Temporal Planning with Temporal Metric Trajectory Constraints

Andrea Micheli
Fondazione Bruno Kessler
Trento, 38123, Italy
amicheli@fbk.eu

Enrico Scala
Fondazione Bruno Kessler
Trento, 38123, Italy
escala@fbk.eu

Abstract

In several industrial applications of planning, complex temporal metric trajectory constraints are needed to adequately model the problem at hand. For example, in production plants, items must be processed following a "recipe" of steps subject to precise timing constraints. Modeling such domains is very challenging in existing action-based languages due to the lack of sufficiently expressive trajectory constraints.

We propose a novel temporal planning formalism allowing quantified temporal constraints over execution timing of action instances. We build on top of instantaneous actions borrowed from classical planning and add expressive temporal constructs. The paper details the semantics of our new formalism and presents a solving technique grounded in classical, heuristic forward search planning. Our experiments prove the proposed framework superior to alternative state-of-the-art planning approaches on industrial benchmarks, and competitive with similar solving methods on well known benchmarks took from the planning competition.

1 Introduction

Several real world industrial problems, such as the automatic synthesis of a controller for an industrial plant, yield computational problems that would be naturally represented and solved using action-based planning (e.g., PDDL (Fox and Long 2003)). However, even very expressive language proposals (Gerevini et al. 2009; Fox and Long 2006) are inadequate to express and reason over the temporal constraints that are often an essential part of this class of problems.

Consider for example an electroplating plant (Phillips and Unger 1976), where the items need to be treated in a sequence of chemical baths ("recipe") with immersion timings constrained in precise time windows. The problem is to orchestrate a set of hoists moving a certain number of items according to a given recipe whilst keeping the immersion timing constraints in check, and ensuring hoists do not collide among each other. This problem is known in the literature as the Hoist Scheduling Problem (HSP) (Manier and Bloch 2003). For non-cyclic productions (e.g. heterogeneous recipes or the plant need to change production) this problem becomes a natural fit for planning techniques. This problem is traditionally solved by domain-specific scheduling techniques; our interest is tackling it from a *domain independent* standpoint.

In the HSP, it is very natural to model the hoist behaviors using the durative-actions of PDDL 2.1 (Fox and Long 2003). Unfortunately, the cross-action temporal constraints (e.g., the time window between the immersion of an item and its picking), cannot be easily expressed in current action-based representations, even with the use of the state-trajectory constraints offered by PDDL3 (Gerevini et al. 2009). This is in stark difference with scheduling, where temporal constraints are usually imposed over the intervals or the events of the schedule, not the states. Other planning formalisms, such as timeline-based temporal planning (Frank and Jónsson 2003; Cesta et al. 2009), do offer rich temporal constraint support, but are not well suited for problems with prominent causal structure.

Motivated by real-world industrial applications (e.g., the HSP), this paper proposes a novel planning framework aimed at taking the best from both action-based and timeline-based approaches. We start with a classical planning model where all actions are instantaneous and extend it with "temporal knowledge". A temporal knowledge is a set of quantified temporal metric axioms predicating over the execution timings of action instances. This way, the encoding of the HSP becomes very natural: the hoist dynamic is modeled using classical actions, while the immersion timings are preserved through our temporal knowledge using a formula with precise temporal metric constraints. Differently from action-based planning - where temporal constraints are implied by action-selection, and from timeline-based planning - where, instead, the actions are the implicit part of the problem, our paradigm collocates both planning goals and temporal constraints on the same level.

We make the following contributions. First, we provide a formal account to encode quantified temporal formulae over actions combined with classical planning; we detail the syntax and the semantics of this formalism, along with compilation schemata for encoding temporal constructs presented in other planning formalisms. Second, we present a heuristic search framework to handle the arising planning problem in a systematic and practical way. This encompasses the definition of a search space that intertwines planning with reasoning over our temporal knowledge. To explore this

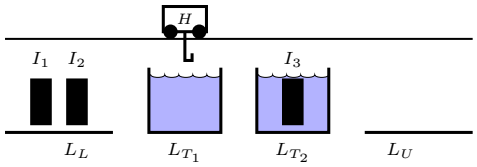


Figure 1: The Hoist Scheduling Problem example.

search space we present a simple yet effective relaxation-based heuristic that is informed of the temporal structure. Finally, we provide empirical evidence of the usefulness of the proposal across a variety of domains inspired by industrial applications, and well known planning benchmarks. On industrial domains especially, the proposed framework makes planning feasible over problems where PDDL 2.1 state-of-the-art temporal planners only solve very small instances. On well known PDDL 2.1 problems, our proposal proves competitive with existing approaches using similar techniques, and improves substantially on the state of the art when the very same domains are extended with the richer temporal constraints induced by intermediate action effects.

2 Problem Motivation and Formalization

Let us consider the example situation of figure 1. We have a simple HSP with 2 tanks (L_{T_1} and L_{T_2}) and 1 hoist H. Each item I_x is required to obey the following "recipe": after being picked up from the loading position (L_L) stay in L_{T_1} for 10 to 12 minutes, and then in L_{T_2} for 20 to 21 minutes; finally, reach position (L_U). H takes 1 minute to move between adjacent tanks and negligible time to load and unload items. While an item is in a tank, H is free to move, pick and transport other items. We assume that both the tanks and H can hold at most one item at a time.

Encoding the HSP in PDDL 2.1 [1] is complicated because the time windows are state-dependent temporal constraints. One can use the clip-action encoding technique (Fox, Long, and Halsey 2004), which results in adding extra actions and variables to the domain. For instance, let $unload(I_x, L_{T_2})$ be the action representing the drop-off of item I_x in position L_{T_2}; one needs two durative-actions win_l (with duration 20) and win_u (with duration 21) to model the lower and the upper bounds of the time window in which I_x can stay in L_{T_2} (we use the end-effects of such actions to enforce the picking of I_x in this time window). In addition, we need to force the end of action $unload(I_x, L_{T_2})$ to happen at the same time of the starting time of both win_l and win_u; this can be done using an additional durative action called "clip-action". This construction makes the planning problem much harder to encode, and much harder to solve for existing planners. First, it requires the user to rewrite the domain model instead of enforcing these constraints as goals. Second, it makes the reasoning much more complicated, leading to unnecessary branching points during search, consequence of breaking the actual structure of the problem. Concretely, given an instance of HSP with n hoists, and m positions (all involved in the recipe) for a number k of items, the clip-action encoding results in adding $3 \times (n \times m \times k)$ artificial actions to the original set of actions.

To overcome these modeling limitations, we introduce a new formalism from which we define a Timed Planning Problem (TPP). For convenience, we use TPP to refer to both the planning problem, and the formalism (i.e., the way such a problem can be expressed).

Notation. Our setting is standard First Order Logic (Kleene 1967). We indicate a formula ϕ expressed over variables $v_1, \cdots, v_n$ as $\phi(v_1, \cdots, v_n)$; given $\phi(v_1, \cdots, v_n)$ we write $\phi(x_1, \cdots, x_n)$ for the substitution of each v_i with x_i.

Syntax. We start our formalization by recalling the syntactical definition of a classical planning problem.

Def 1. *A **classical planning problem** is a tuple $P : \langle X, A, I, G\rangle$ where: X is a set of Boolean state variables; A is a set of ground actions[2] and each $a \in A$ is a pair $\langle \mathrm{pre}(a), \mathrm{eff}(a)\rangle$ where $\mathrm{pre}(a)$ is a propositional formula over variables from X and $\mathrm{eff}(a)$ is a set of non-conflicting assignments to variables in X; I, the initial state, is a complete assignment to variables in X; G, the goal, is a propositional formula over X.*

The following definition formalizes the syntax of axioms that express quantified temporal metric constraints on the execution time of actions. We allow any alternation of existential ($\bar{\exists}$) and universal ($\bar{\forall}$) action quantifiers, each binding one action-typed variable. The body of the axiom is a formula of binary temporal constraints (Dechter, Meiri, and Pearl 1991) over such variables.

Def 2. *Given a classical planning problem $P : \langle X, A, I, G\rangle$, a **temporal action axiom** is a formula in the form: $\nabla_1 v_1 \colon a_1 . \cdots . \nabla_n v_n \colon a_n . \phi(v_1, \cdots v_n, \bar{0})$ where $\nabla_i \in \{\bar{\exists}, \bar{\forall}\}$ is an action quantifier; v_i is a time-point variable; $a_i \in A$ is an action; $\phi(v_1, \cdots v_n, \bar{0})$ is an arbitrary Boolean combination of atoms of the form $v_x - v_y \leq k$ with $v_x, v_y \in \{v_1, \cdots v_n, \bar{0}\}$ and $k \in \mathbb{R}$, and $\bar{0}$ is a distinct time-point variable indicating the starting instant of the plan.*

Def 3. *A **Temporal Knowledge (TK)** Γ is a conjunction of temporal action axioms. A **Timed Planning Problem (TPP)** is a pair $\langle P, \Gamma\rangle$, where P is a classical planning problem.*

A plan for a given TPP is *timed* in that each action is associated with an execution time.

Def 4. *Given a classical planning problem $P : \langle X, A, I, G\rangle$, a **timed plan** is a set of pairs $\{\langle a_1, t_1\rangle, \cdots, \langle a_n, t_n\rangle\}$ where $a_i \in A$ is an action and $t_i \in \mathbb{R}_{\geq 0}$, for each $i \in 1 \cdots n$.*

Semantics. We inherit the semantics of the classical planning problem from PDDL 2.1 level 1 (Fox and Long 2003), i.e., its untimed fragment. We just recall that a ***valid classical plan*** is a sequence of actions that, starting from the initial state, are applicable one after the other, and yield a final state that fulfills the goal condition. To reason over the validity of a classical plan within TPP, we define an induced causal plan as a sequence of actions composed of the very same actions appearing in the timed plan, sorted by their timing.

[1] Another option would be PDDL3, but it only supports few temporal metric constructs that are insufficient to express the HSP without a construction analogous to the one for PDDL 2.1.

[2] We only present a ground representation; our results extends to the lifted case and our implementation supports both.

Def 5. *Given a timed plan* $\tau : \{\langle a_1, t_1\rangle, \cdots, \langle a_n, t_n\rangle\}$, *an* ***induced causal plan*** *is an ordered list of actions* $\pi : \langle a_{o(1)}, \cdots, a_{o(n)}\rangle$ *where* $o : \mathbb{N} \to \mathbb{N}$ *is such that for all* $i, j \in \mathbb{N}$ *with* $i < j$, $t_{o(i)} \leq t_{o(j)}$ *holds.*

If two actions are executed at the same time, there might be more than one induced plan; hence, we define the causal validity of a timed plan as the validity of *all* the possible induced causal plans. This handles the problem of non-commutative actions scheduled at the same time.

Def 6. *A timed plan* τ *is* ***causally valid*** *for a* TPP $\langle P, \Gamma\rangle$, *if all its induced causal plans are valid for* P.

We can now define the validity of the temporal action axioms w.r.t. a timed plan τ. Intuitively, being τ a finite set, the semantics of $\bar{\forall} v\colon a$ ($\bar{\exists} v\colon a$) coincide to a conjunction (disjunction) over all the instances of action a in τ.

Def 7. *Given* $\gamma : \nabla_1 v_1\colon b_1.\cdots.\nabla_n v_n\colon b_n.\phi(v_1, \cdots v_n, \bar{0})$, *a timed plan* $\tau : \{\langle a_1, t_1\rangle, \cdots, \langle a_n, t_n\rangle\}$ ***satisfies temporal action axiom*** γ *if the following formula (where* $\bigodot^i = \bigwedge$ *if* $\nabla_i = \bar{\forall}$ *and* $\bigodot^i = \bigvee$ *if* $\nabla_i = \bar{\exists}$*) is satisfiable.*

$$\bigodot^1_{\langle b_1, x_1\rangle \in \tau} \cdots \bigodot^n_{\langle b_n, x_n\rangle \in \tau} \phi(x_1, \cdots, x_n, 0)$$

Def 8. *A timed plan* τ ***is valid*** *for a* TPP $\langle P, \Gamma\rangle$ *if it is causally valid and satisfies all temporal action axioms in* Γ.

Solving a TPP $\langle P, \Gamma\rangle$ where $P : \langle X, A, I, G\rangle$ is the task of finding a valid timed plan using actions from A. Interestingly, this planning problem is decidable when interpreted over integer time (as for the PDDL case (Rintanen 2007)).

Thm 1. TPP *is decidable if interpreted over integer time.*

Proof sketch. We reduce TPP to the satisfiability of a TPTL formula with past operators, that is decidable (Alur and Henzinger 1993; 1994). TPTL offers the usual LTL operators e.g., $\Box$ (for always in the future) and $\Diamond$ (for eventually); the past operators e.g., $\boxminus$ (for always in the past) and $\diamondminus$ (for once in the past); and "freezing quantifiers" of the form "x." referring to the timing of states. The classical planning part of TPP has a known compilation in the LTL fragment of TPTL (Calvanese, De Giacomo, and Vardi 2002), while the temporal action axioms can be encoded using freezing quantifiers in combination with temporal operators. As for our formalism, TPTL allows Boolean combinations of binary temporal constraints expressed over the variables of freezing quantifiers, hence we only need to encode the universal and existential quantification as freezing quantifications. This can be done by translating $\bar{\forall} v\colon a.\alpha$ into $\Box v.(p_a \to [\alpha]) \vee \boxminus v.(p_a \to [\alpha])$ where $[\alpha]$ indicates the recursive compilation of the right-hand side of the axiom, and p_a encodes the states in which action a is triggered. Similarly, $\bar{\exists} v\colon a.\alpha$ translates to $\Diamond v.(p_a \wedge [\alpha]) \vee \diamondminus v.(p_a \wedge [\alpha])$. $\Box$

Example*. To specify the HSP problem of figure 1 in* TPP *we devise an action* $load(I_x, L)$ *for loading an item* I_x *in location* L, *similarly an action* $unload(I_x, L)$ *for the corresponding drop-off, and finally a pair of actions* $move_{\vdash}(L_1, L_2)$ *and* $move_{\dashv}(L_1, L_2)$ *for the movement of the hoist* H *(representing the start and end of the movement, respectively). For each item* I_x, *we impose the following axiom that encodes the recipe for the item itself.*

$$\begin{aligned}&\bar{\forall} l_0\colon load(I_x, L_L).\bar{\exists} u_1\colon unload(I_x, L_{T_1}).\bar{\exists} l_1\colon load(I_x, L_{T_1}).\\ &\bar{\exists} u_2\colon unload(I_x, L_{T_2}).\bar{\exists} l_2\colon load(I_x, L_{T_2}).\bar{\exists} u_3\colon unload(I_x, L_U).\\ &(l_0 \leq u_1) \wedge (10 \leq (l_1 - u_1) \leq 12) \wedge (l_1 \leq u_2) \wedge\\ &(20 \leq (l_2 - u_2) \leq 21) \wedge (l_2 \leq u_3)\end{aligned}$$

Intuitively, each time the hoist loads an item from L_L, *the universal action quantifier ensures the particular sequence of events required by the recipe, with the right schedule. For every pair of positions* $\langle L_1, L_2\rangle$, *the duration of the hoist movement is specified by the following axiom schemata.*

$$\bar{\forall} m_s\colon move_{\vdash}(L_1, L_2).\bar{\exists} m_e\colon move_{\dashv}(L_1, L_2).(m_e - m_s) = tt(L_1, L_2)$$

$$\bar{\forall} m_e\colon move_{\dashv}(L_1, L_2).\bar{\exists} m_s\colon move_{\vdash}(L_1, L_2).(m_e - m_s) = tt(L_1, L_2)$$

Where $tt(L_1, L_2)$ *is the time needed to go from* L_1 *to* L_2. *These axiom schemata leave room for ordering the items production and the movements of the hoist respecting the duration constraints. A fragment of a valid timed plan is:*

$$\begin{aligned}&\{\cdots, \langle move_{\vdash}(L_{T_1}, L_{T_2}), 12\rangle, \langle move_{\dashv}(L_{T_1}, L_{T_2}), 13\rangle,\\ &\langle unload(I_1, L_{T_2}), 13.1\rangle, \cdots, \langle move_{\dashv}(L_{T_2}, L_{T_L}), 15.5\rangle,\\ &\langle load(I_2, L_{T_L}), 15.6\rangle, \cdots, \langle load(I_1, L_{T_2}), 34\rangle, \cdots\}.\end{aligned}$$

TPP as a Target Formalism

We now show how TPP can express interesting features of other languages. First, we encode durative actions of PDDL 2.1 level 3 (without numeric state variables) under the assumption of non-unbounded self-overlapping actions (Rintanen 2007). Handling this feature is complicated both in PDDL and in TPP due to the matching of the starts with their relative ends. Second, we encode problems featuring intermediate actions effects and temporally qualified conditions, constructs like those offered by ANML (Smith, Frank, and Cushing 2008) and (to some extent) NDL (Rintanen 2015).

From PDDL2.1 to TPP. Similarly to our encoding of the HSP, we decompose each durative action da, having duration constraint $L_{da} \leq \texttt{duration} \leq U_{da}$, into a pair of TPP actions ($da_{\vdash}, da_{\dashv}$), and prevent self-overlap by means of a fresh dummy state variable r_{da}. $da_{\vdash}, da_{\dashv}$ are equivalent to the snap actions used in Coles et al. (2010) with the addition that $da_{\vdash}$ and $da_{\dashv}$ set r_{da} to $\top$ and $\bot$, respectively. These two actions retain the causal structure of da (start/end condition and effects), while the temporal aspects are encoded in TPP using the following axioms:

$$\bar{\forall} s\colon da_{\vdash}.\bar{\exists} e\colon da_{\dashv}.(e - s \leq U_{da}) \wedge (s - e \leq -L_{da})$$

$$\bar{\forall} e\colon da_{\dashv}.\bar{\exists} s\colon da_{\vdash}.(e - s \leq U_{da}) \wedge (s - e \leq -L_{da}).$$

For each "over all" condition ϕ of da we conjoin every action precondition with the invariant constraint $r_{da} \to \phi$.

Intermediate effects and temporally qualified conditions into TPP. An intermediate (also called delayed) effect is a set of non-conflicting assignments happening at a specific time after the starting of the action. This can be modeled in TPP with an instantaneous action ie encoding the effect, and by imposing the appropriate axioms: (i) $\bar{\forall} s\colon da_{\dashv}.\bar{\exists} e\colon da_{\dashv}.\bar{\exists} i\colon ie.\cdots$, (ii) $\bar{\forall} e\colon da_{\dashv}.\bar{\exists} s\colon da_{\vdash}.\bar{\exists} i\colon ie.\cdots$, and finally (iii) $\bar{\forall} i\colon ie.\bar{\exists} e\colon da_{\dashv}.\bar{\exists} s\colon da_{\vdash}.\cdots$.

A temporally qualified condition (Smith, Frank, and

Cushing 2008) is a formula that must hold within a specific interval after the starting of an action. It can be encoded in TPP with two intermediate effects that explicit, through a fresh dummy state variable, the start and the end of the interval associated to the condition; similarly to an over-all condition, the encoding then enforces an invariant constraint.

3 Solving Timed Planning Problems

This section presents the heuristic search approach we employ to solve TPP. We first present the design of the search space; then we describe the relaxation-based heuristics devised to steer a best-first-search in this space.

Forward Search Schema and Temporal Reasoning

Inspired by POPF (Coles et al. 2010), our search navigates the state space using forward chaining: start from the initial state and progress over some next state only using those applicable actions leading to a satisfiable TK. Differently from PDDL 2.1, the TK makes the temporal reasoning in our setting much more complicated. As search progresses forward towards a next state s, it is important to assess which (if any) from the actions contained in the prefix (i.e., the sequence of actions leading to s from the initial state) is interpreting some of the quantified variables involved in TK. Dealing with universally quantified variables is straightforward: each matching action instance in the prefix leads to an instantiation of the quantifier. Instead, existential quantification is more complex: we must decide which action instance in the prefix witnesses the quantifier. For example, let $A : \{a_0, a_1, a_2\}$ and $\bar{\forall} x\colon a_0.\bar{\exists} y\colon a_1.y - x > 10 \in$ TK. Say $\langle a_0, a_2\rangle$ is the current prefix and the search selects action a_1. There are two possibilities: either use a_1 to witness the satisfaction of the axiom, or wait for some other occurrence of a_1 in the future. The first, more committed, choice would prevent to position this occurrence of a_1 anytime before $t(a_0) + 10$. The second, less committed, choice would instead allow more freedom in positioning a_1 on the timeline, but require a future occurrence of a_1 to witness the satisfaction of the temporal action axiom. Note that, restricting to either decision results in an incomplete search.

Online temporal reasoning. To handle the above issue systematically, we annotate each state with temporal commitments, and use these annotations to construct a formula (called TN) whose satisfiability encode only course of actions deemed consistent for the TK at hand. Formally, a search state s is a tuple $\langle \mu, \pi, \rho, \sigma\rangle$ where: μ is an assignment for the Boolean variables - this corresponds to a classical planning state; $\pi = \langle \bar{a}_1, \cdots, \bar{a}_k\rangle$ is the sequence of action instances that have been applied to reach s (i.e., the prefix). The over-bar writing (i.e., $\bar{a}$) is used to refer to action instances that can be chosen during search. Let $\bar{a}$ be an action instance, $type(\bar{a})$ denotes its action *type*. ρ is a set of existentially quantified action instances, anticipated to witness some existential quantifier in TK; σ is a set of elements of the form $f_x(\bar{b}_{y_1}, \cdots, \bar{b}_{y_n}) = \bar{a}_x$, where x is an existentially-quantified variable in TK and $y_1, \cdots, y_n$ are the universally quantified variables appearing on the left-hand side of x; $\bar{b}_{y_i}$ is an action instance from π that matches y_i, while $\bar{a}_x$ is an action instance in $\pi \cup \rho$ that matches x. W.l.o.g. we assume that each variable name in TK is unique. Intuitively, σ encodes the Skolemization functions (Hähnle 2001), defined by cases, of some existential quantifier appearing in TK.

Let s be a state, the formula $TN(s)$ is defined over a set of real-valued time-points variables each encoding *when* an action instance $\bar{a}$ (with $\bar{a} \in s.\pi \cup s.\rho$) is scheduled in the resulting timed plan. $tp(\bar{a})$ indicates the time point associated with $\bar{a}$. To preserve causal validity (as per definition 6) and to satisfy all the axioms in TK, the $TN(s)$ formula is the conjunction of a (valid) partial order of π (obtained through the KK algorithm (Bäckström 1998)) and the following encoding of TK. For each axiom $\nabla_1 v_1\colon a_1.\cdots \nabla_n v_n\colon a_n.\phi(v_1, \cdots v_n, \bar{0}) \in$ TK we conjoin the following formula:

$$\bigwedge_{\bar{b}_{o(1)} \in U_1} \cdots \bigwedge_{\bar{b}_{o(m)} \in U_m} \phi(t_1, \cdots t_n, 0)$$

where m is the number of universal quantifiers in the axiom (obviously, $m \leq n$); $o : 1 \cdots m \to 1 \cdots n$ is such that $o(i)$ gives the index of the i-th universal quantifier; each U_i is the set of action instances in π matching the i-th universal quantifier, defined as $\{\bar{b}_{o(x)} \in \pi \mid type(\bar{b}_{o(x)}) = a_{o(x)}\}$; and $t_k = tp(\bar{b}_k)$ if $\nabla_k = \bar{\forall}$, otherwise (i.e., $\nabla_k = \bar{\exists}$) $t_k = tp(\bar{c}_k)$ such that $f_{v_k}(\bar{b}_{o(1)}, \cdots, \bar{b}_{o(z)}) = \bar{c}_k \in \sigma$ where z is the number of universal quantifiers on the left-hand side of ∇_k. Intuitively, for each axiom in TK and for each possible matching of the universally quantified variables (if any) appearing in such axiom, we enforce the relative temporal constraint where the existentially quantified variables are substituted by the bindings defined in σ. If the $TN(s)$ formula is satisfiable, we say that s is *temporally consistent*, otherwise, the state can be pruned.

LAZY-SEARCH and EAGER-SEARCH. To progress the search, we devise two branching strategies. The first (hereinafter, LAZY-SEARCH), postpones the binding of existential quantifiers to goal states only. The second (hereinafter, EAGER-SEARCH), anticipates such temporal commitments explicitly into the state space. In both cases, a timed plan is extracted by taking the action instances in the prefix leading to any goal state s; the execution time of each such an action is a model of $TN(s)$. Practically, the two strategies differ on the algorithm used to compute the successor states.

In LAZY-SEARCH none of the existential quantifier is matched for states that do not satisfy the classical goal. This corresponds to keep σ always empty and checking consistency for axioms only containing universal quantifiers. LAZY-SEARCH starts with $I_{lazy} : \langle I, \emptyset, \emptyset, \emptyset\rangle$, and when a goal state is encountered, it combinatorially explores the possible bindings for existentially-quantified variables. More precisely, let s be a classical goal state. LAZY-SEARCH checks the formula in definition 7 for consistency. Operationally, one can either use a dedicated constraint solver, or explicitly explore the possible bindings for the existentially-quantified variables in TK.

EAGER-SEARCH anticipates the existential matching online, during the search. The pseudocode of its key mechanisms is reported in algorithm 1. SUCC is the top-level function and is called whenever a new action a is selected for application in a state s. It takes in input s and a, and returns

Algorithm 1 EAGER-SEARCH Successor Function

```
1: function SUCC(s - State, a - Action)
2:   μ' ← CLASSICALAPPLY(s.μ, a)
3:   S ← {⟨μ', s.π + ⟨ActionInstance(a)⟩, s.ρ, s.σ⟩}
4:   for all ȳ ∈ s.ρ s.t. type(ȳ) = a do
5:     S ← S ∪ {⟨μ', s.π + ⟨ȳ⟩, s.ρ \ {ȳ}, s.σ⟩}
6:   for all γ ∈ TK and a is universally quantified in γ do
7:     S ← ⋃_{s'∈S} APPLY(s', γ', ∅)
8:   return {s ∈ S | TN(s) is consistent}
9: function APPLY(s - State, γ - Temporal Axiom, b - ∀̄Bind)
10:  S ← ∅
11:  if γ is Quantifier Free then
12:    S ← {s}
13:  else if γ = ∀̄x : a.γ' then
14:    S ← {s}
15:    for all ȳ ∈ π s.t. type(ȳ) = a do
16:      S ← ⋃_{s'∈S} APPLY(s', γ', b ∪ {x = ȳ})
17:  else if γ = ∃̄x : a.γ' then
18:    for all ȳ ∈ π ∪ s.ρ s.t. type(ȳ) = a do
19:      σ' ← s.σ ∪ {f_x(b) = ȳ}
20:      S ← S ∪ APPLY(⟨s.μ, s.π, s.ρ, σ'⟩, γ', b)
21:    ā' ← ActionInstance(a)
22:    σ' ← s.σ ∪ {f_x(b) = ā'}
23:    S ← S ∪ APPLY(⟨s.μ, s.π, s.ρ ∪ {ā'}, σ'⟩, γ', b)
24:  return S
```

the set of successor states computed as follows. First, the algorithm applies the classical effects of action a and stores them in μ'. Then, it computes all the interpretations of a by considering both where a does not serve as witness for any existential quantifier, and where a is interpreting some of the necessary, but unmatched, actions from $s.\rho$ (line 3 to 5). The set of states $\mathcal{S}$ collects such interpretations. For each of these states, the algorithm recursively applies all the temporal action axioms by calling APPLY. APPLY enumerates the bindings between actions from the entire prefix of the plan and variables in the axiom γ in a recursive, top-down fashion. The recursion removes from γ one quantifier at a time, and terminates when a quantifier-free temporal constraint is reached. If the leftmost quantifier in γ is a $\bar{\forall}$, the algorithm iterates over all the matching action instances in the prefix, appending each match in b (line 16). Note that, the recursive invocation of line 16 updates the set of successors at each iteration. Besides the very first, each axiom in TK is applied over all the successor states computed by the APPLY on the previously selected axiom. For existentially quantified variables, APPLY extends the set of successor states with each possible matching action instance in π or ρ. In addition, it considers the possibility of satisfying the existential quantifier with an action that is neither in the prefix π nor in the anticipated action instances ρ yet (lines 21 to 23). This forces to eventually apply such an action instance in that branch of the search. EAGER-SEARCH starts with all the states obtained by recursively applying all the purely-existential axioms in TK to I_{lazy} (the recursion is analogous to lines 6-7 of algorithm 1). Checking the consistency of a TN associated with a goal state is sufficient to guarantee that the prefix is a valid solution plan, as long as $s.\rho$ is empty. Otherwise ($s.\rho$ is not empty), even if the classical goal was reached, the search needs to expand the prefix further; this is to ensure that the commitments in $s.\rho$ are fulfilled.

Both LAZY-SEARCH and EAGER-SEARCH safely prune a state s iff $TN(s)$ is not consistent - we can guarantee that the prefix cannot be extended to a valid plan. As a difference, they explore quite a different search space. Each time a new action is selected, LAZY-SEARCH successor function generates a *single* state (it does not anticipate the binding of existential quantifiers) while EAGER-SEARCH generates much more successor states (this is in general exponential in the number of existential quantified variables). Albeit EAGER-SEARCH might seem not practical, we highlight that (i) its exponential behavior is a very worst-case scenario: occurrences of the same action in the plan are often limited, and when an action is not universally quantified, at most $1+|s.\rho|$ successors are generated; (ii) by expliciting the temporal commitments in the state, the search can be more informed. In fact, EAGER-SEARCH proves superior to LAZY-SEARCH for a consistent number of problems (see section 5).

Relaxation and Heuristics

The search tree induced by the proposed search schemata is prohibitively large to be explored exhaustively, hence we propose relaxation-based heuristics to guide the search. We build on the well known subgoaling relaxation (Haslum and Geffner 2000), also known as h^1, that ignores the interactions within conjunction of propositions into a recursive schema. h^1 is a safe relaxation of classical planning (i.e., it proves unsolvable only when the associated classical planning problem is). Moreover, h^1 also relaxes TPP because classical planning is an obvious relaxation of TPP. From h^1, we concentrate our attention on its additive implementation (h_{add}), known to be very effective for satisficing planning.

$$h_{add}(G) = \begin{cases} 0 & \text{if } s \models G \\ \min_{a \in ach(G)} h_{add}(\text{pre}(a)) + 1 & \text{if } |G| = 1 \\ \sum_{g \in G}(h_{add}(g)) & \text{if } |G| > 1 \end{cases}$$

Clearly, h_{add} knows nothing about TK, hence we extend it in two different directions to make it informed of the axiom structure when employed in EAGER-SEARCH. Recall that, the set $s.\rho$ keeps track of the foresaw action instances that are necessary to fulfill the TK axioms in state s. Given such prediction, a first temporally-aware version of h_{add} (called h_{atk}) is obtained by adding to the heuristic estimate the number of anticipated actions: $h_{atk} = h_{add} + |s.\rho|$.

A further improvement is obtained by working on the underlying relaxation of h^1. For each action $a = type(\bar{a})$ s.t. $\bar{a} \in s.\rho$, we add a fresh predicate p_a to the goal, ensuring that a is the only achiever of p_a. This yields a tighter, state-dependent classical planning relaxation of TPP. We call the resulting additive heuristic h_{dtk}.

Prop 1. *For any search state s,* $h_{dtk}(s) \geq h_{atk}(s) \geq h_{add}(s)$.

Proof sketch. In absence of existential statements within TK, $h_{dtk}(s) = h_{atk}(s) = h_{add}(s)$. When the current state contains at least one unmatched existential quantifier $h_{atk}(s) \geq h_{add}(s)$; moreover, $h_{dtk}(s) \geq h_{atk}(s)$, as h_{dtk} also considers the preconditions of the action necessary to match the existential statement. □

Naturally, h_{dtk} and h_{atk} bring some computational overhead (i.e., both need EAGER-SEARCH) over h_{add}. Their effectiveness is empirically studied in section 5.

4 Related Work

Our formalism is inspired by the timeline-based approach to temporal planning (Frank and Jónsson 2003). This approach uses temporal constraints to express both the timing behavior and, implicitly, the causal structure of the system. Similarly to timelines, TPP allows temporal constraints, but generalizes synchronization-rules with nesting of quantifiers and keeps the causal structure explicit.

TPP is similar in spirit to PDDL3 (Gerevini et al. 2009): both formalisms introduce trajectory constraints. PDDL3 uses not-nested LTL-like formulae, constraining the state trajectory. TPP predicates over action instances, imposing metric temporal requirements with arbitrary quantifier alternations. Clearly, it is possible to encode state-change with dedicated actions in TPP and, symmetrically, use additional predicates to monitor action executions in PDDL3. However, problems such as the HSP cannot be expressed as non-nested PDDL3 trajectory constraints without modifying the problem structure, analogously to the PDDL 2.1 case.

Similarly, PDDL+ (Fox and Long 2006) can encode some action timing constraints using dummy predicates, processes and events. However, even simple constraints are very hard to encode: e.g., $\bar{\forall}x : a.\bar{\forall}y : b.y - x \geq 10$ requires several additional actions, events and processes. Moreover, PDDL+ planners focus on continuous change, not provinding ways to directly exploit the temporal structure of our domains.

MPMTL (To et al. 2017) focuses on continuous change, but also temporal trajectory constraints are possible: the modal operators in the logic are similar to our quantifiers, but there are several differences: first, MPMTL predicates over state variables not actions; second, only future operators are available; finally, the planning language is restricted to 2-level operator nesting. Arbitrary nesting is needed for practical purposes: in the HSP a "recipe" is encoded as a chain of operators (i.e., $\bar{\forall}\bar{\exists}\cdots\bar{\exists}$ corresponding to $\Box\Diamond\cdots\Diamond$).

HTN (Nau et al. 2003) provides facilities to predicate over actions temporally, but TPP is substantially different. TPP leaves the planning domain intact (actions represent the physics (McDermott 2000)) and separated from temporal constraints. Instead, HTN imposes such constraints in the planning domain by modifying the model of the actions. Moreover, HTN is not well suited for metric constraints.

The planning language of TBurton (Wang and Williams 2015) supports the compilation of networks of (restricted) timed automata, and the specification of the goal condition as a Simple Temporal Network (STN) over the automata states. These can be formalized in TPP with purely existential axioms. TPP subsumes the goal language of TBurton as it allows arbitrary quantifier alternations.

TAL (Doherty and Kvarnström 2008) is a logic to express planning control knowledge that can constrain action occurrences via the **occ** statement. These statements, that can be naturally encoded in TPP, are limited to existential qualitative constraints and are not supported by the only planner implementation using TAL (Doherty and Kvarnström 2001).

Karpas et al. (2015) and Marzal, Sebastia, and Onaindia (2016) introduce the idea of "temporal action landmark", that is a set of actions subject to temporal constraints that are in any valid plan. Such landmarks can be expressed in a narrow subset (using only the $\bar{\exists}$ quantifier) of our formalism, and our techniques can be used to solve and exploit these structures. As for actions deemed necessary within the EAGER-SEARCH schema, the existential quantifiers of those landmarks translates into necessary actions during search.

Several works proposed mechanisms for temporal planning. We borrowed the basic search schema of POPF and OPTIC (Coles et al. 2010; Benton, Coles, and Coles 2012) (keeping the time reasoning lifted and the classical planning part explicit), substantially extending it to handle the TPP constructs and providing dedicated heuristics based on subgoaling decomposition. Instead, decision epoch planners (e.g., (Do and Kambhampati 2003; Eyerich, Mattmüller, and Röger 2012)) simplify the search at the cost of making it incomplete for temporally expressive problems (Cushing et al. 2007). By keeping time lifted, our approach complicates the search (e.g., duplicate checking (Coles and Coles 2016)), but proves superior for temporally expressive problems.

5 Experimental Evaluation

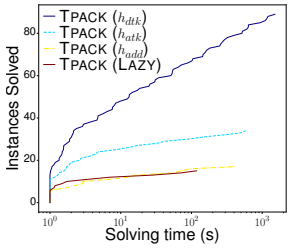
We implemented the proposed approach in a planner (called TPACK) that extends ENHSP (Scala, Haslum, and Thiébaux 2016) with native support for our TK: we allow the specification of the temporal axioms using a lisp-like syntax.

Our experimental analysis includes (whenever applicable) three other planners, ITSAT (Rankooh and Ghassem-Sani 2015), TFD (Eyerich, Mattmüller, and Röger 2012), and OPTIC (Benton, Coles, and Coles 2012) To assess the effect of our heuristics and of the two search schemata, we ran TPACK with 4 different configurations: TPACK (LAZY), implementing LAZY-SEARCH with the h_{add} heuristic, TPACK (h_{dtk}), TPACK (h_{atk}) and TPACK (h_{add}) implementing EAGER-SEARCH with the heuristic in parenthesis. All configurations ran using a best-first-search with $f(s) = g(s) + 4 \times h(s)$ where the g-value is the length of the prefix. During search, TPACK does not explore any state s' iff the search has already encountered another s with the same propositional assignments (i.e. $s.\mu = s'.\mu$) and smaller g-value ($g(s) < g(s')$). To ensure completeness, IDA*(Korf 1985) is run over the same search space in case best-first-search deemed the problem unsolvable. The consistency checking of a TN is implemented by explicitly splitting over its DNF representation. Each disjunct is a Simple Temporal Network (Dechter, Meiri, and Pearl 1991), solved using well known incremental algorithms (Cesta and Oddi 1996).

We consider two classes of problems, namely (i) temporally expressive domains directly inspired by our industrial application experience (HSP and a planning version of the job-shop-scheduling problem (Graham 1966)) and (ii) domains from the International Planning Competition (IPC), and a reformulation of these featuring intermediate effects.

For all the domains and all the planners, we report the number of instances solved. We ran all the experiments on a Xeon E5-2620 2.10GHz with 1800s/15GB of time/memory limits. TPACK and the benchmark instances are available at http://es.fbk.eu/people/amicheli/resources/aaai19.

Industrial domains. The HSP is encoded in TPP following the description in section 2. Instances scale the number of tanks (from 2 to 11) and items to be treated (from 1 to



(a) Cactus plot for HSP.

HSP (b)

#Items	ITSAT clip*	ITSAT cont.*	OPTIC clip	OPTIC cont.	TPACK (h_{dtk})	TPACK (h_{atk})	TPACK (h_{add})	TPACK (Lazy)
1	0	2	1	1	**10**	**10**	**10**	**10**
2	0	0	0	0	**10**	9	2	3
3	0	0	0	0	**10**	5	1	1
4	0	0	0	0	**10**	3	1	1
5	0	0	0	0	**10**	3	1	1
6	0	0	0	0	**9**	1	1	0
7	0	0	0	0	**9**	1	1	0
8	0	0	0	0	**8**	1	1	0
9	0	0	0	0	**7**	1	0	0
10	0	0	0	0	**7**	1	0	0
Tot.	0	2	1	1	**90**	35	18	16

MAJSP (c)

#Jobs	ITSAT clip*	ITSAT cont.*	OPTIC clip	OPTIC cont.	TPACK (h_{dtk})	TPACK (h_{atk})	TPACK (h_{add})	TPACK (Lazy)
1	NA	NA	22	0	56	56	**57**	**57**
2	NA	NA	0	0	41	40	37	**46**
3	NA	NA	0	0	25	25	23	**29**
4	NA	NA	0	0	11	10	10	**12**
Tot.	NA	NA	22	0	133	131	127	**144**

Temporal IPC (d)

Domain (# inst.)	ITSAT	OPTIC	TFD	TPACK (h_{dtk})	TPACK (h_{atk})	TPACK (h_{add})	TPACK (Lazy)
DRIVERLOG (20)	**18**	15	6	14	14	14	5
FLOORTILE (8)	**8**	7	6	4	4	4	2
MAPANALYSER (20)	**20**	0	**20**	**20**	**20**	**20**	0
MATCHCELLAR (10)	**10**	9	**10**	6	3	1	3
SATELLITE (20)	**20**	14	**20**	9	9	7	3
TMS (20)	**20**	1	1	1	1	0	0
Total (98)	**96**	46	63	54	51	46	13

Uncertainty IPC (e)

Domain (# inst.)	ITSAT clip*	ITSAT cont.*	OPTIC clip	OPTIC cont.	TPACK (h_{dtk})	TPACK (h_{atk})	TPACK (h_{add})	TPACK (Lazy)
DRIVERLOG (20)	0	0	0	5	6	**9**	8	7
FLOORTILE (8)	0	0	0	0	**4**	**4**	**4**	1
MAPANALYSER (20)	0	0	0	0	2	5	**6**	0
MATCHCELLAR (10)	4	0	3	**10**	5	1	1	2
SATELLITE (20)	1	0	0	1	**5**	**5**	4	2
TMS (20)	**1**	0	0	0	0	0	0	0
Total (98)	6	0	3	16	22	**24**	23	12

Figure 2: Industrial experiments. Coverage results for HSP and MAJSP (b-c) and for IPC-14 problems (d-e).

10) for a total of 100 instances. The same instances have been encoded in PDDL 2.1 using either the clip-action[3] or the window-container construction (Fox, Long, and Halsey 2004; Smith 2003; Cimatti et al. 2018). OPTIC was the only planner capable to handle the resulting temporally expressive instances. ITSAT parser does not support rational durations, so we used it on a reformulation of the instances where the actions duration was scaled to be integral. This scaling over-approximates the actual problem, and breaks the clip-action construction in general: we highlight this comparison difference in the tables with an asterisk in the affected columns. We devised a planning version of the job-shop scheduling problem by requiring items to be transported by robots to the machines performing the jobs. We refer to this domain as MAJSP. Instances scale with the number of robots (from 1 to 3), items to be processed (from 1 to 4), positions (from 2 to 6) and jobs (from 1 to 4) for a total of 240 instances, formulated both in TPP and PDDL 2.1.

Figures 2b-c report coverage results for the HSP and MAJSP instances, showing the detrimental effect of encoding time windows in PDDL 2.1. As the problem size grows, all the incarnations of TPACK outperform the competitors. In HSP, TPACK (h_{dtk}) solved most of the instances (all up to 5 items); the search with h_{dtk} expanded roughly linearly many nodes with the number of positions, and this translated to substantial speed-ups. This is also evident by observing figure 2a, reporting a coverage-versus-time comparison. In MAJSP, the advantage of using h_{dtk} is not prominent as in the HSP case, because the planning part (i.e., robots allocation) dominates the temporal constraints aspect, so the h_{dtk} overhead did not pay off. We ascribe this issue to the additive nature of h_{dtk} that badly overestimates the interference of the two components (planning and scheduling) of the problem. ITSAT crashed in all MAJSP instances.

IPC with and without intermediate effects. We took 6 domains from the IPC-14 competition (Vallati, Chrpa, and McCluskey 2018), choosing those where a problem generator was available[4]. For each domain, we experimented on both the original and the intermediate effects formulation. The latter results from adding temporal uncertainty to some actions and then using the technique in Cimatti et al. (2018) to compile uncertainty away. The resulting planning problem with intermediate effects is formulated both in PDDL 2.1 (with clip and container actions) and TPP (as per section 2).

As expected, TPACK does not dominate PDDL planners over IPC instances (figure 2d-e), yet its best configuration (EAGER-SEARCH and h_{dtk}) proved competitive with other forward search planners (such as TFD and OPTIC). Our findings show both that TPACK well complement some weaknesses of other heuristic search planners and, more importantly, that the translation PDDL to TPP is not only theoretically possible but also practical. TPACK suffers in domains such as MATCHCELLAR and SATELLITE probably for lack of reasoning mechanisms specifically targeting the structure of durative actions. Interestingly, when the very same problems are augmented with intermediate effects, TPACK's best configuration uniformly manages to solve small and medium sized instances, instead OPTIC manages to solve only the small ones, except for MATCHCELLAR where OPTIC's heuristic seems to provide exceptional guidance.

6 Conclusions

We proposed a novel temporal planning framework that is well-suited for naturally specifying a wide class of industrially-derived problems involving rich temporal metric constraints; we also showed how this framework is general enough to encode temporal constructs of alternative approaches (e.g., intermediate effects). For solving the arising planning problem, we use a forward heuristic search tailored for our rich temporal knowledge constructs. Our experimental findings show our technique being very effective over industrial problems that are largely out of the reach of state-of-the-art planners.

We think these ideas pave the way to several interesting research directions. First, the study of the theoretical properties of our formalism and of its fragments (e.g., complexity and decidability in continuous time) could lead to even more effective solvers. Second, the formalism itself can be extended to account for additional temporal patterns (e.g., durative actions).

[3] We used the clip-action construction instead of the strut-action as ITSAT crashed with all the strut instances. The coverage for OPTIC with strut-actions is identical to the one with clip-actions.

[4] Many of the original instances were out of reach for most of the planners, hence we generated smaller ones.

Acknowledgements. Thanks go to the anonymous reviewers, Alessandro Cimatti and Marco Roveri for their valuable feedback. This work is funded by the Autonomous Province of Trento in the scope of L.P. n.6/1999 with grant MAIS (Mechanical Automation Integration System) n. 2017-D323-00056 del. n. 941 of 16/06/2017.

References

Alur, R., and Henzinger, T. A. 1993. Real-time logics: Complexity and expressiveness. *Inf. Comput.* 104(1):35–77.

Alur, R., and Henzinger, T. A. 1994. A really temporal logic. *J. ACM* 41(1):181–204.

Bäckström, C. 1998. Computational aspects of reordering plans. *JAIR* 9:99–137.

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*.

Calvanese, D.; De Giacomo, G.; and Vardi, M. Y. 2002. Reasoning about actions and planning in LTL action theories. In *KR 2002, Toulouse, France, April 22-25, 2002*, 593–602.

Cesta, A., and Oddi, A. 1996. Gaining efficiency and flexibility in the simple temporal problem. In *TIME 1996, Key West, Florida, USA, May 19-20, 1996*, 45–50.

Cesta, A.; Cortellessa, G.; Fratini, S.; Oddi, A.; and Rasconi, R. 2009. The APSI Framework: a Planning and Scheduling Software Development Environment. In *ICAPS 2009 Application Showcase*.

Cimatti, A.; Do, M.; Micheli, A.; Roveri, M.; and Smith, D. E. 2018. Strong temporal planning with uncontrollable durations. *Artif. Intell.* 256:1–34.

Coles, A., and Coles, A. 2016. Have I been here before? state memoization in temporal planning. In *ICAPS 2016, London, UK, June 12-17, 2016.*, 97–105.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, 42–49.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. S. 2007. When is temporal planning really temporal? In *IJCAI 2007, Hyderabad, India, January 6-12, 2007*, 1852–1859.

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

Do, M. B., and Kambhampati, S. 2003. Sapa: A multi-objective metric temporal planner. *JAIR* 20:155–194.

Doherty, P., and Kvarnström, J. 2001. Talplanner: A temporal logic-based planner. *AI Magazine* 22(3):95–102.

Doherty, P., and Kvarnström, J. 2008. Temporal action logics. In *Handbook of Knowledge Representation*. 709–757.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2012. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Towards Service Robots for Everyday Environments - Recent Advances in Designing Service Robots for Complex Tasks in Everyday Environments*. 49–64.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *JAIR* 20:61–124.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *JAIR* 27:235–297.

Fox, M.; Long, D.; and Halsey, K. 2004. An investigation into the expressive power of PDDL2.1. In *ECAI 2004*, 328–342.

Frank, J., and Jónsson, A. 2003. Constraint-based Attribute and Interval Planning. *Constraints* 8(4):339–364.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.* 173(5-6):619–668.

Graham, R. L. 1966. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal* 45(9):1563–1581.

Hähnle, R. 2001. Tableaux and related methods. In *Handbook of Automated Reasoning*. 100–178.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *AIPS 2000, Breckenridge, CO, USA, April 14-17, 2000*, 140–149.

Karpas, E.; Wang, D.; Williams, B. C.; and Haslum, P. 2015. Temporal landmarks: What must happen, and when. In *ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, 138–146.

Kleene, S. 1967. *Mathematical Logic*. J. Wiley & Sons.

Korf, R. E. 1985. Iterative-deepening-a*: An optimal admissible tree search. In *IJCAI 1985, Los Angeles, CA, USA, August 1985*, 1034–1036.

Manier, M.-A., and Bloch, C. 2003. A classification for hoist scheduling problems. *International Journal of Flexible Manufacturing Systems* 15(1):37–55.

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

McDermott, D. V. 2000. The 1998 AI planning systems competition. *AI Magazine* 21(2):35–55.

Nau, D. S.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: an HTN planning system. *JAIR* 20:379–404.

Phillips, L. W., and Unger, P. S. 1976. Mathematical programming solution of a hoist scheduling program. *AIIE transactions* 8(2):219–225.

Rankooh, M. F., and Ghassem-Sani, G. 2015. ITSAT: an efficient sat-based temporal planner. *JAIR* 53:541–632.

Rintanen, J. 2007. Complexity of concurrent temporal planning. In *ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, 280–287.

Rintanen, J. 2015. Impact of modeling languages on the theory and practice in planning research. In *AAAI*, 4052–4056.

Scala, E.; Haslum, P.; and Thiébaux, S. 2016. Heuristics for numeric planning via subgoaling. In *IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 3228–3234.

Smith, D.; Frank, J.; and Cushing, W. 2008. The ANML language. In *KEPS 2008*.

Smith, D. E. 2003. The case for durative actions: A commentary on PDDL2.1. *JAIR* 20:149–154.

To, S. T.; Johnson, B.; Roberts, M.; and Aha, D. W. 2017. A new approach to temporal planning with rich metric temporal properties. In *ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017.*, 288–296.

Vallati, M.; Chrpa, L.; and McCluskey, T. L. 2018. What you always wanted to know about the deterministic part of the international planning competition (IPC) 2014 (but were too afraid to ask). *Knowledge Eng. Review* 33:e3.

Wang, D., and Williams, B. C. 2015. tburton: A divide and conquer temporal planner. In *AAAI 2015 January 25-30, 2015, Austin, Texas, USA.*, 3409–3417.