Journal of Artificial Intelligence Research 50 (2014) 447-485 Submitted 2/14; published 6/14

Monotone Temporal Planning: Tractability, Extensions and Applications

Martin C. Cooper COOPER@IRIT.FR
Frédéric Maris MARIS@IRIT.FR
Pierre Régnier REGNIER@IRIT.FR
IRIT, Paul Sabatier University
118 Route de Narbonne
31062 Toulouse, France

Abstract

This paper describes a polynomially-solvable class of temporal planning problems. 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 with difference constraints). This class includes temporally-expressive problems requiring the concurrent execution of actions, with potential applications in the chemical, pharmaceutical and construction industries.

We also show that any (temporal) planning problem has a monotone relaxation which can lead to the polynomial-time detection of its unsolvability in certain cases. Indeed we show that our relaxation is orthogonal to relaxations based on the ignore-deletes approach used in classical planning since it preserves deletes and can also exploit temporal information.

1. 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). Identifying tractable classes of planning is important for at least two reasons. Firstly, real-world applications may fall into such classes. Secondly, relaxing an arbitrary instance I so that it falls in the tractable class can provide useful information concerning I in polynomial time.

Temporal planning is an important extension of classical planning in which actions are durative and may overlap. An important aspect of temporal planning is that, unlike classical planning, it permits us to model problems in which the execution of two or more actions in parallel is essential in order to solve the problem (Cushing, Kambhampati, Mausam & Weld, 2007). Although planning has been studied since the beginnings of research in Artificial Intelligence, temporal planning is a relatively new field of research. No tractable classes had specifically been defined in the temporal framework before the research described in this paper. We present a class of temporal planning problems that can be solved in polynomial time. In particular, we considerably extend the theoretical results given in conference papers (Cooper, Maris & Régnier, 2012, 2013b) by considering plans with optimal makespan, by relaxing the assumption that two instances of the same action do not overlap and by introducing the notion of unitary actions. We also give previously unpublished results of experimental trials on benchmark problems. But first, we review previous work in the identification of tractable classes of classical planning problems.

A lot of work has been done on the computational complexity of non-optimal and optimal planning for classical benchmark domains. In the non-optimal case, Helmert (2003, 2006) proved that most of these benchmarks can be solved by simple procedures running in low-order polynomial time. In the optimal case, finding an optimal plan for the famous blocksworld domain is NP-hard (Gupta & Nau, 1992) but Slaney and Thiébaux (2001) proved that this domain is tractable when searching for a non-optimal plan.

Moreover, some planners empirically showed that the number of benchmark problems that can be solved without search may be even larger than the number of tractable problems that have been identified theoretically. The FF planner (Hoffmann, 2005) demonstrated that domains with constant-bounded heuristic plateaus can theoretically be solved in polynomial time using the h+ heuristic. The eCPT planner (Vidal & Geffner, 2005) can solve, by use of inference, many instances of benchmark domains without having to backtrack.

Since the work of Bäckström and Klein (1991a) on the SAS formulation of planning, several studies have also been performed to define tractable classes of planning problems. Many of these results (Bylander, 1994; Bäckström & Nebel, 1995; Erol, Nau & Subrahmanian, 1995; Jonsson & Bäckström, 1998) are based on syntactic restrictions on the set of operators. For example, operators having a single effect, no two operators having the same effect, etc.

Another important body of work focused on the underlying structure of planning problems which can be highlighted using the causal graph, a directed graph that describes variable dependencies (Knoblock, 1994). Jonsson and Bäckström (1995) presented a class of planning problems with an acyclic causal graph and unary operators. In this "3S" class, variables are either Static, Symmetrically reversible, or Splitting; plan existence can be determined in polynomial time while plan generation is provably intractable. Giménez and Jonsson (2008) designed an algorithm that solves these problems in polynomial time while producing a compact macro plan in place of the explicit exponential solution. They also proved that the problem of plan existence for planning problems with multi-valued variables and chain causal graphs is NP-hard. Plan existence for planning problems with binary state variables and polytree causal graphs was also proven to be NP-complete.

Jonsson and Bäckström (1994, 1998) considered optimal and non-optimal plan generation and presented an exhaustive map of complexity results based on syntactic restrictions (using the SAS+ formulation of planning) together with restrictions on the causal graph structure (interference-safe, acyclic, prevail-order-preserving). They present a planning algorithm which is correct and runs in polynomial time under these restrictions. Williams and Nayak (1997) designed a polynomial-time algorithm for solving planning problems with acyclic causal graphs and reversible actions. Domshlak and Dinitz (2001) investigated connections between the structure of the causal graph and the complexity of the corresponding problems in the case of coordination problems for dependent agents with independent goals acting in the same environment. This general problem is shown to be intractable, but some significant subclasses are in NP and even polynomial.

Brafman and Domshlak (2003, 2006) studied the complexity of planning in the propositional STRIPS formalism under the restrictions of unary operators and acyclic graphs. They give a polynomial planning algorithm for domains whose causal graph induces a polytree of bounded indegree. However, they also demonstrated that for singly connected causal graphs the problem is NP-complete. Giménez and Jonsson (2012) gave a polynomial algorithm for the class P(k) of k-dependent planning problems with binary variables and polytree causal graphs for any fixed value of k. They also showed that if, in addition, the causal graph has bounded depth, plan generation is linear in the size of the input. Haslum (2008) defines planning problems in terms of graph grammars. This method reduces the original problem to that of graph parsing, which can be solved in polynomial time under certain restrictions on the grammar. Haslum thus explores novel classes of restrictions that are distinct from previously known tractable classes. Katz and Domshlak (2008) showed that planning problems whose causal graphs are inverted forks are tractable if the root variable has a domain of fixed size. Jonsson (2007, 2009) introduced the class IR of inverted tree reducible planning problems and gave an algorithm that uses macros to solve problems from this class. Its complexity depends on the size of the domain transition graph and it runs in polynomial time for several subclasses of IR. Chen and Giménez (2008) gave a unified framework to classify the complexity of planning under causal graph restrictions. They give a complete complexity classification of all sets of causal graphs for reversible planning problems. The graph property that determines tractability is the existence of a constant bound on the size of strongly connected components.

However, in real application domains, the sequential nature of classical plans is often too restrictive and a temporal plan is required consisting of a set of instances of durative actions which may overlap. Whereas classical planning consists in scheduling action-instances, temporal planning can be seen as scheduling the events (such as the establishment or destruction of fluents) of action-instances subject to temporal constraints capturing the internal structure of actions. A temporal planning framework must therefore be used to formalize temporal relations between events within the same or different actions-instances. 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. Polynomiality follows from the double assumption that each sub-goal fluent can be established by at most one action and also satisfies a monotonicity condition. This allows us to express temporal planning as an instance of the polynomial-time solvable problem $STP^{\neq}$ (Simple Temporal Problem with difference constraints). An $STP^{\neq}$ instance consists of a set of real-valued variables and a set of constraints of the three following forms $x-y < c$, $x-y \leq c$ or $x-y \neq c$, where x,y are any variables and c is any constant. Our tractable class includes temporally-expressive problems requiring the concurrent execution of actions, and has potential industrial applications. We also show how to derive, from an arbitrary (temporal) planning problem, a relaxed version belonging to this tractable class. This can lead to the polynomial-time detection of unsolvability in certain cases. It also provides a polynomial-time heuristic for detecting actions or fluents satisfying certain properties.

The article is organized as follows: Section 2 reviews existing temporal planners and their use of temporal constraints. Section 3 presents our temporal framework. Section 4 introduces the notion of monotonicity of fluents. Section 5 shows how the notion of monotonicity can be extended to monotonicity* in order to define a larger tractable class and presents the main theorem. Section 6 demonstrates how to build a tractable relaxation of any temporal planning problem (or classical planning problem) based on simple temporal problems. Section 7 shows how to determine whether fluents are monotone* using this relaxation and describes a tractable class of temporal planning problems. Section 8 describes experimental trials to validate and identify the limits of this temporal relaxation. Section 9 gives examples of temporal planning problems that can be solved in polynomial time, including a detailed example involving concrete mixing, delivery and use. It is worth noting that all solutions to the examples discussed in Section 9 require concurrent actions. Sections 10 and 11 conclude and discuss avenues of future research.

2. Temporal Constraint Solving in Temporal Planning

After the first temporal planner DEVISER (Vere, 1983), planners such as FORBIN (Dean, Firby & Miller, 1988) quite rapidly used an independent module, called Time-Map Manager (Dean & McDermott, 1987), to handle temporal constraints. The HTN (Hierarchical Tasks Network) planners IxTeT (Ghallab & Alaoui, 1989; Laborie & Ghallab, 1995), TRIPTIC (Rutten & Hertzberg, 1993) and TEST (Reichgelt & Shadbolt, 1990) kept this idea of an Independent module to manage temporal data.

Today's temporal planners are essentially based on one of three types of algorithms: plan-space search, state-space search and GRAPHPLAN (Blum & Furst, 1995).

The plan-space planners HTN and POP (Partially Ordered Planning) were the first to be extended to the temporal framework. In general, they use temporal intervals for the representation of actions and propositions, the causality relation between actions being replaced by a temporal order in partial plans. Conflict handling is then performed by a system inspired by Time-Map Manager. For example, the VHPOP planner (Younes & Simmons, 2003) uses a system of simple temporal constraints (STP: Simple Temporal Problem) (Dechter, Meiri & Pearl, 1991), whereas DT-POP (Schwartz & Pollack, 2004) is based on a system of disjunctive temporal constraints (DTP: Disjunctive Temporal Problem) (Stergiou & Koubarakis, 2000). The advantage of STPs is that they can be solved in polynomial time. DTPs cannot be solved in polynomial time, but allow the user to express temporal constraints such as "A appears before or after B", which lightens the work of the planner.

State-space search planners associate a start instant with each world state. Search can be based first on the instants when an event can occur: each decision of the form "*when* to perform an action" is then taken before all decisions of the form "*which* action is to be performed". This approach is called Decision Epoch Planning. Search can also be based first on finding which actions to use before scheduling these actions in time: all decisions of the form "*when* to perform an action" are taken only after all decisions of the form "*which* action is to be performed" have been taken. This approach is called Temporally Lifted Progression Planning.

GRAPHPLAN has also been extended to temporal domains through the use of solvers, in the planners LPGP (Long & Fox, 2003), TM-LPSAT (Shin & Davis, 2004) and TLP-GP (Maris & Régnier, 2008).

As we have seen, many temporal planners use the resolution of a system of temporal constraints. However, even when this system of constraints can be solved in polynomial time, as is the case for simple temporal constraints, the PSPACE complexity of classical planning remains. Indeed, certain planners even solve a system of disjunctive temporal constraints, which is known to be NP-hard. The tractable classes of classical planning (discussed in Section 1) have not been explicitly extended to temporal planning. In this paper we present what is to our knowledge the first tractable class of temporal planning problems. Its solution algorithm is based on solving a system of simple temporal constraints.

3. Definitions

We study temporal propositional planning in a language based on the temporal aspects of PDDL2.1 (Fox & Long, 2003). 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. Note that Add(a) ∩ Del(a) may be non-empty. Indeed, it is not unusual for a durative action to establish a fluent at the beginning of the action and destroy it at its end. We can also observe that the duration of an action, the time between the first and last events of the action, does not need to be explicitly stored.

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 action LOAD(m,c) shown in Figure 1 represents loading a batch of concrete c in a mixer m. We have Cond(LOAD(m,c)) = {Fluid(c), Empty(m), At-factory(m)}. We can see from the figure that the mixer must be empty at the start of the loading, whereas the concrete must be fluid and the mixer at the factory during the whole duration of the loading. We have Del(LOAD(m,c)) = {Empty(m)} and Add(LOAD(m,c) = {On(m,c)}. We can see from the figure that as soon as loading starts, the mixer is no longer empty and at the end of loading the mixer contains the concrete.

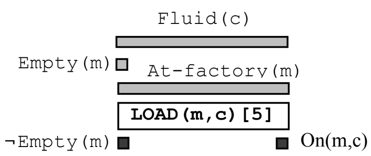


Figure 1: An example of the representation of a durative action

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 action-instances 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 (or action-instance) 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), ($f |\!\rightarrow a$) and ($f \rightarrow\!| \, a$) 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). For example, the internal structure of the fixed-length action LOAD(m,c) shown in Figure 1 is defined by constraints such as

$$\tau(\text{Fluid(c)} \rightarrow\!| \ \text{LOAD(m,c)}) - \tau(\text{Fluid(c)} \ |\!\rightarrow \text{LOAD(m,c)}) = 5$$

$$\tau(\text{LOAD(m,c)} \rightarrow \text{On(m,c)}) - \tau(\text{Fluid(c)} \rightarrow\!| \ \text{LOAD(m,c)}) = 0$$

As in PDDL2.1, we consider that the length of time between events in Events(a) is not necessarily fixed and that Constr(a) is a set of interval constraints on pairs of events, such as $\tau(f \rightarrow\!| \, a) - \tau(f |\!\rightarrow a) \in [\alpha, \beta]$ 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 $\alpha_a(e_1, e_2) = \beta_a(e_1, e_2)$. Similarly, the absence of any constraint can be modelled by the interval $[-\infty, +\infty]$. We now introduce two basic constraints that all temporal plans must satisfy.

inherent constraints on the set of action-instances 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 action-instances A: for all $a_i, a_j \in A$, for all positive fluents $f \in \text{Del}(a_i) \cap \text{Add}(a_j)$, $\tau(a_i \rightarrow \neg f) \neq \tau(a_j \rightarrow f)$.

The inherent constraints define the internal structure of each action-instance, whereas the contradictory-effects constraints ensure that the truth-value of each fluent never becomes undefined during the execution of a temporal plan. For example, if a plan contains an instance a of the action LOAD(m,c) shown in Figure 1 and an instance b of another action CLEAN(m) with Empty(m) $\in$ Add(CLEAN(m)), then the temporal plan must satisfy the contradictory-effects constraint

$$\tau(a \rightarrow \neg \text{EMPTY(m)}) \neq \tau(b \rightarrow \text{EMPTY(m)}).$$

Definition 3.1. 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 3.2. $P = <A,\tau>$, where A is a finite 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) $A \subseteq A'$, 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 $f \in \text{Cond}(a_i)$ is true when it is required, and

(4) all goal fluents $g \in G$ are true at the end of the execution of P.

(5) P is robust under infinitesimal shifts in the starting times of actions.

Events are instantaneous, whereas actions are not only durative but may also be of variable length. Thus a temporal plan P does not schedule its action-instances directly but schedules all the events in its action-instances.

Condition (5) in Definition 3.2 means that we disallow plans which require perfect synchronisation between different actions. Fox, Long and Halsey (2004) show how this condition can be imposed within PDDL2.1. We require that in all plans fluents are established *strictly* before the beginning of the interval over which they are required. The only exception to this rule is when a fluent f is established and required by the same action a. We allow the possibility of perfect synchronization within an action, which means that we can have $\tau(a \rightarrow f) = \tau(f|\!\rightarrow a)$. Similarly, fluents can only be destroyed *strictly* after the end of the interval over which they are required. The only exception to this rule is when a fluent f is required and destroyed by an action a, in which case we can have $\tau(f \rightarrow\!| a) = \tau(a \rightarrow \neg f)$. For example, the fluent Empty(m) is simultaneously required and destroyed by the action LOAD(m,c) shown in Figure 1.

Since a set of actions can be viewed as a set of action-instances in which each action occurs exactly once, we can apply constraints, such as the inherent and contradictory-effects constraints, to a set of actions rather then a set of action-instances. We now look in more detail at the type of constraints that we impose on the relative times of events within an action-instance.

Definition 3.3. An *interval constraint* $C(x,y)$ on real-valued variables x,y is a binary constraint of the form $x-y \in [a,b]$ where a,b are real constants.

Definition 3.4. (Jeavons & 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. A binary constraint $C(x,y)$ is *max-closed* if for all pairs of values (x_1,y_1), (x_2,y_2) which satisfy C, $(\max(x_1,x_2),\max(y_1,y_2))$ also satisfies C.

Lemma 3.5. Let $A = \{a_1,\ldots, a_n\}$ be a set of actions and A' a set of action-instances in which each action a_i $(i=1,\ldots, n)$ occurs $t_i \geq 1$ times. Let τ be a real-valued function on the set of events in A'. For each $e \in \text{Events}(a_i)$, let $e[j]$ $(j=1,\ldots,t_i)$ represent the occurrence of event e within instance number j of a_i. For $i \in \{1,\ldots,n\}$, define the real-valued functions $\tau_{\min},\tau_{\max}$ on the set of events in the set of actions A by $\tau_{\min}(e) = \min\{\tau(e[j]) \mid j=1,\ldots, t_i\}$ and $\tau_{\max}(e) = \max\{\tau(e[j]) \mid j=1,\ldots, t_i\}$. If τ satisfies the inherent constraints on A', then both $\tau_{\min}$ and $\tau_{\max}$ satisfy the inherent constraints on A.

Proof: All interval constraints are both min-closed and max-closed (Jeavons & Cooper, 1995). By applying the definition of min-closedness (respectively, max-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 $\tau_{\min}$ ($\tau_{\max}$) 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 $j=1,\ldots,t_i$, then $\tau_{\min}(e_1) - \tau_{\min}(e_2) \in [\alpha_a(e_1, e_2), \beta_a(e_1, e_2)]$ and $\tau_{\max}(e_1) - \tau_{\max}(e_2) \in [\alpha_a(e_1, e_2), \beta_a(e_1, e_2)]$. Hence if τ satisfies the inherent constraints on A', then $\tau_{\min}$ and $\tau_{\max}$ satisfy the inherent constraints on A. □

Definition 3.6. 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, for each positive fluent f, of a new fluent *notf* to replace occurrences of $\neg f$ in conditions of actions (Ghallab, Nau & Traverso, 2004). It is important to note, however, that this transformation may not conserve other properties of the instance. By the assumption that all problems are positive, 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.

For simplicity of presentation, we assume throughout this paper that the set of actions A has undergone the filtering operation consisting of eliminating those actions a from A which cannot possibly be executed since Cond(a) is not a subset of $I \cup$ Add(A).

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 specialised so that it applies to a specific subset of the positive fluents. In the next section, we apply it to the subset of positive fluents which may be required for the realisation of the goal.

Definition 3.7. A set of actions $A = \{a_1,...,a_n\}$ is *establisher-unique* (*EU*) relative to a set of positive fluents S if for all $i \neq j$, Add(a_i) $\cap$ Add(a_j) $\cap S = \emptyset$, *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 sub-goals 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. Establisher-uniqueness alone cannot prevent minimal plans from being of exponential size (Bäckström & Klein, 1991b).

4. Monotone Planning

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

Definition 4.1. 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 $<I,A,G>$).

Example 4.2: Consider the two actions shown in Figure 2: LIGHT-MATCH and LIGHT-CANDLE. The action LIGHT-MATCH requires that the match be live, in order to light it. The match remains lit until it is blown out at the end of the action. A constraint in Constr(LIGHT-MATCH) imposes that the duration of the action, i.e. τ(LIGHT-MATCH $\rightarrow$ $\neg$Match-lit) – τ(LIGHT-MATCH $\rightarrow$ Match-lit), is between 1 and 10 time units. The second action LIGHT-CANDLE requires that the match be lit during two time units for the candle to be lit. For an initial state I = {live, ¬Match-lit} and a set of goals G = {Candle-lit}, it is clear that all temporal plans for this problem involve executing the two actions in parallel with the start (respectively, end) of LIGHT-MATCH being strictly before (after) the start (end) of LIGHT-CANDLE. There is only one match available, which means that LIGHT-MATCH can be executed at most once. This means that the fluent Match-lit is −monotone since it cannot be established after being destroyed. This same fluent Match-lit is not +monotone since it is destroyed after being established.

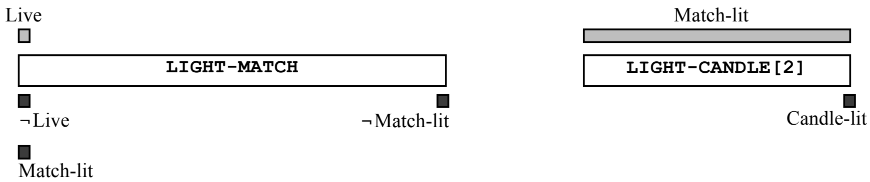


Figure 2: An example of a set of actions which allows us to light a candle using a single match.

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 4.1.

Lemma 4.3. If $f \notin$ Add(A) ∩ Del(A), then f is both −monotone and +monotone relative to the positive temporal planning problem $<I,A,G>$.

Certain physical actions or chemical reactions are irreversible. Examples include bursting a balloon, killing a fly, adding milk to a cup of coffee or burning fuel. Since there can be no action to destroy the corresponding fluents Burst, Fly-dead, Milk-added, Fuel-burnt, these fluents are necessarily both −monotone and +monotone by Lemma 4.3. A similar remark holds for fluents that may be true in the initial state but for which there is no action which establishes them, such as Fly-alive, for example. In Example 4.2, the fluent Live is both −monotone and +monotone since there is no action to establish it, and the fluent Candle-lit is −monotone and +monotone since there is no action to destroy it.

We now introduce three other sets of constraints, the −authorisation constraints being applied to –monotone fluents f and the +authorisation constraints to +monotone fluents. The causality constraints on fluent f are only valid if there is a unique action-instance which establishes f.

–authorisation constraints on the positive fluent f and the set of action-instances A: for all $a_i \neq a_j \in A$, if $f \in \mathrm{Del}(a_j) \cap \mathrm{Cond}(a_i)$, then $\tau(f \rightarrow\mid a_i) < \tau(a_j \rightarrow \neg f)$; for all $a_i \in A$, if $f \in \mathrm{Del}(a_i) \cap \mathrm{Cond}(a_i)$, then $\tau(f \rightarrow\mid a_i) \leq \tau(a_i \rightarrow \neg f)$.

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

causality constraints on the positive fluent f and the set of action-instances A: for all $a_i \neq a_j \in A$, if $f \in (\mathrm{Cond}(a_j) \cap \mathrm{Add}(a_i)) \setminus I$, then $\tau(a_i \rightarrow f) < \tau(f \mid\rightarrow a_j)$; for all $a_i \in A$, if $f \in (\mathrm{Cond}(a_i) \cap \mathrm{Add}(a_i)) \setminus I$ then $\tau(a_i \rightarrow f) \leq \tau(f \mid\rightarrow a_i)$.

Within the same action-instance a_i, perfect synchronisation is possible between the events $f \rightarrow\!| \; a_i$ and $a_i \rightarrow \neg f$. Indeed, one way of ensuring that an action a is executed at most once in any temporal plan is to create a fluent $f_a \in \text{Cond}(a) \cap \text{Del}(a) \cap I$ which is simultaneously required and deleted at the start of a and which is established by no action. For example, when a is the action LIGHT-MATCH in Example 4.2, f_a is the fluent live. On the other hand, by condition (5) of Definition 3.2 of a temporal plan, we cannot have perfect synchronisation between events in distinct action-instances. This explains why the −authorisation constraints impose the strict inequality $\tau(f \rightarrow\!| \; a_i) < \tau(a_j \rightarrow \neg f)$ when $a_i \neq a_j$ but only the non-strict inequality $\tau(f \rightarrow\!| \; a_i) \leq \tau(a_j \rightarrow \neg f)$ when $a_i = a_j$. A similar remark holds for the perfect synchronisation of the events $a_i \rightarrow f$ and $f \,|\!\rightarrow a_j$ which is only permitted by the causality constraints when $a_i = a_j$.

Definition 4.4. A temporal plan $<A,\tau>$ for a positive temporal planning problem $<I,A',G>$ is *monotone* if each pair of action-instances (in A) satisfies the +authorisation constraints for all +monotone fluents and satisfies the –authorisation constraints for all –monotone fluents.

Definition 4.5. Given a temporal planning problem $<I,A,G>$, the *set of sub-goals* is the minimum subset SG of $\text{Cond}(A) \cup G$ satisfying

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

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

We can determine SG and then A^r in polynomial time and the result is unique. To see this consider the simple algorithm which initialises SG to G and then repeatedly adds to SG the set of fluents F which is the union of $(\text{Cond}(a) \setminus SG)$ over all actions $a \in A$ such that $\text{Add}(a) \cap (SG \setminus I) \neq \emptyset$, until $F=\emptyset$. This simple algorithm has worst-case time complexity $O(n^3)$, where n is the total number of events in the actions of A, and produces a unique result which is clearly the minimum set of fluents satisfying the two conditions of Definition 4.5. Note that this algorithm is similar to the standard method of relevance detection used in GRAPHPLAN (Blum & Furst, 1995).

In order to state our theorem, we require a more relaxed definition of the set of sub-goals and the reduced set of actions to take into account the case in which fluents in the initial state are destroyed and re-established. Let SG^p (the set of possible sub-goals) denote the minimal set of fluents satisfying

1. $G \subseteq SG^p$
2. for all actions $a \in A$, if $\text{Add}(a) \cap SG^p \neq \emptyset$ then $\text{Cond}(a) \subseteq SG^p$.

Let A^p be the set of actions $\{ a \in A \mid \text{Add}(a) \cap SG^p \neq \emptyset \}$. The difference between A^r and A^p is that A^p is the set of actions which could occur in a minimal temporal plan in which fluents in the initial state can be destroyed and re-established. As with SG and A^r, SG^p and A^p are unique and can be determined in $O(n^3)$ time.

If each fluent in $\text{Cond}(A^r) \cup G$ 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 $f \in \text{Cond}(A^r) \cup G$, whether f is + or –monotone).

The following theorem contains minor improvements and corrections compared to the conference version of the present paper (Cooper, Maris & Régnier, 2012). Since it is a corollary of Theorem 5.6 (proved in the following section), we omit its proof.

Theorem 4.6. Given a positive temporal planning problem $<I,A,G>$, let SG and A^r be, respectively, the set of sub-goals and the reduced set of actions, with A^p defined as above. Suppose that Constr(A^r) are interval constraints, the set of actions A^r is establisher-unique relative to $SG \setminus I$, 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 $<I,A^p,G>$. Then $<I,A,G>$ has a temporal plan P if and only if

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

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

(3) all fluents $g \in G \cap \mathrm{Del}(A^r) \cap \mathrm{Add}(A^r)$ 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.

5. Extending Monotonicity of Fluents

In this section we introduce the notion of monotonicity*, thus allowing us to define a larger tractable class of temporal planning problems.

Definition 5.1. A plan is *minimal* if removing any non-empty subset of action-instances produces an invalid plan. 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 minimal 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 minimal temporal plan for $<I,A,G>$. A fluent is *monotone** (relative to $<I,A,G>$) if it is either + or –monotone* (relative to $<I,A,G>$).

Example 5.2. To give an example of a monotone* fluent which is not monotone, consider the following planning problem in which all actions are instantaneous:

Start_vehicle: k → o

Drive: o → d, ¬o

Unload: d → p

with $I = \{k\}$, $G = \{p\}$. The fluents represent that I have they ignition key (k), the engine is on (o), the destination has been reached (d) and that the package has been delivered (p). There is only one minimal plan, namely Start_vehicle, Drive, Unload, but there is also the non-minimal plan Start_vehicle, Drive, Start_vehicle, Unload in which the fluent o is established, destroyed and then re-established. Hence o is –monotone* but not –monotone.

A +monotone (–monotone) fluent is clearly +monotone* (–monotone*) since in any plan, including minimal plans, after having been established (destroyed) it is never destroyed (re-established). In order to prove the equivalent of Theorem 4.6 for monotone* fluents, we first require another definition and two minor technical results.

Definition 5.3. A minimal temporal plan $<A,\tau>$ for a positive temporal planning problem $<I,A',G>$ is *monotone** if each pair of action-instances in A satisfies the +authorisation constraints for all +monotone* fluents and satisfies the –authorisation constraints for all –monotone* fluents.

The following lemma follows directly from Definition 5.1 of the monotonicity* of a fluent along with the fact that a fluent cannot be simultaneously established and destroyed in a temporal plan.

Lemma 5.4. Suppose that the positive fluent f is monotone* relative to a positive temporal planning problem $<I,A',G>$. Let $<A,\tau>$ be a minimal temporal plan for $<I,A',G>$ with actions $a_i, a_j \in A$ such that $f \in \mathrm{Add}(a_i) \cap \mathrm{Del}(a_j)$. If f is +monotone* relative to this problem, then $\tau(a_j \rightarrow \neg f) < \tau(a_i \rightarrow f)$. If f is –monotone* relative to this problem, then $\tau(a_i \rightarrow f) < \tau(a_j \rightarrow \neg f)$.

Proposition 5.5. If each fluent in Cond(A) is monotone* relative to a positive temporal planning problem $<I,A,G>$, then all minimal temporal plans for $<I,A,G>$ are monotone*.

Proof: Let P be a minimal 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 P is a plan and f cannot be re-established once it is destroyed. Consider secondly a positive +monotone* fluent f. By Lemma 5.4, the +authorisation constraint is satisfied for f in P. □

We can now give our main theorem which generalizes Theorem 4.6 to monotone* fluents.

Theorem 5.6. Given a positive temporal planning problem $<I,A,G>$, let SG and A^r be, respectively, the set of sub-goals and the reduced set of actions. Suppose that all constraints in Constr(A^r) are interval constraints, the set of actions A^r is establisher-unique relative to $SG \setminus I$, each fluent in $\mathrm{Cond}(A^r) \cup G$ is monotone* relative to $<I,A^r,G>$ and each fluent in $I \cap (\mathrm{Cond}(A^r) \cup G)$ is –monotone* relative to $<I,A^p,G>$. Then $<I,A,G>$ has a temporal plan P if and only if

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

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

(3) all fluents $g \in G \cap \mathrm{Del}(A^r) \cap \mathrm{Add}(A^r)$ are +monotone* relative to $<I,A^r,G>$

(4) the set of authorisation, inherent, contradictory-effects and causality constraints (given in Sections 3 and 4) has a solution τ over the set of actions A^r (where the +authorisation constraints apply to each +monotone* fluent and the –authorisation constraints apply to each –monotone* fluent).

Proof: (⇒) If $<I,A,G>$ has a temporal plan, then it clearly has a minimal plan P. A^r is the set of those actions which establish sub-goals $f \in SG \setminus I$. By definition, $SG = \mathrm{Cond}(A^r) \cup G$. Since A^r is establisher-unique relative to $SG \setminus I$, each sub-goal $f \in SG \setminus I$ has a unique action which establishes it. Hence each action in A^r must occur in the plan P. Furthermore, $(\mathrm{Add}(A) \setminus \mathrm{Add}(A^r)) \cap (\mathrm{Cond}(A^r) \setminus I) = \varnothing$ by Definition 4.5. It follows that (2) is a necessary condition for a temporal plan P to exist.

Let P^p be a version of P in which we only keep actions in A^p. P^p is a valid temporal plan since, by definition of A^p, no fluent in $(\mathrm{Cond}(A^p) \cup G)$ can be established by actions in $A \setminus A^p$. Indeed, since P was assumed to be minimal, we must have $P^p = P$. Now let P' be a version of P in which we only keep actions in A^r. By Definition 4.5, 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. Thus to show that P' is also a valid temporal plan we only need to show that any establishment of a fluent $f \in I \cap (\mathrm{Cond}(A^r) \cup G)$ in P by an action $a \in A^p \setminus A^r$ is unnecessary. By hypothesis, f is −monotone* relative to $<I,A^p,G>$ and hence f cannot be established in P after having been destroyed. Since $f \in I$, this means that the establishment of f in P was unnecessary. Hence P' is a valid temporal plan. Indeed, since P was assumed to be minimal, we must have $P'=P$.

We have seen that P contains exactly the actions in A^r. Hence, all fluents $g \in G$ (which are necessarily positive by our hypothesis of a positive planning problem) that are either not present in the initial state I or are deleted by an action in A^r must be established by an action in A^r. It follows that (1) is a necessary condition for P to be a valid temporal plan. Consider $g \in G \cap \mathrm{Del}(A^r) \cap \mathrm{Add}(A^r)$. From Lemma 5.4, we can deduce that g cannot be −monotone*, since g is true at the end of the execution of P. Thus (3) holds. Let $P_{\min}=<A^r,\tau_{\min}>$ be the version of the temporal plan $P=<A^r,\tau>$ in which we only keep one instance of each action $a_i \in A^r$ (and no instances of the actions in $A \setminus A^r$) and $\tau_{\min}$ 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 3.5. We will show that $P_{\min}$ satisfies the authorisation, inherent, contradictory-effects and causality constraints.

We know that P is a temporal plan for the problem $<I,A,G>$. Hence it is also a temporal plan for the problem $<I,A^r,G>$, since it uses only actions from A^r. By hypothesis, all fluents in $\mathrm{Cond}(A^r)$ are monotone relative to $<I,A^r,G>$. Therefore, by Proposition 5.5, the temporal plan P is 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 3.5 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 $f \in (\mathrm{Cond}(a_j) \cap \mathrm{Add}(a_i)) \setminus I$, where $a_i, a_j \in A^r$. Since $a_j \in A^r$, we know that $\mathrm{Add}(a_j) \cap (SG \setminus I) \neq \varnothing$ and hence that $\mathrm{Cond}(a_j) \subseteq SG$, by the definition of the set of sub-goals SG. Since $f \in \mathrm{Cond}(a_j)$ we can deduce that $f \in SG$. In fact, $f \in SG \setminus I$ since we assume that $f \notin I$. It follows that if $f \in \mathrm{Add}(a)$ for some $a \in A$, then $a \in A^r$. But we know that A^r is establisher-unique (relative to SG). Hence, since $f \in \mathrm{Cond}(a_j) \subseteq \mathrm{Cond}(A^r)$ and $f \in \mathrm{Add}(a_i)$, 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 conditions (1)-(4) 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), $g \in (I \setminus \mathrm{Del}(A^r)) \cup \mathrm{Add}(A^r)$. If $g \notin \mathrm{Del}(A^r)$, then g is necessarily true at the end of the execution of P. On the other hand, if $g \in \mathrm{Del}(a_j)$ 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* $f \in \mathrm{Cond}(a_j)$ 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), $f \in I \cup \text{Add}(A^r)$, so we only need to consider the case in which $f \notin I$ but $f \in \text{Add}(a_i)$ 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 $f \in \text{Cond}(a_j)$, where $a_j \in A^r$, such that f is not –monotone*. By the assumptions of the theorem, f is necessarily +monotone* and $f \notin I$. First, consider the case $f \notin \text{Del}(A^r) \cap \text{Add}(A^r)$. By Lemma 4.3, f is –monotone (and hence –monotone*) which contradicts our assumption. Therefore $f \in \text{Del}(a_k) \cap \text{Add}(a_i)$, 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. □

The *makespan* of a temporal plan $P = \langle A,\tau \rangle$ is the time interval between the first and last events of P, i.e. $\max\{\tau(e) \mid e \in \text{Events}(A)\} - \min\{\tau(e) \mid e \in \text{Events}(A)\}$. The problem of finding a plan with minimum makespan is *polytime approximable* if there is a polynomial-time algorithm which, given a temporal planning problem $\langle I,A,G \rangle$ and any $\varepsilon > 0$, finds a temporal plan whose makespan is no more than $M_{\text{opt}} + \varepsilon$, where M_{opt} is the minimum makespan of all temporal plans for $\langle I,A,G \rangle$.

If the constraints in Constr(a) impose that the time interval between each pair of events in Events(a) is fixed, then we say that action a is *rigid*.

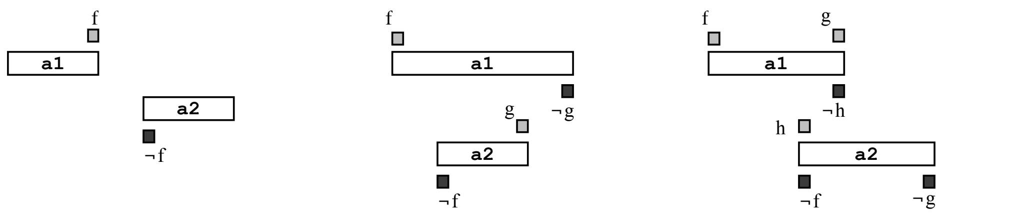
We express complexities in terms of the total number n of events in the actions in A. Without loss of generality, we assume that all actions in A contain at least one event and all fluents occur in at least one event. Hence the number of actions and fluents are both bounded above by n.

Theorem 5.7: Let $\Pi^{\text{EUM*}}$ be the class of positive temporal planning problems $\langle I,A,G \rangle$ in which A is establisher-unique relative to $\text{Cond}(A) \cup G$, all fluents in $\text{Cond}(A) \cup G$ are monotone* relative to $\langle I,A,G \rangle$ and all fluents in $I \cap (\text{Cond}(A) \cup G)$ are –monotone* relative to $\langle I,A,G \rangle$. Then $\Pi^{\text{EUM*}}$ 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 or of minimal cost, if each action has an associated non-negative cost, in the same complexity. Furthermore, if all actions in A are rigid then the problem of finding a plan with minimum makespan is polytime approximable.

Proof: The fact that $\Pi^{\text{EUM*}}$ can be solved in time $O(n^3)$ and space $O(n^2)$ follows almost directly from Theorem 5.6 and the fact that the set of authorisation, inherent, contradictory-effects and causality constraints form an $\text{STP}^{\neq}$, a simple temporal problem with difference constraints (Koubarakis, 1992). An instance of $\text{STP}^{\neq}$ 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 difference constraints (i.e. constraints of the form $x_j - x_i \neq d$). Here, the only difference constraints are the contradictory-effects constraints of which there are at most n^2, so $k=O(n^2)$. Furthermore, as pointed out in Section 4, the calculation of SG and A^r is $O(n^3)$.

Establisher-uniqueness tells us exactly which actions must belong to minimal temporal plans. Then, as we have seen in the proof of Theorem 5.6, 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 or of minimal cost, if each action has an associated cost, by solving the set of authorisation, inherent, contradictory-effects and causality constraints.

Now suppose that all actions in A are rigid. We will express the problem of minimising makespan while ignoring the contradictory-effects constraints as a linear program. We will then show that it is always possible to satisfy the contradictory-effects constraints (without violating the other constraints) by making arbitrarily small perturbations to the start times of actions. We assume that the events in a single action-instance satisfy the contradictory-effects and authorisation constraints; since actions are rigid this can be checked independently for each action in A in polynomial time. Let P be a temporal plan for $<I,A,G>$ which has minimum makespan. We showed in the proof of Theorem 5.6 that $P_{\min}$, obtained from P by keeping only one instance of each action in P, is also a valid temporal plan. Since makespan cannot be increased by eliminating action-instances from a temporal plan, $P_{\min}$ also minimises makespan. Let C be the set of inherent, authorisation and causality constraints that $P_{\min}$ must satisfy. C contains equality constraints of the form $x_j - x_i = d$, where d is a constant and x_j, x_i are times of events in the same action, and constraints of the form $x_j - x_i < d$, where d is a constant and x_j, x_i are times of events in different actions. We introduce two other variables τ_{begin}, τ_{end} and we denote by C_{opt} the set of constraints C together with the constraints $\tau_{\text{begin}} \le \tau(e)$ and $\tau(e) \le \tau_{\text{end}}$ for all $e \in \text{Events}(A^{\text{r}})$. The linear program which minimises $\tau_{\text{end}} - \tau_{\text{begin}}$ subject to the constraints C_{opt} minimises makespan but does not take into account the contradictory-effects constraints $C_{\neq}$ that must be satisfied by a valid temporal plan. Let P_{LP} be a solution to this linear program. Its makespan is clearly no greater than M_{opt}, the minimum makespan of all temporal plans (since valid temporal plans must satisfy the constraints $C \cup C_{\neq}$). Let δ be the minimum difference $d - (x_j - x_i)$ in P_{LP} over all constraints of the form $x_j - x_i < d$ in C. Let Δ be the minimum non-zero difference $|d - (x_j - x_i)|$ in P_{LP} over all contradictory-effects constraints $x_j - x_i \neq d$ in $C_{\neq}$. Suppose that $A^{\text{r}} = \{a_1,\dots,a_{\text{m}}\}$. Let P be identical to P_{LP} except that we add $\mu_i = \min\{\varepsilon,\delta,\Delta\}(i-1)/m$ to $\tau(e)$ for all $e \in \text{Events}(a_i)$. By construction, all contradictory-effects constraints which were violated in P_{LP} are satisfied in P, all contradictory-effects constraints which were satisfied in P_{LP} are still satisfied in P, and all strict inequalities which were satisfied in P_{LP} are still satisfied in P. The inequalities $\tau_{\text{begin}} \le \tau(e)$ are still satisfied in P. Finally, in order to guarantee satisfying the inequalities $\tau(e) \le \tau_{\text{end}}$, it suffices to add ε to τ_{end}. The resulting solution P corresponds to a valid temporal plan whose makespan is no more than $M_{\text{opt}} + \varepsilon$. The result then follows from the fact that linear programming is solvable in $O(n^{3.5}L)$ time by Karmarkar's interior-point algorithm, where n is the number of variables and L the number of bits required to encode the problem (Karmarkar, 1984). □



(a) (b) (c)

Figure 3: (a) all instances of action a_1 occur strictly before all instances of action a_2, (b) all instances of a_2 are contained in all instances of a_1 (c) all instances of a_1 overlap all instances of a_2.

In the class Π^{EUM*}, all sub-goal fluents $f \in SG$ are true over a single interval Int(f, P) during the execution of a temporal plan P. If f is +monotone*, then Int(f, P) is necessarily of the form $[t_1,\infty)$ where t_1 is the moment when f is first established. If f is –monotone*, then Int(f, P) is necessarily of the form $[t_1, t_2]$ where t_1 is again the moment when f is first established (or 0 if $f \in I$) and t_2 is the first moment after t_1 when f is destroyed (or $[t_1,\infty)$ if f is never destroyed). The class Π^{EUM*} is solvable in polynomial time due to the fact that establisher-uniqueness ensures that there is no choice concerning which actions to include in the plan and monotonicity* ensures that the only choice concerning the time of events is within an interval. Given these two restrictions it is quite surprising that a large range of industrial planning problems fall in this class (Cooper, Maris & Régnier, 2012, 2013b). EU monotone planning is a sufficiently powerful modelling language to allow us to impose constraints such as an action occurs at most once in a plan or that all instances of event e_1 occur before all instances of event e_2. To illustrate this, Figure 3 shows how we can impose precedence, containment or overlapping constraints between actions a_1 and a_2 by the introduction of, respectively, one, two or three fluents $f, g, h \in I$ which occur only in the events shown in Figure 3. By Lemma 4.3, these fluents f, g, h are all necessarily both –monotone and +monotone in all temporal plans.

6. Temporal Relaxation

Relaxation is ubiquitous in Artificial Intelligence. A *valid* relaxation of an instance I has a solution if I has a solution. Hence when the relaxation has no solution, this implies the unsolvability of the original instance I. A *tractable* relaxation can be built and solved in polynomial time.

The traditional relaxation of propositional non-temporal planning problems consisting of ignoring deletes has two drawbacks. Firstly, it is traditionally used with a forward search, which is not valid in temporal planning unless some specific transformation has been applied beforehand to the set of actions (Cooper, Maris & Régnier, 2013a). Secondly, it does not use information which may be essential for the detection of unsolvability of the original instance, namely the destruction of fluents and temporal information such as the relative duration of actions. In this section we present a valid tractable relaxation inspired by EU monotone temporal planning. In the following section we show how to use our temporal relaxation to detect monotonicity* of fluents. There are other possible applications, such as the detection of action landmarks (actions which occur in each solution plan) (Karpas & Domshlak, 2009), which immediately leads to a lower bound on the cost of a plan when each action has an associated cost (Cooper, de Roquemaurel & Régnier, 2011).

The traditional relaxation of classical non-temporal planning problems consists of ignoring deletes of actions. By finding the cost of an optimal relaxed plan, this relaxation can be used to calculate the admissible h+ heuristic. As shown by Betz and Helmert (2009), h+ is very informative but unfortunately NP-hard to compute (Bylander, 1994) and also hard to approximate (Betz & Helmert, 2009). As this relaxation does not use information which may be essential for the detection of un-solvability of the original instance (namely the destruction of fluents), a lot of research has been carried out to take some deletes into account (Fox & Long, 2001; Gerevini, Saetti & Serina, 2003; Helmert, 2004; Helmert & Geffner, 2008; Keyder & Geffner, 2008; Cai, Hoffmann & Helmert, 2009). Another recent approach (Haslum, Slaney & Thiébaux, 2012; Keyder, Hoffmann & Haslum, 2012) consists in enriching the classical relaxation with a set of fact conjunctions. Finally the red-black relaxation (Katz, Hoffmann & Domshlak, 2013a, 2013b) generalizes delete-relaxed planning by relaxing only a subset of the state variables.

Unfortunately, these relaxations do not directly generalize to temporal planning, since techniques based on a combination of ignoring deletes and forward search are not valid in temporal planning unless some specific transformation has been applied beforehand to the set of actions. 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, Kambhampati, Mausam & Weld, 2007). A typical example of a temporally-expressive problem is cooking: several ingredients must be cooked simultaneously in order to be ready at the same moment. In a previous paper (Cooper, Maris & Régnier, 2010), we identified a subclass of temporally expressive problems, known as temporally-cyclic, which require cyclically-dependent sets of actions in order to be solved.

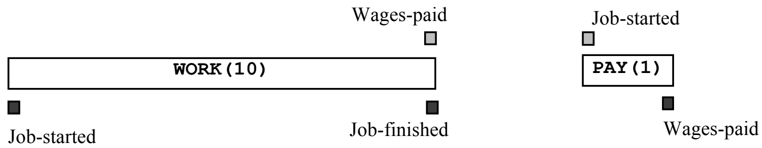


Figure 4: An example of a temporally cyclic temporal planning problem.

A simple temporally-cyclic problem is shown in Figure 4, where $I = \varnothing$ and $G = \{\text{Job-finished}\}$. A condition for a workman to start work is that he is paid (at the end of the job) whereas his employer will only pay him after he has started work. In a valid temporal plan for this problem the actions PAY and WORK must be executed in parallel with the execution of action PAY contained within the interval over which action WORK is executed. After applying the traditional ignore-deletes relaxation, forward chaining from the initial state would not be able to start either of the two actions since both of them have a missing condition. Thus, certain proposed techniques, although very useful in guiding heuristic search (Eyerich, Mattmüller & Röger, 2009; Do & Kambhampati, 2003), are not valid for temporally cyclic problems. Different solutions exist to get round the problem of temporal cycles. For example, we gave a polynomial-time algorithm to transform a temporally-cyclic problem into an equivalent acyclic one (Cooper, Maris & Régnier, 2013a). Other transformations have been proposed in the literature (Long & Fox, 2003; Coles, Fox, Long & Smith, 2008) which also eliminate the possibility of temporal cycles, although this was not an explicitly-stated aim in the descriptions of these transformations: temporal cycles are avoided by decomposing durative actions into instantaneous actions denoting the start and end of the action. Intermediate conditions can also be managed by splitting actions into component actions enclosed within an "envelope" action (Smith, 2003). In each case, ignoring deletes in the transformed problem followed by forward search provides a valid relaxation. If the original problem is not temporarily cyclic, then ignoring deletes followed by forward search is a valid relaxation.

In this section, we present an alternative form of relaxation, which we call TR (for Temporal Relaxation), inspired by EU monotone planning, comprising an $\text{STP}^{\neq}$ instance which has a solution only if the original temporal planning instance has a solution. It is incomparable with the relaxation based on ignoring deletes, as we will show through temporal and non-temporal examples, in the sense that there are instances that can be detected as unsolvable using EU monotone relaxation but not by ignoring deletes (and vice versa).

By applying the following simple rule until convergence we can transform (in polynomial time) any temporal planning problem $\mathcal{P}$ into a relaxed version $\mathcal{P}'$ which is EU relative to the set of sub-goals *SG*: if a sub-goal fluent *f* is established by two distinct actions, then delete *f* from the goal *G* and from Cond(a) for all actions a. As a consequence, *f* is no longer a sub-goal and SG has to be recalculated. Clearly, $\mathcal{P}'$ is a valid relaxation of $\mathcal{P}$. From now on we assume the temporal planning problem is EU relative to *SG*.

We denote by A^{LM} the set of action landmarks that have been detected (Karpas & Domshlak, 2009). Action landmarks are also known as indispensable actions (Cooper, de Roquemaurel & Régnier, 2011). Establisher-uniqueness implies that we can easily identify many such actions, in particular the set of actions A^{r} which establish sub-goals not present in the initial state *I*.

We cannot assume in the $\mathrm{STP}^{\neq}$ that a single instance of each action will be sufficient. For each action landmark a and for each event $e \in \mathrm{Events}(a)$, we introduce two variables $\tau_{\mathrm{first}}(e)$, $\tau_{\mathrm{last}}(e)$ representing the times of the first and last occurrences of event e in the plan. The constraints of our temporal relaxation TR include versions of the internal, contradictory-effects, authorization and causality constraints (which we give below) together with the following obvious constraint:

intrinsic TR-constraints: $\forall a \in A^{\mathrm{LM}}$, for all events $e \in \mathrm{Events}(a)$, $\tau_{\mathrm{first}}(e) \leq \tau_{\mathrm{last}}(e)$.

In the conference version of this paper in which we described a preliminary version of TR (Cooper, Maris & Régnier 2013b), we made the assumption that no two instances of the same action can overlap. Under this assumption, for $e_1, e_2 \in \mathrm{Events}(a)$, the first occurrences of e_1, e_2 in a plan correspond to the same instance of action a. A similar remark holds for the last occurrences of e_1, e_2. It turns out that we do not need to make this assumption in order to apply in TR each inherent constraint in Constr(a) independently to the values of $\tau_{\mathrm{first}}(e)$ and $\tau_{\mathrm{last}}(e)$ (for each $e \in \mathrm{Events}(a)$). Indeed, according to Lemma 3.5, assuming that Constr(A^{r}) are interval constraints, if each instance of action a satisfies its inherent constraints, then both τ_{first} and τ_{last} satisfy the inherent constraints on events in action a:

inherent TR-constraints: $\forall a \in A^{\mathrm{LM}}$, $\forall\ e_1,e_2 \in \mathrm{Events}(a)$, $\tau_{\mathrm{first}}(e_1) - \tau_{\mathrm{first}}(e_2) \in [\alpha_a(e_1,e_2), \beta_a(e_1,e_2)]$ and $\tau_{\mathrm{last}}(e_1) - \tau_{\mathrm{last}}(e_2) \in [\alpha_a(e_1,e_2), \beta_a(e_1,e_2)]$.

The contradictory-effects constraints in TR are as follows:

contradictory-effects TR-constraints: $\forall a_i, a_j \in A^{\mathrm{LM}}$, for all positive fluents $f \in \mathrm{Del}(a_i) \cap \mathrm{Add}(a_j)$, $\forall \mathrm{L1},\mathrm{L2} \in \{\mathrm{first},\mathrm{last}\}$, $\tau_{\mathrm{L1}}(a_i \to \neg f) \neq \tau_{\mathrm{L2}}(a_j \to f)$.

For each positive fluent *f* which is known to be −monotone*, we apply in TR the following modified version of the −authorisation constraints on *f*:

−authorisation TR-constraints: $\forall\ a_i \neq a_j \in A^{\mathrm{LM}}$, if $f \in \mathrm{Del}(a_j) \cap \mathrm{Cond}(a_i)$, then $\tau_{\mathrm{last}}(f \to\mid a_i) < \tau_{\mathrm{first}}(a_j \to \neg f)$; for all $a_i \in A^{\mathrm{LM}}$, if $f \in \mathrm{Del}(a_i) \cap \mathrm{Cond}(a_i)$, then $\tau_{\mathrm{last}}(f \to\mid a_i) \leq \tau_{\mathrm{first}}(a_i \to \neg f)$.

For each positive fluent *f* which is known to be +monotone*, we apply in TR the following modified version of the +authorisation constraints on *f*:

+authorisation TR-constraints: $\forall a_i, a_j \in A^{\mathrm{LM}}$, if $f \in \mathrm{Del}(a_j) \cap \mathrm{Add}(a_i)$, then $\tau_{\mathrm{last}}(a_j \to \neg f) < \tau_{\mathrm{first}}(a_i \to f)$.

We check that every condition and every goal can be established, i.e. $\mathrm{Cond}(A^{\mathrm{LM}}) \subseteq I \cup \mathrm{Add}(A)$ and $G \subseteq (I \setminus \mathrm{Del}(A^{\mathrm{LM}})) \cup \mathrm{Add}(A)$. If not, we consider that the relaxation TR has no solution.

We also apply in TR the following causality constraints for each positive fluent f:

causality TR-constraints: $\forall\ a_i \neq a_j \in A^{\mathrm{LM}}$, if $f \in (\mathrm{Cond}(a_j) \cap \mathrm{Add}(a_i)) \setminus I$ then $\tau_{\mathrm{first}}(a_i \rightarrow f) < \tau_{\mathrm{first}}(f| \rightarrow a_j)$; for all $a_i \in A^{\mathrm{LM}}$, if $f \in (\mathrm{Cond}(a_i) \cap \mathrm{Add}(a_i)) \setminus I$ then $\tau_{\mathrm{first}}(a_i \rightarrow f) \leq \tau_{\mathrm{first}}(f| \rightarrow a_i)$.

We also apply the following goal constraints for each $g \in G$:

goal TR-constraints: $\forall a_i, a_j \in A^{\mathrm{LM}}$, if $g \in \mathrm{Del}(a_j) \cap \mathrm{Add}(a_i)$, then $\tau_{\mathrm{last}}(a_j \rightarrow \neg g) < \tau_{\mathrm{last}}(a_i \rightarrow g)$.

Of course, these causality and goal constraints are necessary conditions for the existence of a plan only if A^{LM} is EU relative to $(\mathrm{Cond}(A^{\mathrm{LM}}) \setminus I) \cup (G \cap \mathrm{Del}(A))$.

Definition 6.1: An action $a \in A$ is *unitary* for a temporal planning problem $<I,A,G>$ if each minimal temporal plan for the $<I,A,G>$ contains at most one instance of a.

If an action $a \in A^{\mathrm{LM}}$ is known to be unitary, then in TR, for each event $e \in \mathrm{Events}(a)$, we replace the two variables $\tau_{\mathrm{first}}(e)$, $\tau_{\mathrm{last}}(e)$ in the above constraints by a unique variable $\tau(e)$.

Thus TR consists in first eliminating from G and Cond(a) (for all $a \in A$) all fluents established by two distinct actions, then, after checking that $\mathrm{Cond}(A^{\mathrm{LM}}) \subseteq I \cup \mathrm{Add}(A)$ and $G \subseteq (I \setminus \mathrm{Del}(A^{\mathrm{LM}})) \cup \mathrm{Add}(A)$, solving the $\mathrm{STP}^{\neq}$ consisting of the intrinsic, inherent, contradictory-effects, −authorisation, +authorisation, causality and goal TR-constraints, given above.

TR is a valid relaxation since the constraints of TR must clearly be satisfied by any temporal plan. Furthermore, if a plan exists then a minimal plan necessarily exists and in minimal plans there is at most one instance of each unitary action. Under assumptions of establisher-uniqueness and monotonicity*, TR is in fact a solution procedure for the tractable class described in Theorem 5.7.

The temporal relaxation TR can be significantly strengthened by the prior identification of unitary actions. We therefore present some lemmas which cover several simple but common cases in which actions can be identified as unitary. We first give a lemma which allows us to simplify certain actions.

Lemma 6.2: Suppose that $f \in I$ and that f is −monotone* in the positive temporal planning problem $<I,A,G>$. Let A' be identical to the set of actions A except that f has been deleted from Add(a) for each action a. Then all minimal temporal plans for $<I,A,G>$ are minimal temporal plans for $<I,A',G>$ and vice versa.

Proof: If P is a minimal temporal plan for $<I,A,G>$, then, since f is −monotone*, it cannot be established after having been destroyed in P. It follows that all establishments of f in P are unnecessary since they can only occur when $f \in I$ was already true. Hence, P is also a plan for $<I,A',G>$. It is also minimal for $<I,A',G>$ since all conditions and goals are identical in both problems. A minimal temporal plan for $<I,A',G>$ is necessarily a temporal plan for $<I,A,G>$, since all conditions and goals are positive, and is again necessarily minimal since all conditions and goals are identical in both problems. □

We assume in the rest of the paper that in TR the set of actions A has been simplified as indicated in Lemma 6.2.

Lemma 6.3: If f is –monotone*, $f \in$ Cond(a), $f \in$ Del(a) and a simultaneously requires and destroys f (i.e. Constr(a) contains the constraints $\tau(f \mid\rightarrow a) = \tau(f \rightarrow\mid a) = \tau(a \rightarrow \neg f)$), then a is unitary.

Proof: Let P be a minimal temporal plan containing a and consider the instance of a in P which first destroys f. By condition (5) of Definition 3.2 of a temporal plan, no two instances of a can be synchronised so that they destroy f simultaneously, so this instance is unique. By –monotonicity*, f cannot later be established in P. Hence no other instance of a can require (and destroy) f after this instant. It follows that the minimal temporal plan P can contain at most one instance of a. Hence a is unitary. □

Lemma 6.4: Let $<I,A,G>$ be a positive temporal planning problem and $a \in A$ an action such that a is rigid or no two instances of a can overlap in a temporal plan for $<I,A,G>$. In each of the three following cases a is unitary in $<I,A,G>$: (1) all fluents in Add(a) are monotone*, (2) Add(a) $\subseteq$ $G\backslash$Cond(A), or (3) Add(a) = $\{h\}$ for some fluent $h \notin G$, where there is a unique action b such that $h \in$ Cond(b), and furthermore b is unitary.

Proof: Let P be a minimal temporal plan for $<I,A,G>$ containing action a. First, consider case (1) in which all fluents in Add(a) are monotone*. Monotone* fluents never need to be established more than once in a minimal plan. This is because in minimal plans, once a +monotone* fluent has been established, it cannot be destroyed and, once a –monotone* fluent has been established, it can be destroyed but not established again. It follows that all but the first establishment of each fluent in Add(a) by a is unnecessary in P. Whether a is rigid or no two instances of a can overlap in P, all the first establishments of each of the fluents in Add(a) correspond to the same instance of a. All other instances of a can thus be deleted from P without destroying its validity. Hence a is unitary.

Now consider case (2), i.e. Add(a) $\subseteq$ $G\backslash$Cond(A). All but the last establishment of each fluent in Add(a) by a is unnecessary in P, since no fluents in Add(a) are conditions of actions in A. Whether a is rigid or no two instances of a can overlap in P, all the last establishments of each of the fluents in Add(a) correspond to the same instance of a. All other instances of a can thus be deleted from P without destroying its validity. Hence a is unitary.

Now consider case (3). Since b is unitary, the minimal plan P contains at most one instance of b. Only the instance of a which last establishes h before it is required by the unique instance of b can actually be necessary. All other instances of a can be deleted from P without destroying its validity. Hence a is unitary. □

It is often the case that no two instances of an action a can be executed in parallel, for example due to limited resources. It is therefore quite common when modelling a temporal planning problem to forbid that two instances of the same action a overlap. This can be achieved by introducing a fluent $f \notin$ Cond(A$\backslash\{a\}$) $\cup$ Add(A$\backslash\{a\}$) $\cup$ Del(A$\backslash\{a\}$) $\cup$ G, adding f to I and placing the events $f \mid\rightarrow a$, $f \rightarrow\mid a$ and $a \rightarrow \neg f$ at the beginning of a and the event $a \rightarrow f$ at the end of a. Alternatively, we can place the event $a \rightarrow f$ at the beginning of a and the events $f \mid\rightarrow a$, $f \rightarrow\mid a$ and $a \rightarrow \neg f$ at the end of a, in which case we do not need to have $f \in I$. In either case, we say that f is a *non-overlap* fluent for a. We can now state a more general version of Lemma 6.4.

Lemma 6.5: Let $<I,A,G>$ be a positive temporal planning problem and $a \in A$ an action such that f is a *non-overlap* fluent for a. In each of the three following cases a is unitary in $<I,A,G>$: (1) all fluents in $\mathrm{Add}(a) \setminus \{f\}$ are monotone*, (2) $\mathrm{Add}(a) \setminus \{f\} \subseteq G \setminus \mathrm{Cond}(A)$, or (3) $\mathrm{Add}(a) \setminus \{f\} = \{h\}$ for some fluent $h \notin G$, where there is a unique action b such that $h \in \mathrm{Cond}(b)$, and furthermore b is unitary.

Proof: Let P be a minimal temporal plan for $<I,A,G>$ containing action a. No two instances of a can overlap in P. As in the proof of Lemma 6.4, we only need to keep a single instance of a: in case (1) this is the first instance of a, in case (2) the last instance of a, and in case (3) the last instance of a before h is required by b. Hence a is unitary. □

The temporal relaxation TR uses two types of information not used by the ignore-deletes relaxation: the destruction of fluents and temporal information. We give two very simple examples to illustrate this.

Example 6.6: The simplest possible example showing that TR can detect the unsolvability of a planning problem that cannot be detected by the ignore-deletes relaxation consists of an initial state $I = \{f\}$, a goal $G = \{f, g\}$ and a single action which simultaneously establishes g and destroys f. Unsolvability is detected by TR since the condition $G \subseteq I \setminus \mathrm{Del}(A^{\mathrm{LM}}) \cup \mathrm{Add}(A)$ is not satisfied.

Example 6.7: Consider again the problem of lighting a candle using a single match described in Example 4.2. Suppose now that the match is very short and will only burn for at most two time units. The problem is clearly establisher-unique. Furthermore, both actions belong to A^{r} and are hence landmarks. We can deduce that LIGHT-MATCH is unitary by Lemma 6.3 and that LIGHT-CANDLE is unitary by Lemma 6.4 (case (2)). Thus, in TR there is a single variable $\tau(e)$ for each event $e \in \mathrm{Events}(A)$. As we have already seen, Match-lit is −monotone, so TR contains the −authorisation constraint $\tau(\text{match-lit} \rightarrow\mid \text{LIGHT-CANDLE}) < \tau(\text{LIGHT-MATCH} \rightarrow \neg\text{match-lit})$. It also contains the causality constraint $\tau(\text{LIGHT-MATCH} \rightarrow \text{match-lit}) < \tau(\text{match-lit} \mid\rightarrow \text{LIGHT-CANDLE})$. The two inherent constraints $\tau(\text{LIGHT-MATCH} \rightarrow \neg\text{match-lit}) - \tau(\text{LIGHT-MATCH} \rightarrow \text{match-lit}) \leq 2$ and $\tau(\text{match-lit} \rightarrow\mid \text{LIGHT-CANDLE}) - \tau(\text{match-lit} \mid\rightarrow \text{LIGHT-CANDLE}) = 2$ then provide a contradiction. No form of relaxation which does not take into account the duration of actions can detect the unsolvability of this problem, since the identical problem with different durations given in Example 4.2 has a solution.

Example 6.8: We now give a generic example involving the choice between two alternatives in which the temporal relaxation TR can detect unsolvable problems that cannot be detected by ignoring all deletes. We can illustrate our generic example by a simple non-temporal planning problem P with initial state $I = \{\mathrm{f}\}$, goal $G = \{\mathrm{g,h}\}$ and the following two actions:

B: $\mathrm{f} \rightarrow \neg\mathrm{f}, \mathrm{g}$

C: $\mathrm{f} \rightarrow \neg\mathrm{f}, \mathrm{h}$

The fluents have many possible interpretations, including: f = I have a packet, g = I have sent the packet to Destination1, h = I have sent the packet to Destination2. Clearly this problem has no solution, but this is not discovered by the ignore-deletes relaxation (which cannot take into account the fact that I no longer have the packet once I have sent it somewhere).

To show that TR has no solution we give a proof for the general case in which A^{LM} is EU, actions B,C $\in A^{LM}$ are instantaneous, f$\in$(Cond(B) $\cap$ Del(B) $\cap$ Cond(C) $\cap$ Del(C) $\cap$ I)\Add(A), g $\in$ Add(B) $\cap$ (G\$I$) and h $\in$ Add(C) $\cap$ (G\$I$). The fluent f is −monotone by Lemma 4.3 since there is no action to establish it. TR has no solution since we obtain the following contradiction by a sequence of −authorisation, inherent, intrinsic, −authorisation, inherent and intrinsic (respectively) constraints: $\tau_{last}(f \rightarrow\!| B) < \tau_{first}(C \rightarrow \neg f) = \tau_{first}(f \rightarrow\!| C) \leq \tau_{last}(f \rightarrow\!| C) < \tau_{first}(B \rightarrow \neg f) = \tau_{first}(f \rightarrow\!| B) \leq \tau_{last}(f \rightarrow\!| B)$.

We should point out that improved versions of the ignore-deletes relaxation which retain some information concerning deletes would also be able to detect the unsolvability of this simple problem. For example, the transformation of Keyder, Hoffmann and Haslum (2012) can detect unsolvability by introducing a special fluent representing the conjunction of g and h .

Example 6.9: We now show that the temporal relaxation TR can detect unsolvable problems which are not necessarily establisher-unique. In this example, all actions are instantaneous and hence we present it in the form of a non-temporal planning problem P with initial state I = {j,m,d}, goal G = {g} and the following three actions:

Buy: j, m → h, ¬d, ¬m
Sell: h → m, ¬h
Mort2: d, h → m, ¬d, g

We can interpret the fluents as follows: j = I have a job, m = I have money, d = I am debt-free, h = I own a house, g = I have taken out a second mortgage. For example, the action Buy is possible only if I have a job and money to put down a deposit on a house; the result is that I own a house but I am in debt and no longer have money. The goal is to take out a second mortgage via the action Mort2.

This problem has no solution, but this fact is not detected by the standard relaxation consisting of ignoring destructions of fluents. To set up TR, we first determine the action landmarks A^{LM} = {Buy, Mort2} easily identified as landmarks by the rules given by Cooper, de Roquemaurel and Régnier (2011) since they establish the sub-goals h and g, respectively, not present in the initial state. Observe that A^{LM} is EU relative to the set of sub-goals {g,h} and retains some destructions of fluents. By applying Lemma 6.3 to the fluent d (which is the –monotone by Lemma 4.3), we can deduce that Mort2 is unitary. Then we can deduce from Lemma 6.4 (case (3)) that Buy is unitary, since Add(Buy) = {h} and Mort2 is the only action requiring h. Thus, in TR there is a single variable $\tau(e)$ for each event $e \in$Events(A^{LM}). TR contains the following constraints: τ(d →| Mort2) = τ(h |→ Mort2) by an internal TR-constraint in Mort2; τ(Buy → h) = τ(Buy → ¬d) by an internal TR-constraint in Buy; τ(Buy → h) < τ(h |→ Mort2) by the causality TR-constraint on h; τ(d →| Mort2) < τ(Buy → ¬d) by the –authorisation TR-constraint, since d is –monotone. This set of four constraints has no solution, from which we can deduce that P has no solution. This example shows that temporal relaxation can be useful even in non-temporal planning problems which are not establisher-unique.

The above examples show that EU monotone relaxation TR can be stronger than the relaxation based on ignoring deletes for two reasons: TR uses temporal information, for example concerning the duration of actions, and retains destructions of fluents. To see that ignoring deletes can be stronger than EU monotone relaxation, consider a problem in which the unique goal g is produced by a unique action a such that Cond(a) = {f} where the fluent f is produced by two distinct actions b and c. In the EU monotone relaxation, the fluent f is deleted from Cond(a), since it is established by two distinct actions, and the relaxed version of the problem is immediately solvable by a plan containing the single action a. Ignoring deletes, on the other hand, can detect the unsolvability of the original problem in certain cases, for example, if actions b and c are instantaneous, b is the only action that establishes some fluent $p \in \mathrm{Cond}(c) \setminus I$ and c is the only action that establishes some fluent $q \in \mathrm{Cond}(b) \setminus I$.

An obvious application of temporal relaxation is the detection of action landmarks by the following classic technique which applies to any valid relaxation (Hoffmann, Porteous & Sebastia, 2004; Cooper, de Roquemaurel & Régnier, 2011). Let $\mathcal{P}[-a]$ represent the planning problem $\mathcal{P}$ without a particular action a. If the temporal relaxation of $\mathcal{P}[-a]$ has no solution, then we can conclude that a is an action landmark for $\mathcal{P}$.

In the following sections we investigate other applications of temporal relaxation concerning the detection of different forms of monotonicity. The basic idea is that if H is a hypothesis to be tested and H can be expressed as the conjunction of $\mathrm{STP}^{\neq}$ constraints, then we can add H to the constraints of the temporal relaxation TR. We thus obtain an $\mathrm{STP}^{\neq}$ instance which we denote by TR[H]: if TR[H] has no solution then H cannot be true in any solution to the planning problem. In each case, the complexity of solving TR[H] is $O(n^3)$ time and $O(n^2)$ space, where n is the total number of events in the actions in A (as we have already seen in the proof of Theorem 5.7).

7. Detecting Monotonicity* Using Temporal Relaxation

A subclass 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 recognize 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. In this section we show how our temporal relaxation can be used to detect monotonicity* of certain fluents. Unfortunately, the following theorem shows that, in general, detection of monotonicity* is as difficult as temporal planning.

Theorem 7.1. Determining whether a fluent of a temporal planning problem $\langle I,A,G \rangle$ is monotone (or 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 $\langle I,A,G \rangle$ has no solution, then all fluents are trivially monotone (and hence monotone*) by Definition 4.1, since they are neither established nor destroyed in any plan. 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 $\langle I,A,G \rangle$: f_1 is monotone (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 $\langle I,A,G \rangle$ 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. In this section we give rules which can be applied in polynomial time. Given Theorem 7.1, 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 5.7, represents a tractable class, since it can be both recognized and solved in polynomial time.

To detect the +monotonicity (+monotonicity*) of a fluent f it suffices to give a proof that f cannot be destroyed in a (minimal) plan after being established. In the conference version of this paper we gave rules to provide such a proof, based on knowledge of the monotonicity of other fluents (Cooper, Maris & Régnier, 2012). It turns out that there is a simpler and more general proof rule (although computationally more expensive) which involves solving an STP$^{\neq}$ for each pair of actions a, b such that $f \in \mathrm{Add}(a) \cap \mathrm{Del}(b)$. If the set of actions is establisher-unique, then there is at most one such action a. To try to prove that b cannot destroy f after a establishes f, we set up a relaxation TR[Before(a, f, b)] consisting of the temporal relaxation TR of the planning problem together with a single hypothesis constraint: $\mathrm{Before}(a,f,b) = \{\tau_{\mathrm{first}}(a \rightarrow f) < \tau_{\mathrm{last}}(b \rightarrow \neg f)\}$. We can consider the case in which such a pair of actions a,b does not exist (see Lemma 4.3) as simply a special case of this rule. Note, however, that the fact that TR[Before(a,f, b)] has a solution is a necessary but not a sufficient condition for the existence of a valid temporal plan in which b destroys f after a establishes f. Indeed, Theorem 7.1 tells us that it is highly unlikely that a polynomial-time algorithm exists for determining whether a fluent is monotone*.

To detect the −monotonicity* of a fluent f we need to prove that f cannot be established in a minimal plan after being destroyed. In the corresponding STP$^{\neq}$ TR[After(a, f, b)], the hypothesis is: $\mathrm{After}(a,f,b) = \{\tau_{\mathrm{first}}(b \rightarrow \neg f) < \tau_{\mathrm{last}}(a \rightarrow f)\}$. We assume that when setting up the temporal relaxation TR[Before(a,f, b)] or TR[After(a,f, b)] we apply the rules given in the previous section for the identification of unitary actions. This implies that implicitly we are only considering minimal plans and hence that we detect monotonicity* rather than monotonicity.

Lemma 7.2. Suppose that the set of actions A is EU. If TR[Before(a,f, b)] has no solution for any pair of actions $a,b \in A$ such that $f \in \mathrm{Add}(a) \cap \mathrm{Del}(b)$, then f is +monotone* relative to $\langle I,A,G \rangle$. If TR[After(a,f, b)] has no solution for any pair of actions $a, b \in A$ such that $f \in \mathrm{Add}(a) \cap \mathrm{Del}(b)$, then f is −monotone* relative to $\langle I,A,G \rangle$.

In order to apply Theorem 5.6, we also have to prove that all fluents in $I \cap (\mathrm{Cond}(A^{\mathrm{r}}) \cup G)$ are −monotone* relative to $\langle I,A^{\mathrm{p}},G \rangle$. A plan for the problem $\langle I,A^{\mathrm{p}},G \rangle$ necessarily includes all actions from A^{LM}, but may or may not include actions from $A^{\mathrm{p}} \setminus A^{\mathrm{LM}}$. As a consequence of this, we can only impose constraints on actions in A^{LM}. Indeed, under the extra hypothesis that the set of actions A^{p} is establisher-unique, the corresponding STP$^{\neq}$ is identical to TR[After(a,f, b)].

Lemma 7.3. If the set of actions A^{p} is establisher-unique and the temporal relaxation TR[After(a,f, b)] has no solution for any pair of actions $a, b \in A^{\mathrm{p}}$ such that $f \in \mathrm{Add}(a) \cap \mathrm{Del}(b)$, then f is −monotone* relative to $\langle I,A^{\mathrm{p}},G \rangle$.

We now give a simple lemma to detect certain +monotone* fluents based on the notion of unitary action. We assume that Lemmas 6.3, 6.4 and 6.5 are used to detect unitary actions.

Lemma 7.4. If A is establisher-unique and action $a \in A$ is unitary, then all fluents $f \in \mathrm{Add}(a) \cap (G \setminus (I \setminus \mathrm{Del}(A^{\mathrm{LM}})))$ are +monotone* relative to the temporal planning problem $\langle I,A,G \rangle$.

Proof: Let $f \in$ Add(a) $\cap$ ($G \setminus (I \setminus$ Del(A^{LM}))) and let P be a minimal plan for <I,A,G>. All fluents in $G \setminus (I \setminus$ Del(A^{LM})) must be established in P. Thus P must contain an instance of action a, since A is establisher-unique. Indeed, since a is unitary, P contains exactly one instance of a. Therefore, f is established exactly once in P, and furthermore cannot later be destroyed in P since f is a goal fluent. It follows that f is +monotone*. □

Example 7.5. Consider the following simple example of a planning problem with instantaneous actions:

Wash_hair: $\rightarrow \neg$d, c

Dry_clean_hair: c $\rightarrow$ d

where d means dry hair and c means clean hair, $I = \emptyset$ and G = {d,c}. Note that we impose the condition that hair must be clean before it can be dried. The fluent d is not monotone since there is a solution plan Wash_hair, Dry_clean_hair, Wash_hair, Dry_clean_hair (in which the last two actions are clearly redundant) which destroys, establishes, destroys and re-establishes d, but this plan is clearly not minimal. We can deduce from Lemma 6.4 (case(2)) that Dry_clean_hair is unitary. Lemma 7.4 then tells us that d is +monotone* since Add(Dry_clean_hair) $\cap$ ($G \setminus (I \setminus$ Del(A^{LM}))) = {d}.

The following theorem now follows from Theorem 5.7 together with the fact that each STP$^{\neq}$ can be solved in polynomial time. Recall that a class is tractable if it can be recognised and solved in polynomial time.

Theorem 7.6. Let Π_1 be the class of positive temporal planning problems <I,A,G> in which Constr(A^r) are interval constraints, A^r is establisher-unique relative to SG, all fluents in Cond(A^r) $\cup$ G are monotone* relative to <I,A^r,G> and all fluents in $I \cap$ (Cond(A^r) $\cup$ G) are –monotone* relative to <I,A^p,G>, where monotonicity* of all fluents can be detected by applying Lemmas 7.3 and 7.4. Then Π_1 is tractable.

We have already seen in the proof of Theorem 5.7 that each temporal relaxation can be solved in O(n^3) time and O(n^2) space, where n is the total number of events in the actions in A. The number of temporal relaxations to solve, in order to prove that a temporal planning problem belongs to Π_1, is proportional to the number of triples (a, f, b) such that $a, b \in A^p$ and $f \in$ Add(a) $\cap$ Del(b). The number of pairs (f, b) such that $b \in A^p$ and $f \in$ Del(b) is bounded above by n. If A^p is establisher-unique, then there is at most one action that $a \in A^p$ such that $f \in$ Add(a). Therefore, the complexity of recognizing Π_1 is O(n^4) time and O(n^2) space. In the conference version of this paper (Cooper, Maris & Régnier, 2012) we gave simple rules that can be used to recognize a subclass of Π_1 in O(n^2) time and O(n) space.

We now discuss further rules for the detection of monotone* fluents. We will show that more monotone* fluents can be detected in polynomial time but at the cost of greater computational complexity.

We say that an action-instance *a usefully produces* a fluent h during the execution of a plan if h was false just before being established by a. We say that *a usefully produces the required fluent h* if a usefully produces h and either $h \in G$ or the fluent h is the condition of some action c in the plan such that $\tau(a \rightarrow h) < \tau(h \mid\rightarrow c)$. We can now state the following general proposition.

Proposition 7.7. Suppose that the set of actions A is EU relative to the set of sub-goals and let $a \in A$ be the unique action that establishes sub-goal f. **(a)** If $\forall b \in A$ such that $f \in \text{Del}(b)$, there is no minimal plan in which some instance of b destroys f after some instance of a establishes f, and such that the instance of a which first establishes f and the last instance of b which last destroys f both usefully produce required fluents, then f is +monotone*. **(b)** If $\forall b \in A$ such that $f \in \text{Del}(b)$, there is no minimal plan in which some instance of a establishes f after some instance of b destroys f, and such that the instance of a which last establishes f and the instance of b which first destroys f both usefully produce required fluents, then f is –monotone*.

Proof: **(a)** Let P be a minimal plan in which some instance of b destroys f after some instance of a establishes f. Then, by the hypothesis of the proposition, either the instance of a which first establishes f in P or the last instance of b which last destroys f in P does not usefully produce a required fluent. Hence P cannot be minimal, since we could delete either this instance of b or this instance of a from P to leave another valid plan. This contradiction shows that f is +monotone*. The proof of case **(b)** is similar. □

We now give a lemma which allows us to deduce one of the hypotheses of Proposition 7.7 and hence to deduce that a fluent f is +monotone* or that it is –monotone*. To simplify the expression of the lemma, we suppose that there is a goal-achieving action a_G that must be executed at the end of all plans and such that $\text{Cond}(a_G) = G$. This simply means that goal fluents h do not need to be treated as special cases.

Lemma 7.8. Suppose that A is EU relative to the set of sub-goals SG and let $a \in A$ be the unique action that establishes fluent $f \in SG$. Let $b \in A$ be such that $f \in \text{Del}(b)$.

(a) Let $h \in SG \cap \text{Add}(a)$ and $h' \in SG \cap \text{Add}(b)$. If any of the following conditions hold, then there is no minimal plan P in which the last destruction of f by an instance of b occurs after the first establishment of f by an instance of a, and in which this instance of b usefully produces the required fluent h' and this instance of a usefully produces the required fluent h:

(1) for all actions c,c' such that $h \in \text{Cond}(c)$, $h' \in \text{Cond}(c')$, $\text{TR}[\text{Before}(a,f,b) \cup \text{For}(a,\text{first},h,c) \cup \text{For}(b,\text{last},h',c')]$ has no solution, where $\text{For}(x,\text{L},h,c) = \{\tau_{\text{L}}(x \rightarrow h) < \tau_{\text{last}}(h \mid\!\rightarrow c)\}$.

(2) $\text{Constr}(b)$ imposes a fixed interval between the destruction of f and the establishment of h' by b, h' is monotone* and for all actions c,c' such that $h \in \text{Cond}(c)$ and $h' \in \text{Cond}(c')$, $\text{TR}[\text{Before}(a,f,b) \cup \text{Once}(b) \cup \text{For}(a,\text{first},h,c) \cup \text{For}(b,\text{last},h',c')]$ has no solution, where $\text{Once}(x) = \{\tau_{\text{first}}(E) = \tau_{\text{last}}(E) \mid E \in \text{Events}(x)\}$.

(b) Let $h \in SG \cap \text{Add}(a)$ and $h' \in SG \cap \text{Add}(b)$. If any of the following conditions hold, then there is no minimal plan P in which the last establishment of f by an instance of a occurs after the first destruction of f by an instance of b, and in which this instance of a usefully produces the required fluent h and this instance of b usefully produces the required fluent h':

(1) for all actions c,c' such that $h \in \text{Cond}(c)$, $h' \in \text{Cond}(c')$, $\text{TR}[\text{After}(a,f,b) \cup \text{For}(a,\text{last},h,c) \cup \text{For}(b,\text{first},h',c')]$ has no solution.

(2) $\text{Constr}(b)$ imposes a fixed interval between the destruction of f and the establishment of h' by b, h is monotone* and for all actions c,c' such that $h \in \text{Cond}(c)$ and $h' \in \text{Cond}(c')$, $\text{TR}[\text{After}(a,f,b) \cup \text{Once}(a) \cup \text{For}(a,\text{last},h,c) \cup \text{For}(b,\text{first},h',c')]$ has no solution.

Proof: (a) We suppose that A is EU relative to SG, $f \in SG \cap \text{Add}(a) \cap \text{Del}(b)$, $h \in SG \cap \text{Add}(a)$ and $h' \in SG \cap \text{Add}(b)$.

(1) If TR[Before(a,f,b) ∪ For(a,first,h,c) ∪ For(b,last,h',c')] has no solution for all actions c,c' such that $h \in \text{Cond}(c)$, $h' \in \text{Cond}(c')$, then it cannot be the case that in a minimal plan P the first establishment of f by a occurs before the last destruction of f by an instance of b, and the instance of a which first establishes f usefully produces the required fluent h in P, and the instance of b which last destroys f usefully produces the required fluent h' in P.

(2) If h' is monotone*, then only the instance of b which first establishes h' can usefully produce h' in P. Since there is a fixed interval between the destruction of f and the establishment of h' by b, it is necessarily the same instance of b which first destroys f. Since it is the instance of b which last destroys f which is assumed to usefully produce h', we can deduce that all instances of b are synchronised to destroy f at exactly the same moment. But then this contradicts Condition (5) of the Definition 3.2 of a temporal plan, unless there is only one instance of b in P. The result follows from the same argument as in case (2) with the extra constraint Once(b) that there is only one instance of b in P. □

The proof of part **(b)** of the lemma is similar. □

Example 7.9. Consider the following EU temporal planning problem in which all actions are instantaneous:

Check: p → g, o

Drive: p, o → a, ¬g

Take: g → p

and I = {g} and G = {a}. One interpretation of these actions and fluents is: Have_Engine_checked (*Check*), Drive_to_destination (*Drive*), Take_Petrol (*Take*), Have_petrol (p), At_garage (g), Engine_OK (o), Arrived (a). The fluent g is not monotone since there is a plan *Take*, *Check*, *Drive*, *Check* (in which the last action is clearly redundant) which establishes, destroys, and establishes g. However, g is −monotone* since in a minimal plan action *Check* cannot usefully produce a fluent $h \in$ Add(*Check*) = {g, o} after action *Drive* has destroyed g. In the case h=g, this is by Lemma 7.8(b)(1): *Take* is the only action such that g ∈ Cond(*Take*), and TR[After(*Check*,g,*Drive*) ∪ For(*Check*,last,g, *Take*)] has no solution. In the case h = o, this is by Lemma 7.8(b)(2) since o is monotone* (by Lemma 4.3) and TR[After(*Check*,*g*,*Drive*) ∪ Once(*Check*)] has no solution. Note that since a, p are monotone by Lemma 4.3, we can deduce from Lemma 6.4 (case (1)) that actions *Drive* and *Take* are unitary. It then follows from Lemma 6.4 (case (3)) that action *Check* is also unitary. We therefore impose in TR that the actions *Check*, *Drive* and *Take* occur only once; it follows that we could have deduced that g is −monotone* directly from Lemma 7.2 without having to use Lemma 7.8.

Combining Proposition 7.7 and Lemma 7.8 allows us to define a tractable class of temporal planning problems which is larger than the class described in Theorem 7.6.

Theorem 7.10. Let Π_2 be the class of positive temporal planning problems <I,A,G> in which Constr(A^r) are interval constraints, A^r is establisher-unique relative to SG, all fluents in Cond(A^r) ∪ G are monotone* relative to <I,A^r,G> and all fluents in I ∩ (Cond(A^r) ∪ G) are −monotone* relative to

<I,A^p,G>, where monotonicity* of all fluents can be deduced from Proposition 7.7, Lemmas 7.4 and 7.8. Then Π_2 is tractable.

The number of temporal relaxations to solve, in order to prove that a temporal planning problem belongs to Π_2, is proportional to the number of septuples (a,f,b,c,h,c',h') such that $a,b,c,c' \in A^p$, $f \in \mathrm{Add}(a) \cap \mathrm{Del}(b)$, $h \in \mathrm{Add}(a) \cap \mathrm{Cond}(c)$ and $h' \in \mathrm{Add}(b) \cap \mathrm{Cond}(c')$. We have seen in Section 5 that, assuming $\overline{A}$ is establisher-unique, the number of triples (a,f,b) satisfying $a,b \in A^p$ and $f \in \mathrm{Add}(a) \cap \mathrm{Del}(b)$ is bounded above by n, the total number of events in the actions in A. The number of pairs (c,h) such that $c \in A^p$ and $h \in \mathrm{Cond}(c)$ is again bounded above by n. Therefore, the number of relaxations to be solved is $O(n^3)$. We have seen in the proof of Lemma 4.3 that each temporal relaxation can be solved in $O(n^3)$ time and $O(n^2)$ space. It follows that the complexity of recognizing Π_2 is $O(n^6)$ time and $O(n^2)$ space.

8. Experiments on IPC-2011 Benchmarks

We conducted experiments on the benchmark problems from the temporal deterministic track of the 7th international planning competition IPC-2011, in order to test the applicability of our proposed temporal relaxation TR as well as the relative utility of our various lemmas for the detection of monotonicity*. The main drawback of our temporal relaxation is that it only concerns EU fluents (i.e. fluents f for which there is a single action a with $f \in \mathrm{Add}(a)$). Indeed, the first step in setting up TR is to remove all fluents which are not EU from the goal. All goal fluents are removed for all problems in the following domains: **floortile**, **matchcellar**, **parking, pegsol, openstacks, sokoban, storage, turnandopen**. We therefore concentrated our experiments on the three domains **crewplanning**, **parcprinter**, **tms**. For each problem in each domain, let SG^p denote the set of possible sub-goals in the original unrelaxed problem. For each of the 20 problems in each of these domains, we determined the set $EUSG^p$ of possible sub-goals SG^p which are EU. Not all of these EU possible sub-goals of the unrelaxed problem remain possible sub-goals in TR, since in TR we remove all fluents which are not EU from conditions of all actions. We calculated the set of possible sub-goals in the relaxed problem, which we call SG^p(rel) to distinguish this set from the set of possible sub-goals SG^p in the unrelaxed problem. Moreover, we calculated the set of actions which produce possible sub-goals in the relaxed problem, which we call A^p(rel) to distinguish this set from A^p in the unrelaxed problem. The first three columns of Table 8.1 show the minimum, mean and maximum percentages of possible sub-goals which remain possible sub-goals in TR (i.e. the ratio $|\, SG^p(\mathrm{rel}) \,| \,/\, |\, SG^p \,|$) over the 20 problems. We then used the fluents in $EUSG^p$ and, in particular, the ones from SG^p(rel) to test the relative frequency of monotonicity*. The results for SG^p(rel) are shown in the next three columns of Table 8.1. In the **parcprinter** domain, all $EUSG^p$ fluents were detected to be monotone*. For each problem in the **crewplanning** domain, almost all fluents in $EUSG^p$ were detected to be monotone*. In each problem in the **tms** domain, 37% of the fluents in $EUSG^p$ were detected to be monotone*, and 50% of SG^p(rel). These results indicate that in certain temporal planning problems EU monotone* fluents are quite common, but that in others TR can provide no useful information since no goals are EU.

IPC 2011 Domain	SG^p(rel)			Monotone* SG^p(rel)			Unitary A^p(rel)		
	MIN	MEAN	MAX	MIN	MEAN	MAX	MIN	MEAN	MAX
Crewplanning	21%	36%	94%	87%	95%	98%	39%	56%	71%
Parcprinter	4%	8%	15%	100%	100%	100%	56%	72%	94%
Tms	60%	60%	60%	50%	50%	50%	54%	54%	54%

Table 8.1. Results of experiments on three domains from IPC 2011.

The last three columns of Table 8.1 give the number of actions in A^p(rel) which we detected as unitary in TR. The identification of unitary actions by Lemmas 6.3, 6.4 and 6.5 can be achieved in linear time and provides useful information which is used by TR in the detection of monotonicity*. On average, over half the actions in A^p(rel) were found to be unitary.

We detected monotonicity* by applying our lemmas in increasing order of computational complexity: Lemma 4.3, Lemma 7.4, Lemma 7.2, and then Lemma 7.8. In all domains, the majority of fluents which are recognised as monotone* are recognised as such by applying Lemma 4.3, some fluents are recognised as monotone* by applying Lemma 7.4 and all other monotone* fluents are recognised by Lemma 7.2 (which uses the temporal relaxation TR). We found no monotone* fluents which required Lemma 7.8 to be detected. The significantly greater complexity of applying Lemma 7.8 compared with Lemma 7.2 means that it is not worth applying systematically to all problems. On the other hand, our experiments have confirmed that Lemmas 4.3, 7.4 and 7.2 are all effective for the detection of monotonicity*. Table 8.2 shows for each of the three domains, the minimum, mean and maximum percentage of fluents in SG^p which are detected by each of these three lemmas.

IPC 2011 Domain		Monotone* SG^p(rel)			
		Lemma 4.3	Lemma 7.4	Lemma 7.2	ALL
Crewplanning	MIN	80%	0%	0%	87%
	MEAN	87%	0%	7%	95%
	MAX	95%	0%	13%	98%
Parcprinter	MIN	88%	0%	0%	100%
	MEAN	95%	0%	5%	100%
	MAX	100%	0%	12%	100%
Tms	MIN	29%	21%	0%	50%
	MEAN	29%	21%	0%	50%
	MAX	29%	21%	0%	50%

Table 8.2. Percentages of fluents detected as monotone* by three different lemmas.

To illustrate the degree of variation between different problems within the same domain, in Figures 8.3, 8.4 and 8.5 we show the details of each of the 20 problems in the three different domains. For each problem we show the number of monotone* fluents in SG^p(rel) detected by different lemmas.

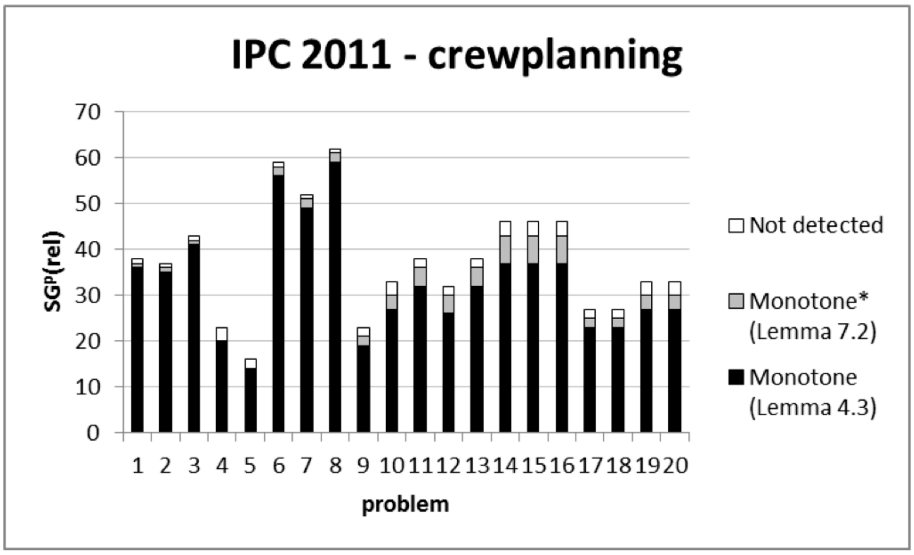


Figure 8.3. The number of fluents in SG^{P}(rel) detected as monotone* by different lemmas in the **crewplanning** domain.

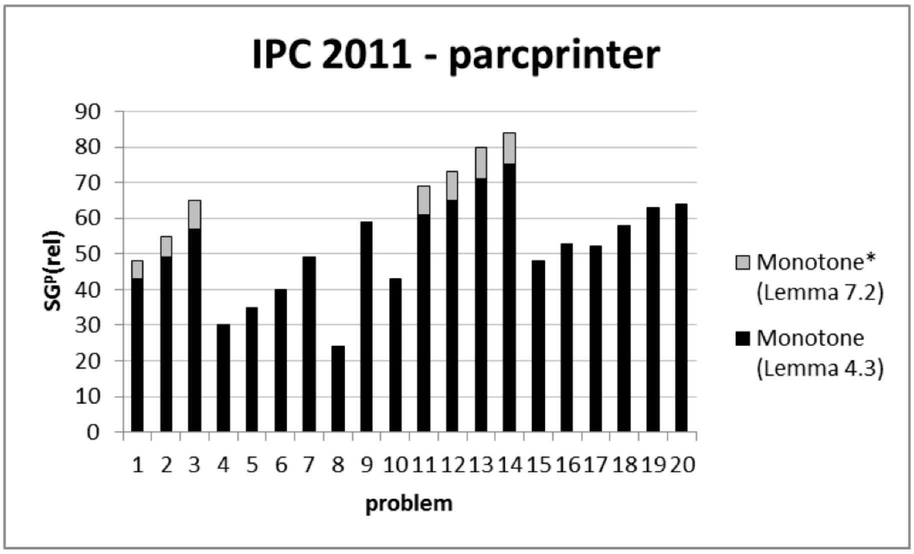


Figure 8.4. The number of fluents in SG^{P}(rel) detected as monotone* by different lemmas in the **parcprinter** domain.

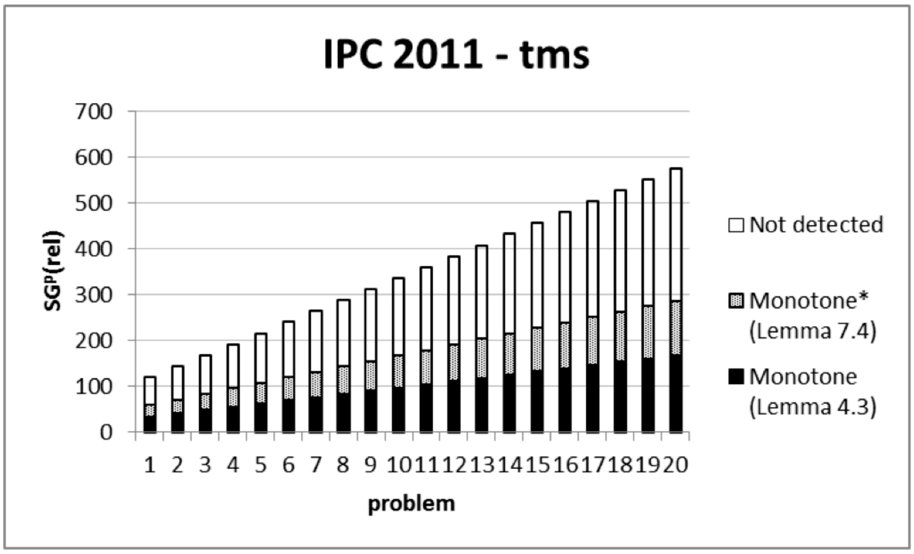


Figure 8.5. The number of fluents in SG^P(rel) detected as monotone* by different lemmas in the **tms** domain.

As a general conclusion of our experimental trials, we have seen that in many problems TR provides no useful information since all goal fluents are removed. Nevertheless, we have identified various benchmark domains in which it can be applied. The fact that a large percentage of fluents were found to be monotone* and a large percentage of actions were found to be unitary demonstrates the potential importance of these notions beyond their use in the temporal relaxation TR. These experimental trials together with our investigation of specific examples (such as Example 7.9 or the Temporal Cement Factory domain described in the following section) seem to indicate that integrating the detection of unitary actions into TR provides as much information as the more computationally expensive approach of Lemma 7.8.

9. Examples of Applications of EU Monotone Planning

We have previously shown that EU monotone planning has potential applications in various industrial settings, such as the construction or the chemical and pharmaceutical industries (Cooper, Maris & Régnier, 2012, 2013b). For example, the Temporal Chemical Process domain, described in detail by Cooper, Maris and Régnier (2013b), involves different kinds of operations on chemicals that are performed in the industrial production of compounds. For each raw material, there is an operator that can *activate* its source. Then, this raw material can be *catalysed* in different ways to *synthesize* different products. These products can be *mixed* and *reacted* using the raw material once again to produce the desired compound. 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 (which is often the case due to industrial constraints).

We now give in detail an example from the construction industry to show how the detection of unitary actions can greatly speed up the recognition of such problems. The Temporal Cement Factory planning domain (Cooper, Maris & Régnier, 2013b) allows us to plan concrete mixing, delivery and use. An action of duration 30 time units *makes and times* a batch of concrete which is fluid from time unit 3 to 30 (after which it sets). At the same time, a concrete-mixer must be *cleaned*, in order for the concrete to be *loaded*, then *driven* to a building site, where it is *unloaded*. The concrete must then be *used* while it is still fluid. This process is illustrated by the temporal plan given in Figure 5. This set of actions A (illustrated in the temporal plan shown in Figure 5) are all landmarks. The initial state I and the goal G are given by

I = `{At-factory(m), Available(c)}`

G = `{Delivered(m,c,s), Used(c)}`

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

SG = `{Delivered(m,c,s), Used(c), Fluid(c), At(m,s), Available(c), On(m,c), At-factory(m), Empty(m)}`

A^r = `{USE(c), UNLOAD(m,c,s), DRIVE(m,s), LOAD(m,c), CLEAN(m), MAKE-AND-TIME-CONCRETE(c)}`

For all $a_i \neq a_j \in A$, we have Add(a_i) $\cap$ Add(a_j) $\cap$ $SG = \varnothing$. Hence, by Definition 3.3, the set of actions A is EU relative to SG. We can immediately remark that no actions delete the fluents `Used(c)`, `Delivered(m,c,s)` and `At(m,s)`, and no actions add the fluents `Available(c)` and `At-factory(m)`. Thus, by Lemma 4.3, each of these fluents are both –monotone and +monotone relative to <I,A,G>. By Lemma 7.2, we can deduce that `On(m,c)` is –monotone since the temporal relaxation TR[After(`LOAD(m,c)`,`On(m,c)`,`UNLOAD(m,c,s)`)] has no solution. By a similar argument, `Fluid(c)` is also –monotone by Lemma 7.2.

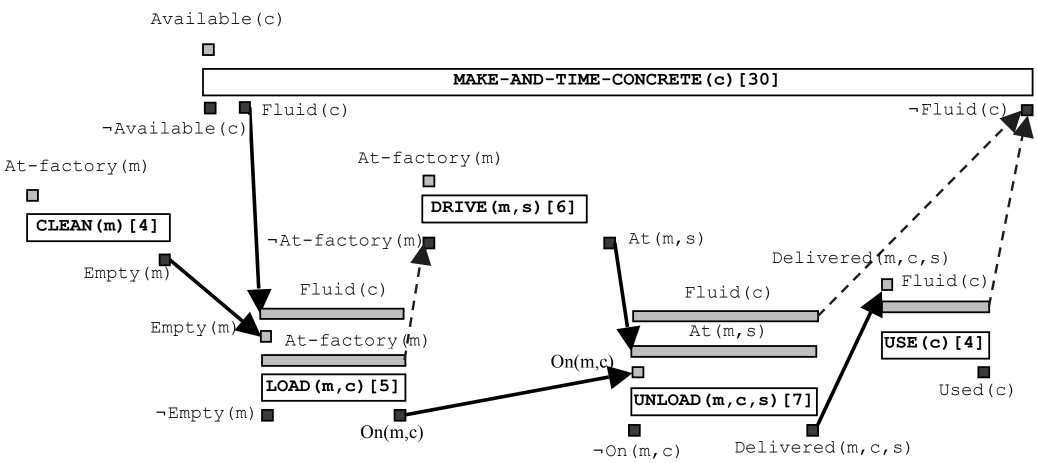


Figure 5: Ready-mix Concrete Delivery Temporal Plan

We can then detect unitary actions. From Lemma 6.4 (case (1)), we can deduce that `UNLOAD(m,c,s)` is unitary. Then, applying Lemma 6.4 (case (3)), with h = `On(m,c)` and b =

`UNLOAD(m,c,s)`, tells us that `LOAD(m,c)` is unitary. Finally, applying Lemma 6.4 (case (3)), with h = `Empty(m)` and b = `LOAD(m,c)`, tells us that `CLEAN(m)` is unitary. Since `CLEAN(m)` is unitary, we effectively add the constraint Once(`CLEAN(m)`) to TR and with this constraint Lemma 7.2 is now sufficient to detect that `Empty(m)` is monotone*. Thus, using the new notion of unitary action and the linear-time rules to detect such actions given in Section 6, we can prove monotonicity* of all fluents without needing to use the computationally expensive Lemma 7.8 as we previously proposed (Cooper, Maris & Régnier, 2013b).

It is now possible to apply Theorem 5.6, since A is EU, all fluents are monotone* and all fluents in I are –monotone*. It follows that TR is a solution procedure for this problem. The problem $<I,A,G>$ has a solution-plan, found by TR, shown in Figure 5. We represent non-instantaneous actions by a rectangle. Conditions are written above an action, and effects below; causality constraints are represented by bold arrows, and –authorisation constraints by dotted arrows.

This example can be extended to the generic case in which there are several sites, several batches of concrete and several mixers. It is monotone and remains EU provided that the goals (via the fluents `Delivered(m,c,s)`) specify which mixer `m` is to deliver which batch `c` to which building site `s`. All such instances can be solved in polynomial time by Theorem 7.10.

10. 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 Lemma 4.3, is covered by the PA tractable class of Jonsson and Bäckström (1998).

An obvious question is whether both establisher-uniqueness and monotonicity are necessary to obtain tractability. An affirmative answer to this question follows from intractability results in non-temporal planning: Bäckström and Klein (1991b) showed that establisher-uniqueness alone cannot prevent minimal plans being of exponential size, and Jonsson and Bäckström (1998) showed that under conditions implying monotonicity of all fluents (the class BA^{+} in their terminology), planning is NP-hard.

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, remark that Theorem 5.6 still holds if the inherent constraints are arbitrary min-closed constraints, since this was the only property required of the constraints in the proof of Theorem 5.6. An example of such a constraint $C(x,y)$ is a binary interval constraint with variable bounds: $y–x \in [f(x,y),g(x,y)]$, 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. The shift-monotonic constraints used by Pralet and Verfaillie (2012) in the scheduling of agile satellites are a subclass of such constraints since in shift-monotonic constraints both $f(x,y)$ and $g(x,y)$ are monotone increasing functions of x and monotone decreasing functions of y. The consistency of a set of shift-monotonic constraints can be tested in time $O(n^3)$.

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, Kambhampati, Mausam & Weld, 2007). The cement factory planning problem given in Section 9 is an example of a temporally-expressive problem, since concurrency of actions is required in any solution. Indeed, in industrial environments, concurrency of actions is often used to keep storage space and turn-around times within given limits. In a previous paper (Cooper, Maris & Régnier, 2013a), we 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 but commonly occurring example was given in Figure 4, concerning an agreement between an employer and employee. The tractable class of temporal planning problems described in Theorem 7.6 contains both temporally-expressive and temporally-cyclic problems. For example, the temporally-cyclic problem given in Figure 4 is establisher-unique and all fluents are both + and −monotone (this follows from Lemma 4.3 since no fluents are destroyed by either action).

Most temporal planning problems will not fall into the tractable class of EU monotone problems. Even so, it may be that certain sub-problems do fall into this class. Given a temporal planning problem <*I*,*A*,*G*>, we can test in polynomial time, for each fluent f, whether the sub-problem <*I*,*A*,*{f}*> satisfies the conditions of Theorem 7.6 (i.e. EU monotone, with monotonicity detectable using the temporal relaxation TR). If this is the case, then we can find in polynomial time a plan P_f which establishes the fluent *f*. This plan P_f can then be considered as an action which could be added to the set of actions *A* in order to facilitate the solution of the original problem <*I*,*A*,*G*>.

Our work is related to the literature regarding landmarks. Porteous, Sebastia and Hoffmann (2001) and Keyder, Richter and Helmert (2010) define a landmark as a fact that must be true at some point in every valid solution-plan. Landmarks have been used in planning in two main ways. The first one is the conception of heuristic functions to guide search algorithms (Richter, Helmert & Westphal, 2008; Richter & Westphal, 2010; Helmert & Domshlak, 2009). Another use of landmarks is to partition the problem into easier subproblems whose goals are disjunctions of landmarks (Hoffmann, Porteous & Sebastia, 2004; Sebastia, Onaindia & Marzal, 2006). More recently, Vernhes, Infantes and Vidal (2013) define a landmark-based meta best-first search algorithm.

Landmarks have also been used for the detection of unsolvable temporal planning problems (Marzal, Sebastia & Onaindia, 2008). A graph is built by adding causal relationships between the extracted landmarks. Then temporal intervals are associated with each landmark and these intervals, together with the causal relationships, define a set of constraints. Finally, a CSP solver checks the consistency of this set and indicates that the problem has no solution when an inconsistency is found. Unlike the set of constraints in our temporal relaxation TR, this set of constraints does not fall into a tractable class. Further research is required to determine whether certain of these constraints could be usefully combined with the $STP^{\neq}$ constraints of our temporal relaxation TR to obtain an even stronger tractable relaxation.

In the general case, verifying that a fact is a landmark is PSPACE-complete (Hoffmann, Porteous & Sebastia, 2004). However, some landmarks can be found more efficiently by using various techniques: Porteous and Cresswell (2002) and Hoffmann, Porteous and Sebastia (2004) present methods for detecting landmarks and relations between landmarks based on backchaining from the goals in the relaxed planning graph, whereas Zhu and Givan (2003) use forward propagation in the same graph and Richter, Helmert and Westphal (2008) use the domain transition graph, a graph whose nodes represent the possible values of the variable and edges represent the possible transitions between values induced by actions.

The notion of monotonicity* introduced in this paper depends on the relative order of the establishment and the destruction of the same fluent within a minimal plan. Our experiments have demonstrated that many fluents in benchmark problems are indeed monotone*. An interesting avenue of future research would be to investigate, both theoretically and empirically, the relative order of the establishment and the destruction of *different* fluents within a minimal temporal plan. This is again closely related to research on landmarks. Different orderings between landmarks have been studied in non-temporal planning. Some of these orderings are guaranteed to hold in every solution-plan and do not prune the solution space (they are sound): "Natural" (Koehler & Hoffmann, 2000), "Necessary" and "Greedy-necessary" (Hoffmann, Porteous & Sebastia, 2004). The natural ordering is the most general and then greedy-necessary ordering and necessary ordering. Others orderings such as "Reasonable", "Obedient-Reasonable" (Koehler & Hoffmann, 2000) are not sound (it is possible that no solution-plan respects these orderings) but they may prune the solution space. All these orderings between landmarks are defined assuming instantaneous actions and would need to be redefined in the temporal framework. Further research is required to determine whether landmark orderings could usefully be extended to incorporate orderings between all types of events in temporal plans (the establishment or destruction of a fluent by an action landmark, or the beginning or end of an interval over which a fluent is required by an action landmark).

11. Conclusion

We have presented a class of temporal planning problems which can be solved in polynomial time and which has a number of possible applications, notably in the chemical, pharmaceutical and construction industries. The notion of monotonicity in temporal planning is an essential part of the definition of this class. We extended our basic notion of monotonicity to monotonicity* by considering only minimal plans.

We have also shown that all planning problems have a relaxation based on EU monotone planning which is an interesting alternative to the standard relaxation produced by ignoring deletes. It also provides a means of detecting action landmarks and monotone* fluents.

Further research is required to discover other possible application areas and, on a practical level, to develop tools to help users find a model of a problem involving only monotone* fluents when such a model exists. On a theoretical level, an interesting avenue of future research is the extension of the tractable classes presented in this paper by relaxing the condition of establisher-uniqueness so that a fluent can be established by more than one action provided that there is only one action that can establish it at any given moment.

Acknowledgements

This research was supported by ANR Project ANR-10-BLAN-0210. We gratefully acknowledge the help of the reviewers whose constructive suggestions led to significant improvements in the presentation of this paper.

References

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

Bäckström C. & Klein I. (1991b) Planning in polynomial time: the SAS-PUBS class, *Computational Intelligence* 7 (3), 181-197.

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

Baier J. A. & Botea A. (2009). Improving planning performance using low-conflict relaxed plans. *Proc. 19th International Conference on Automated Planning and Scheduling (ICAPS'2009).*

Betz C. & Helmert M. (2009). Planning with h+ in Theory and Practice. *KI 2009: Advances in Artificial Intelligence*, LNCS Vol. 5803, 9-16.

Blum A.L. & Furst M.L. (1995) A.L. Blum, M.L. Furst, Fast planning through planning-graphs analysis, in: *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montréal, Québec, Canada, 1636-1642.

Bonet B., Loerincs G. & Geffner H. (1997) A Robust and Fast Action Selection Mechanism for Planning, *Proceedings AAAI-97/IAAI-97*, 714-719.

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

Brafman R.I. & Domshlak C. (2006) Factored Planning: How, When, and When Not. *Proc. 21st National Conference on Artificial Intelligence*, 809-814.

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

Cai D., Hoffmann J. & Helmert M. (2009). Enhancing the context-enhanced additive heuristic with precedence constraints. *Proc. 19th International Conference on Automated Planning and Scheduling (ICAPS'2009)*, 50–57.

Chen H. & Giménez O. (2008) Causal Graphs and Structurally Restricted Planning. *Proc. 18th International Conference on Automated Planning and Scheduling (ICAPS'2008)*, 36-43.

Coles A., Fox M., Long D. & Smith A. (2008) Planning with Problems Requiring Temporal Coordination, *Proc. of AAAI 2008*, 892-897.

Cooper M.C., de Roquemaurel M. & Régnier, P. (2011) A weighted CSP approach to cost-optimal planning, *Artificial Intelligence Communications* 24(1), 1-29.

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

Cooper M.C., Maris F. & Régnier, P. (2012) Tractable monotone temporal planning, *Proceedings ICAPS 2012*, 20-28.

Cooper M.C., Maris F. & Régnier, P. (2013a) Managing Temporal Cycles in Planning Problems Requiring Concurrency, *Computational Intelligence* 29(1), 111-128.

Cooper M.C., Maris F. & Régnier P. (2013b) Relaxation of Temporal Planning Problems, *International Symposium on Temporal Representation and Reasoning (TIME)*, 37-44.

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*, 1852-1859.

Dean T., Firby J. & Miller D. (1988) Hierarchical Planning involving deadlines, travel time and ressources. *Computational Intelligence* 6(1), 381-398.

Dean T. & McDermott D.V. (1987) Temporal Data Base Management. *Artificial Intelligence* 32(1), 1-55.

Dechter R., Meiri I. & Pearl J. (1991) Temporal Constraint Networks, *Artificial Intelligence* 49(1-3), 61-95.

Do M.B. & Kambhampati S. (2003) Sapa: A Multi-objective Metric Temporal Planner, *Journal of Artificial Intelligence Research* 20, 155-194.

Domshlak C. & Dinitz Y. (2001) Multi-agent off-line coordination: Structure and complexity. *Proceedings of 6th European Conference on Planning, ECP'2001*, 277-288.

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

Eyerich P., Mattmüller R. & Röger G. (2009) Using the Context-enhanced Additive Heuristic for Temporal and Numeric Planning, *Proceedings ICAPS 2009*, 130-137.

Fox, M. & Long, D. (2001). Stan4: A hybrid planning strategy based on subproblem abstraction. *The AI Magazine* 22(3), 81–84.

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

Fox M., Long D. & Halsey K. (2004). An Investigation into the Expressive Power of PDDL2.1, *Proc. 16th European Conference on Artificial Intelligence*, 328-342.

Gerevini A. & Cristani M. (1997) On Finding a Solution in Temporal Constraint Satisfaction Problems. *Proc. 15th International Joint Conference on Artificial Intelligence*, 1460-1465.

Gerevini, A., Saetti, A. & Serina, I. (2003). Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research* 20, 239–290.

Ghallab M. & Alaoui A.M. (1989) Managing Efficiently Temporal Relations Through Indexed Spanning Trees. *Proc. 11th Int. Joint Conference on Artificial Intelligence*, 1297-1303.

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

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

Giménez O. & Jonsson A. (2012). The influence of *k*-dependence on the complexity of planning. *Artificial Intelligence* 177-179, 25-45.

Haslum P. (2008) A New Approach To Tractable Planning. *Proceedings of ICAPS'2008*, 132-139.

Haslum P., Slaney J. & Thiébaux S. (2012). Incremental lower bounds for additive cost planning problems. *Proc. 22nd International Conference on Automated Planning and Scheduling (ICAPS'2012)*, 74–82.

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

Helmert M. (2004). A planning heuristic based on causal graph analysis. *Proc. 14th International Conference on Automated Planning and Scheduling (ICAPS'2004)*, 161–170.

Helmert M. (2006) New Complexity Results for Classical Planning Benchmarks. *Proc. 16th International Conference on Automated Planning and Scheduling (ICAPS'2006)*, 52-61.

Helmert M. & Geffner H. (2008). Unifying the causal graph and additive heuristics. *Proc. 18th International Conference on Automated Planning and Scheduling (ICAPS'2008)*, 140–147.

Helmert M. & Domshlak C. (2009) Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? *Proc. International Conference on Automated Planning and Scheduling (ICAPS 2009), 162-169*.

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

Hoffmann J., Porteous J. & Sebastia L. (2004) Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research* 22, 215–278.

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

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

Jonsson A. (2009) The Role of Macros in Tractable Planning. *Journal of Artificial Intelligence Research* 36, 471-511.

Jonsson P. & Bäckström C. (1994) Tractable planning with state variables by exploiting structural restrictions. *Proc. AAAI'1994*, 998-1003.

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

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

Karmarkar N. (1984) A new polynomial time algorithm for linear programming. *Combinatorica* 4 (4) 373–395.

Karpas E. & Domshlak C. (2009) Cost-optimal planning with landmarks, *International Joint Conference on Artificial Intelligence (IJCAI'2009)*, 1728–1733.

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

Katz M., Hoffmann J. & Domshlak C. (2013a). Who said we need to relax All variables? *Proc. 23rd International Conference on Automated Planning and Scheduling, (ICAPS'2013).*

Katz M., Hoffmann J. & Domshlak C. (2013b). Red-Black Relaxed Plan Heuristics. *Proc. 27th AAAI Conference on Artificial Intelligence (AAAI'2013).*

Keyder E. & Geffner H. (2008). Heuristics for planning with action costs revisited. *Proc. 18th European Conference on Artificial Intelligence (ECAI'2008)*, 588–592.

Keyder E., Hoffmann J. & Haslum P. (2012) Semi-Relaxed Plan Heuristics, *Proc. 22nd International Conference on Automated Planning and Scheduling, ICAPS'2012*, 128-136.

Keyder E., Richter S. & Helmert M. (2010) Sound and Complete Landmarks for And/Or Graphs. *Proceedings of the European Conference on Artificial Intelligence (ECAI),* 335-340.

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

Koehler J. & Hoffmann J. (2000) On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *Journal of Artificial Intelligence Research* 12, 338–386.

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

Laborie P. & Ghallab M. (1995) Planning with Sharable Resource Constraints. *Proc. 14th International Joint Conference on Artificial Intelligence*, 1643-1651.

Long D. & Fox M. (2003) Exploiting a graphplan framework in temporal planning, *Proc. 13th International Conference on Automatic Planning and Scheduling*, 52-61.

Maris F. & Régnier P. (2008) TLP-GP: Solving Temporally-Expressive Planning Problems, *TIME 2008*, 137-144.

Marzal E., Sebastia L. & Onaindia E. (2008) Detection of unsolvable temporal planning problems through the use of landmarks. *Proceedings of ECAI'2008*. 919-920.

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

Porteous J. & Cresswell S. (2002) Extending landmarks analysis to reason about resources and repetition. *Proceedings of PLANSIG'2002*, 45–54.

Porteous J., Sebastia L. & Hoffmann J. (2001) On the Extraction, Ordering, and Usage of Landmarks in Planning. *Recent Advances in AI Planning. European Conference on Planning (ECP 2001)*, 37–48.

Pralet C. & Verfaillie G (2012) Time-Dependent Simple Temporal Networks, *Proc. 18th International Conference on Principles and Practice of Constraint Programming*, 608-623.

Reichgelt H. & Shadbolt N. (1990). A Specification Tool for Planning Systems. *Proc. 9th European Conference on Artificial Intelligence*, 541-546.

Richter S., Helmert M. & Westphal M. (2008) Landmarks revisited. *Proc. 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*. 975-982.

Richter S. & Westphal M. (2010) The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research (JAIR)* 39: 127-177.

Rintanen J. (2007) Complexity of Concurrent Temporal Planning. *Proc. 17th International Conference on Automated Planning and Scheduling (ICAPS'2007)*, 280-287.

Rutten E. & Hertzberg J. (1993) Temporal Planner = Nonlinear Planner + Time Map Manager. *Artificial Intelligence Communications 6(1)*, 18-26.

Schwartz P. & Pollack M.E. (2004) Planning with Disjunctive Temporal Constraints. *Proc. ICAPS'04 Workshop on Integrating Planning into Scheduling*, 67-74.

Sebastia L., Onaindia E. & Marzal E., (2006) Decomposition of planning problems. *AI Communications* 19:49–81.

Shin J. & Davis E. (2004) Continuous Time in a SAT-Based Planner. *Proc. 19th National Conference on Artificial Intelligence (AAAI'04)*, 531-536.

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

Smith D.E. (2003) The Case for Durative Actions: A Commentary on PDDL2.1, *Journal of Artificial Intelligence Research* 20, 149-154.

Stergiou K. & Koubarakis M. (2000) Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence* 120(1):81–117.

Vere S. (1983) Planning in Time: Windows and Durations for Activities and Goals. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 5, 246-267.

Vernhes S., Infantes G. & V. Vidal V. (2013) Problem Splitting using Heuristic Search in Landmark Orderings. *Proc. 23rd International Joint Conference on Artificial Intelligence (IJCAI'2013*), 2401-2407.

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

Williams B.C. & Nayak P. (1997) A reactive planner for a model-based executive. *Proc. 15th International Joint Conference on Artificial Intelligence*, 1178-1185.

Younes H.L.S. & Simmons R.G. (2003) VHPOP: Versatile Heuristic Partial Order Planner. *Journal of Artificial Intelligence Research* 20, 405-430.

Zhu L. & Givan R. (2003) Landmark extraction via planning graph propagation. *ICAPS'2003 Doctoral Consortium*, 156–160.