Handling PDDL3.0 State Trajectory Constraints with Temporal Landmarks

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

Abstract

Temporal landmarks have been proved to be a helpful mechanism to deal with temporal planning problems, specifically to improve planners performance and handle problems with deadline constraints. In this paper, we show the strength of using temporal landmarks to handle the state trajectory constraints of PDDL3.0. We analyze the formalism of TempLM, a temporal planner particularly aimed at solving planning problems with deadlines, and we present a detailed study that exploits the underlying temporal landmark-based mechanism of TempLM for representing and reasoning with trajectory constraints.

Introduction

In planning, a *landmark* is a fact that must be true in any solution plan. Since the influential work presented in (Hoffmann, Porteous, and Sebastia 2004) on the use of landmarks in planning, there have been multiple investigations that exploit landmarks for cost-optimal planning (Helmert and Domshlak 2009; Karpas and Domshlak 2009), satisficing planning (Richter and Westphal 2010) or more recently on goal recognition (Pereira, Oren, and Meneguzzi 2017).

Although the use of landmarks in temporal contexts has been less explored, there are two works that address the exploitation of temporal landmarks in planning. In (Karpas et al. 2015), the authors define temporal fact landmarks (facts that must hold between two time points) and temporal action landmarks, which state that some event (the start or end of an action) must occur at some time point. The temporal information is captured in a Simple Temporal Network (Dechter, Meiri, and Pearl 1991) over the symbolic time points associated with each landmark. This approach is aimed at deriving temporal landmarks and constraints from planning problems and integrating them into domain-independent temporal planners in order to improve their performance. Overall, the results reveal that there is some benefit from using temporal landmarks in concurrent temporal planning problems like the TMS domain and in some non-temporally expressive domains in terms of solution quality.

Differently, the approach presented in (Marzal, Sebastia, and Onaindia 2014; 2016), called TempLM, studies the use of temporal landmarks for solving planning problems with deadline constraints. In this approach, a temporal landmark is defined as a fact that must be achieved in a solution plan to satisfy the problem deadline constraints. TempLM discovers the causal (non-temporal) landmarks of the problem and then exploits deadlines to infer new (temporal) landmarks that must be accomplished to meet the deadlines. The limitation of TempLM is that it relies on the specification of deadlines for the goal propositions of the problem and it requires an upper time bound for the plan, which is automatically derived from the problem deadlines. This way, when there are no deadlines in the problem, a sufficiently large estimated value must be provided as upper bound of the plan. TempLM shows an excellent performance in overconstrained problems with tight deadlines, which would clearly degrade with larger plan upper bounds.

Nonetheless, the key contribution of TempLM is its internal representation of temporal landmarks, which can be easily used to handle all the state trajectory constraints defined in PDDL3.0 (Gerevini et al. 2009). In this paper, we show that the interval representation of temporal landmarks of TempLM, along with the constraints defined between landmarks and their intervals, constitute a very suitable framework for representing and reasoning with trajectory constraints. The next section summarizes the main features of TempLM, highlighting the representation of a landmark and the propagation of temporal constraints. The following section presents the 10 modal operators that specify the state trajectory constraints in PDDL3.0; for each operator, we show the corresponding landmark representation and the set of constraints that would be needed in TempLM in order to account for the constraint. Subsequently, we present an illustrative example that shows the powerful inference engine of TempLM when handling trajectory constraints. In the last section, we discuss the advantages and limitations of TempLM and we stress the utilization of temporal landmarks for representing other constraints in temporal planning.

Overview of TempLM

TempLM is a framework specifically aimed at solving temporal planning problems with deadline constraints (Marzal, Sebastia, and Onaindia 2014; 2016). It assumes 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) as well as the

`within` constraint introduced in PDDL3.0 (Gerevini et al. 2009).

A **temporal planning problem with deadline constraints** is a tuple $\mathcal{P} = \langle P, O, I, G, D \rangle$, where P is a set of propositions, I is the initial state, O is a set of durative actions in PDDL2.1, I is the initial state, G is a goal description and D is a set of deadline constraints of the form (p, t), denoting that proposition p must be achieved within t time units. 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).

A **temporal plan** Π is a set of pairs (a, t), where $a \in O$ and t is the start execution time of a. For a given proposition p, $start(p)$ and $end(p)$ denote the time points when p is asserted and deleted, respectively, by any action a in Π. The duration (makespan) of a temporal plan Π is $dur(\Pi) = \max_{\forall (a,t) \in \Pi} \big(t + dur(a)\big)$. Additionally, the upper bound of the temporal plan Π is set as $T_\Pi = \max_t (p, t)\ , \forall (p, t) \in D$.

TempLM extracts first the non-temporal landmarks of a problem $\mathcal{P}$ (Hoffmann, Porteous, and Sebastia 2004) and then discovers a new set of (temporal) landmarks through the deadline constraints in D. A **temporal landmark** of a problem $\mathcal{P}$ is a proposition of P that must hold in every plan that solves $\mathcal{P}$ in order to satisfy D. In this paper, we focus exclusively on describing the internal representation of temporal landmarks in TempLM. We refer the reader to the works in (Marzal, Sebastia, and Onaindia 2014; 2016) for details of the landmark extraction process.

Temporal Landmarks

We introduce a running example on the classical `depots` domain (Figure 1) in order to show the relevant information of the temporal landmarks. The scenario consists of a depot `D0`, where the pallet `P0` and the truck `T0` are located; a crate `C0` is in pallet `P0`. There is also a distributor `D1` which contains create `C1` in pallet `P1`, and a distributor `D2` that contains the pallets `P2` and `P3`. Finally, the crate `C2` is in pallet `P4` located in distributor `D3`. The time a truck takes to travel between depots or distributors is shown on the edges. For simplicity, the hoists of the original domain have been eliminated: crates can be loaded into the truck as long as they are clear (have nothing on top) and can be unloaded from the truck to be put on top of another crate or onto a pallet.

Landmarks are characterized by their temporal intervals and relationships between them. Landmarks form a **Temporal Landmarks Graph (TLG)**, a directed graph $G = (V, E)$ where V are landmarks and E is a set of ordering constraints of the form $l_i \prec_n l_j$ or $l_i \prec_d l_j$ that denote a *necessary* or *dependency* ordering, respectively, meaning that l_i must happen before landmark l_j in every solution plan.

Let's assume the goal of the problem is `((at C0 D2),40)`, which will be specified as `(within 40 (at C0 D2))`. Then, `(at C0 D2)` is a temporal landmark and likewise all the propositions of the initial state and goal state. In this situation, a dependency ordering `(at T0 D0)` $\prec_d$ `(at T0 D2)` is established, which denotes that even though there are two possible ways of reaching the goal - through distributor `D1` or `D3`- `(at T0 D0)` must always be satisfied before `(at T0 D2)` in any solution plan. However, if the goal were `((at C0 D2),25)` then the TLG would contain the necessary ordering `(at T0 D0)` $\prec_n$ `(at T0 D3)` because in this case it is mandatory that truck `T0` travels through `D3` to reach the goal in time. Another interesting aspect is that since the surface which `C0` must be stacked on is not known, the propositions `(on C0 P2)` and `(on C0 P3)` are not landmarks.

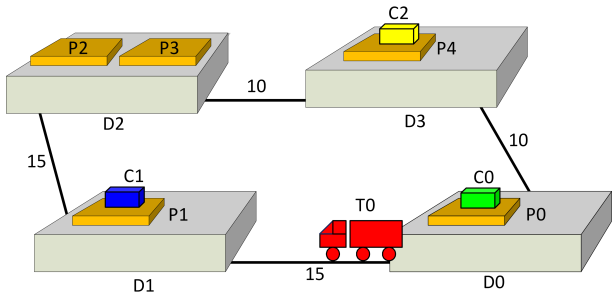


Figure 1: Illustrative example

Landmarks are also annotated with various *temporal intervals* that represent the validity of the corresponding temporal proposition (Marzal, Sebastia, and Onaindia 2008). Three types of intervals are identified:

- The **generation interval** of a landmark l is denoted by $[min_g(l), max_g(l)]$. $min_g(l)$ represents the earliest time point when landmark l can start in the plan. This value is determined by the time of the first proposition layer when l appears in a Temporal Relaxed Planning Graph (TRPG). $max_g(l)$ represents the latest time point when l must start in order to satisfy the deadlines D of a problem $\mathcal{P}$ 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 hold in the plan. Initially, this interval is set as $min_v(l) = min_g(l)$ and $max_v(l) = T_\Pi$.
- 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 $max_n(l) = T_\Pi$.

Let us assume the `load` and `unload` actions in the example of Figure 1 have a duration of two time units each; and that the problem goal is $g =$ `((at C0 D2),25)`, being this the only deadline constraint of the problem. Figure 2 shows the initial TLG for this goal. Thus, $T_\Pi = 25$ and the generation interval of g is:

- $max_g(g) = 25$ because the latest time at which g must be generated in order to satisfy the deadline is 25

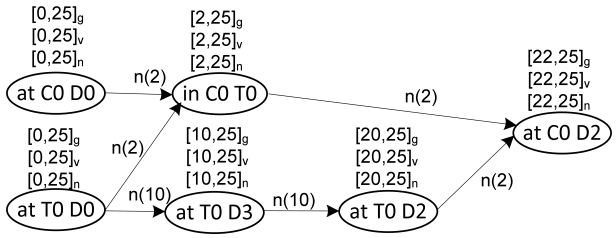


Figure 2: Initial TLG for the goal `(within 25 (at C0 D2))`

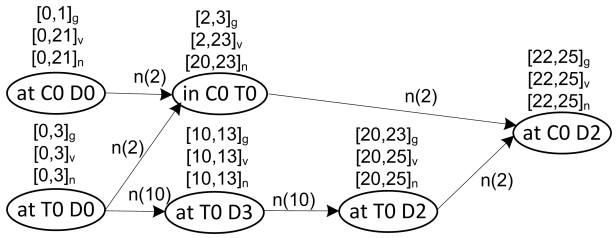


Figure 3: TLG for the goal `(within 25 (at C0 D2))` after constraint propagation

- $min_g(g) = 22$ because the first appearance of `(at C0 D2)` in the TRPG is at level 22: 20 (shortest route) + 2 (`unload`) (note that the first TRPG layer that contains the effects of `load` is at level 2).

For landmark l_1 = `(at T0 D0)` we have that $min_g(l_1) = 0$ and $max_g(l_1) = 25$; for landmark l_2 = `(at T0 D3)`, $min_g(l_2) = 10$ and $max_g(l_2) = 25$; and for l_3 = `(at T0 D2)`, $min_g(l_3) = 20$ and $max_g(l_3) = 25$. Likewise, the validity intervals would initially take on the same values as for the generation intervals.

Propagation of temporal constraints

Once the intervals of the temporal landmarks are initialized in the TLG, constraints are propagated and the landmark intervals are updated accordingly.

Causal relationships. The ordering constraints $l_i \prec_n l_j$ or $l_i \prec_d l_j$ represent causal relationships, where $l_i \in Cond(a)$ and $l_j \in Eff(a')$ for two actions $a, a' \in O$. If $a = a'$ then it is a direct causal relationship represented by $l_i \prec_n l_j$. In any other case, $l_i \prec_d l_j$ represents an indirect causal relationship that involves more than one action. The necessary and dependency orderings are transitively propagated across the TLG creating further constraints. Particularly, for two landmarks involved in a causal relationship, a separating temporal distance between the time point when l_i is required and the time point when l_j is needed is defined according to the duration of the action(s) involved in the causal transition. Restricting our attention to the simple case when $a = a'$, we have that: (a) $dist(l_i, l_j) = dur(a)$ if $l_i \in SCond(a)$ and $l_j \in EEff(a)$; (b) $dist(l_i, l_j) = \epsilon$ if $l_i \in ECond(a)$ and $l_j \in EAdd(a)$; and (c) $dist(l_i, l_j) = -dur(a)$ if $l_i \in ECond(a)$ and $l_j \in SAdd(a)$[1].

In the initial TLG for the goal g = `((at C0 D2),25)` shown in Figure 2, we can observe that nodes are landmarks labeled with the three temporal intervals and edges are labeled with a necessary or dependency ordering constraint plus a temporal distance. For example, `(in C0 T0)` $\prec_{n(2)}$ `(at C0 D2)` means it is necessary to have the crate `C0` into the truck `T0` at least 2 time units (duration of the action `unload`) before having the crate `C0` at `D2`. In this case, `(in C0 T0)` is a $SCond$ and `(at C0 D2)` is an $EEff$ of the same `unload` action, respectively. Figure 2 does not picture any dependency ordering because the deadline to have `C0` at `D2` is 25, which compels `T0` to reach `D2` via `D3`. However, the TLG for the goal `(within 40 (at C0 D2))` shown in Figure 4 pictures a dependency ordering `(at T0 D0)` $\prec_{d(20)}$ `(at T0 D2)` . The distance 20 denotes that `T0` must be in `D0` 20 times units before reaching `D2`, which is the minimal distance to reach `D2` from `D0`. Given that the deadline for the goal is 40 in this case, `T0` can reach `D2` through `D3` or `D1` but this information is not known yet. This is the reason of the dependency ordering in Figure 4, which means that at least two `drive` actions are involved in this causal relationship.

We apply an *interval constraint propagation* that restricts the domain of the temporal intervals accordingly to the type of interval and the distance of the causal relationship. The min endpoints of the intervals are propagated forward in time and the max endpoints are propagated backward along time. A causal relationship of the form $l_i \prec_{\{n,d\}} l_j$ between two landmarks l_i and l_j (l_i is required to generate l_j) implicitly defines the following interval constraints:

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

Thus, the min_v of l_j is subject to the minimum validity of l_i plus the duration of the action(s) that separates both landmarks. Likewise, the latest time when l_i must start in the plan depends on the latest time when l_j is required minus the temporal distance determined by the duration of the action(s) that are needed to generate l_j from l_i. Back to the example of Figure 1 with goal g = `((at C0 D2),25)`, TempLM will update the generation intervals of the landmarks as shown in Table 1. Note that the order of the max_g propagation goes backwards from `D2` (the destination depot) through `D3` to finally reach `D0`. The final TLG after propagation is shown in Figure 3.

Mutex relationships. Given $l_i \prec_{\{d,n\}} l_j$, if l_i and l_j are mutex (Blum and Furst 1997) then l_i and l_j cannot overlap in any way. The propagation of the mutex relationships updates $max_v(l_i)$ to ensure l_i does not overlap with l_j. Thus, $max_v(l_i)$ is updated to the minimum value among the current validity endpoints of l_i and the latest time when l_j must start in the plan minus the temporal distance between both landmarks landmarks. Particularly:

[1]The definition of $dist(l_i, l_j)$ is also dependent on the first or the last time when l_i is required. For the sake of simplicity, we define a general concept of distance. More details on this can be found in (Marzal, Sebastia, and Onaindia 2014).

Causal constraint	Interval update
(at T0 D2) $\prec_{n(2)}$ (at C0 D2)	max_g(at T0 D2) = $25 - 2 = 23$
(at T0 D3) $\prec_{n(10)}$ (at T0 D2)	max_g(at T0 D3) = $23 - 10 = 13$
(at T0 D0) $\prec_{n(10)}$ (at T0 D3)	max_g(at T0 D0) = $13 - 10 = 3$

Table 1: Update of generation intervals for the goal (within 25 (at C0 D2))

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

Mutex landmarks	Interval update
((at T0 D3),(at T0 D2))	max_v(at T0 D3) = $\min(25, 20, 23 - 10) = 13$
((at T0 D0),(at T0 D3))	max_v(at T0 D0) = $\min(25, 10, 13 - 10) = 3$

Table 2: Update of validity intervals

Search

TempLM searches in the space of *partial plans*. Nodes are represented by a pair (Π, S_t), where Π is a conflict-free partial plan and S_t is the state reached at time $t = dur(\Pi)$ after executing Π in I. Nodes are expanded by finding the earliest start time of the set of applicable actions in S_t. Each node is associated to a TLG. A newly inserted action may cause the appearance of new temporal landmarks in the TLG of a node and the propagation of the temporal constraints will update the landmarks intervals. As a result of the propagation, if for a given deadline (p, t) it turns out that $max_g(p) > t$ or some inconsistency is found in the endpoints of the landmarks intervals, the node is pruned.

State trajectory constraints in PDDL3.0

PDDL3.0 is the language used at the Fifth International Planning Competition (IPC-2006)[2]. This extended language introduces new expressive functionalities such as strong and soft *constraints on plan trajectories* and soft problem goals or *preferences* (Gerevini et al. 2009). In this work, we will exclusively focus on the *strong state trajectory constraints* and we will analyze how these are represented, interpreted and handled when using temporal landmarks.

State trajectory constraints are used to express conditions that must be met by the entire sequence of states visited during the execution of a plan. They are expressed through temporal modal operators over first order formulae involving state predicates. Actually, all the constraints expressed with the temporal modal operators of PDDL3.0 specify a temporal interval at which the state predicate must hold along the sequence of states of the plan execution. In the following, we analyze the semantics of the ten modalities of constraints introduced in PDDL3.0 as well as how they would be encoded in a framework based on temporal landmarks.

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

The operator `at end`

The syntax of this constraint is (at end <GD>), where <GD> is a goal descriptor (a first-order logic formula). It is used to identify conditions that must hold in the final state when the plan has been executed, making them equivalent to traditional goal conditions. Whenever a goal condition with no temporal modal operator is specified in a problem file, it is assumed to be an (at end) condition, thus preserving the standard meaning for existing goal specifications. The semantics of this operator is shown in formula 1.

$$\langle (S_0, 0), (S_1, t_1), \ldots, (S_n, t_n) \rangle \models (\texttt{at end}\ \phi) \\ \text{iff } S_n \models \phi \quad (1)$$

For example, (at end (at truck1 cityA)) indicates that truck1 must be in cityA at the goal state. The operator (at end ϕ) defines an interval $[t_i, t_j]$ for the occurrence of the goal condition ϕ such that $0 \leq i \leq n$ and $j = n$. Note that the fulfillment of ϕ throughout $[t_i, t_j]$ does not necessarily imply that there must be a single occurrence of ϕ. Particularly, the expression (at end ϕ) refers to the last appearance of ϕ in the plan so that the constraint will be met for such occurrence of ϕ as long as $j = n$.

Taking into account the above considerations, a constraint of the form (at end ϕ) allow us to make the following implications regarding the information of temporal landmarks:

1. ϕ is a temporal landmark since it is a mandatory condition to be satisfied in a particular time interval
2. given that ϕ is needed at the end, $max_n(\phi) = t_n$, which implies that $max_v(\phi) = t_n$ because $max_n(\phi) \leq max_v(\phi)$
3. the non-compliance of the constraint in the TLG of a node cannot be used to prune partial plans during search. Only when the plan is complete, the non-compliance of this constraint will be used to discard a plan as a valid solution.

The operator `always`

The semantics of a constraint (always <GD>) is shown in formula 2, which expresses that the goal condition must hold in every state in order for the modal formula to hold over the trajectory.

$$\langle (S_0, 0), (S_1, t_1), \ldots, (S_n, t_n) \rangle \models (\texttt{always}\ \phi) \\ \text{iff } \forall i : 0 \leq i \leq n \cdot S_i \models \phi \quad (2)$$

A constraint (always ϕ) expresses that the goal condition ϕ must be true throughout the plan. For instance, if the problem requires to have a pallet1 clear all the time, we will use the constraint (always (clear pallet1)). Thereby, the expression (always ϕ) defines a temporal interval $[t_i, t_j]$ for ϕ such that $i = 0$ and $j = n$. In this case,

it is clear that there must be a single occurrence of ϕ that holds over $[t_i, t_j]$.

The temporal landmark information that can be inferred through a constraint `(always` ϕ`)` is the following:

1. ϕ is a temporal landmark as the constraint denotes a proposition that must be true in every solution plan over the interval $[0, t_n]$
2. ϕ is needed throughout the interval $[0, t_n]$ so $min_n(\phi) = 0$ and $max_n(\phi) = t_n$, which in turn implies that $min_v(\phi) = 0$ and $max_v(\phi) = t_n$ because $[min_n(\phi), max_n(\phi)] \in [min_v(\phi), max_v(\phi)]$
3. it allows pruning a search node when adding a new action in its TLG entails a modification of the necessity or validity interval of ϕ. For example, if the TLG of a node contains `(always (clear pallet1))` and an action $\alpha =$ `(drop P1 T1 pallet1 distributor)` is added to the node, then the node will be pruned because the value of $max_v($`(clear pallet1)`$)$ is modified since α deletes `(clear pallet1)`. Additionally, any partial plan of the tree which does not contain ϕ will be also pruned.

The operator `at-most-once`

The syntax of this operator is `(at-most-once <GD>)` and the semantics is expressed in the formula 3.

$$\begin{aligned}&\langle (S_0,0),(S_1,t_1),\ldots,(S_n,t_n)\rangle \models (\texttt{at-most-once}\ \phi)\\ &\text{iff } \forall i: 0 \le i \le n \cdot\\ &\quad \text{if } S_i \models \phi \text{ then } \exists j:\ j \ge i \cdot \forall k: i \le k \le j \cdot S_k \models \phi\\ &\quad \text{and } \forall k: k > j \cdot S_k \models \neg\phi\end{aligned} \tag{3}$$

A constraint `(at-most-once` ϕ`)` obviously denotes that ϕ must occur `at most once` in the plan, if any. That is, this constraint does not impose a mandatory occurrence of ϕ but if it happens then only a single occurrence of ϕ must appear in the plan. Consequently, the single occurrence of ϕ will be valid over an interval $[t_i, t_j]$, where $0 \le i \le n$ and $j \ge i$.

Regarding a temporal landmark representation, a constraint of the form `(at-most-once` ϕ`)` leads to the following derivations:

1. ϕ cannot be labeled as a temporal landmark since a mandatory occurrence is not imposed
2. if ϕ is a landmark then we know that $max_g(\phi) \le t_n$, which indicates that ϕ must be obtained before completion of the plan
3. it prevents having more than one occurrence of ϕ so any node that violates this condition will be pruned. This has some implications when solving conflicts that involve adding a new occurrence of ϕ. For example, let's assume that $max_v(\phi) = d$ and $max_n(\phi) = d'$ such that $d' > d$. In this case, a new occurrence of ϕ is needed to satisfy the necessity interval. This conflict is solvable in TempLM by introducing another instance of ϕ (ϕ') as long as ϕ' is consistent with the intervals of the rest of the landmarks in the TLG of the node (Marzal, Sebastia, and Onaindia 2016). However, such a conflict would be unsolvable if a constraint `(at-most-once` ϕ`)` exists in the planning problem specification.

The operator `sometime`

The semantics of a constraint `(sometime <GD>)` is presented in formula 4. As the name and semantics express, a constraint `(sometime` ϕ`)` indicates that ϕ must occur *at least once* in the plan. Every single occurrence of ϕ must hold over an interval $[t_i, t_j]$, where $0 \le i \le n$ and $j \ge i$.

$$\begin{aligned}&\langle (S_0,0),(S_1,t_1),\ldots,(S_n,t_n)\rangle \models (\texttt{sometime}\ \phi)\\ &\text{iff } \exists i: 0 \le i \le n \cdot S_i \models \phi\end{aligned} \tag{4}$$

A constraint of the form `(sometime` ϕ`)` allows us to derive the following information related to temporal landmarks:

1. ϕ is a temporal landmark as it must necessarily occur in the plan at least once
2. it must hold $max_g(\phi) \le t_n$ to ensure that ϕ occurs at least in the last state of the plan trajectory
3. similarly to the `(at-end` ϕ`)` constraint, the non-compliance of this constraint in the TLG of a node cannot be used to prune nodes during search. Once the plan construction is finished, we will be able to discard it as a valid solution in case ϕ never holds in the plan.

The operator `within`

The operator `within` is used to express deadlines. The syntax of this operator is `(within <num> <GD>)`, where `<num>` is any numeric literal (in STRIPS domains it will be restricted to integer values) and `<GD>` has the same meaning as in all the previous operators. The semantics associated to this operator is shown in formula 5.

$$\begin{aligned}&\langle (S_0,0),(S_1,t_1),\ldots,(S_n,t_n)\rangle \models (\texttt{within}\ t\ \phi)\\ &\text{iff } \exists i: 0 \le i \le n \cdot\ S_i \models \phi \wedge t_i \le t\end{aligned} \tag{5}$$

For example, `(within 10 (at T0 D3))` specifies that truck `T0` must be in depot `D3` by time 10 at the latest. The semantics of the operator `within` does not state the specific occurrence of the goal to which the constraint is applied in case that `(at T0 D3)` is achieved more than once in the plan. More specifically, the definition states that, if a goal is achieved more than once in the plan, it suffices one appearance of `(at T0 D3)` to fulfill the `within` constraint. On the other hand, there is no indication in the semantics that the goal condition must persist until the goal state; that is, the above constraint is satisfied as long as `(within 10 (at T0 D3))` is met in the plan irrespective of the final location of truck `T0`.

The information of temporal landmarks that can be derived from a constraint `(within t` ϕ`)` (for a particular occurrence of ϕ) is:

1. ϕ is a temporal landmark as it must necessarily occur in the plan at least once

2. it must always be true that $max_g(\phi) \leq t$
3. given a partial plan $(\Pi, S_{t'})$ such that $t' \geq t$, the node will be pruned if ϕ does not hold in Π

Operators `always-within`, `sometime-after` and `sometime-before`

These three operators share a similar syntax and semantics as they all involve two goal conditions in the constraint. The syntax is as follows: `(always-within <num> <GD <GD>)`, `(sometime-after <GD <GD>)` and `(sometime-before <GD <GD>)`. The constraints only differ in the temporal interval specified for the occurrence of the second goal condition. The semantics of the three operators are shown in formulas 6, 7 and 8.

$$\langle (S_0,0),(S_1,t_1),\ldots,(S_n,t_n)\rangle \models (\texttt{always-within}\ t\ \phi\ \psi)$$
$$\text{iff } \forall i : 0 \leq i \leq n \text{ if } S_i \models \phi$$
$$\text{then } \exists j :\ i \leq j \leq n \cdot S_j \models \psi \text{ and } t_j - t_i \leq t \tag{6}$$

$$\langle (S_0,0),(S_1,t_1),\ldots,(S_n,t_n)\rangle \models (\texttt{sometime-after}\ \phi\ \psi)$$
$$\text{iff } \forall i \cdot 0 \leq i \leq n \text{ if } S_i \models \phi$$
$$\text{then } \exists j :\ i \leq j \leq n \cdot S_j \models \psi \tag{7}$$

$$\langle (S_0,0),(S_1,t_1),\ldots,(S_n,t_n)\rangle \models (\texttt{sometime-before}\ \phi\ \psi)$$
$$\text{iff } \forall i \cdot 0 \leq i \leq n \text{ if } S_i \models \phi$$
$$\text{then } \exists j :\ 0 \leq j < i \cdot S_j \models \psi \tag{8}$$

The semantics of the three operators express the following characteristics:

- the constraints are not restricted to a single occurrence of ϕ and ψ
- the constraints apply if and only if ϕ occurs in the plan
- the constraints imply that for every occurrence of ϕ there must exist at least one occurrence of ψ that satisfies the corresponding temporal requirement
- it is not mandatory that every occurrence of ψ meets the constraint as long as there exists at least one occurrence of ψ that does meet the constraint for every ϕ

Specifically, a constraint of the form `(always-within` t ϕ ψ`)` indicates that ψ must hold within t time units from the occurrence of ϕ. A constraint `(sometime-before` ϕ ψ`)` is met if ψ holds before ϕ and a constraint `(sometime-after` ϕ ψ`)` is satisfied if ψ holds after ϕ.

Regarding the information of temporal landmarks, we can infer the following derivations:

1. for the three operators: if ϕ is a temporal landmark, then ψ is a temporal landmark too as it must necessarily occur in the plan at least once
2. for the operator `always-within`: it must hold that $\forall\phi\ \exists\psi\ :\ max_g(\psi) \leq max_g(\phi) + t$. Thus, assuming that ϕ_1 is the first occurrence of ϕ, for the remainder occurrences $\phi_i\ :\ i > 1$, if $max_g(\phi_i) \leq max_g(\phi_1) + t$ then the same occurrence of ψ will satisfy all ϕ_i; otherwise, for occurrences $\phi_j : j > 1$ such that $max_g(\phi_j) > max_g(\phi_1) + t$ a different occurrence of ψ, say ψ', will be needed to satisfy the constraint of ϕ_j.
3. for the operator `sometime-after`, it must hold that $\forall\phi\ \exists\psi : max_v(\psi) \geq max_g(\phi)$
4. for the operator `sometime-before`, it must hold that $\forall\phi\ \exists\psi : max_g(\psi) \leq max_g(\phi)$
5. the existence of a constraint `(always-within` t ϕ ψ`)` allows discarding a node $(\Pi, S_{t'})$, $t' > t$, if Π contains ϕ but not ψ
6. a constraint `(sometime-before` ϕ ψ`)` will allow to immediately prune a node which contains ϕ but not ψ
7. a constraint `(sometime-after` ϕ ψ`)` can only be used to prune nodes that contain finished plans in which ϕ holds and ψ does not.

The operator `hold-during`

The semantics of a constraint `(hold-during <num> <num> <GD>)` is expressed in formula 9, indicating that ϕ must hold during the interval $[u_1, u_2)$. More particularly, formula 9 explains three cases: when $[u_1, u_2)$ falls entirely within the plan trajectory (first case); when $[u_1, u_2)$ falls partially within the plan trajectory (second case); and when $[u_1, u_2]$ falls outside the plan trajectory (third case).

$$\langle (S_0,0),(S_1,t_1),\ldots,(S_n,t_n)\rangle \models (\texttt{hold-during}\ u_1\ u_2\ \phi)$$
$$\text{iff}\ \ \text{if}\ \ t_n > u_1\ \ \text{then}$$
$$\forall i \cdot 0 \leq i \leq n\ \cdot\ \text{if}\ u_1 \leq t_i < u_2\ \text{then}\ S_i \models \phi,$$
$$\forall j \cdot 0 \leq j < n\ \cdot\ \text{if}\ t_j \leq u_1 < t_{j+1}\ \text{then}\ S_j \models \phi$$
$$\text{if } t_n \leq u_1\ \text{then}\ S_n \models \phi \tag{9}$$

A constraint of the form `(hold-during` u_1 u_2 ϕ`)` allows us to derive the following information related to temporal landmarks:

1. ϕ is a temporal landmark since it must necessarily occur in the plan at least once
2. for the first case, given that it is mandatory for ϕ to hold between u_1 and u_2, we have that $min_n(\phi) \leq u_1$ and $u_2 < max_n(\phi)$; that is, `(hold-during` u_1 u_2 ϕ`)` determines that ϕ is needed at least between $[u_1, u_2)$
3. for the second case, given that it is mandatory for ϕ to hold between u_1 and t_n, we have that $min_n(\phi) \leq u_1$ and $max_n(\phi) = t_n$; that is, `(hold-during` u_1 u_2 ϕ`)` determines that ϕ is needed at least between $[u_1, t_n)$
4. for the third case, given that it is mandatory for ϕ to hold at t_n, we have that $min_n(\phi) = max_n(\phi) = t_n$; that is, `(hold-during` u_1 u_2 ϕ`)` determines that ϕ is needed at t_n
5. the search process will prune any node in which some restriction modifies the necessity interval $[u_1, u_2)$ of ϕ (for the first case) or modifies the necessity interval $[u_1, t_n)$ of ϕ (for the second case), or it will discard finished plans that do not contain ϕ (for the last case)

The operator `hold-after`

The semantics of `(hold-after <num> <GD>)` imposes that the goal condition ϕ must hold in a state after t time units have elapsed from the initial state at time 0 (see formula 10). Note that the semantics does not say that ϕ must exclusively hold after time t so it could be the case that ϕ also holds before t. On the other hand, if t is a time later than the finish time of the plan at t_n then ϕ must just hold in the last state (second *if* in formula 10).

$$\langle (S_0,0),(S_1,t_1),\ldots,(S_n,t_n)\rangle \models (\texttt{hold-after}\ t\ \phi)$$
$$\text{iff} \quad \text{if } t_n > t \text{ then } \exists i : 0 \le i \le n \cdot S_i \models \phi \text{ and } t_i > t$$
$$\text{if } t_n \le t \text{ then } S_n \models \phi \tag{10}$$

A constraint of the form `(hold-after` t ϕ`)` allows us to derive the following information related to temporal landmarks:

1. ϕ is a temporal landmark as it must necessarily occur in the plan at least once
2. the constraint $max_v(\phi) \ge t$ must be met
3. this constraint can only be used to discard finished plans that do not contain ϕ

Tables 3 and 4 summarize the landmarks derived from the PDDL3.0 modal operators and the updates applied on the endpoints of the landmarks intervals, respectively.

Constraint	Landmark
`(at end` l`)`	l
`(always` l`)`	l
`(at-most-once` l`)`	-
`(sometime` l`)`	l
`(within` t l`)`	l
`(always-within` t l_i l_j`)`	if l_i is a landmark, then l_j is a landmark
`(sometime-before` l_i l_j`)`	if l_i is a landmark, then l_j is a landmark
`(sometime-after` l_i l_j`)`	if l_i is a landmark, then l_j is a landmark
`(hold-during` t_1 t_2 l`)`	l
`(hold-after` t l`)`	l

Table 3: Creation of temporal landmarks

Application examples

In this section we present some practical examples that show the behaviour of TempLM when handling several PDDL3.0 state trajectory constraints on the scenario introduced in Figure 1. The actions of this domain are:

```
(:durative-action drive
  :parameters (?truck - truck ?loc-from - place
               ?loc-to - place ?driver - driver)
  :duration (= ?duration (time-to-drive ?loc-from ?loc-to))
  :condition (and (at start (at ?truck ?loc-from))
                  (at start (link ?loc-from ?loc-to)))
  :effect (and (at start (not (at ?truck ?loc-from)))
               (at end (at ?truck ?loc-to))))
```

Constraint	Our Model
`(at end` l`)`	$max_n(l) = max_v(l) = t_n$
`(always` l`)`	$min_n(l) = min_v(l) = t_0$ $max_n(l) = max_v(l) = t_n$
`(at-most-once` l`)`	$max_g(l) \le t_n$
`(sometime` l`)`	$max_g(l) \le t_n$
`(within` t l`)`	$max_g(l) \le t$
`(always-within` t l_i l_j`)`	$max_g(l_j) \le max_g(l_i) + t$
`(sometime-before` l_i l_j`)`	$max_v(l_j) \ge max_g(l_i)$
`(sometime-after` l_i l_j`)`	$max_g(l_i) \le max_g(l_j)$
`(hold-during` t_1 t_2 l`)`	first case: $min_n(l) \le t_1 \;\; max_n(l) > t_2$ second case: $min_n(l) \le t_1 \;\; max_n(l) = t_n$ third case: $min_n(l) = max_n(l) = t_n$
`(hold-after` t l`)`	$max_v(l) \ge t$

Table 4: Temporal constraints on the endpoints of the intervals

The `drive` action allows a `?truck` to move between two locations `?loc-from` and `?loc-to`, which are a depot or a distributor. The truck can move without carrying any crates. The duration of this action is given by the time to drive between the two locations.

```
(:durative-action load
  :parameters (?obj - crate ?truck - truck
               ?surf - surface ?loc - place)
  :duration (= ?duration 2)
  :condition (and
   (at start (at ?obj ?loc))(at start (on ?obj ?surf))
   (over all (at ?truck ?loc))(at start (clear ?obj))
   (over all (at ?surf ?loc)))
  :effect (and
   (at start (not (at ?obj ?loc)))(at end (clear ?surf))
   (at start (not (on ?y ?z)))(at end (in ?obj ?truck))))
```

The `load` action is used to load a crate `?obj`, which is onto the surface `?surf` of the location `?loc`, into the `?truck`. As a side effect, the surface where the crate was found is cleared.

```
(:durative-action unload
  :parameters (?obj - crate ?truck - truck
               ?surf - surface ?loc - place)
  :duration (= ?duration 2)
  :condition (and
   (over all (at ?truck ?loc))(at start (in ?obj ?truck))
   (over all (at ?surface ?loc))(at start (clear ?surf)))
  :effect (and
   (at start (not (in ?obj ?truck)))(at end (at ?obj ?loc))
   (at start (not (clear ?surf))) (at end (on ?obj ?surf))))
```

The `unload` action puts a crate `?obj` which is into a `?truck` onto a surface `?surf` located at the same place `?loc` than the `?truck`.

We will now show the temporal information that can be extracted when applying several state trajectory constraints on this problem.

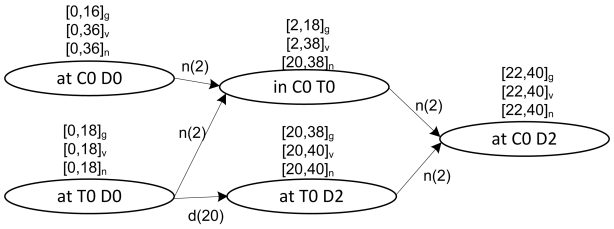


Figure 4: Initial TLG for the goal `(within 40 (at C0 D2))`

Example with a `within` constraint. Let's assume the problem goal is `(within 20 (at C0 D2))`. In this case, TempLM finds that min_g(`at C0 D2`) $=$ 22 (see the calculation of this value in page 2) and max_g(`at C0 D2`) $= 20$ so the TLG will not be generated and TempLM will return 'unsolvable problem'.

Let's now suppose that the problem goals are `(within 25 (at C0 D2))` and `(within 35 (at C1 D2))`. The initial TLG before propagation for this problem is shown in Figure 5[3]. As explained in section *Overview of TempLM*, `(at T0 D3)` is a landmark because it is the only way to satisfy `(within 25 (at C0 D2))`. On the other hand, `(at T0 D1)` is a landmark too because `T0` must go by distributor `D1` to load crate `C1`. Then, the min endpoint of the validity interval of `(at T0 D2)` is updated to: min_v(`at T0 D2`) $=$ $\max(min_v$(`at T0 D2`)$, min_v$(`at T0 D3`) $+$ $10, min_v$(`at T0 D1`)$) + 15) = 30$. After propagating this interval modification, we have that min_v(`at C0 D2`) $= 32$, which obviously entails an interval inconsistency because max_g(`at C0 D2`) $= 25 < min_v$(`at C0 D2`) $= 32$. This is an indication that the only way of achieving `(at C0 D2)` at time 25 is traveling through distributor `D3`. In turn, this means that `(at T0 D1)` is ordered after `(at T0 D2)` (i.e., `C1` is transported after `C0`), thus causing another inconsistency in the landmark `(at C1 D2)` because `C1` will not be delivered in time. This is also an indication that the problem is unsolvable and TempLM will detect this situation before even starting the search process.

Example with an `always` constraint. In this case, we will assume that the only goal is `(within 40 (at C0 D2))`. Figure 4 shows the initial TLG for this goal[4]. In the figure, we can observe a dependency ordering between `(at T0 D0)` and `(at T0 D2)`, representing that `T0` will reach `D2` after `D0` (in this case, since the deadline is at 40 we don't know yet whether the route of `T0` to reach `D2` must go through `D1` or `D3`). Another observation is that the pallet on which `C0` will be stacked is unknown, reason why `(on C0 P2)` and `(on C0 P3)` are not *landmarks*.

If we define `(always (clear P2))`, `(clear P2)` becomes a landmark with validity interval $[0, 40]$. During the search process, two actions that achieve the effect `(at C0 D2)` are found: `(unload C0 T0 P2 D2)` and `(unload C0 T0 P3 D2)`. Given that the application of `(unload C0 T0 P2 D2)` modifies the value of max_v(`at C0 D2`) when crate `C0` is unloaded in `P2`, and that an `always` constraints compels max_v(`at C0 D2`) $= t_n$ throughout the plan, the only viable option is to use the action `(unload C0 T0 P3 D2)` and TempLM would discard the node that unloads `C0` in `P2`.

Example with an `always-within` constraint. Following with a problem that contains the single goal `(within 40 (at C0 D2))`, let's suppose that we add `(always-within 22 (in C0 T0) (at C0 D2))`. Since the deadline for the problem goal `(at C0 D2)` is not very tight (max_g(`at C0 D2`) $= 40$), the new constraint does not affect the max endpoint of the generation interval of the goal. However, new information could be inferred during the search process. For instance, if `(in C0 T0)` is achieved at $t = 5$ then max_g(`at C0 D2`) $= 27$, which would allow us to infer that `(at T0 D3)` must be now a landmark.

Example with a `hold-during` constraint. Assuming we have the same goal as above `(within 40 (at C0 D2))`, let's suppose the truck `T0` must go through some maintenance repair in depot `D0` before starting the delivery. We define the restriction `(hold-during 0 10 (at T0 D0))` to denote that `T0` must stay at `D0` for 10 time units for the maintenance work. This restriction does not alter the initial necessity interval of `(at T0 D0)`, which is $[0, 18]$ as can be seen in Figure 4 (18 is the latest time that `T0` can stay in `D0` in order to achieve the goal at 40). Nodes that comprise partial plans in which `T0` is not in `D0` up to time 10 will be eliminated during the search process; that is, nodes that include a drive action `(drive T0 D0 X)` between 0 and 10.

Example with an `at end` constraint. Assuming we have the same goal as above, `(within 40 (at C0 D2))`, this examples shows a situation in which besides satisfying the goal, the truck `T0` must end the transportation at distributor `D3`. This implies defining also the constraint `(at end (at T0 D3))`, which makes `(at T0 D3)` become a landmark with validity interval $[0, t_n]$, max_n(`at T0 D3`) $= t_n$ and introduces the ordering `(at T0 D2)` $\prec_d$ `(at T0 D3)`. This ordering is motivated because `(at T0 D2)` and `(at T0 D3)` are mutex and `(at T0 D3)` must happen at the end due to the constraint.

During the plan construction, given that the goal deadline is at time 40 and hence truck `T0` can reach distributor `D2` either traversing `D1` or `D3`, the search tree will comprise two branches that follow these two alternatives. Let's analyze the impact of constraint `(at end (at T0 D3))` in the second branch, the one that traverses `D3`. In this case, `T0` must go through `D3` to reach `D2`, which implies `(at T0 D3)` $\prec_n$ `(at T0 D2)` and max_v(`at T0 D3`) $< t_n$. Then, a conflict arises because the TLG contains the landmark `(at T0 D3)` with max_n(`at T0 D3`) $= t_n$. As explained in the section of

[3]Only the most relevant landmarks are displayed

[4]The landmarks of the initial state are not shown for the sake of simplicity

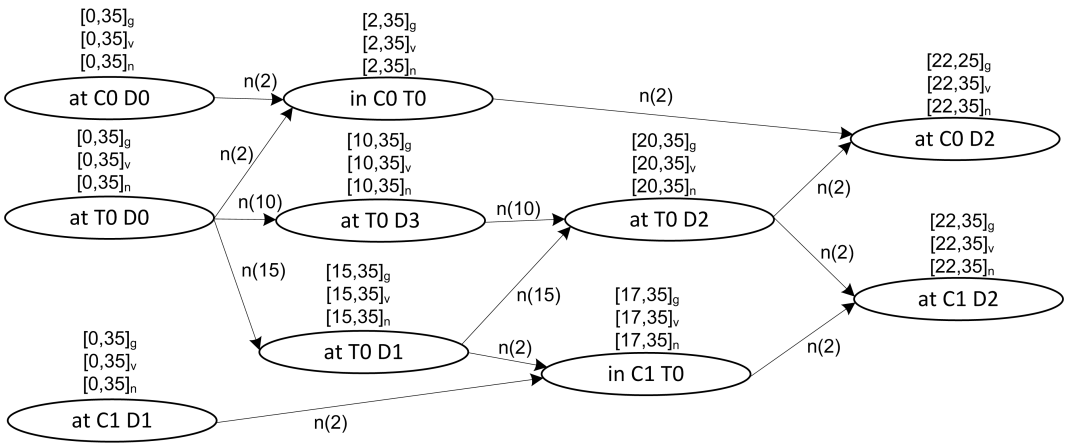


Figure 5: Initial TLG for the goals `(within 25 (at C0 D2))` and `(within 35 (at C1 D2))` before propagation

the `at-most-once` modal operator, when a landmark is found to be needed beyond its maximum validity, TempLM solves this conflict by introducing a new occurrence of the landmark `(at T0 D3)`, and this new occurrence is the one that will be ordered before `(at T0 D2)`. Thus, the final TLG will contain `(at T0 D3)` $\prec_n$ `(at T0 D2)` $\prec_d$ `(at T0 D3)`.

Example with an `at-most-once` constraint. In this example, the goal is to switch the location of two packages between distributors: `(within 50 (at C1 D3))` and `(within 50 (at C2 D1))`. Figure 6 shows the initial TLG for this problem. We can observe there is an inconsistency between the validity intervals of `(at T0 D3)` and `(at T0 D1)` because these two landmarks are mutex and cannot co-exist (the truck `T0` cannot be simultaneously in distributor `D1` and distributor `D3`). TempLM is not able to decide the order of these two landmarks with the current deadlines, being thus possible `(at T0 D3)` $\prec_d$ `(at T0 D1)` or `(at T0 D1)` $\prec_d$ `(at T0 D3)`.

Let's suppose the problem includes now the constraint `(at-most-once (at T0 D3))` and that a node $n = (\Pi, S_t)$ which contains the ordering `(at T0 D3)` $\prec_d$ `(at T0 D1)` in Π is found during the search process. Π embodies a plan where `T0` drops first by `D3` to load crate `C2`, which in turn implies that `T0` will need to get back to `D3` to unload `C1`, thus violating the constraint `(at-most-once (at T0 D3))`. Therefore, the node n will be discarded. In this example, the only feasible solution is a plan that contains `(at T0 D1)` $\prec_d$ `(at T0 D3)`, meaning that by the time `T0` reaches distributor `D3` to load crate `C2`, the truck already contains the crate `C1` to be unloaded in `D3`.

Discussion: beyond PDDL3.0

The exposition presented in the two previous sections reveal that the temporal landmarks formalism of TempLM is a very appropriate mechanism to deal with state trajectory constraints. It is certainly true that the functioning of TempLM is conditioned to the upper time bound of the plan T_Π, which can be set as the maximum value of all the deadlines constraints defined in the problem or as any particular value, and that the less restrictive T_Π is, the less information will be extracted from the trajectory constraints. Nevertheless, considering that constraints `at end`, `sometime`, `sometime-after` and `hold-after` are only applicable over finished plans, and that constraints `sometime-before`, `at-most-once` and `always` are easily checkable in any partial plan regardless the deadlines of the problem, we can conclude that the constraints that mostly affect the behaviour and performance of a temporal planner are `within`, `always-within` and `hold-during`, which all define a deadline constraint. Interestingly, adapting makespan-minimization heuristics to account for state trajectory constraints is still a challenging and unexplored line of investigation.

Besides the potential of temporal landmarks to handle trajectory constraints, we envision some further functionalities. For instance, one is not allowed to express in PDDL3.0 that a proposition ψ must hold within t time units from the end of another proposition ϕ. The specification of state trajectory constraints that involve two propositions ϕ and ψ is always related to the occurrence time of the first proposition ϕ, irrespective of ϕ is true or not when ψ holds. Handling a constraint of the form "ψ must `hold within` t time units from the end of ϕ" will be easily encoded with the temporal constraint $max_g(\psi) \leq max_v(\phi) + t$.

Another interesting issue is to be able to specify *persistence* of facts. Persistence would be expressed with `(within <num> (always <GD>))`, which requires nesting of the modalities and this is not allowed in standard PDDL3.0 syntax[5]. If PDDL3.0 were extended to include, for example, a modal operator like `(persistence t φ)`, this would be easily encoded in TempLM as $max_n(\phi) \geq max_g(\phi) + t$.

Last but not least, TempLM can also be adapted to the particular features of any temporal model; e.g., Allen's interval algebra (Allen 1983). Intervals of the algebra would be represented by means of the landmarks intervals and the

[5]Personal communication with Derek Long

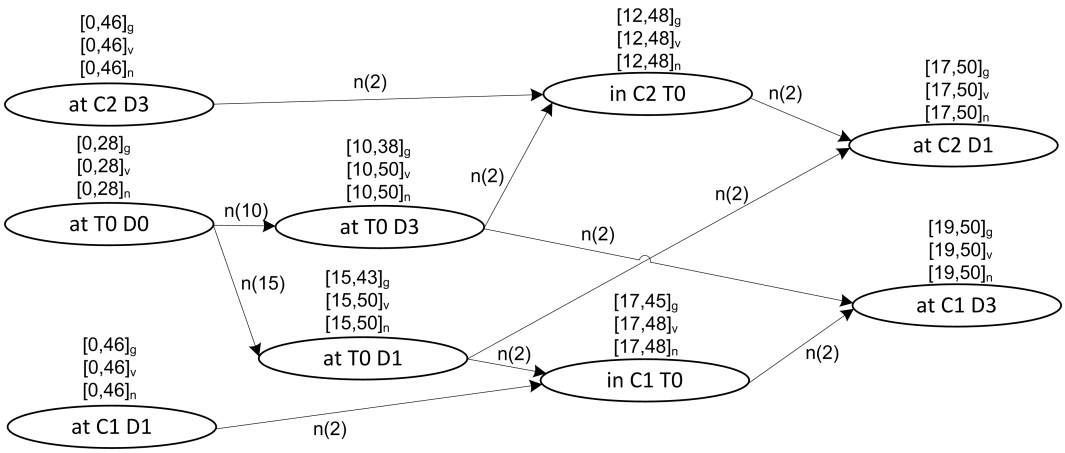


Figure 6: Initial TLG for the goal `(within 50 (at C1 D3))` and `(within 50 (at C2 D1))`

13 base relations would be captured by setting the appropriate temporal constraints between the max_g and max_v of the temporal landmarks. For instance, `(overlaps` ϕ ψ`)` would be encoded as $max_v(\phi) \geq max_g(\psi)$; and `(during` ϕ ψ`)` as $max_g(\phi) \geq max_g(\psi)$ and $max_v(\phi) \leq max_v(\psi)$.

A practical application of state trajectory constraints is the delivery of perishable goods such as fish or seafood. Companies must not only meet the delivery deadlines but also consider the best transport means for each product. Hence, depending on the type of product (fresh, frozen or long-term preserving fish products) and the temperature of the refrigerated transport (ice-cooled or machine-cooled wagons), the amount of time goods are exposed to particular temperatures must not exceed a time limit so as to ensure freshness, nutritional value and food preservation of the fishing goods.

All in all, we can conclude that the temporal formalism of TempLM offers a great flexibility to express any kind of temporal constraints in temporal planning problems.

Acknowledgements

This work has been partly supported by the Spanish MINECO under project TIN2014-55637-C2-2-R and the Valenciam project PROMETEO II/2013/019.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

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

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

Fox, M., and Long, D. 2003. PDDL2.1 : An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and 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., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. Int. Conference on Automated Planning and Scheduling (ICAPS-09)*, 162–169.

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

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

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 1728–1733.

Karpas, E.; Wang, D.; Williams, B. C.; and Haslum, P. 2015. Temporal landmarks: What must happen, and when. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, (ICAPS-15)*, 138–146.

Marzal, E.; Sebastia, L.; and Onaindia, E. 2008. Detection of unsolvable temporal planning problems through the use of landmarks. In *Proc. of the European Conference on Artificial Intelligence (ECAI-08)*, 919–920.

Marzal, E.; Sebastia, L.; and 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 (ICAPS-14)*, 172–180. AAAI Press.

Marzal, E.; Sebastia, L.; and Onaindia, E. 2016. Temporal landmark graphs for solving overconstrained planning problems. *Knowledge Based Systems* 106:14–25.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2017. Landmark-based heuristics for goal recognition. In *Proc. of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*. AAAI Press.

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