On the Use of Temporal Landmarks for Planning with Deadlines

Eliseo Marzal, Laura Sebastia and Eva Onaindia
Universitat Politècnica de València
Camino de Vera s/n
E46022-Valencia (Spain)
{emarzal, listarin, onaindia}@dsic.upv.es

Abstract

In this paper we present a temporal planning approach for handling problems with deadlines. The model relies on the extraction of temporal landmarks from the problem and the construction of a landmarks graph as a skeleton of the solution plan. A temporal landmark is a proposition that must be achieved in a solution plan to satisfy the problem deadline constraints. Each temporal landmark is associated to three temporal intervals, which are updated and propagated according to the landmarks orders and the deadline constraints. Then, the partial plans in the search tree that are not compliant with the information comprised in this graph are pruned. The experimental results will show that this approach is helpful to quickly detect unsolvable problems and it is also very effective to solve problems with deadlines in comparison to other state-of-the-art planners.

Introduction

This paper presents TempLM, a new approach to temporal planning with deadline constraints. In temporal planning, it does not always follow that the plan with the shortest makespan will be compliant with the achievement time for individual goals. Goal deadlines are often in constraint-based planning, manufacturing operations in supply-chain activities, delivery of goods or workflow-based systems.

The introduction of PDDL3.0 (Gerevini et al. 2009) in the Fifth International Planning Competition (IPC5) was aimed at dealing with richer temporal problems defining state trajectory constraints and preferences. The state trajectory constraints included, among others, the operator within to express deadlines. Two planners participated at IPC5 in the time constraints track, namely MIPS-XXL (Edelkamp, Jabbar, and Nazih 2006) and SGPLAN5 (Chen, Wah, and Hsu 2006), and no one exhibited a good performance or complete accuracy in avoiding deadline violations. Later, OPTIC (Benton, Coles, and Coles 2012), a planner that also handles soft-deadlines with continuous cost functions, showed to outperform MIPS-XXL and SGPLAN on the benchmark temporal planning problems with preferences. For our purposes, we run OPTIC on the IPC5 domains with time constraints and we confirmed it also outperformed the two aforementioned IPC5 planners on problems with goal deadlines. We thus took OPTIC as the reference temporal planner to compare TempLM with.

Both OPTIC and TempLM handle deadlines expressed through the within operator or with Timed Initial Literals (TILs) (Hoffmann and Edelkamp 2005) and also other modal operators defined in PPDL3.0, like always-within, sometime-after or sometime-before. Moreover, TempLM can easily be adapted to the particular features of any temporal model (i.e. Allen’s interval algebra (Allen 1984)). Nevertheless, expressing deadlines with within or TILs suffices for our purposes as we are particularly interested in analyzing the behavior of the two temporal planning approaches when dealing with hard deadlines. We must also say that, unlike OPTIC, TempLM does not handle preferences as it was specifically designed for dealing with goal deadlines.

TempLM builds upon the work in (Marzal, Sebastia, and Onaindia 2008), which presents a CSP-based consistency checker to detect unsolvability in planning problems with deadlines. TempLM exploits the same definition of temporal landmark: a proposition that must be true in a solution plan as well as the requirements for logical and temporal consistency. As the intervals are updated and propagated, solving tightly-constrained problems and detecting unsolvability are the tasks in which planners typically find more difficulties. TempLM builds a graph of temporal landmarks which structurally resembles a solution plan. Likewise, temporal landmarks are annotated with three temporal intervals which define the temporal occurrence of a landmark in a solution plan as well as the requirements for logical and temporal consistency. As the intervals are updated and propagated, and constraints are violated, plans are discarded from the set of solution plans. We apply a simple heuristic search guided by an adapted version of the $h_{LM-cut}$ heuristic (Helmert and Domshlak 2009) to temporal planning problems. The experiments will show that TempLM outperforms OPTIC in solving tightly-constrained and unsolvable problems while obtaining similar or better solutions in loosely-constrained problems from the IPCs.

This paper is organized as follows. Section Preliminary concepts gives some basic definitions. Section Overview in-
Introduces the key aspects of TempLM, which are explained in detail in the next three sections. Section Experiments compares the results of OPTIC and TempLM in several different settings. Finally, we conclude and outline some further work.

**Preliminary concepts**

A planning problem \( P = \langle P, O, I, G \rangle \) is characterized by an initial state \( I \), a goal description \( G \) and a set of actions \( O \) that can be applied in the domain of the problem. The set of all propositions in a planning problem is denoted by \( P \).

In this paper, we assume a subset of the semantics of the temporal model of PDDL2.1 (Fox and Long 2003), the TILs defined in PDDL2.2 (Hoffmann and Edelkamp 2005) and some of the state trajectory constraints introduced in PDDL3.0 (Gerevini et al. 2009). We give here some definitions that will be used throughout the paper.

**Definition 1** A durative action \( a \in O \) contains the following elements:

- Conditions \( Cond(a) \), which are divided into start conditions \( SCond(a) \), end conditions \( ECond(a) \) and invariant conditions \( Inv(a) \).

- Duration. The duration of an action is a positive value represented by \( dur(a) \in \mathbb{R}^+ \).

- Effects \( Eff(a) \), which are divided into the start effects \( SEff(a) = \{ SAdd(a) \cup SDel(a) \} \); and the end effects \( EEff(a) = \{ EAdd(a) \cup EDel(a) \} \). We also define \( AddEff(a) = SAdd(a) \cup EAdd(a) \) and \( DelEff(a) = SDel(a) \cup EDel(a) \).

The structure of a durative action \( a \) determines two distances between a condition \( c \) and an effect \( e \) of the action. We define \( \text{dis}_{E}(c, e) \) as the distance between the time point when \( e \) is asserted and the earliest time point when \( c \) is needed; and \( \text{dis}_{I}(c, e) \) as the distance between the time point when \( e \) is asserted and the latest time point when \( c \) is needed\(^1\). This is detailed in Table 1.

**Definition 2** A temporal plan \( \Pi \) is a set of pairs of the form \( (a, t) \), where \( a \in O \) and \( t \) is the start execution time of action \( a \). Given a proposition \( p \), we denote by \( start(p) \) and \( end(p) \) the time points when \( p \) is asserted and deleted, respectively, by an action in \( \Pi \). The duration (makespan) of a temporal plan \( \Pi \) is \( \text{dur} = \max_{(a,t) \in \Pi} (t + dur(a)) \). That is, the duration of the temporal plan is the end time of the action to finish last.

**Definition 3** A temporal planning problem with deadline constraints is a tuple \( \mathcal{P} = \langle P, O, I, G, D \rangle \), where \( D \) is a set of deadline constraints of the form \( (l, t) \), denoting that \( l \) must be achieved within \( t \) time units.

We are particularly interested in those problems that impose a deadline \( T_{H} \) for the whole plan. We thus assume that \( D \) contains a set of constraints so that we can establish a deadline \( t_{g} \) for each \( g \in G \), and, then: \( T_{H} = \max_{g \in G} (t_{g}) \). In PDDL, there are two basic ways for expressing a deadline \( t_{g} \) over a goal \( g \):\(^2\)

- Explicitly, by means of the modal operator of PDDL3 within, as stated in (Gerevini et al. 2009): \( \langle S_0, 0 \rangle, \langle S_1, t_1 \rangle, \ldots, \langle S_n, t_n \rangle \rangle \Rightarrow (\text{within } t_{g} \text{ } g) \text{ iff } \exists i : 0 \leq i \leq n \bullet S_i \Rightarrow g \land t_i \leq t_{g} \). Therefore, given a deadline expressed as \( (\text{within } t_{g} \text{ } g) \), a constraint \( (g, t_{g}) \) is added to \( D \).

- Implicitly, by means of the TILs of PDDL2.2, which is a way to express time windows. A TIL is defined by a pair \( (t, l) \), where \( t \) is the rational-valued time of occurrence of the literal \( l \) (Hoffmann and Edelkamp 2005). Furthermore, a TIL \( (t, (\text{not } l)) \) corresponds to a deadline if \( t \) is present in the initial state or added by another TIL, never added by any action, and never reinstated by any other TIL \( (t', l) \), such that \( t < t' \) (Coles et al. 2012). In general, given a set of actions \( A \) that assert \( g \) and some TIL \( (t_i, l_i) \) that restrict the execution of these actions, the constraint \( (g, t_{g}) \) to be added to \( D \) is computed as follows:

\[
    t_{g} = \max (t_i + \max_{a \in A} (\text{dis}_{E}(l_i, g)))
\]

For example, let \( (t, (\text{not } p)) \) be the only TIL defined in the problem and let \( a \in O \) be the only action that has \( g \) as an effect, say \( g \in EAdd(a) \). Additionally, \( p \in SCond(a), \text{dur}(a) = d \) and \( p \in I, g \in G \). In this case, the deadline constraint that would be added to \( D \) is \( (g, t+d) \), given that \( \text{dis}_{E}(p, g) = \text{dur}(a) = d \).

**Overview**

Given \( \mathcal{P} = \langle P, O, I, G, D \rangle \), our goal is to obtain a plan that achieves \( G \) from \( I \), while satisfying the constraints in \( D \) (including the plan deadline \( T_{H} \) ). TempLM consists of two components: knowledge compilation and search.

**Knowledge Compilation.** We distinguish two steps in this component. In the first step, we generate a graph that represents a plan sketch to a satisfactory solution. Initially,

<table>
<thead>
<tr>
<th>Condition</th>
<th>Effect</th>
<th>( dis_{E}^{c} )</th>
<th>( dis_{I}^{c} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCond</td>
<td>SAdd</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
</tr>
<tr>
<td>SCond</td>
<td>EAdd</td>
<td>( \text{dur}(a) )</td>
<td>( \text{dur}(a) )</td>
</tr>
<tr>
<td>Inv</td>
<td>SAdd</td>
<td>( \varepsilon )</td>
<td>( -\text{dur}(a) )</td>
</tr>
<tr>
<td>Inv</td>
<td>EAdd</td>
<td>( \text{dur}(a) )</td>
<td>( \varepsilon )</td>
</tr>
<tr>
<td>ECond</td>
<td>SAdd</td>
<td>( -\text{dur}(a) )</td>
<td>( \varepsilon )</td>
</tr>
<tr>
<td>ECond</td>
<td>EAdd</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
</tr>
</tbody>
</table>

1\( \varepsilon \) is used at PDDL2.1 level 3 to express the duration of an instantaneous action, an amount so small that it makes no sense to split it (Garrido, Fox, and Long 2002).

2\The definition of deadlines could be generalized for any other proposition, it is not exclusive of goal propositions.

3This definition states that, if a goal is achieved more than once in the plan, it suffices one appearance to fulfill the constraint. However, abusing from this notation, we consider that this constraint, when applied to a goal \( g \), refers to the last appearance of \( g \). Operator hold-after does not imply persistence. Persistence would be expressed with \( (\text{within deadline (always goal)}) \), which requires nesting of the modalities and this is not allowed in standard PDDL3.0 syntax. (Derek Long, personal communication).
this graph is constructed by extracting the (non-temporal) landmarks of the problem (Hoffmann, Porteous, and Sebastia 2004). Next, new (temporal) landmarks derived from the deadline constraints are added to the LG. In the second step, landmarks are annotated with three temporal intervals (Marzal, Sebastia, and Onaindia 2008), which denote the validity of the temporal proposition. At this step, we also analyze other constraints involving temporal landmarks that will determine the consistency of the graph and thus the solvability of the problem.

Search. We apply a search algorithm guided by a heuristic that exploits the information of the landmarks together with the temporal and causal knowledge compiled in the landmarks graph. The information comprised in the graph will allow us to prune inconsistent sub-plans.

It is important to highlight that the output of the knowledge compilation phase is a LG comprising information that any solution plan of a problem should contain. In particular, the non-temporal landmarks of the graph are the propositions that must be true in any solution plan, and the temporal landmarks are the propositions that must appear in the plan in order to satisfy the deadline constraints. For both types of landmarks, their temporal intervals show when these propositions must be achieved in the plan. Consequently, if a sub-plan not compliant (or consistent) with the information contained in the LG is encountered during the search phase then the search node that represents such a plan will be pruned.

Knowledge compilation: Landmarks extraction

In this section, we detail the first part of our knowledge compilation process, the extraction of non-temporal and temporal landmarks.

Non-temporal landmarks

In a non-temporal setting, a landmark (Hoffmann, Porteous, and Sebastia 2004) is defined as a proposition that must be true at some point during the execution of any solution plan. The process for the extraction of non-temporal landmarks that we use in this paper was introduced in (Marzal, Sebastia, and Onaindia 2011). It is a combination of several existing methods that returns more landmarks than any other known technique. The key concepts of this approach are the first achievers and dependency labels. Let \( \Gamma^d \) be the set of relaxed plans\(^4\) that reach a proposition \( p \) at a time \( t \). The first achievers of \( p \) at \( t \) are defined as follows:

\[
fa^d(p) = \{ a \in \Gamma^d_t : p \in AddEff(a) \}
\]

The idea behind the first achievers is that this set contains all of the actions that may achieve \( p \) at time \( t \). Additionally, the set of dependency labels for each proposition \( p \) at a time point \( t \), denoted by \( labels^d(p) \), is defined as follows:

\[
labels^d(p) = p \cup \left( \bigcap_{a \in fa^d(p)} \left( \bigcup_{c} \text{Cond}(a) \right) \right)
\]

where \( C = \bigcup_{a \in fa^d(p)} \text{Cond}(a) \)

The labels keep information about the propositions that must necessarily be true in the plan before \( p \) becomes true at \( t \). These labels are computed forwards from the initial state.

The definition of a landmark derives from the concept of first achievers and dependency labels (when the duration of all actions is one time unit):

**Definition 4** A landmark \( l \) is a proposition that belongs to either one of the following sets:

1. the intersection of the conditions of the first achievers of all goals \( g \in G \) at \( t_g \):
   \[
   \bigcap_{a \in fa^d(g)} \forall a \in fa^d(g).
   \]
   The set of landmarks that belong to this set become in turn subgoals of the problem and the process is repeated again.
   This iterative process allows to derive new landmarks.
2. the set of dependency labels of all goals \( g \in G \) at \( t_g \), that is, the set formed with labels\(^d\)(\( g \)) for every goal \( g \).

Landmarks can be (partially) ordered according to the order in which they must be achieved. This information will be later used during the search to find a skeleton of the solution plan. The method in (Hoffmann, Porteous, and Sebastia 2004) obtains a set of necessary orderings of the form \( l \prec_a l' \) during the landmarks extraction, denoting that for achieving \( l' \), \( l \) must be true in the intermediate preceding state. Additionally, a set of dependency orderings of the form \( l \prec_{d} l' \) is computed by a process inspired by the works described in (Porteous and Cresswell 2002) and (Richter, Helmert, and Westphal 2008). Generally speaking, a dependency ordering indicates that \( l \) must be achieved before \( l' \), but not necessarily in the state immediately prior to the state in which \( l' \) is achieved. The set of landmarks along with the necessary and dependency orderings between them define a landmarks graph (LG).

Temporal landmarks

Let \( \Pi^P \) be the set of all the solution plans for a planning problem \( P \) without deadline constraints. By definition, the intersection of the propositions achieved along the execution of the plans in \( \Pi^P \) define the set of non-temporal landmarks. Let’s assume that \( P' \) is a new problem where some deadline constraints, \( D \), are defined in \( P \). Then, we have that \( \Pi^P \subseteq \Pi^P \). That is, due to the existence of \( D \), the number of solution plans for \( P' \) can be smaller and, consequently, more landmarks can be found in the set \( \Pi^P \). This set of new landmarks are called temporal landmarks.

Informally speaking, a temporal landmark is a proposition that must be achieved in any solution plan that satisfies \( D \). The set of non-temporal landmarks extracted are also considered as temporal landmarks. Moreover, we need to establish when a temporal landmark must be achieved in order to satisfy \( D \). We denote by \( [min_d(l), max_d(l)] \) the generation interval of a temporal landmark, where:

- \( min_d(l) \) is the earliest time point when landmark \( l \) can start in the plan. This value is determined by the time of the first fact layer when \( l \) appears in a TRPG (Temporal Relaxed Planning Graph) (Coles et al. 2008).

---

\(^4\)This set can be calculated by means of a relaxed planning graph (Hoffmann and Nebel 2001).
• $\text{max}_g(l)$ represents the latest time point when $l$ must start in order to satisfy $D$. The propositions involved in a deadline constraint are initially assigned a $\text{max}_g$ value. Eventually, the $\text{max}_g$ value is propagated to other landmarks across the LG.

Given a set of deadline constraints $D$, where $(l, t) \in D$, in a first step of the process for extracting temporal landmarks, $l$ is added as a temporal landmark and $\text{max}_g$ is set to be prior or equal to $t$: $\text{max}_g(l) \leq t$.

The second step of the temporal landmarks extraction exploits the idea that the existence of a deadline constraint in a planning problem rules out the applicability of some actions due to the time restrictions, as explained at the beginning of this section. Given a proposition $l$ in the LG, more landmarks can be easily obtained by consulting the propositions that $l$ has some dependencies with at time $\text{max}_g(l)$, which is the latest time for $l$ to appear in the plan. That is, the propositions included in $\text{labels}_{\text{max}_g(l)}(l)$ are considered as new temporal landmarks. This second step is repeated while new temporal landmarks are found.

Once the set of temporal landmarks is extracted, we must set the appropriate orders. In particular, let $l_i$ be a landmark in the graph, $l$ a temporal landmark to be inserted in the graph, and $A_l$ the set of actions that achieve $l$ at $t$ where, if $\exists(t, t') \in D$, then $t = t'$; otherwise, $t = T_l$. There is a necessary ordering between $l_i$ and $l$ ($l_i \prec_l l$) if $l_i \in \bigcap \text{Cond}(A_l)$. Likewise, there is a dependency ordering between $l_i$ and $l$ ($l_i \prec_d l$) if $l_i \in \text{labels}^l(l)$. The definition of orders between landmarks may cause the appearance of cycles in the graph, which indicate that a landmark must be achieved more than once in the plan. In this case, a cycle is broken by adding another instance of the same proposition, as explained in (Hoffmann, Porteous, and Sebastia 2004).

**Knowledge compilation: Propagation of temporal information**

Once we have the LG composed of both temporal and non-temporal landmarks, we know which information (propositions) must necessarily appear in any solution plan that satisfies the deadline constraints. We also know the last time point when some temporal landmarks (actually, those that appear in $D$) must be achieved. The second step of the knowledge compilation is to propagate this information across the LG, in order to progressively refine the intervals that define a landmark. This will allow us to derive new temporal landmarks by applying the second step of the temporal landmarks extraction over the $\text{max}_g$ time points that will be updated during the propagation.

Let’s recall that $\text{start}(l)$ and $\text{end}(l)$ denote the time points of the temporal occurrence of a landmark $l$ in a solution plan. Besides the generation interval, we define the validity interval $(\text{min}_v(l), \text{max}_v(l))$: it is the maximum interval of validity of landmark $l$ in the plan. That is, $\text{min}_v(l) \leq \text{start}(l) \leq \text{end}(l) \leq \text{max}_v(l)$. Initially, $\text{min}_v(l) = \text{min}_g(l)$ and $\text{max}_v(l) = T_l$; these values will be updated when the information in the graph is propagated.

### Updating the generation and validity intervals

**Necessary and dependency orderings.** A causal relationship between two landmarks $l_i$ and $l_j$, such that $l_i \prec_{(d,n)} l_j$, implicitly establishes some temporal constraints between the endpoints of the intervals. For example, let $(g, t)$ be a deadline constraint in $D$; we know that $\text{max}_g(g) = t$. Let’s assume that an action $a$ has a duration of $d_i$; adds $g$ at the end and $p$ is a start condition. Then, given that $p \prec_n g$, we can infer that $\text{max}_g(p) \leq \text{max}_g(g) - d$ and that $\text{min}_v(g) \geq \text{min}_v(p) + d$. In general, we can establish that, if $l_i \prec_n l_j$, then (see Table 1):

$$\begin{align*}
\text{max}_g(l_i) &\leq \text{max}_g(l_j) - \text{dis}_E(l_i, l_j) \\
\text{min}_v(l_j) &\geq \text{min}_v(l_i) + \text{dis}_E(l_i, l_j)
\end{align*}$$

A dependency relationship between two landmarks $l_i$ and $l_j$, $l_i \prec_c l_j$, actually means there exists a sequence of actions to reach $l_j$ from $l_i$, not just a single action. In this case, we define $\text{DIS}_E(l_i, l_j)$ and $\text{DIS}_L(l_i, l_j)$ as a generalization of $\text{dis}_E$ and $\text{dis}_L$, which are recursively computed as the minimum distance between all the propositions in the path from a state that contains $l_i$ to a state that contains $l_j$ (Marzl, Sebastia, and Onaindia 2008). Particularly, these values are computed as:

$$\text{DIS}_{(E,L)}(l_i, l_j) = \min(\text{DIS}_{E,L}(l_i, l) + \text{dis}_{(E,L)}(l, l_j))$$

where $l$ is a condition of some action $a$ and $l_j$ is an $AddEff$ of $a$. That is, $l$ is a proposition in the path between $l_i$ and $l_j$. In case of $l_i = l_j$ then $\text{DIS}_{(E,L)}(l_i, l_j) = 0$.

From these distances, we can establish some relationships between the endpoints of the intervals of a pair of landmarks when $l_i \prec_{(d,n)} l_j$. These relationships, called propagation constraints, are propagated across the LG every time the interval of a landmark is modified:

$$\begin{align*}
\text{max}_g(l_i) &\leq \text{max}_g(l_j) - \text{DIS}_E(l_i, l_j) \\
\text{min}_v(l_j) &\geq \min_v(l_i) + \text{DIS}_E(l_i, l_j)
\end{align*}$$

**Mutex relationships.** If two landmarks $l_i$ and $l_j$ are mutex (Blum and Furst 1997), this means that they will not coexist in any state of a solution plan. Therefore, we can use this information for updating the validity intervals in the LG. If $l_i$ and $l_j$ are mutex at time $t$ and there exists a causal relationship between both landmarks of the form $l_i \prec_{(d,n)} l_j$, we set:

$$\text{max}_v(l_i) = \min(\text{max}_v(l_i), \text{max}_g(l_j) - \text{DIS}_L(l_i, l_j))$$

This way, the end time of the validity interval of $l_i$ is set equal to the latest time point when $l_j$ must be true ($\text{max}_g(l_j)$) minus the minimum distance between them. Moreover, the constraint $\text{max}_v(l_i) \leq \text{min}_v(l_j)$ is also added to the set of propagation constraints. This information will be then propagated to the rest of the graph.

**Necessity interval**

Along with the validity and generation intervals, we define the necessity interval $(\text{min}_n(l), \text{max}_n(l))$, which represents the set of time points when $l$ is required as a condition for an action to achieve other landmarks. That is, $\text{start}(l) \leq \text{min}_n(l) \leq \text{max}_n(l) \leq \text{end}(l)$.

In this case, the actual duration of the actions is considered.
The necessity interval of a given landmark \( l_i \) is computed by taking into account all the necessary orderings in which \( l_i \) is needed as condition to generate another landmark:

\[
\begin{align*}
\min_n(l_i) &= \min_{\forall l_j : l_i \prec_n l_j} (\min_v(l_j) - \text{DIS}^E(l_i, l_j)) \\
\max_n(l_i) &= \max_{\forall l_j : l_i \prec_n l_j} (\max_v(l_j) - \text{DIS}^L(l_i, l_j))
\end{align*}
\]

That is, the (resp. end) time of the necessity interval of \( l_i \) is set to the earliest (resp. latest) time point in which \( l_i \) is necessary to ensure the validity (resp. generation) of every temporal landmark \( l_j \) for which there exists a necessary ordering with \( l_i \).

If the end point of validity interval of a landmark \( l_i \) is updated due to a mutex relationship, the end time point of the necessity interval is updated accordingly: \( \max_n(l_i) = \min(\max_v(l_i), \max_n(l_i)) \).

**Interval Consistency**

Figure 1 shows the relationships between the temporal occurrence of \( l \) in the plan and its temporal intervals. It also shows the relationships between the endpoints of the intervals of \( l \) (Marzial, Sebita, and Onaindia 2008):

\[
\begin{align*}
\min_n(l) &\leq \min_v(l) \leq \min_n(l) \\
\min_n(l) &\leq \max_v(l) \leq \max_n(l) \\
\max_n(l) &\leq \max_v(l)
\end{align*}
\]

The relationship \( \min_n(l) \leq \min_v(l) \) always holds, given that \( \min_n(l) \) is the earliest time point when \( l \) can start in the plan, determined by the time of the first fact layer when \( l \) appears in a TRPG. Although \( \min_v(l) \) is initially set to \( \min_n(l) \), it can move forward in time due to the relationships with other landmarks.

The inclusion and propagation of the deadline constraints and the causal relationships in the LG causes a modification in the endpoints of the intervals, as explained in the previous subsections. If one of the relationships above is violated during the construction of the LG, it means that the problem is unsolvable, in which case TempLM will report that no solution plan is found. For example, this is the case when \( \min_v(l) > \min_n(l) \), which implies that \( l \) is needed before it is generated. Finding these inconsistencies allow us to promptly detect that the constraints defined in the problem lead to an unsolvable problem, even before starting the search. This will be shown in the experiments section.

Finally, the LG can be viewed as a Simple Temporal Network (Dechter, Meiri, and Pearl 1991), where the nodes are the time points of the intervals. Additionally, note that, as in constrained-based planning (Frank and Jonsson 2003), enforcing the interval consistency relationships allows to exploit the landmarks properties in strongly constrained problems thus facilitating an early detection of unsolvability.

**Example**

This section introduces an example on the driverlog domain (Long and Fox 2003) to show the construction of the LG and the calculation of the landmarks intervals. The initial state is depicted in Figure 2, along with the goal deadlines. Figure 3 shows the LG obtained for this problem (after the propagation process). In this case, only the propositions from the initial and goal state are non-temporal landmarks, given that there are two trucks for transporting the packages. The deadline constraints impose that \( \max_g((P2 S3)) = 50 \), so only the actions that achieve \((P2 S3)\) at 50 are considered. Therefore, the propositions in bold ((driving D1 T1), (at T1 S2), (in P2 T1)) and (at T1 S3)), which belong to labels\(^50\)(at P2 S3), are temporal landmarks.

Once the landmarks are inserted in the LG, the necessity and dependency orderings are established. For example, \((T1 T3) \prec_n (P2 S3)\), because the only action that adds \((P2 S3)\) at 50, \( a \ = \text{UNLOAD P2 S3 T1} \), with \( \text{dur}(a) = 2 \), has \( (T1 S3) \) as a start condition. Moreover, \( \max_g (T1 S3) \) is set to \( 50-2=48 \), given that \( \text{DIS}^E \) is 2 in this case. Similarly, as Figure 3 shows, there is a dependency order between \((T1 S2)\) and \((T1 S3)\) because there are several paths to achieve \((T1 S3)\) from \( (T1 S2) \), so a necessary order cannot be established. This dependency order determines that \( \max_g (T1 S2) = 48 - 20 = 28 \), as Equation 1 states. In this case, \( \text{DIS}^E \) is 20, which represents the shortest distance between \((T1 S2)\) and \((T1 S3)\), as Figure 2 shows. From all these orders, the start time point of the validity intervals is also updated. For example, following Equation 2, \( \min_v (T1 S3) = 24 + 20 = 44 \). Thus, \( \min_n (T1 S3) \) is greater than its \( \min_g \) value, meaning that \((T1 S3)\) cannot be actually reached before 44.

Given that \((T1 S2)\) and \((T1 S3)\) are mutex and there is a dependency order between them, we update the end
point of the validity interval of (at T1 S2), which is initially set to \( T_{II} = 200 \), as Equation 3 states: \( \max_{n}(at \ T1 \ S2) = \min(200, 48 - 20) = 28 \). As Figure 3 indicates, (at T1 S2) will only be valid in the plan during the interval [24, 28]. There are some intervals that cannot be updated, like the validity interval of (driving D1 T1), because, in this case, there is no action in a solution plan that satisfies the goal deadlines that deletes this proposition.

Let’s now show how an inconsistency in the graph is found. What would happen if the deadline of (at P2 S3) were 40 time units instead of 50? Its \( \min_{n} \) would take the same value shown in Figure 3 (46) because the value of \( \min_{n} \) is propagated from the initial state to the goals. Then, the validity interval of (at P2 S3) would be [46, 200] and its generation interval would be [35, 40]. Obviously, there is an inconsistency between these two intervals because \( \max_{n}(at \ P2 \ S3) \) is not contained within the validity interval. It is important to note that this inconsistency would not be found in the TRPG expansion; the first time layer at which (at P2 S3) appears is 35 (\( \min_{g} \)). Therefore, any deadline greater than 35 would be found as satisfiable by the TRPG, whereas our LG would identify any deadline lower than 46 (\( \min_{n} \)) as unsolvable.

**Searching for a solution plan**

In this section, we summarize some aspects of the search process. We apply a search process in the space of partial plans. A node in the search tree is denoted by a pair \((\Pi, S_{t})\), where \( t = \text{dur}(\Pi) \). \( \Pi \) represents a conflict-free partial plan and \( S_{t} \) represents the state reached (at time \( t \)) after the execution of \( \Pi \) from \( I \). Search is guided by: (1) the temporal and causal knowledge compiled in the LG, used to prune the plans that are not compliant with the information in the graph and (2) a (non-admissible) heuristic based on action landmarks for sub-optimal temporal planning.

**Evaluation function**

The search process is guided by a standard evaluation function \( f(n) = g(n) + h(n) \). In temporal planning problems, the parallelism of the plans must be taken into account to obtain an accurate \( g(n) \). Therefore, given a node \( n = (\Pi, S_{t}) \), we define \( g(n) = \text{dur}(\Pi) \). This way, the \( g \)-value of a node accurately reflects the cost of its plan.

The heuristic used in our algorithm is a temporal approximation of the well-known landmark cut heuristic \( h^{LM-cut} \) (LM-cut for short). This is an admissible heuristic based on delete relaxation that provides one of the best-known poly-time approximations of \( h^{+} \) (Helmert and Domshlak 2009). LM-cut has been successfully used in optimal sequential planning ((Helmert and Domshlak 2009), (Domshlak, Katz, and Lefler 2010), (Bonet and Helmert 2010), (Bonet and Castillo 2011)). Generally speaking, LM-cut computes a set of groups of actions, called cuts, each of them corresponding to an action landmark of the planning problem \( P \); without at least one action of each cut, \( P \) cannot be solved. Thus, the value of the LM-cut estimate is the sum of the cost of all the cuts; i.e. the number of cuts.

When adapting the LM-cut heuristic to a temporal context, the cost of an action must be its duration. Therefore, the heuristic value is an estimate based on the makespan of a sequential plan. This value is clearly an overestimate of the actual plan makespan as it ignores the possible overlapping of the actions. The good results exhibited by the LM-cut in propositional planning led us to adopt the LM-cut heuristic as an approximation for sub-optimal temporal planning. As for further work, we intend to test admissible heuristics for plan makespan, as those introduced in (Haslum 2009).

**Search process**

The search starts with the basic node \((\emptyset, I)\). As the search goes on, new nodes are created in the search tree as a result of the application of an action in the parent node. Given a node \((\Pi, S_{t})\), the whole set of applicable actions in \( S_{t}, O_{t}, \)
Definition 5 Given a node $(\Pi, S_t)$ and an action $a_i \in O_t$, the earliest start time of $a_i$ (denoted by $t_{i}^{\Pi, t}$) is the first time point from $I$ where $a_i$ is applicable and does not interfere with any other action in $\Pi$. Action $a_i$ is said to interfere with the action $(a_j, t_{a_j}) \in \Pi$ if any of the following situations holds:

- $\exists p \in Inv(a_j)$ and $a_i$ deletes $p$ within the execution interval of $a_j$
- $\exists q \in SCond(a_j)$ and $a_i$ deletes $q$ between the time point when $q$ is produced and $t_{a_j}$
- $\exists r \in ECond(a_j)$ and $a_i$ deletes $r$ between the time point when $r$ is produced and $t_{a_j} + dur(a_j)$

In other words, if $a_i$ interferes with $a_j$ (already in $\Pi$), then $a_j$ is not introduced in $\Pi$. However, both actions (joint with a third action $a_k$ that re-achieves $p$, $q$ or $r$ (resp.)) can be found in a plan of another branch of the search tree if actions are inserted in this order: $a_i, a_k, a_j$.

The node resulting from applying $a_i$ in $S_t$ is denoted by $(\Pi', S_{t'}_r)$, where $\Pi' = \Pi \cup (a_i, t_{i}^{\Pi, t})$ and $t' = dur(\Pi')$. $S_{t'}$ is computed in two steps:

1. $S_{t'}^{I} = S_t - SDel(a_i) \cup SAdd(a_i)$
2. $S_{t'}^{V} = S_{t'}^{I} - EDel(a_i) \cup EAdd(a_i)$

The definition of $S_{t'}$ is only expressed in terms of the action $a_i$ to be inserted in $\Pi$ but, actually, all the actions concurrent with $a_i$ are considered in the calculation of $S_{t'}$. The next step is to discard, among the nodes resulting from the applicable actions $O_t$, those that are not compliant with the information in the LG. A search node is ruled out if one of these two situations holds:

1. Temporal intervals: If the start($l$) and/or end($l$) of a landmark $l$ in a plan $\Pi$ is not consistent with its validity and/or generation interval, then $\Pi$ is pruned. Thus, the corresponding node is discarded if any of the following conditions holds:
   (a) $start(l) < min_v(l)$
   (b) $start(l) > max_v(l)$
   (c) $end(l) > max_v(l)$ (only applicable if end($l$) is known)

2. Causal relationships: Given a pair of landmarks $l_i$ and $l_j$ in a plan $\Pi$, such that $l_i \prec_{(d,n)} l_j$, if start($l_i$) $\geq$ start($l_j$), $\Pi$ is pruned.

TempLM inherits the foundations of landmarks and temporal reasoning for confirming the logical and temporal consistency of plans. Note that nodes in the tree represent valid partial plans so any state for which the constraints expressed in (1) or (2) cannot be satisfied is immediately pruned.

<table>
<thead>
<tr>
<th></th>
<th>TempLM-noLG</th>
<th>TempLM</th>
<th>OPTIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MakeSp</td>
<td>Time</td>
<td>MakeSp</td>
</tr>
<tr>
<td>Satellite</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PW1*</td>
<td>176.69</td>
<td>0.4</td>
<td>176.69</td>
</tr>
<tr>
<td>PW2*</td>
<td>176.52</td>
<td>1.68</td>
<td>176.52</td>
</tr>
<tr>
<td>PW3*</td>
<td>106.77</td>
<td>0.64</td>
<td>106.77</td>
</tr>
<tr>
<td>PW4*</td>
<td>171.85</td>
<td>7.37</td>
<td>171.85</td>
</tr>
<tr>
<td>PW5*</td>
<td>180.46</td>
<td>41.86</td>
<td>180.46</td>
</tr>
<tr>
<td>PW7*</td>
<td>- T.E.</td>
<td>134.38</td>
<td>1467.77</td>
</tr>
<tr>
<td>PW8*</td>
<td>- T.E.</td>
<td>- T.E.</td>
<td>119.94</td>
</tr>
<tr>
<td>PW9*</td>
<td>- T.E.</td>
<td>18.99</td>
<td>15.67</td>
</tr>
</tbody>
</table>

Table 2: Results for the domains from IPC4 (time in secs.).

Experiments

In this section, we present the experiments we performed to compare TempLM with OPTIC. As we were also interested in the impact of the knowledge compilation component of our approach, we created two configurations: one version which only uses the search stage, without the LG (TempLM-noLG), and a second one which corresponds to TempLM as described in the previous sections. We selected 5 domains from the International Planning Competitions (IPC):

- Two domains from the IPC4, Satellite and PipesWorld, with deadlines encoded by means of TIL (Table 2, group Satellite/PipesWorld-IPC4). We also generated some additional problems with non-reachable deadlines (Table 2, group Satellite/PipesWorld-Unsolv., indicated with a *).
- The Trucks domain from the IPC6, with deadlines encoded by means of the modal operator within (Table 3, group Trucks-IPC6). We also generated some additional problems with tighter time deadlines (Table 3, group Trucks-Tight, indicated with a *).
- Two domains from the IPC3, DriverLog and ZenoTravel. As goal deadlines were not defined in IPC3, for each original problem, we generated three versions with a different value of $T_{i1}$, obtained by constraining the makespan of
the solutions w.r.t. the makespan of the best solution plan found by LPG (Gerevini, Saetti, and Serina 2003) (indicated with a *). Then, loosely-constrained problems were generated by increasing the makespan of the best LPG solutions a percentage between 20-40, tightly-constrained problems by increasing the makespan between 0 and 20 percent of the best LPG solution by establishing non-reachable deadlines.

We selected these domains because they fit our requirements: deadlines expressed by means of TIL or within, with no metric features or ADL. Tables 2, 3 and 4 show the results for all problems where at least one planner found a solution. We show the makespan and the time in seconds needed for solving it with OPTIC and both configurations of TempLM. As for unsolvable problems, we present the time needed to detect it. We restricted the execution time to 30 minutes ("T.E." stands for "Time Exhausted"). The experiments were performed in an Intel-Core-i5-3.2-GHz with 16GB-RAM.

In general, OPTIC solved fewer problems compared to both configurations, TempLM-noLG and TempLM. This is specially remarkable in unsolvable problems, where OPTIC could only detect 7 out of 20 whereas TempLM detected all of them. The most costly unsolvable instance took 0.33 seconds in OPTIC; this is likely to be the case that at least one of the goal deadlines were not reached before their deadlines in the TRPG. In these cases, TempLM was as fast as OPTIC. For the unsolvable problems which OPTIC returned T.E., TempLM solved a few instances very rapidly, because it detected an inconsistency in the landmarks intervals before even starting the search. As shown in the explanatory example, the earliest times of propositions recorded in the LG are more accurate than in the TRPG. For the rest of unsolvable problems in which TempLM took longer, unsolvability was detected during the search. We also highlight that TempLM-noLG found 9 unsolvable problems, an indication that the search process is sufficiently efficient to rapidly explore the search tree. In summary, the high number of unsolvable problems identified by TempLM relies on the usefulness of the LG to check consistency of the landmarks intervals as well as to discard the search nodes not compliant with the graph.

Table 3: Results for the domains from IPC6 (time in secs.).

<table>
<thead>
<tr>
<th>Trucks</th>
<th>TempLM-noLG</th>
<th>TempLM</th>
<th>OPTIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF1</td>
<td>843.2</td>
<td>0.03</td>
<td>843.2</td>
</tr>
<tr>
<td>PF2</td>
<td>1711.4</td>
<td>1.33</td>
<td>1711.4</td>
</tr>
<tr>
<td>PF3</td>
<td>1470.1</td>
<td>4.74</td>
<td>1470.1</td>
</tr>
<tr>
<td>PF4</td>
<td>2629.4</td>
<td>24.84</td>
<td>2629.4</td>
</tr>
<tr>
<td>PF5</td>
<td>1671.6</td>
<td>321.97</td>
<td>1671.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trucks</th>
<th>TempLM-noLG</th>
<th>TempLM</th>
<th>OPTIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF1</td>
<td>843.2</td>
<td>0.03</td>
<td>843.2</td>
</tr>
<tr>
<td>PF2</td>
<td>1711.4</td>
<td>1.33</td>
<td>1711.4</td>
</tr>
<tr>
<td>PF3</td>
<td>1470.1</td>
<td>4.74</td>
<td>1470.1</td>
</tr>
<tr>
<td>PF4</td>
<td>2629.4</td>
<td>24.84</td>
<td>2629.4</td>
</tr>
<tr>
<td>PF5</td>
<td>1671.6</td>
<td>321.97</td>
<td>1671.6</td>
</tr>
</tbody>
</table>

Table 4: Results for the domains from IPC3 (time in secs.).

<table>
<thead>
<tr>
<th>Zeno</th>
<th>TempLM-noLG</th>
<th>TempLM</th>
<th>OPTIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1U*</td>
<td>-</td>
<td>0.51</td>
<td>-</td>
</tr>
<tr>
<td>Z2U*</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Z3U*</td>
<td>-</td>
<td>T.E.</td>
<td>-</td>
</tr>
<tr>
<td>Z4U*</td>
<td>-</td>
<td>T.E.</td>
<td>-</td>
</tr>
<tr>
<td>Z5U*</td>
<td>-</td>
<td>T.E.</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Zeno</th>
<th>TempLM-noLG</th>
<th>TempLM</th>
<th>OPTIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1T*</td>
<td>592</td>
<td>0.25</td>
<td>592</td>
</tr>
<tr>
<td>Z2T*</td>
<td>592</td>
<td>0.13</td>
<td>592</td>
</tr>
<tr>
<td>Z3T*</td>
<td>393</td>
<td>8.31</td>
<td>393</td>
</tr>
<tr>
<td>Z4T*</td>
<td>542</td>
<td>16.86</td>
<td>542</td>
</tr>
<tr>
<td>Z5T*</td>
<td>529</td>
<td>63.51</td>
<td>522</td>
</tr>
<tr>
<td>Z6T*</td>
<td>323</td>
<td>32.84</td>
<td>320</td>
</tr>
</tbody>
</table>

With respect to the original problems from the IPC4 and IPC6 in Tables 2 and 3, in general, TempLM obtains plans with a shorter makespan in less time. In the Trucks domain, the makespan of OPTIC plans is rather larger. This is, though, mitigated in the Tight problems because the existing deadline constraints lead the planner to find plans of better makespan but at the expense of a worse performance. In contrast, one of the key benefits of TempLM is that the more constrained the problem is, the more efficient the search.

Regarding the problems from the IPC3, OPTIC showed to have more difficulties in problems with tight deadlines, being unable to solve 6 of these problems. In the Driverlog and ZenoTravel domains, TempLM outperformed OPTIC both in terms of number of problems solved and makespan.

TempLM also shows better performance than TempLM-noLG. In most of the problems, both obtain plans of very similar makespan, but TempLM requires less time. Here, the LG shows to be very beneficial for discovering unsolvable problems and pruning the search space. We must also no-
tice that TempLM shows significantly higher times in some problems like PF7 of Satellite or PF5 of Trucks. These are loosely-constrained problems, which are the cases in which the exploitation of landmarks reveals less effective.

We can conclude that the knowledge compilation component of TempLM turns out to be very helpful to discover unsolvable problems and obtain an efficient temporal planning system. Additionally, our approach works, in general, better than OPTIC in problems with deadline constraints.

**Conclusions**

We have presented TempLM, a domain-independent temporal planning model to solve problems with deadline constraints. The key contribution of our work is the design and incorporation of a knowledge compilation component into the planning system. TempLM puts the emphasis on representing, modeling and exploiting the temporal knowledge of the problem rather than simply delegating resolution to a search process. We have empirically demonstrated that TempLM is capable to solve more problems with deadline constraints than the planner OPTIC. We have also shown that the use of the LG helps reduce the search space and allows for a promptly detection of unsolvable problems.

As for further work, we are developing a method for using information from the valid sub-plans as a feedback to the LG, which in turn may give rise to new helpful information for solving the problem.

**Acknowledgements**

We thank anonymous reviewers for their very useful comments and suggestions and Derek Long for solving our doubts about the modal operators in PDDL3. This work has been partially supported by Consolider Ingenio 2010 CSD2007-00022 and Spanish Government Project MICINN TIN2011-27652-C03-01 and Valencian Government Project Prometeo II/2013/019.

**References**


