring the production and combination of several molecules, given that there is a unique way to obtain them (often imposed by industrial, economical or ecological reasons).

Discussion

The results in this paper can also be applied to non-temporal planning since, for example, a classical STRIPS planning problem can be modelled as a temporal planning problem in which all actions are instantaneous. It is worth pointing out that the tractable class of classical planning problems in which all actions are establisher-unique and all fluents are detectable as (both + and –) monotone by applying only Rule 1, is covered by the PA tractable class of (Jonsson, Bäckström, 1998).

For simplicity of presentation and for conformity with PDDL2.1, we have considered that inherent constraints between the times of the events within the same action-instance are all interval constraints. We can, however, generalise our tractable classes to allow for arbitrary min-closed constraints since this was the only property required of the constraints in the proof of Theorem 1. An example of such a constraint C(x,y) is a binary interval constraint with variable bounds: $y\text{-}x \in [f(x,y),g(x,y)]$, which is min-closed provided that f(x,y) is a monotone increasing function of x and g(x,y) is a monotone decreasing function of y. Another example of a min-closed constraint is the ternary constraint $(x+y)/2 \leq z$, which could be used, for example, to impose that an effect takes place in the latter half of an action.

An important aspect of temporal planning, which is absent from non-temporal planning, is that certain temporal planning problems, known as temporally-expressive problems, require concurrency of actions in order to be solved (Cushing et al. 2007). A typical example of a temporally-expressive problem is cooking: several ingredients or dishes must be cooked simultaneously in order to be ready at the same moment. In industrial environments, concurrency of actions is often used to keep storage space and turn-around times within given limits. (Cooper et al. 2010) identified a subclass of temporally expressive problems, known as temporally-cyclic, which require cyclically-dependent sets of actions in order to be solved. A simple example of this type of problem is the construction of two pieces of software, written by two different subcontractors, each needing to know the specification of the other program in order to correctly build the interface between the two programs. The tractable class of temporal planning problems described in Theorem 4 contains both temporally-expressive and temporally-cyclic problems. This follows from that fact that, as illustrated by the example given in (Cooper et al. 2010) it is possible to construct an example of a temporally-cyclic problem which is establisher-unique and in which no fluents are destroyed by any action (and hence, by Lemma 2, all fluents are both + and –monotone). The chemical process planning problem given in Section 5 is another example of a problem which is temporally-expressive since concurrency of actions is required in any solution.

Conclusion

We have presented a class of temporal planning problems which can be solved in polynomial time. We have identified a number of possible applications in the chemical industry. Further research is required to discover other possible application areas and, on a theoretical level, to uncover other rules to prove the monotonicity of fluents.

References

Bäckström C., Klein I. (1991) Parallel non-binary planning in polynomial time. Proceedings IJCAI'1991, pp. 268-273.

Bäckström C., Nebel B. (1995) Complexity results for SAS+ planning). Computational Intelligence 11(4), pp. 625-655.

Brafman R.I., Domshlak C. (2003) Structure and Complexity in Planning with Unary Operators. Journal of Artificial Intelligence Research 18, pp. 315-349.

Brafman R.I., Domshlak C. (2006) Factored Planning: How, When, and When Not". Proceedings of the 21st National Conference on Artificial Intelligence.

Bylander T. (1994) The Computational Complexity of Propositional STRIPS Planning. Artificial Intelligence, 69(1-2), pp.165-204.

Chen H., Giménez O. (2008) Causal Graphs and Structurally Restricted Planning. Proceedings of the 18th International Conference on Automated Planning and Scheduling, ICAPS'2008.

Cooper M.C., Maris F., Régnier P. (2010) Solving temporally cyclic planning problems, International Symposium on Temporal Representation and Reasoning (TIME), p. 113-120.

Cushing W., Kambhampati S., Mausam, Weld D.S. (2007) When is Temporal Planning Really Temporal? Proceedings of 20th International Joint Conference on Artificial Intelligence, IJCAI'2007, pp. 1852-1859.

McDermott D. (1998) PDDL, The Planning Domain Definition Language. Technical Report, http://cs-www.cs.yale.edu/ homes/dvm/.

Domshlak C., Dinitz Y. (2001) Multi-agent off-line coordination: Structure and complexity. Proceedings of 6th European Conference on Planning, ECP'2001.

Erol K., Nau D.S., Subrahmanian V.S. (1995) Complexity, decidability and undecidability results for domain-independent planning. Artificial Intelligence, 76(1-2), pp.75-88.

Fox M., Long D. (2003) PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains, Journal of Artificial Intelligence Research 20, pp. 61-124.

Gerevini A., Cristani M. (1997) On Finding a Solution in Temporal Constraint Satisfaction Problems. Proceedings of 15th International Joint Conference on Artificial Intelligence, IJCAI'1997, pp. 1460-1465.

Ghallab M., Nau D.S., Traverso P. (2004) Automated Planning: Theory and Practice, Morgan Kaufmann.

O. Giménez, A. Jonsson (2008) The complexity of planning problems with simple causal graphs. Journal of AI Research 31, pp. 319-351.

Haslum P. (2007) Reducing Accidental Complexity in Planning Problems. Proceedings of IJCAI'07, pp. 1898-1903.

Haslum P. (2008) A New Approach To Tractable Planning. Proceedings of ICAPS'2008.

Helmert M. (2003) Complexity results for standard benchmark domains in planning. Artificial Intelligence 143 (2), pp. 219-262.

Helmert M. (2006) New Complexity Results for Classical Planning Benchmarks. Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS'2006, pp. 52-61.

Hoffmann J. (2005) Where Ignoring Delete Lists Works, Local Search Topology in Planning Benchmarks. Journal of Artificial Intelligence Research 24, pp. 685-758.

Jeavons P., Cooper M.C. (1995) Tractable constraints on ordered domains, Artificial Intelligence 79, pp. 327-339.

Jonsson A. (2007) The Role of Macros in Tractable Planning Over Causal Graphs. Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI'2007, pp. 1936-1941.

Jonsson P., Bäckström C. (1994) Tractable planning with state variables by exploiting structural restrictions. Proceedings of AAAI'1994, pp. 998-1003.

Jonsson P., Bäckström C. (1995) Incremental Planning. In New Directions in AI Planning: 3rd European Workshop on Planning, EWSP'1995, pp. 79-90.

Jonsson P., Bäckström C. (1998) State-variable planning under structural restrictions: Algorithms and complexity. Artificial Intelligence, 100(1-2), pp. 125- 176.

Katz M., Domshlak C. (2008) New Islands of Tractability of Cost-Optimal Planning. Journal of Artificial Intelligence Research, 32, pp. 203-288.

Knoblock C.A. (1994) Automatically Generating Abstractions for Planning. Artificial Intelligence, 68(2), pp. 243-302.

Koubarakis M. (1992) Dense Time and Temporal Constraints with ≠. Proceedings of 3rd International Conference on Principles of Knowledge Representation and Reasoning, KR'1992, pp. 24-35.

Rintanen J. (2007) Complexity of concurrent temporal planning. Proceedings of the 17th International Conference on Automated Planning and Scheduling, ICAPS, pp. 280-287.

Slaney J., Thiébaux S. (2001) Blocks World revisited. Artificial Intelligence 125, pp. 119-153.

Vidal V., Geffner H. (2005) Solving Simple Planning Problems with More Inference and No Search. Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming, CP'05, p. 682-696.

Williams B.C., Nayak P. (1997) A reactive planner for a model-based executive. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, pp. 1178-1185.