# Temporal Landmark Graphs for Solving Overconstrained Planning Problems

Eliseo Marzal[a], Laura Sebastia[a,*], Eva Onaindia[a]

[a] *Universitat Politècnica de València, Valencia, 46022, Spain*

**Abstract**

This paper presents TempLM, a novel approach for handling temporal planning problems with deadlines. The proposal revolves around the concept of temporal landmark, a proposition that must be necessarily true in all solution plans to achieve the problem goals within their deadlines. The temporal landmarks extracted from the problem form a landmarks graph where nodes are landmarks and edges represent temporal as well as causal relationships between landmarks. The graph comprises information about which propositions and when these propositions must be achieved in a solution plan, information that is later used to guide the search process as well as reduce the search space. Thus, the partial plans of the search tree that are not compliant with the information comprised in this graph are pruned. We present an exhaustive experimentation evaluation in overconstrained and unsolvable problems and we compare the performance of TempLM with other state-of-the-art planners. The results will show the efficiency of TempLM in the detection of unsolvable problems.

*Keywords:* Automated planning, Temporal planning, Deadlines, Landmarks

## 1. Introduction

Planning is the art of building algorithms that synthesize a course of action, or plan, that satisfies a set of goals from an initial situation. The aim of automated planning algorithms is to select the actions and the order in which these actions should be taken to achieve the problem goals. In temporal settings, the existence of durative actions (actions with durations) puts the emphasis on which actions to select and how to organize such actions so as to obtain the optimal makespan plan (plan of minimal duration). Temporal planning problems are harder to solve due to the temporal relationships between the action intervals, augmenting the choices of ordering, separation or overlapping of the actions.

In many real-world temporal planning problems that present, additionally, deadline constraints, the optimal makespan plan does not necessarily have to be compliant with the

*Corresponding author

*Email addresses:* emarzal@dsic.upv.es (Eliseo Marzal), lstarin@dsic.upv.es (Laura Sebastia), onaindia@dsic.upv.es (Eva Onaindia)

achievement time of the individual goals. Goal deadlines frequently occur in constraint-based planning, manufacturing operations in supply-chain activities, delivery of goods or work flow-based systems. In general, handling deadline constraints in temporal planning has not been much exploited. Finding the proper actions along with their correct temporal occurrences to meet the problem deadlines requires integrating the planning machinery to select the actions with temporal reasoning methods to satisfy the deadlines. Consider, for example, a real-world application like the regional transportation of fish and seafood by distributor companies. The objective is to daily deliver fresh fish, frozen products or long-term preserving fish products to supermarkets and restaurants. The planner will need to choose between ice-cooled or machine-cooled wagons for delivering goods according to the range of temperatures each wagon is suited for and the freshness expiration of the goods. Besides these timing restrictions, the planner must also select the appropriate transport route in terms of the trip duration and the maximum allowed time that a product can be onto a wagon under a certain temperature or outside a wagon in case of unloading and reloading (Sen (2010)). This real-world application highlights the need to make appropriate choices about the wagon and the route to use for the fish transportation as well as about the temporal organization of the actions.

Deadline constraints have been widely studied in time-constrained problems within the field of Constraint Satisfaction Problem (CSP) (Rossi et al. (2006)). In these problems, the objective is to find a schedule for a given set of actions in order to satisfy the deadlines. In contrast, in temporal planning, the problem also involves selecting the right actions to satisfy the constraints, which introduces a higher level of difficulty that is emphasized when dealing with overconstrained problems or unsolvable problems. In these cases, it is crucial to appropriately exploit the information of the planning problem in order to promptly detect and discard the actions that are not compliant with the deadline constrains.

Within the planning community, the first attempt to handle planning problems with deadline constraints was motivated by the introduction of the PDDL3.0 (Planning Domain Definition Language) language in the Fifth International Planning Competition (IPC-2006)[1] (Gerevini et al. (2009)). PDDL3.0 includes the operator *within* to express deadlines as well as some specific operators such as *sometime-before*, *sometime-after* and *always-within*. The two planners that took part in the constraints domain track in the IPC-2006, namely MIPS-XXL (Edelkamp et al. (2006)) and SGPLAN5 (Chen et al. (2006)), did not exhibit a good performance or complete accuracy in avoiding deadline violations. No further advances were done in temporal planning with deadline constraints, with the notable exception of OPTIC (Benton et al. (2012)), a planner which outperforms MIPS-XXL and SGPLAN5 on the benchmark of temporal planning problems with soft-constraints and which also handles deadlines expressed with the *within* operator.

This paper contributes with a novel approach for solving deadline overconstrained planning problems. Our planner, TempLM, revolves around the notion of temporal landmark, thus extending the definition of a fact which must be true in every solution plan (landmark) (Hoffmann et al. (2004)) to a temporal context. This paper presents the evolution of TempLM, which started with the theoretical design of a CSP-based consistency checker to detect unsolvability in planning problems with deadlines (Marzal et al.

[1] http://www.icaps-conference.org/index.php/Main/Competitions

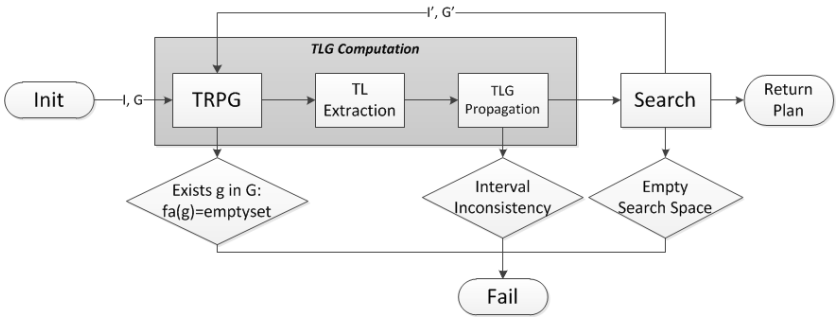


Figure 1: Scheme of TempLM

(2008)). In a later work (Marzal et al. (2014)), we establish the foundations of the temporal landmarks graph (TLG) and the relationships between the end points of the temporal landmarks. Subsequently, a first operational version of TempLM (Marzal et al. (2015)) stresses the role of the search engine in the process of finding a solution in overconstrained problems.

In this paper, we present the last version of TempLM, which has been extensively tested in a wide variety of problems from the different IPCs. We compare the performance of TempLM against the planner OPTIC in loosely-constrained, tightly-constrained and unsolvable problems. Moreover, we also compare TempLM with a recent approach focused on solving concurrent temporal problems (Karpas et al. (2015)), which enhances the incorporation of temporal action landmarks, i.e., durative actions that must necessarily occur in every solution plan. We conducted several experiments to assess the behaviour of this recent approach in tightly-constrained and unsolvable problems and the results show that TempLM is much more efficient. Thereby, in this paper we particularly highlight the contributions that make TempLM be much more superior in the resolution of overconstrained and unsolvable temporal planning problems.

The structure of this paper follows the flow of Figure 1, which outlines the main stages of TempLM. Section 2 presents some basic notions on temporal planning and section 3 explains the process for extracting temporal landmarks and building the TLG, labeled as *TL Extraction* in Figure 1. Subsequently, section 4 details the propagation of the temporal constraints across the TLG. Once the TLG is built, the next stage is applying the Search process to find a solution plan (section 5). One of the novelties of TempLM lies in the continual planning approach which continuously updates the TLG with the information obtained during search and viceversa, as can be observed in Figure 1. Overall, TempLM is an iterative multi-stage satisfiability planning approach that enables a rapid detection of unsolvable problems (unsolvability points are marked as diamonds in Figure 1). This is shown in section 6. Finally, we present a thorough empirical evaluation of TempLM and we assess the performance of TempLM compared to another state-of-the-art temporal planner, OPTIC, and a novel approach that exploits temporal landmarks for solving temporal planning problems as well.

## 2. Preliminary concepts

A *temporal planning problem* $\mathcal{P} = \langle P, O, I, G \rangle$ 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 $P$. We assume a subset of the semantics of the temporal model of PDDL2.1 (Fox and Long (2003)), the Time-Initial Literals (TILs) defined in PDDL2.2 (Hoffmann and Edelkamp (2005)) and some of the state trajectory constraints introduced in PDDL3.0 (Gerevini et al. (2009)).

A **durative action** $a \in O$ in PDDL2.1 (Fox and Long (2003)) is defined as a tuple $\langle dur(a), Cond(a), Eff(a) \rangle$ where $dur(a) \in \mathcal{R}^+$ is the duration of the action; $Cond(a) = SCond(a) \cup ECond(a) \cup Inv(a)$ (conditions to hold at the start, at the end or overall the duration of $a$); $Eff(a) = SEff(a) \cup EEff(a)$ (effects produced at the start or end of the execution of $a$), which are divided into $SEff(a) = \{SAdd(a) \cup SDel(a)\}$ (propositions added and deleted at the start of $a$), and $EEff(a) = \{EAdd(a) \cup EDel(a)\}$ (propositions added and deleted at the end of $a$).

We define two gaps between a condition $c$ and an effect $e$ of a durative action $a$: (1) $gap^a_E(c, e)$ is the gap between the time point when $e$ is asserted and the earliest time point when $c$ is needed; and (2) $gap^a_L(c, e)$ is the gap between the time point when $e$ is asserted and the latest time point when $c$ is needed[2]. Specifically:

| Condition | Effect | $gap^a_E$ | $gap^a_L$ | Condition | Effect | $gap^a_E$ | $gap^a_L$ |
|---|---|---|---|---|---|---|---|
| $SCond$ | $SAdd$ | $\varepsilon$ | $\varepsilon$ | $SCond$ | $EAdd$ | $dur(a)$ | $dur(a)$ |
| $Inv$ | $SAdd$ | $\varepsilon$ | $-dur(a)$ | $Inv$ | $EAdd$ | $dur(a)$ | $\varepsilon$ |
| $ECond$ | $SAdd$ | $-dur(a)$ | $-dur(a)$ | $ECond$ | $EAdd$ | $\varepsilon$ | $\varepsilon$ |

**Definition 2.1.** A **temporal plan** $\Pi$ is a set of pairs $(a, t)$, where $a \in O$ and $t$ is the start execution time of $a$. Given a proposition $p$, we denote by $start(p)$ and $end(p)$ the time points when $p$ is asserted and deleted, respectively, by any action $a$ in $\Pi$. The duration (makespan) of a temporal plan $\Pi$ is $dur(\Pi) = max_{\forall (a,t) \in \Pi}\big(t + dur(a)\big)$.

**Definition 2.2.** 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 $(t, p)$, denoting that proposition $p$ must be achieved within $t$ time units.

In PDDL, there are two basic ways for expressing a deadline $t_g$ over a goal $g$: explicitly, by means of PDDL3.0 modal operator within, as stated in Gerevini et al. (2009); and, implicitly, by means of the PDDL2.2 TILs to express time windows (Marzal et al. (2014)). We assume that $D$ is a set of constraints such that we are able to establish a deadline $t_g$ for each $g \in G$, and, hence, $T_\Pi = max_{\forall g \in G}(t_g)$.

## 3. Extraction of Temporal Landmarks

The notion of landmark has been widely used in automated planning and has proved to be a very helpful source of information to guide search towards a solution Hoffmann et al. (2004).

[2] $\varepsilon$ 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 et al. (2002)).

**Definition 3.1.** Let $\mathcal{P} = \langle P, O, I, G \rangle$ be a temporal planning problem. A **landmark** of $\mathcal{P}$ is a proposition of $P$ that must be satisfied in every plan that solves $\mathcal{P}$.

Let's now assume that we add some deadline constraints $D$ to $\mathcal{P}$, thus defining a temporal planning problem with deadline constraints $\mathcal{P}' = \langle P, O, I, G, D \rangle$. The landmarks extracted for $\mathcal{P}$ will remain being landmarks in $\mathcal{P}'$ because a landmark is a true assertion irrespective of the temporal features of the problem. However, a new set of landmarks can be discovered in $\mathcal{P}'$ due to the existence of deadline constraints.

**Definition 3.2.** Given $\mathcal{P} = \langle P, O, I, G, D \rangle$, a **temporal landmark** of $\mathcal{P}$ is a proposition of $P$ that must hold in every plan that solves $\mathcal{P}$ in order to satisfy $D$.

The method we use for extracting the *landmarks* of a problem is a combination of several existing techniques that returns more landmarks than any other known technique (details can be found in Marzal et al. (2011)). The extraction of *temporal landmarks* (Marzal et al. (2014)) draws upon the concepts of *first achievers* and *labels* of a proposition, which are calculated by using a **temporal relaxed planning graph (TRPG)**. A TRPG is a directed, layered graph that contains proposition levels $P_i$, the set of propositions that appear in the relaxation after $i$ time units, and action levels $A_i$, the set of actions that are applicable after $i$ time units. $TRPG(A_t)$ represents the action level at time $t$.

**Definition 3.3.** The **first achievers** of $p \in P$ at time $t$ is the set of actions that may achieve $p$ at $t$:
$fa^t(p) = \{a \in TRPG(A_t) : p \in SAdd(a) \cup EAdd(a)\}$

On the other hand, the *dependency labels* keep information about the propositions that must necessarily be true in the plan before $p$ is achieved at time $t$.

**Definition 3.4.** The **dependency labels** of $p \in P$ at time $t$, denoted by $labels^t(p)$, is given by:

$$labels^t(p) = \{p\} \cup \left( \bigcap_{\forall a \in fa^t(p)} labels_a^t(a) \right), where \;\; labels_a^t(a) = \left( \bigcup_{\forall c \in Cond(a)} labels^{t-dur(a)}(c) \right)$$

Both landmarks and temporal landmarks are combined together in a single structure named Temporal Landmarks Graph. In the following, we will refer to any type of landmark simply as a landmark, unless stated otherwise.

**Definition 3.5.** A **Temporal Landmarks Graph (TLG)** is a directed graph $G = (V, E)$ where the set of nodes $V$ are landmarks and an edge in $E$ is a tuple of the form $(l_i, l_j, \prec_{\{n,d\}})$ which represents a necessary or dependency ordering constraint between the landmarks $l_i$ and $l_j$, denoting that $l_i$ must happen before $l_j$ in a solution plan.

We note that the TLG may contain several instances of the same landmark in case the landmark has to be achieved several times along the plan.

Landmarks are also annotated with various *temporal intervals* that represent the validity of the corresponding temporal proposition (Marzal et al. (2008)). The **generation interval** of a landmark is denoted by $[min_g(l), max_g(l)]$. $min_g(l)$ represents the earliest

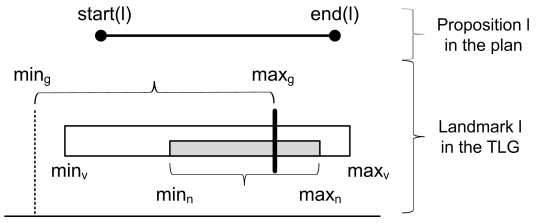


Figure 2: Connection between the temporal occurrence of a proposition $l$ in the plan (top) and the temporal landmark $l$ in the TLG (bottom)

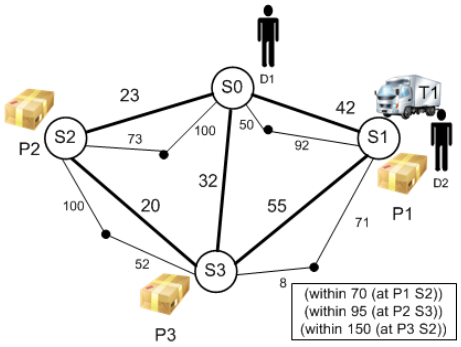


Figure 3: Initial state and goals of the explanatory example.

time point when landmark $l$ can start in the plan. This value is determined by the time of the proposition layer when $l$ appears in the TRPG. $max_g(l)$ represents the latest time point when $l$ must start in order to satisfy $D$ and it is initialized as $max_g(l) = T_\Pi$. The **validity interval** of a landmark $l$ is denoted by $([min_v(l), max_v(l)])$ and it represents the longest time that $l$ can be in the plan. Initially, it is set as $min_v(l) = min_g(l)$ and $max_v(l) = T_\Pi$. Finally, the **necessity interval** of a landmark $l$ is denoted by $([min_n(l), max_n(l)])$ and it represents the set of time points when $l$ is required as a condition for an action to achieve other landmarks. Initially, $min_n(l) = min_g(l)$ and $max_n(l) = T_\Pi$. Figure 2 shows the relationships between the temporal intervals of a landmark $l$ in the TLG and the temporal occurrence of such landmarks in the plan (Marzal et al. (2008)).

Roughly speaking, given a set of deadline constraints $D$, where $(t, l) \in D$, two ways of finding temporal landmarks can be defined. The **first method** is to directly extract the temporal landmarks from the constraints defined in $D$, insert them in the TLG, and update the generation interval of the landmarks. Specifically, $\forall (t, l) \in D : TL = TL \cup l,\ max_g(l) = t$. The **second method** 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. Given a landmark $l$ in the TLG, the propositions which $l$ has some dependencies with at time $max_g(l)$ are the propositions that must be necessarily achieved in the plan in order to obtain $l$ at time $max_g(l)$. Formally, $\forall l \in TL : TL = TL \cup labels^{max_g(l)}(l)$. This second method is repeated as long as new temporal landmarks are found.

Figure 3 shows the initial and goal state of a temporal planning problem. The deadline constraints, $D$, are shown in a box. This is a problem of the driverlog domain (Long and Fox (2003)) which involves trucks moving between locations and delivering packages. We can observe there are several ways of transporting P1 to S2, either going through S0 or through S3. However, as P1 must be delivered within 70 time units (t.u.), only the route through S0 can be used (going through S3 takes at least 75 t.u.). Thereby, (driving D2 T1) and (at T1 S0) are temporal landmarks. Moreover, we have that (within 70 (at P1 S2)) $\in D$ and so the proposition $l$ =(at P1 S2) is identified as a temporal landmark by the first method such that $max_g(l)$ =70. On the other hand, the set $labels^t$(at P1 S2) at $t = 70$ contains the propositions (driving D2 T1) and (at T1 S0) and so both propositions

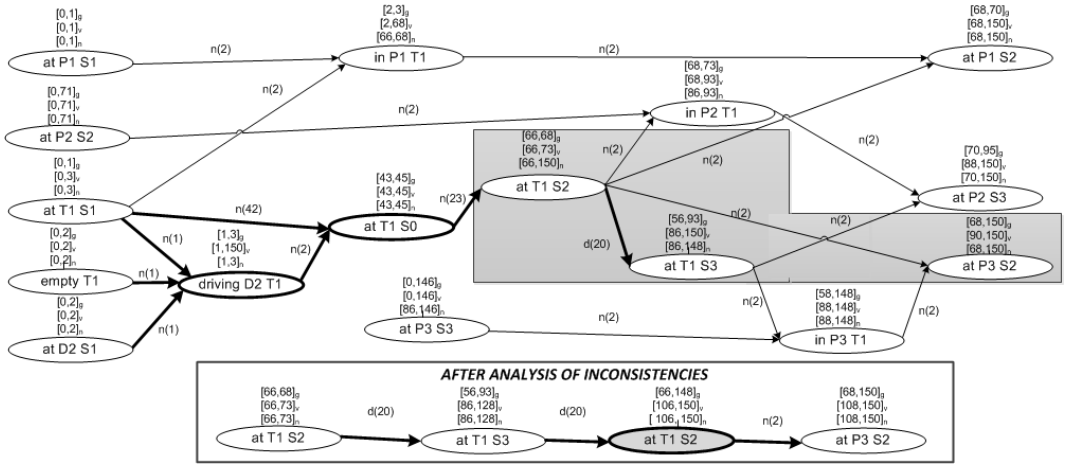


Figure 4: Landmarks Graph for the Example (top: after the propagation stage, bottom: modifications after the analysis of inconsistencies)

become temporal landmarks by applying the second method (see both landmarks in Figure 4 (top) circled in bold).

Once the intervals of the landmarks are initialized as indicated above, the extraction algorithm computes the set of *necessary* and *dependency* orders between landmarks (Marzal et al. (2014)). There is a necessary order between $l_i$ and $l$ ($l_i \prec_n l$) if $l_i \in \bigcap Cond(A_{max_g(l)})$. Likewise, there is a dependency order between $l_i$ and $l$ ($l_i \prec_d l$) if $l_i \in labels^{max_g(l)}(l)$. For example, in Figure 4 (top), there is a necessary order between propositions (driving D2 T1) and (at T1 S1) with proposition (at T1 S0) because the only way to achieve (at T1 S0) at $max_g = 45$ is by using the action `Drive(T1 S1 S0 D2)` whose conditions are satisfied with the propositions (driving D2 T1) and (at T1 S1), respectively. And there is a dependency order between (at T1 S2) and (at T1 S3) because landmark (at T1 S2) belongs to the labels of (at T1 S3) at $t = 93$.

## 4. Propagation of Temporal Constraints

When a first version of the TLG is built, the next step is to propagate the constraints and update the temporal intervals of the landmarks accordingly. This is performed in two stages.

### *4.1. Analysis of causal and mutex relationships*

A causal relationship of the form $l_i \prec_{\{d,n\}} l_j$ between two landmarks $l_i$ and $l_j$ implicitly establishes some constraints between the endpoints of the temporal intervals. Specifically:

$$min_v(l_j) = \max(min_v(l_j), min_v(l_i) + GAP_E(l_i, l_j))$$
$$max_g(l_i) = \min(max_g(l_i), max_g(l_j) - GAP_E(l_i, l_j))$$

Moreover, if $mutex(l_i, l_j)$ (Blum and Furst (1997)):

$max_v(l_i) = \min(max_v(l_i), min_v(l_j), max_g(l_j) - GAP_L(l_i, l_j))$
$max_g(l_i) = \min(max_v(l_i), max_g(l_i))$

where $GAP_E$ and $GAP_L$ are the general case of $gap_E$ and $gap_L$, respectively; i.e., the gap between two landmarks $l_i$ and $l_j$ connected through a dependency order ($l_i \prec_d l_j$):

$$GAP_{\{E,L\}}(l_i, l_j) = min(GAP_{\{E,L\}}(l_i, l) + gap_{\{E,L\}}(l, l_j))$$

where $\exists a \in O : l \in Cond(a) \wedge l_j \in SAdd(a) \cup EAdd(a)$. In case of $l_i = l_j$ then $GAP_{\{E,L\}}(l_i, l_j) = 0$.

These constraints are propagated across the TLG when the interval of a landmark is modified. Specifically, $min_v$ values are propagated forwards from the initial state whereas $max_g$ and $max_v$ values are propagated backwards from the goal.

In Figure 4 (top), it holds (at T1 S3) $\prec_n$ (at P2 S3) and $max_g$(at P2 S3)=95 because of the deadline in $D$. The action that causes the necessary order is $a$ =`Unload(P2 S3 T1)` with $dur(a) = 2$; therefore, $max_g$(at T1 S3)=95-2=93, indicating that, in order to achieve (at P2 S3) at 95, (at T1 S3) must be true at 93; or, equivalently, $gap^a_E = 2$. Similarly, (in P2 T1)$\prec_n$(at P2 S3) and (at T1 S3)$\prec_n$(at P2 S3) in Figure 4 (top), which implies that $min_v$(at P2 S3)=max($min_v$(in P2 T1)+2, $min_v$(at T1 S3)+2)=88, indicating that the first time point when (at P2 S3) can be valid is 88, which is a more accurate bound in contrast with $min_g$(at P2 S3), which indicates that the first time it is achieved in the TRPG is 70.

On the other hand, given that (at T1 S2) and (at T1 S3) are mutex and the existence of a dependency order between them, we update $max_v$(at T1 S2) to the value 73, indicating that (at T1 S2) will be no longer valid from 73 onwards. However, this proposition is necessary to achieve (at P3 S2) between 90 and 150. Next section describes how to solve this inconsistency.

### *4.2. Computation of the necessity intervals*

The necessity interval of a landmark $l_i$ is computed by taking into account all the necessary orders for which $l_i$ is needed as a condition to generate another landmark. Thus, $\forall l_j : \exists l_i \prec_n l_j$, the necessity interval of $l_i$ is calculated as:

$min_n(l_i) = \min_{\forall l_j : \exists l_i \prec_n l_j} (min_v(l_j) - GAP_E(l_i, l_j))$
$max_n(l_i) = \max_{\forall l_j : \exists l_i \prec_n l_j} (max_g(l_j) - GAP_L(l_i, l_j))$

If a landmark $l$ is needed beyond its validity interval, that is, $[min_n(l), max_n(l)] \not\subseteq [min_v(l), max_v(l)]$, then it implies that some information is missing in the TLG. Specifically, this case induces that another instance of the landmark $l$, namely $l'$, must be added to the TLG to support the fragment of $[min_n(l), max_n(l)]$ that is not supported by $[min_v(l), max_v(l)]$. Once $l'$ is created, its temporal intervals are initialized. Then, we analyze the landmarks with which $l$ maintains some dependency. For all propositions $l_i$ whose propagation of $max_g(l_i)$ updates $max_n(l)$, the constraint $l' \prec_n l_i$ is straightforwardly inferred. For the rest of landmarks $l_j$ that do not need $l$ at $max_n(l)$ for being generated, the relation between $l$ and $l_j$ is maintained. Finally, the new orderings are inserted in the TLG and propagated. For example, as explained above, $max_v$(at T1 S2)=73 due to a mutex relationship and this is inconsistent with $max_n$(at T1 S2)=150. In this case, $l$ is the first landmark that appears in Figure 4 (bottom) and $l'$ is the landmark in grey. $l'$ is added to the TLG, where $l_i$ =(at P3 S2) and $l_j$ =(at T1 S3), and all the intervals are updated accordingly.

## 5. Searching for a solution plan

TempLM applies a search process in the space of partial plans in order to find a solution plan for a problem $\mathcal{P} = \langle P, O, I, G, D \rangle$ (Marzal et al. (2014)). Given a node $n$ of the search tree, a successor of $n$ results from adding an executable action $a$ to the partial plan of $n$, provided that $a$ does not cause conflicts with the actions of $n$. Hence, the plan of the successor node will contain a newly added action, information which can be exploited to find new temporal landmarks in the TLG (Marzal et al. (2015)).

Let's assume, for instance, that we have another truck `T2` at location `S1` in the problem of Figure 3. In this case, the TLG of the problem would only comprise the propositions in $I$ and $G$ because it is not possible to determine which truck must be used to transport each package. However, if package `P1` were loaded in `T1` then we could infer that we must use `T1` to transport `P1` to `S2` because otherwise there would not be enough time to unload `P1` from `T1`, load it again in `T2` and transport it to `S2` in time. In this situation, a new landmark (at T1 S2) would be discovered when the action (load P1 T1 S1) along with its effect `(in P1 T1)` is inserted in the plan. Thereby, we can associate a distinct TLG to each search node $n$ accordingly to the plan of $n$. This will provide a helpful guidance during search because the TLG at each node is a much more informative graph than the TLG of the problem (root node).

A node in the search tree is defined as $n = (\Pi, S_t, TLG_\Pi)$, where $\Pi$ is the conflict-free partial plan of $n$, $S_t$ is the state reached at time $t = dur(\Pi)$ after the execution of $\Pi$ from $I$ and $TLG_\Pi$ is the refined TLG after taking into account the information in $\Pi$. TempLM applies a A$^\star$ search algorithm guided by the standard evaluation function $f(n) = g(n) + h(n)$, where $g(n) = dur(\Pi)$ accurately reflects the cost of $n$ since $dur(\Pi)$ accounts for the parallelism of $\Pi$; and $h(n)$ is a non-admissible heuristic based on action landmarks for sub-optimal temporal planning. Particularly, given $n = (\Pi, S_t, TLG_\Pi)$, we apply the following evaluation function, $f(n) = g(n) + h(n) = dur(\Pi) + h^{LM-cut}_{temp}(S_t, G)$.

$h^{LM-cut}_{temp}$ is a temporal approximation of the well-known landmark cut heuristic $h^{LM-cut}$, which has been successfully used in optimal sequential planning (Helmert and Domshlak (2009)). When adapting the LM-cut heuristic to a temporal context, the cost of an action is its duration. Then, $h(n)$ is an estimate based on the makespan of a sequential plan that neglects overlapping actions, which, obviously, is an overestimate of the actual plan makespan. Consequently, we can affirm TempLM does not guarantee to find the optimal solution, the one with shortest duration.

Algorithm 1 details the search process of TempLM, which starts with the root node $(\emptyset, I, TLG)$. When the node $n = (\Pi, S_t, TLG_{\Pi*})$ with the lowest $f(n)$ value is selected from the open list $SS$[3], we first check whether the end conditions of the actions in $\Pi$ that finish at $t$ are fulfilled in state $S_t$; otherwise, the successors of $n$ are not generated (steps 5 to 7 of Algorithm 1). If $n$ represents a solution plan, $\Pi$ is returned and the process stops (steps 8 to 10). Otherwise, $TLG_{\Pi*}$ of $n$ is replaced by $TLG_\Pi$, which is the result of refining $TLG_{\Pi*}$ by including the new landmarks derived from the plan $\Pi$ (step 11, which will be detailed in section 5.2). Then, we generate the successors of $n$ (step 12), as described in Algorithm 2. Finally, for each successor $n'$ of $n$, $f(n')$ is calculated and $n'$ is inserted in $SS$, which is implemented as a priority queue.

[3] $TLG_{\Pi*}$ refers to the TLG of the parent node. This will be later explained.

**Algorithm 1** Algorithm for searching a solution plan

1: **Input:** $\mathcal{P}, TLG$
2: $SS \leftarrow \{(\emptyset, I, TLG)\}$ {Initialization of the search space}
3: **while** $SS \neq \emptyset$ **do**
4: $n = (\Pi, S_t, TLG_{\Pi*}) \leftarrow Pop(SS)$
5: **if** $\exists a \in \Pi : end(a) = t \wedge ECond(a) \notin S_t$ **then**
6: discard $n$ and return to step 4
7: **end if**
8: **if** $isSolution((\Pi, S_t, TLG_{\Pi*}))$ **then**
9: **return** $\Pi$
10: **end if**
11: **if** $TLG_\Pi = refinementTLG(\Pi, S_t, TLG_{\Pi*})$ **then**
12: $Suc_t = calculateSuccessors(\Pi, S_t, TLG_\Pi)$
13: **for** $n' = (\Pi', S_{t'}, TLG_\Pi) \in Suc_t$ **do**
14: push $(SS, (\Pi', S_{t'}, TLG_\Pi), f(n'))$
15: **end for**
16: **end if**
17: **end while**
18: **return** no solution

*5.1. Computation of successors*

Algorithm 2 computes the successors of a node $(\Pi, S_t, TLG_\Pi)$. First, we calculate the set of initially applicable actions in $S_t$ (step 3 of Algorithm 2) as: $O_t = \{a \in O : SCond(a) \cup Inv(a) \subseteq S_t\}$. Only reversible actions that lead to an already visited node are ruled out during the node expansion. By definition, an action $a \in O_t$ is always applicable in $S_t$ of a node $(\Pi, S_t, TLG_\Pi)$ at time $t$ but $a$ may be also executable before $t$ in $\Pi$. This consideration may suppose a significant difference in the plan makespan. For this reason, the algorithm computes the earliest start time of each new action (step 5).

**Definition 5.1.** Given a node $(\Pi, S_t, TLG_\Pi)$ and an action $a_i \in O_t$, the **earliest start time** of $a_i$ (denoted by $t_E^{a_i}$) is the first time point from $I$ where $a_i$ is applicable and does not interfere with any other action in $\Pi$. Action $(a_i, t_E^{a_i})$ is said to **interfere** with the action $(a_j, t_{a_j}) \in \Pi$ if any of the following situations holds: (1) $\exists p \in Inv(a_j)$ and $a_i$ deletes $p$ within the execution interval of $a_j$; (2) $\exists q \in SCond(a_j)$ and $a_i$ deletes $q$ between the time point when $q$ is produced and $t_{a_j}$; and (3) $\exists r \in ECond(a_j)$ and $a_i$ deletes $r$ between the time point when $r$ is produced and $t_{a_j} + dur(a_j)$.

The function *createSucc* in Algorithm 2 creates a new successor $(\Pi', S_{t'})$, where $\Pi' = \Pi \cup (a, t_E^a)$ (plan resulting from adding $(a, t_E^a)$ in $\Pi$ at time $t_E^a$), $t' = dur(\Pi')$ and $S_{t'}$ is the state reachable after executing $\Pi'$. Computing $S_{t'}$ requires executing all actions $a' \in \Pi'$ that start or end in the interval $(t_E^a, t')$ and updating the states at each time instant accordingly to $SAdd(a')$, $SDel(a')$, $EAdd(a')$ and $EDel(a')$.

If the information of the propositions in $\Pi'$ is compliant with the landmarks of $TLG_\Pi$, the TLG of the parent node, then the successor is a valid node (step 7 of Algorithm 2) and will be inserted in the successor list. Otherwise, the node will be invalid, meaning that $\Pi'$ is not a correct plan that leads to a solution plan (see section 6 for details). Note

**Algorithm 2** Function *calculateSuccessors* of a node

**Input:** $(\Pi, S_t, TLG_\Pi)$
$successorsList = \emptyset$
$O_t = initiallyApplicableActions(S_t)$
**for** $a \in O_t$ **do**
  $t^a_E = earliestStartTime(\Pi, S_t, a)$
  $(\Pi', S_{t'}) = createSucc(\Pi, S_t, a, t^a_E)$
  **if** $\neg invalidNode(\Pi', S_{t'}, TLG_\Pi)$ **then**
    append$(successorsList, (\Pi', S_{t'}, TLG_\Pi))$
  **end if**
**end for**
**return** $successorsList$

that we check the validity of the successor node $(\Pi', S_{t'})$ with respect to $TLG_\Pi$ and not $TLG_{\Pi'}$. The reason is that the TLG refinement is a very costly process that we only apply when a node is selected for expansion from the open list $SS$ (Algorithm 1), thus avoiding re-calculating the TLG of nodes that might never be expanded. Although we may lose some valuable information, this results in an important cost saving that largely benefits the performance of TempLM.

### *5.2. TLG refinement*

The key principle of the TempLM search procedure is that instead of guiding the search with the TLG of the root node, each node is associated to its own TLG, which will reflect the information comprised in the plan of the node. Specifically, let $n = (\Pi, S_t, TLG_\Pi)$ and $n' = (\Pi', S_{t'}, TLG_{\Pi'})$, a successor of $n$. Since $\Pi'$ contains one more action than $\Pi$, $TLG_{\Pi'}$ will most likely change w.r.t. $TLG_\Pi$ so the aim is to compute $TLG_{\Pi'}$ as a refinement of $TLG_\Pi$ by taking into account the newly added action.

We simulate the execution of $\Pi'$ from $I$, obtaining the set of propositions that are true in $S'_t$ and the set of *all propositions* that have been deleted from $I$, say $L_{del}$. If it exists $l_i \in L_{del}$, and $l_j$ is a landmark of $TLG_\Pi$ that is not yet achieved in $\Pi'$, and $l_i \prec_{n,d} l_j$ is a relationship in $TLG_\Pi$, then we have to *repair* the graph of $n'$ so as to restore the support of $l_j$ and ensure that $l_j$ will be achievable. The result of this refinement operation is $TLG_{\Pi'}$, a more informed and adapted graph to the plan $\Pi'$ of $n'$.

The TLG refinement process consists in extracting new temporal landmarks for supporting $l_j$ from the information comprised in $\Pi'$. This local extraction of landmarks follows exactly the same steps explained in section 3 and so the first step is to build a TRPG from $I'$ to $G'$ where:

- $I'$ is the initial state, $I' = S_{t'}$. Instead of placing the propositions of $I'$ at $t = 0$, the first proposition layer of the TRPG, each proposition is set at the specific time point at which it is achieved in $\Pi'$. This way, the TRPG reflects a reliable picture of the current situation that represents the plan $\Pi'$ .
- $G'$ is the goal state, the set of landmarks of $TLG_\Pi$ that are not present in $\Pi'$ and whose support has been invalidated by a proposition of $L_{del}$ (i.e., $l_j$).

By using the TRPG from $I'$ to $G'$, we are able to calculate the new set of temporal landmarks ($labels^{max_g(l_j)}(l_j)$); that is, the necessary propositions to achieve $l_j$ at $max_g(l_j)$. $TLG_{\Pi'}$ is then completed with the causal relationships between landmarks and, finally, constraints are propagated across the TLG. The local refinement of $TLG_{\Pi'}$ may fail in two situations, in which case the node $n'$ is discarded: (1) if $l_j \in G'$ is not reached in the TRPG before $max_g(l_j)$ and (2) if, during the constraint propagation, an inconsistency is found.

Let's consider again the example introduced at the beginning of Section 5 where a new truck `T2` is located at `S1` and we have only the goal (at P1 S2). And let's assume a plan $\Pi$ composed solely of the action `(Board D2 T1 S1)` (1 t.u.). We can find two successors of $\Pi$: (1) `(Drive T1 S1 S0 D2)` at t=1 with a duration of 42 t.u.; and (2) `(Load P1 T1 S1)` at t=0 with a duration of 2 t.u. Successor (1) is discarded because if we apply `(Drive T1 S2 S0 D2)` then (at P1 S2) would be unachievable within 70 t.u.; that is, (at P1 S2) is unachievable within 70 because `P1` is not loaded in `T1` in successor (1) and although we have a second truck, `T2`, we have no other driver for driving `T2`. This leads us to successor (2), where we apply the TLG refinement. Since $G' = \{$(at P1 S2)$\}$ and the dependency order (at P1 S1) $<_{d(68)}$ (at P1 S2) is broken due to the introduction of the action `(Load P1 T1 S1)` in $\Pi'$, the proposition (at P1 S2) must be re-achieved such that $max_g$(at P1 S2)=70. Following the steps described above, we find two new landmarks, (at T1 S0) and (at T1 S2) as a consequence of the action of loading `P1` in `T1`. Thereby, TempLM infers that `T1` must go first to `S0` (42 t.u.) and then to `S2` (23 t.u.) in order to unload `P1` in `S2` on time (2 (load)+ 42 (S1-S0) + 23 (S0-S2) + 2 (unload))=69.

### *5.3. Correctness and complexity of the search algorithm*

The search algorithm that we introduce is correct, as the following theorem shows.

**Theorem:** Given a problem $\mathcal{P} = \langle P, O, I, G, D \rangle$, Algorithms 1 and 2 guarantee that, if a plan $\Pi$ is returned, then $\Pi$ is a correct solution for $\mathcal{P}$.

*Proof:* $\Pi$ is correct if the actions in $\Pi$ are successively applicable in the state $S_t$ of each node $n$ of the tree and $G$ holds in the final state and $D$ is satisfied. The applicability of the actions in $\Pi$ is guaranteed by Algorithm 2 since this algorithm calculates in each node $n$ of the tree the state $S_t$ that results from executing the plan contained in $n$ and only creates successors of $n$ with actions that area applicable in $S_t$ (step 3, Algorithm 2). The satisfaction of $G$ is guaranteed by Algorithm 1, step 8, which only stops when $G \subseteq S_t$. And the satisfaction of $D$ is guaranteed by step 7 of Algorithm 2, which only generates nodes compliant with the landmarks of the TLG and, consequently, with $D$.□

With respect to the complexity of these algorithms, we can distinguish two aspects. First, the search itself, which is a non-admisible A*, has a complexity of $O(b^{(T_\Pi / min\_dur\_action)})$, where $b$ is the branching factor of the problem and *min_dur_action* is the minimum duration of all the actions in the problem. Second, the TLG refinement applied in each expanded node consists in (1) computing a local TRPG ($O(|P||O|)$) and (2) propagating the information in the TLG ($O(|V|^2)$), where $|V|$ is the number of landmarks in the TLG.

## 6. Detection of unsolvable problems

Figure 1 shows the stages of TempLM and the three points where an unsolvable problem can be identified. In the following, we analyze the three situations of unsolvability.

The **first point** occurs during the construction of the TRPG, where an unsolvable problem is found if $\exists g \in G : fa(g)^{max_g(g)} = \emptyset$. This condition indicates there is no sequence of actions that reach $g \in G$ independently and, therefore, we can affirm the problem is unsolvable.

The **second point** occurs during the propagation of the temporal constraints across the TLG, particularly when an inconsistency between the endpoints of the temporal intervals of a landmark is found (Figure 2 bottom shows the relationships that must exist between the temporal intervals of a landmark). Two types of unsolvable inconsistencies identified in the TLG allow us to discard the corresponding node:

1. $max_g(l) < min_v(l)$, meaning that $l$ has to be generated before it can actually be achieved in the plan (due to the relationships of previous landmarks).
2. A temporal interval of $l$ becomes empty: $min_g(l) > max_g(l) \vee min_v(l) > max_v(l) \vee min_n(l) > max_n(l)$. This is an indication that the proposition $l$ will never be achieved in the plan.

For example, what would happen if the deadline of $l =$(at P2 S3) were 80 instead of 95?. Since its validity interval is $[88, 150]$, an inconsistency is found with its generation interval $[70, 80]$. Specifically, the deadline constraint is $max_g$(at P2 S3)=80 and $max_g(l) < min_v(l)$. It is important to remark that this inconsistency will not be discovered during the TRPG construction because the first time layer at which (at P2 S3) appears is 70; i.e. $min_g$(at P2 S3)=70. Therefore, any deadline greater than 70 would be found as *satisfiable* during the construction of the TRPG. However, our TLG would detect any deadline lower than 88 ($min_v$) and hence the problem would be labeled as unsolvable.

The **third point** where an unsolvable problem is detected occurs during the search of a plan. If the search space eventually becomes empty, we can affirm the problem is unsolvable. The logical and temporal consistency of the plan of a search node is based on its TLG. This way, there are two points where a node can be discarded: (1) when the TLG refinement fails (see Section 5.2) and (2) when the node is found invalid due to one of the conditions exposed in Definition 6.1.

**Definition 6.1.** A node $(\Pi, S_t, TLG_\Pi)$ is said to be **invalid** 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) $start(l) > max_g(l)$
   (c) $end(l) > max_v(l)$ (applicable if $end(l)$ is known)
2. **Causal relationships:** Given a pair of landmarks $l_i$ and $l_j$ in $\Pi$, such that $l_i \prec_{\{d,n\}} l_j$ in its TLG, if $start(l_i) \geq start(l_j)$, $\Pi$ is pruned.

It is important to remark that the three points for detecting unsolvability ensure that an unsolvable problem will be always identified provided that TempLM is given enough time, because it will eventually exhaust the search space.

Consider the following partial plan generated during the search process to solve our running example:

```
t=0:  (Board D2 T1 S1) (1 t.u.)
t=1:  (Drive T1 S1 S3 D2) (55 t.u.)
```

The proposition $l$=(at T1 S3) is inserted by the action (Drive T1 S1 S3 D2) at $t = 56$; that is, $start(l) = 56$. However, according to the TLG of Figure 4, we observe that $min_v(l) = 86$; consequently, the node is discarded because $start(l) < min_v(l)$. This is the right decision considering that (in P1 T1) has not yet been achieved in the plan, and $max_g$(in P1 T1)=3, which would be impossible to fulfill in this plan because T1 is no longer at S1, where P1 is.

## 7. Experimental evaluation

We created two configurations of our approach: TempLM-TLG, which only uses the TLG of the initial state of the problem, and TempLM-REF, which incorporates the TLG refinement, thus creating a TLG per node of the search space. We selected eleven domains from the International Planning Competitions (IPC[4]) that fit our requirements: deadlines expressed by means of TIL or *within*, with no metric features or ADL. Specifically, we selected one domain from the IPC-2000, Logistics; five domains from the IPC-2002, DriverLog, Depots, Rovers, ZenoTravel and Satellite; two domains from the IPC-2004, Satellite and PipesWorld and three domains from the IPC-2011, Parking, Pegsol and Floortile.

We performed three sets of experiments: (1) a comparison of the two TempLM configurations with OPTIC, a state-of-the-art planner that handles deadlines expressed with TILs and the within operator (Benton et al. (2012)), over a set of tight and unsolvable problems; (2) a comparison of the two TempLM configurations with OPTIC over a set of problems with randomly generated deadlines; and (3) a comparison of the two TempLM configurations with the approach presented in Karpas et al. (2015). All the experiments were censored after 30 minutes and they were performed in an Intel-Core-i5-3.2-GHz with 16GB-RAM.

**First experiment.** We created a set of overconstrained problems and a set of unsolvable problems for every domain[5]. Using the original problems from the IPCs as a baseline, tight problems were generated by defining a deadline to all the goals of a problem $\mathcal{P}$ as follows: $\forall g \in G, max_g(g) = bpm + bpm * v$, where $bpm$ is the makespan of the best solution plan found by any planner to $\mathcal{P}$ and $v$ is a random value in [0, 0.20]. Subsequently, unsolvable problems were generated by tightening the existing deadlines up to the point where no plan was found[6].

The results obtained for the set of tight problems are presented in Figures 5, 7, 9, 11 and 13, where we show the makespan (Mk) of the resulting plan and the computation

[4]http://www.icaps-conference.org/index.php/Main/Competitions

[5]Except Satellite from IPC-2002, given that we used the IPC-2004 version of this domain.

[6]The generated problems are available at `https://goo.gl/gLTKEC`

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | Mk | Time | Mk | Time | Mk | Time |
| **Driver** | | | | | | |
| D01T | **91** | **0.05** | **91** | 0.12 | 92 | 1.31 |
| D02T | 47 | 5.18 | **40** | 26.25 | **40** | **11.76** |
| D03T | **51** | **58.4** | **51** | 163.82 | 80 | 309.37 |
| D04T | 49 | 69.56 | 49 | 141.74 | 49 | **23.42** |
| D05T | **98** | 302.85 | **98** | **131.02** | 144 | 130.58 |
| D06T | 128 | 264.72 | 128 | 1108.77 | **126** | **158.84** |
| D07T | **37** | **24.90** | **37** | 55.79 | - | T.E. |
| D08T | **98** | **97.74** | **98** | 344.28 | 120 | 389.63 |
| D09T | **76** | **24.61** | **76** | 94.782 | - | T.E. |
| D10T | 49 | 81.69 | 49 | 142.73 | 49 | **21.46** |
| **Zeno** | | | | | | |
| Z01T | 592 | **0.12** | 592 | 0.45 | 592 | 3.45 |
| Z02T | 592 | **0.11** | 592 | 0.38 | 592 | 3.43 |
| Z03T | **393** | **8.8** | **393** | 18.12 | - | T.E. |
| Z04T | 542 | 17.52 | 542 | 34.07 | **536** | 625.35 |
| Z05T | **522** | **24.95** | 529 | 8.49 | - | T.E. |
| Z06T | **320** | **34.36** | **320** | 38.29 | - | T.E. |
| Z07T | **333** | **26.54** | **333** | 37.31 | - | T.E. |
| Z08T | **665** | **35.14** | **665** | 54.21 | - | T.E. |
| Z09T | **430** | **1144.29** | 559 | 1547.08 | 576 | 1729.69 |
| Z10T | **665** | 380.92 | **665** | **33.43** | - | T.E. |

Figure 5: Results for the tight problems (1)

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | D | Time | D | Time | D | Time |
| **Driver** | | | | | | |
| D01U | R | 0.01 | R | 0.01 | U | 0.14 |
| D02U | R | 0.01 | R | 0.01 | U | 0.13 |
| D03U | S | 295.13 | S | **9.33** | - | T.E. |
| D04U | - | T.E. | S | **679.35** | - | T.E. |
| D05U | - | T.E. | S | **1095.74** | - | T.E. |
| D06U | R | 0.01 | R | 0.01 | U | 0.16 |
| D07U | S | **395.29** | S | 792.49 | - | T.E. |
| D08U | R | 0.01 | R | 0.01 | U | 0.11 |
| D09U | S | 351.14 | S | **39.11** | - | T.E. |
| D10U | S | 346.03 | S | **34.99** | - | T.E. |
| **Zeno** | | | | | | |
| Z01U | R | 0.01 | R | 0.01 | U | 0.14 |
| Z02U | S | 1.11 | S | **0.27** | - | T.E. |
| Z03U | - | T.E. | S | **14.41** | - | T.E. |
| Z04U | S | 704.15 | S | **31.37** | - | T.E. |
| Z05U | S | 592.26 | S | **5.16** | - | T.E. |
| Z06U | S | 575.35 | S | **4.77** | - | T.E. |
| Z07U | - | T.E. | S | **104.71** | - | T.E. |
| Z08U | R | 0.01 | R | 0.01 | U | 0.16 |
| Z09U | - | T.E. | S | **96.46** | - | T.E. |
| Z10U | - | T.E. | S | **36.25** | - | T.E. |

Figure 6: Results for the unsolvable problems (1)

time (in secs.) required by each approach. For the unsolvable problems, presented in Figures 6, 8, 10, 12 and 14, we also show where the unsolvability was detected: during the construction of the TRPG (R), when propagating the temporal constraints in the TLG (G) or during the search (S). The U symbol in the OPTIC results indicates that OPTIC was capable of detecting the unsolvable problem. The symbol T.E. stands for Time Exhausted.

With respect to tight problems, TempLM outperforms OPTIC in almost all the domains, except in the Driverlog and Satellite domains, where OPTIC improves TempLM in some of the problems in terms of computational time and makespan (Figures 5 and 13). However, in the Zeno, Depots, Logistics and Floortile domains, OPTIC is only capable of solving a few problems. In the rovers domain, both configurations of TempLM obtain a makespan shorter than OPTIC in almost all of the problems whereas this difference is not so remarkable in the Pipesworld and Pegsol domains. In these last three domains, TempLM-TLG is much faster than OPTIC. In general, the plans returned by TempLM-REF are the same as the plans of TempLM-TLG although TempLM-REF is more costly due to the TLG refinement process. The explanation for this is also accompanied with the figures in Table 1. Although the number of expanded and generated nodes is always lower in TempLM-REF, calculating a TLG in each search node considerably increases the computation time in some instances. However, thanks to the TLG refinement, TempLM-REF solves a few more instances than TempLM-TLG.

With respect to the unsolvable problems, TempLM-REF clearly outperforms both TempLM-TLG and OPTIC, in the number of detected problems as well as computation time except in the Pegsol domain (Figures 6, 8, 10, 12 and 14). OPTIC is only able to detect 19 unsolvable problems out of 90, whereas TempLM-REF is able to detect all of them. Particularly, TempLM-REF identifies 16 out of these 19 unsolvable problems during the TRPG expansion, two during the TLG generation and one during the search process (this latter case is the only one which OPTIC detects earlier). The benefit of the

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | Mk | Time | Mk | Time | Mk | Time |
| **Pipes** | | | | | | |
| PI01T | 6 | **0.16** | 6 | 0.24 | 6 | 1.44 |
| PI02T | 18 | **0.56** | 18 | 6.09 | 18 | 46.22 |
| PI03T | 14 | **0.59** | 14 | 2.78 | 14 | 3.5 |
| PI04T | **16** | **7.39** | **16** | 17.41 | 20 | 245 |
| PI05T | **12** | **1.84** | **12** | 6.89 | 14 | 4.12 |
| PI06T | 14 | **1.3** | 14 | 5.18 | 14 | 118.85 |
| PI07T | 12 | **2.12** | 12 | 4.42 | 12 | 1179.89 |
| PI08T | 14 | **3.09** | 14 | 6.57 | 14 | 4.29 |
| PI09T | 18 | 22.46 | 18 | 71.88 | 18 | **4.5** |
| PI10T | **22** | 1234.37 | 24 | 653.88 | - | T.E. |
| **Pegsol** | | | | | | |
| PE01T | 7 | **0.4** | 7 | 3.26 | 7 | 21.46 |
| PE02T | **6** | **1.01** | **6** | 1.94 | 8 | 56.92 |
| PE03T | 7 | **0.49** | 7 | 4.13 | 7 | 53.28 |
| PE04T | **9** | **28.86** | **9** | 107.61 | - | T.E. |
| PE05T | **8** | **25.46** | **8** | 47.66 | - | T.E. |
| PE06T | 10 | **22.9** | 10 | 167.47 | 10 | 236.86 |
| PE07T | 7 | **0.1** | 7 | 0.99 | 7 | 9.13 |
| PE08T | 7 | **1.94** | 7 | 14.73 | 7 | 118.56 |
| PE09T | 7 | **4.01** | 7 | 25.47 | 8 | 1300.06 |
| PE10T | 10 | **84.84** | 10 | 514.46 | - | T.E. |

Figure 7: Results for the tight problems (2)

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | D | Time | D | Time | D | Time |
| **Pipes** | | | | | | |
| PI01U | R | **0** | R | **0** | U | 3.45 |
| PI02U | S | **0.85** | S | 1.92 | - | T.E. |
| PI03U | S | 31.79 | S | **27.50** | - | T.E. |
| PI04U | S | 54.99 | S | **35.26** | - | T.E. |
| PI05U | S | 397.79 | S | **190.22** | - | T.E. |
| PI06U | S | 368.97 | S | **133.28** | - | T.E. |
| PI07U | S | 135.78 | S | **33.38** | - | T.E. |
| PI08U | S | 1120.21 | S | **81.6** | - | T.E. |
| PI09U | - | TE | S | **157.41** | - | T.E. |
| PI10U | - | TE | S | **282.34** | - | T.E. |
| **Pegsol** | | | | | | |
| PE01U | G | **0.01** | G | **0.01** | U | 87.2 |
| PE02U | S | **8.86** | S | 9.17 | - | T.E. |
| PE03U | S | **71.69** | S | 946.65 | - | T.E. |
| PE04U | S | **63.52** | S | 807.14 | - | T.E. |
| PE05U | S | **14.21** | S | 150.32 | U | 27.53 |
| PE06U | R | 0.01 | R | 0.01 | U | 0.11 |
| PE07U | R | 0.01 | R | 0.01 | U | 0.11 |
| PE08U | G | **0.01** | G | **0.01** | U | 183.09 |
| PE09U | S | **3.27** | S | 50.43 | - | T.E. |
| PE10U | S | **72.77** | S | 992.32 | - | T.E. |

Figure 8: Results for the unsolvable problems (2)

TLG refinement process is noticeable in these problems, where the computational time required by the search process is cut off drastically thanks to the nodes discarded when the TLG refinement fails. The only exception is the Pegsol domain, where TempLM-REF is more costly than TempLM-TLG. The reason can be found in Table 1; in this domain, the reduction in the number of expanded and generated nodes between TempLM-TLG and TempLM-REF is not so remarkable as in the other domains, so the TLG refinement does not payoff in this domain. Note that the number of generated and expanded nodes in Table 1 coincides for the unsolvable problems because the search space is exhausted. We do not show results of generated and expanded nodes in the unsolvable problems of Depots domain because TempLM-REF only detects unsolvability by using TLG.

Following the results in Figures 5 to 14, we conclude that TempLM-REF is the only approach that identifies all unsolvable problems. In the set of overconstrained or tight problems, TempLM-REF is more costly than TempLM-TLG but it solves a few more instances and returns the same solutions than TempLM-TLG. All in all, TempLM is a promising approach for overconstrained problems and particularly for rapidly detecting unsolvability.

**Second experiment.** We created a set of problems with random deadlines for the problem goals. We applied the same formula for $max_g(g)$ of the first experiment but now each $g$ is associated to a different deadline. More specifically, the parameter $v$ takes a value higher than 0.30 for generating the loose problems and a value in between [0, 0.30] for generating the tight problems. Table 2 shows the overall number of problems generated for each domain[7], the number of problems solved by at least one of three approaches (second column) and how many out of these solved problems turned out to

[7]In this second experiment, the Satellite and Pipesworld domains from IPC-2004 were not used because deadlines are specified through TILs and it turns out difficult to generate correct random deadlines.

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | Mk | Time | Mk | Time | Mk | Time |
| **Floor** | | | | | | |
| F01T | **15** | **0.68** | **15** | 3.19 | - | T.E. |
| F02T | **12** | **0.92** | **12** | 4.97 | - | T.E. |
| F03T | **22** | **4.02** | **22** | 24.61 | - | T.E. |
| F04T | **19** | **1.03** | **19** | 4.66 | - | T.E. |
| F05T | **19** | **1.3** | **19** | 6.58 | - | T.E. |
| F06T | **20** | **1.03** | **20** | 4.83 | - | T.E. |
| F07T | - | T.E. | **22** | **50.69** | - | T.E. |
| F08T | **20** | **2.48** | **20** | 5.88 | - | T.E. |
| F09T | **21** | **5.31** | **22** | 11.73 | - | T.E. |
| F10T | - | T.E. | **19** | **49.37** | - | T.E. |
| **Rovers** | | | | | | |
| R01T | 47 | 0.50 | 47 | **0.45** | 47 | 2.1 |
| R02T | 57 | **2.10** | 57 | 6.59 | 57 | 3.02 |
| R03T | 50 | 0.07 | 50 | 0.66 | **45** | 270.04 |
| R04T | **93** | **0.8** | **93** | 4.25 | - | T.E. |
| R05T | **45** | **0.31** | **45** | 0.82 | 104 | 1021.41 |
| R06T | **125** | 22.28 | **125** | **10.54** | - | T.E. |
| R07T | **73** | 10.15 | **73** | 22.16 | 78 | 9.34 |
| R08T | **88** | **5.76** | **88** | 6.49 | 93 | 33.18 |
| R09T | **145** | **30.29** | **145** | 53.074 | - | T.E. |
| R10T | - | T.E. | **115** | **14.98** | 130 | 226.3 |

Figure 9: Results for the tight problems (3)

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | | Time | | Time | | Time |
| **Floor** | | | | | | |
| F01U | S | **441.49** | S | 525.28 | - | T.E. |
| F02U | S | 186.67 | S | **77.25** | - | T.E. |
| F03U | S | 303.27 | S | **246.24** | - | T.E. |
| F04U | S | 215.75 | S | **119.86** | - | T.E. |
| F05U | S | 251.63 | S | **168.54** | - | T.E. |
| F06U | G | **0.01** | G | **0.01** | - | T.E. |
| F07U | - | T.E. | S | **1254.86** | - | T.E. |
| F08U | S | 300.66 | S | **290.63** | - | T.E. |
| F09U | S | 232.22 | S | **136.81** | - | T.E. |
| F10U | S | 369.11 | S | **295.33** | - | T.E. |
| **Rovers** | | | | | | |
| R01U | G | 0.01 | G | 0.01 | - | T.E. |
| R02U | S | 886.62 | S | **77.92** | - | T.E. |
| R03U | G | 0.01 | G | 0.01 | - | T.E. |
| R04U | G | 0.01 | G | 0.01 | - | T.E. |
| R05U | G | 0.01 | G | 0.01 | - | T.E. |
| R06U | R | 0.01 | R | 0.01 | U | 0.14 |
| R07U | R | 0.01 | R | 0.01 | U | 0.17 |
| R08U | G | 0.01 | G | 0.01 | - | T.E. |
| R09U | S | 553.67 | S | **89.94** | - | T.E. |
| R10U | R | 0.01 | R | 0.01 | - | T.E. |

Figure 10: Results for the unsolvable problems (3)

| | **Unsolvable** | | | | **Tight** | | | |
|---|---|---|---|---|---|---|---|---|
| | TempLM-TLG | | TempLM-REF | | TempLM-TLG | | TempLM-REF | |
| | Exp | Gen | Exp | Gen | Exp | Gen | Exp | Gen |
| Driverlog | 202626 | 202626 | **91047** | **91047** | 78851 | 289003 | **51115** | **166063** |
| ZenoTravel | 122865 | 122865 | **12903** | **12903** | 30184 | 283006 | **14653** | **161912** |
| Satellite-TIL | 38222 | 38222 | **26060** | **26060** | 5090 | 191338 | **1260** | **17542** |
| Pipes-TIL | 100553 | 100553 | **18920** | **18920** | 100553 | 100553 | **18920** | **18920** |
| Pegsol | 59147 | 59147 | **56620** | **56620** | 16550 | 77714 | **13007** | **62190** |
| Floortile | 456292 | 456292 | **82441** | **82441** | 2513 | 9024 | **1844** | **6650** |
| Rovers | 151545 | 151545 | **8132** | **8132** | 2291 | 16769 | **1462** | **12670** |
| Depots | - | - | - | - | 1783 | 7016 | 1783 | 7016 |
| Logistics | 264420 | 264420 | **38203** | **38203** | 34034 | 127996 | **34022** | 127996 |

Table 1: Number of generated and expanded nodes for unsolvable and tight problems

be loose, tight or unsolvable. Table 3 shows the number of problems of each type solved by the three approaches.

As we can observe in Table 3, TempLM-TLG is the approach that solves more instances including the three problem types. OPTIC deals better with loose problems and its performance degrades as more tight deadlines are involved in the problems. In contrast, TempLM-REF behaviour is the opposite of OPTIC. The results of Table 3 also show that the performance of the three approaches vary along the domains. Thus, OPTIC results are better in domains where there exist several different plans for a problem such as Driverlog, Parking and Satellite domains. On the other hand, TempLM-TLG and TempLM-REF perform better in domains which involve many interactions among the goals such as Pegsol, Floortile, Depots or Logistics domains.

Figures 15 to 20 compare the makespan and computation time of the problems (loose and tight) that were solved by the three approaches. We selected three domains in which the approaches exhibit a different performance: Pegsol domain shows the worst computational time for OPTIC (Figure 16), TempLM-REF is outperformed by OPTIC

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | Mk | Time | Mk | Time | Mk | Time |
| **Depots** | | | | | | |
| DE01T | 27 | **0.21** | 27 | 0.53 | 27 | 2.31 |
| DE02T | 27 | **0.11** | 27 | 0.48 | 27 | 2.27 |
| DE03T | 27 | **0.11** | 27 | 0.49 | 27 | 2.31 |
| DE04T | **41** | **31.73** | **43** | 103.13 | 74 | 373.13 |
| DE05T | **31** | **128.05** | - | T.E. | - | T.E. |
| DE06T | **31** | **123.95** | - | T.E. | - | T.E. |
| DE07T | **36** | **75.18** | - | T.E. | - | T.E. |
| DE08T | **57** | **121.61** | - | T.E. | - | T.E. |
| DE09T | **57** | **120.74** | - | T.E. | - | T.E. |
| DE10T | **57** | **122.44** | - | T.E. | - | T.E. |
| **Logistics** | | | | | | |
| L01T | **43** | **2.10** | **43** | 0.61 | - | T.E. |
| L02T | **43** | **0.08** | **43** | 0.23 | - | T.E. |
| L03T | **43** | **0.08** | **43** | 0.25 | 56 | 4.08 |
| L04T | 22 | **0.03** | 22 | 0.06 | 22 | 1.59 |
| L05T | **43** | **0.07** | **43** | 0.18 | - | T.E. |
| L06T | **47** | **44.55** | **47** | 153.72 | - | T.E. |
| L07T | **77** | **40.69** | **77** | 130.34 | - | T.E. |
| L08T | **68** | **180.01** | **68** | 350.52 | - | T.E. |
| L09T | **56** | **6.02** | **56** | 18.33 | - | T.E. |
| L10T | **67** | **78.88** | **67** | 279.30 | - | T.E. |

Figure 11: Results for the tight problems (4)

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | | Time | | Time | | Time |
| **Depots** | | | | | | |
| DE01U | - | T.E. | S | **595,62** | - | T.E. |
| DE02U | G | **0.01** | G | **0.01** | - | T.E. |
| DE03U | - | T.E. | S | **628.37** | - | T.E. |
| DE04U | G | **0.01** | G | **0.01** | - | T.E. |
| DE05U | G | **0.01** | G | **0.01** | - | T.E. |
| DE06U | - | T.E. | S | **303.23** | - | T.E. |
| DE07U | G | **0.01** | G | **0.01** | - | T.E. |
| DE08U | G | **0.01** | G | **0.01** | - | T.E. |
| DE09U | G | **0.01** | G | **0.01** | - | T.E. |
| DE10U | G | **0.01** | G | **0.01** | - | T.E. |
| **Logistics** | | | | | | |
| L01U | G | 0.01 | G | 0.01 | - | T.E. |
| L02U | S | 59.35 | S | **27.36** | - | T.E. |
| L03U | R | 0.01 | R | 0.01 | - | T.E. |
| L04U | G | 0.01 | G | 0.01 | - | T.E. |
| L05U | S | 62.56 | S | 20.55 | - | T.E. |
| L06U | G | 0.01 | G | 0.01 | - | T.E. |
| L07U | S | 328.85 | S | **23.80** | - | T.E. |
| L08U | S | 489.02 | S | **30.79** | - | T.E. |
| L09U | S | 161.89 | S | **9.78** | - | T.E. |
| L10U | S | 161.50 | S | **10.03** | - | T.E. |

Figure 12: Results for the unsolvable problems (4)

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | Mk | Time | Mk | Time | Mk | Time |
| **Sat TIL** | | | | | | |
| S01T | 176.69 | **0.4** | 176.69 | 1.5 | 176.69 | 2.47 |
| S02T | 210.47 | 0.86 | 210.47 | 3.12 | **191.28** | 6.2 |
| S03T | 106.77 | **0.64** | 106.77 | 2.46 | 106.77 | 4.53 |
| S04T | 207.18 | 35.13 | **165.92** | 45.84 | 169.18 | 72.85 |
| S05T | **180.46** | 34.95 | **180.46** | 124.30 | 183.91 | 16.28 |
| S06T | - | T.E. | **145.72** | 307.89 | 183.91 | 16.28 |
| S07T | **134.38** | 1467 | - | T.E. | 140.13 | 13.85 |
| S08T | - | T.E. | - | T.E. | **119.94** | **35.2** |
| S09T | 176.69 | **0.45** | 176.69 | 0.65 | 176.69 | 2 |
| S10T | **189.77** | **2.89** | **189.77** | 8.34 | - | T.E. |

Figure 13: Results for the tight problems (5)

| | TempLM-TLG | | TempLM-REF | | OPTIC | |
|---|---|---|---|---|---|---|
| | | Time | | Time | | Time |
| **Sat TIL** | | | | | | |
| S01U | R | 0.01 | R | 0.01 | U | 0.28 |
| S02U | S | **914** | S | 1425 | - | T.E. |
| S03U | R | 0.01 | R | 0.01 | U | 0.13 |
| S04U | G | **0.01** | G | **0.01** | - | T.E. |
| S05U | R | 0.01 | R | 0.01 | U | 0.17 |
| S06U | R | **0.01** | R | **0.01** | - | T.E. |
| S07U | R | 0.01 | R | 0.01 | U | 0.2 |
| S08U | G | **0.01** | G | **0.01** | - | T.E. |
| S09U | S | **0.07** | S | 0.1 | - | T.E. |
| S10U | R | 0.01 | R | 0.01 | U | 0.11 |

Figure 14: Results for the unsolvable problems (5)

in Driverlog (Figure 18) and Rovers domain shows similar results for both approaches (Figure 20). With respect to makespan, TempLM-REF always returns the best results. The values in these figures show a similar trend as the results in Figures 5, 7, 9, 11 and 13. To sum up, we can draw the following conclusions:

1. Makespan: TempLM-TLG and TempLM-REF show almost identical values and they both clearly outperform OPTIC
2. Computation time: the tendency is consistent with the data of Table 3; that is, OPTIC shows a better behaviour in those domains in which it solves more instances (Driverlog and Satellite) and likewise for TempLM in the rest of domains. The time of TempLM-TLG is generally below TempLM-REF except in the Satellite domain where the TLG refinement is very efficient in pruning useless nodes.

These results confirm the conclusions extracted from our first analysis. That is, TempLM-REF is the approach that works better with problems with deadlines.

| Domain | Total Generated | Total Solved | Total Loose | Total Tight | Total Unsolvable |
|---|---|---|---|---|---|
| Driverlog | 60 | 30 | 22 | 6 | 2 |
| ZenoTravel | 60 | 22 | 11 | 10 | 1 |
| Parking | 60 | 31 | 22 | 9 | 0 |
| Pegsol | 60 | 60 | 37 | 15 | 8 |
| Floortile | 40 | 15 | 11 | 4 | 0 |
| Rovers | 51 | 46 | 31 | 15 | 0 |
| Satellite | 60 | 20 | 10 | 10 | 0 |
| Depots | 66 | 30 | 10 | 10 | 10 |
| Logistics | 64 | 46 | 22 | 14 | 10 |
| **Total** | 521 | 300 | 176 | 93 | 31 |

Table 2: Classification of problems and number of solved problems

| Domain | Loose | | | Tight | | | Unsolvable | | |
|---|---|---|---|---|---|---|---|---|---|
| | TempLM-TLG | TempLM-REF | OPTIC | TempLM-TLG | TempLM-REF | OPTIC | TempLM-TLG | TempLM-REF | OPTIC |
| Driverlog | 16 | 14 | 18 | 6 | 6 | 6 | 2 | 2 | 1 |
| ZenoTravel | 10 | 9 | 9 | 11 | 10 | 8 | 1 | 1 | 1 |
| Parking | 4 | 3 | 22 | 5 | 4 | 6 | 0 | 0 | 0 |
| Pegsol | 37 | 32 | 23 | 12 | 12 | 15 | 8 | 7 | 0 |
| Floortile | 11 | 11 | 0 | 4 | 4 | 0 | 0 | 0 | 0 |
| Rovers | 24 | 23 | 24 | 14 | 13 | 13 | 0 | 0 | 0 |
| Satellite | 7 | 6 | 9 | 9 | 7 | 8 | 0 | 0 | 0 |
| Depots | 7 | 2 | 5 | 10 | 4 | 5 | 7 | 10 | 0 |
| Logistics | 22 | 22 | 7 | 14 | 14 | 3 | 10 | 10 | 0 |
| **Total** | 138 | 122 | 117 | 85 | 74 | 64 | 28 | 31 | 2 |

Table 3: Number of solved problems for each domain

**Third experiment.** In the recent ICAPS-2015, another approach that also uses temporal landmarks was presented by Karpas et al. (2015). Unlike TempLM, which is mainly focused on solving overconstrained temporal problems, this new approach is focused on solving concurrent temporal problems. Two types of temporal landmarks are defined: temporal action landmarks and temporal fact landmarks. A temporal action landmark consists of a set of events (start/end of actions) and the time point in which one of these events must occur. On the other hand, temporal fact landmarks are a set of propositions which must hold for some time. Moreover, a Simple Temporal Network (STN) with a set of simple temporal constraints between the time points associated with the landmarks is maintained. This information is then compiled into the domain and problem files and can be later used by any temporal planner. The empirical results presented in Karpas et al. (2015) show that temporal landmarks are very helpful for planners to solve problems with complex temporal interactions. However, according to our experience, this approach is not that efficient when solving overconstrained problems or detecting unsolvable problems. We designed a third experiment in which we used the approach of Karpas et al. (2015) to extract the temporal landmarks of the following instances of the Driverlog, Zeno and Floortile domains: D03T, D04U, Z05T, Z07U, F03T, F05U and F06U from Tables 5, 9, 6 and 10. After running OPTIC with the seven compiled domain and problem files, we found that it could only solve instance F03T.

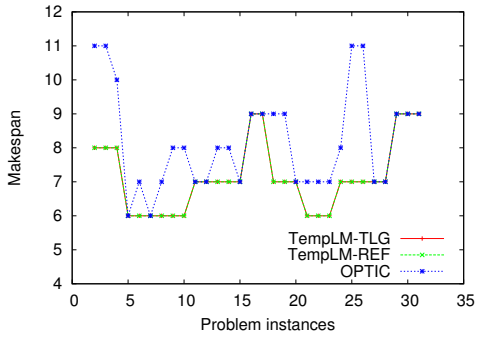


Figure 15: Pegsol (makespan).

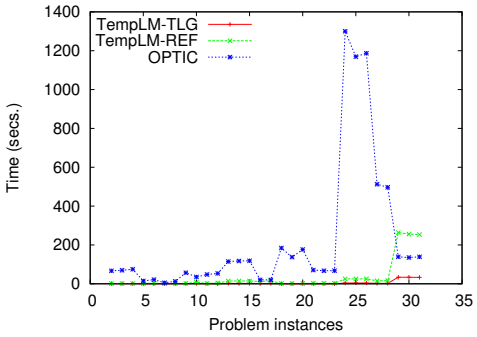


Figure 16: Pegsol (time).

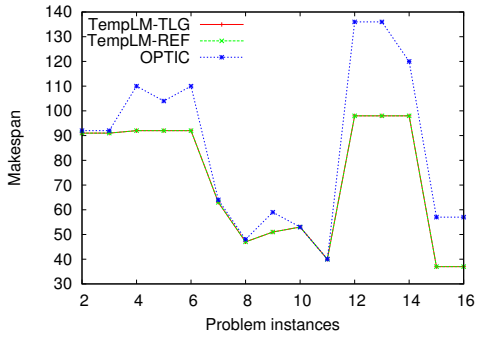


Figure 17: Driverlog (makespan).

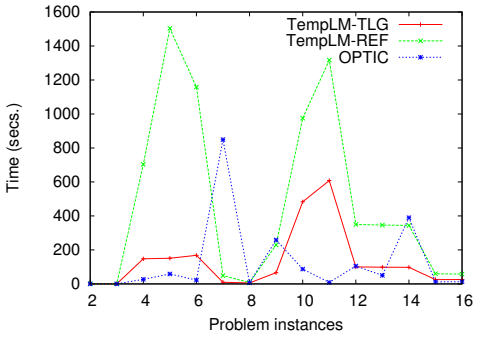


Figure 18: Driverlog (time).

## 8. Conclusion

This paper presents TempLM, a temporal planning approach for solving planning problems with deadlines that draws upon the concept of temporal landmark. A temporal landmark is a proposition that is found to necessarily happen in a solution plan in order to satisfy the deadlines of the problem goals. The set of temporal landmarks extracted from a problem along with their relationships form a temporal landmarks graph (TLG) that is conveniently used to take decisions during the construction of the solution plan and for guiding the search process. A key feature of our approach is that as long as new actions are inserted in a plan and decisions about how to achieve subgoals are taken, new temporal landmarks appear as a consequence of these decisions. This way, the particular action scheduling of a partial plan in the search space determines the future propositions to be achieved and when these propositions must be achieved, updating the TLG accordingly. Our TempLM-REF approach exploits this feature, named TLG refinement, in all the nodes of the search tree resulting in a very competitive approach that exhibits an excellent behaviour in the detection of unsolvable problems and the resolution of overconstrained problems. A rapid detection of unsolvability in overconstrained temporal problems is a relevant issue in many real-world problems that has been traditionally neglected in the temporal planning community. Even there exist

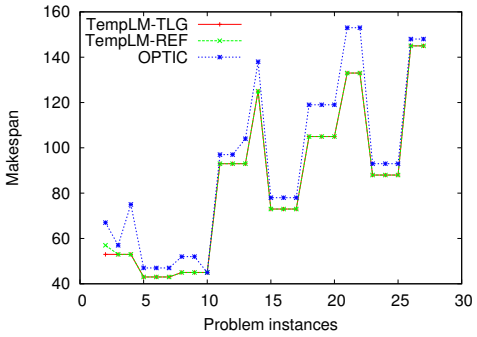


Figure 19: Rovers (makespan).

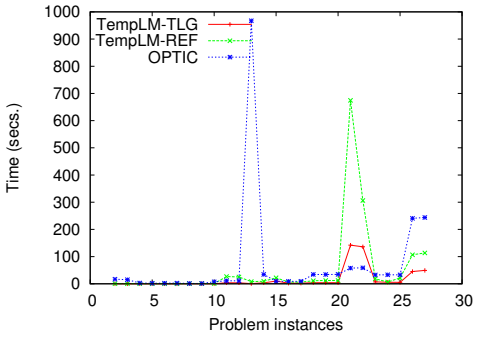


Figure 20: Rovers (time).

approaches that also make use of temporal landmarks, they are more focused on solving complex temporal problems rather than identifying unsolvability. For all this, we think our proposal represents a first solid step in the detection of unsolvable temporal planning problems with deadline constraints.

## Acknowledgements

We thank Derek Long for solving our doubts about the modal operators in PDDL3 and Erez Karpas for supplying the compiled domain and problem files with their temporal landmarks. This work has been partially supported by Spanish Government Project MINECO TIN2014-55637-C2-2-R.

## References

Benton, J., Coles, A. J., Coles, A., 2012. Temporal planning with preferences and time-dependent continuous costs. In: Proc. Int. Conference on Automated Planning and Scheduling.

Blum, A., Furst, M., 1997. Fast planning through planning graph analysis. Artificial Intelligence 90 (1-2), 281–300.

Chen, Y., Wah, B. W., Hsu, C.-W., 2006. Temporal planning using subgoal partitioning and resolution in SGPlan. Journal of Artificial Intelligence Research 26, 323–369.

Edelkamp, S., Jabbar, S., Nazih, M., 2006. Large-scale optimal pddl3 planning with mips-xxl. In: 5th International Planning Competition Booklet. pp. 28–30.

Fox, M., Long, D., 2003. PDDL 2.1 : An extension to pddl for expressing temporal planning domains. Journal of Artificial Intelligence Research 20, 61–124.

Garrido, A., Fox, M., Long, D., 2002. A temporal planning system for durative actions of pddl2. 1. Proc. of the European Conference on Artificial Intelligence, 586–590.

Gerevini, A., Haslum, P., Long, D., Saetti, A., Dimopoulos, Y., 2009. Deterministic planning in the 5th International Planning Competition: PDDL3 and experimental evaluation of the planners. Artificial Intelligence 173 (5-6), 619–668.

Helmert, M., Domshlak, C., 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In: Proc. Int. Conference on Automated Planning and Scheduling.

Hoffmann, J., Edelkamp, S., 2005. The deterministic part of ipc-4: An overview. Journal of Artificial Intelligence Research 24, 519–579.

Hoffmann, J., Porteous, J., Sebastia, L., 2004. Ordered landmarks in planning. Journal of Artificial Intelligence Research 22, 215–287.

Karpas, E., Wang, D., Williams, B. C., Haslum, P., 2015. Temporal landmarks: What must happen, and when. In: Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015. pp. 138–146.

Long, D., Fox, M., 2003. The 3rd international planning competition: Results and analysis. Journal of Artificial Intelligence Research 20, 1–59.

Marzal, E., Sebastia, L., Onaindia, E., 2008. Detection of unsolvable temporal planning problems through the use of landmarks. In: Proc. of the European Conference on Artificial Intelligence. pp. 919–920.

Marzal, E., Sebastia, L., Onaindia, E., 2011. Full Extraction of Landmarks in Propositional Planning Tasks. In: Symposium of the Italian Association for Artificial Intelligence. Vol. 6934. pp. 383–388.

Marzal, E., Sebastia, L., Onaindia, E., 2014. On the use of temporal landmarks for planning with deadlines. In: Proc. of the 24th International Conference on Automated Planning and Scheduling. AAAI Press, pp. 172–180.

Marzal, E., Sebastia, L., Onaindia, E., 2015. Temporal landmarks for overconstrained planning problems with deadlines. In: Proc. of the 27th IEEE International Conference on Tools with Artificial Intelligence. IEEE, p. in press.

Rossi, F., van Beek, P., Walsh, T. (Eds.), 2006. Handbook of Constraint Programming. Vol. 2 of Foundations of Artificial Intelligence. Elsevier.

Sen, D., 2010. Advances in Fish Processing Techology. Allied Publishers PVT LTD.