# Robustness Envelopes for Temporal Plans

Michael Cashmore[1], Alessandro Cimatti[2], Daniele Magazzeni[1], Andrea Micheli[2], Parisa Zehtabi[1]
[1]King's College London, United Kingdom
{michael.cashmore, daniele.magazzeni, parisa.zehtabi}@kcl.ac.uk,
[2]Fondazione Bruno Kessler, Italy
{cimatti, amicheli}@fbk.eu

**Abstract**

To achieve practical execution, planners must produce temporal plans with some degree of run-time adaptability. Such plans can be expressed as Simple Temporal Networks (STN), that constrain the timing of action activations, and implicitly represent the space of choices for the plan executor.

A first problem is to verify that all the executor choices allowed by the STN plan will be successful, i.e. the plan is valid. An even more important problem is to assess the effect of discrepancies between the model used for planning and the execution environment.

We propose an approach to compute the "robustness envelope" (i.e., alternative action durations or resource consumption rates) of a given STN plan, for which the plan remains valid. Plans can have boolean and numeric variables as well as discrete and continuous change. We leverage Satisfiability Modulo Theories (SMT) to make the approach formal and practical.

## 1 Introduction

Planning is the problem of automatically synthesizing a course of actions to achieve a desired goal. In many applications, time and continuous resources need to be modeled, using languages such as PDDL2.1 (Fox and Long 2003). In this setting, plans can be represented as Simple Temporal Networks (STNs) (Dechter, Meiri, and Pearl 1991), so that the specific timing of action execution is not constrained at planning time (e.g. with a fixed schedule). This gives the executor the ability to choose such timings, but some runtime reasoning is required to respect the plan constraints. A vast literature is concerned with the efficient execution of STN plans (e.g. (Muscettola, Morris, and Tsamardinos 1998)).

When employing STN plans, one first issue is to verify that all executor choices allowed by the plan will be successful, i.e. the plan is valid. An even more important issue is to assess the effect of any discrepancy between the model used for planning and the execution environment (e.g. a difference in the consumption rate of a resource). A valid plan is guaranteed to be successful for any choice of the executor — but only under the assumption that the domain model faithfully represents the actual execution environment. Unfortunately, this may not be the case.

In this paper, we tackle both problems in a unified framework. First, we propose techniques to formally validate an STN plan and to synthesize its least restrictive validity bounds in a PDDL2.1 setting. Existing works are limited to the generation of valid-by-construction STN plans (e.g (Frank and Jónsson 2003; Cesta et al. 2009)) or to their validation for purely temporal timelines (Cesta et al. 2010).

Second, we propose an algorithm to formally synthesize all the values of a set of parameters in the planning model that preserve the validity of a given STN plan. This tells us to which extent an STN plan is robust to discrepancies in the values of such parameters between the formal model and the execution environment.

Both of these problems aim at synthesizing a representation of all the variations either in the plan execution or in the domain model that retain the formal validity of the plan according to the planning language semantics. We call such a representation the "robustness envelope" of the plan.

We uniformly approach these problems by adopting the framework of Satisfiability Modulo Theory (SMT): we encode all the possible execution traces of the STN plan as an SMT formula. We use real-valued parameters to symbolically represent the quantities subject to the synthesis in the plan and in the problem, and we use quantifier elimination techniques to synthesize a closed-form of the set of parameter values. The approach is made practical by the availability of efficient SMT solvers, and can be applied to temporal plans with discrete as well as continuous change.

The paper is structured as follows. In Section 2 we give the needed SMT background. Section 3 formalizes the problems we tackle, while in Section 4 we describe the encoding we use to solve them. Section 5 presents an under-approximation technique aimed at simplify robustness envelopes. In Section 6 we discuss related work and Section 7 experimentally analyzes our techniques. Finally, in Section 8 we draw conclusions and discuss future work.

## 2 Logical Background

Given a first-order formula $\psi$ in a background theory $T$ the satisfiability modulo theory (SMT) problem consists in deciding whether there exists a model (i.e. an assignment to the free variables in $\psi$) that satisfies $\psi$. For example, consider the formula $(x \leq y) \wedge (x + 3 = z) \vee (z \geq y)$ in the theory of real numbers ($x, y, z \in \mathbb{R}$). The formula is satisfiable and a valid model is $\{x := 5,\ y := 6,\ z := 8\}$.

An SMT solver (Barrett et al. 2009) is a decision procedure which solves the satisfiability problem for a formula expressed in a decidable subset of first-order logic.

SMT solvers can support different theories. A widely used theory is Linear Real Arithmetic (LRA). A formula in LRA is an arbitrary Boolean combination, or universal ($\forall$) and existential ($\exists$) quantification, of atoms in the form $\sum_i a_i x_i \bowtie c$ where $\bowtie \in \{>, <, \geq, \leq, \neq, =\}$, every $x_i$ is a real variable and every $a_i$ and $c$ are real constants. We denote with QF_LRA the quantifier-free fragment.

In order to deal with quantifiers in LRA, many techniques have been developed and implemented in SMT solvers. Several techniques have been developed for removing quantifiers from an LRA formula (Monniaux 2008): they transform any LRA formula containing quantifiers into an *equivalent* QF_LRA formula. These techniques formally eliminate variables from an LRA formula at a cost that is doubly exponential in time and space in the original formula size, and are extremely useful for synthesis tasks.

## 3 Formalization

We start by defining our planning language: we adopt the full PDDL 2.1 (Fox and Long 2003) with continuous change.

**Definition 1** *A **planning problem** $\mathcal{P}$ is a tuple $\langle P, V, A, I, G\rangle$, where $P$ is a set of propositions; $V$ is a set of real variables, called fluents; $A$ is a set of durative and instantaneous actions; $I : P \cup V \rightarrow \{\top, \bot\} \cup \mathbb{R}$ is the total function describing the initial state of the predicates and the fluents. $G : P \cup V \rightarrow \{\top, \bot\} \cup \mathbb{R}$ is a (possibly partial) function indicating the goal condition. A durative action $a$ is a tuple $\langle pre_a, \mathit{eff}_a, dur_a\rangle$, where $pre_a$ is a set of conditions for the actions partitioned in three subsets $pre_{\vdash a}$, $pre_{\leftrightarrow a}$ and $pre_{\dashv a}$ of at-start, over-all and at-end conditions; $\mathit{eff}_a$ is the set of action effects, partitioned in seven sets: $\mathit{eff}^+_{\vdash a}$ (positive starting effects), $\mathit{eff}^-_{\vdash a}$ (negative starting effects), $\mathit{eff}^{num}_{\vdash a}$ (numeric starting effects), $\mathit{eff}^+_{\dashv a}$ (positive ending effects), $\mathit{eff}^-_{\dashv a}$ (negative ending effects), $\mathit{eff}^{num}_{\dashv a}$ (numeric ending effects) and $\mathit{eff}^{num}_{\leftrightarrow a}$ (continuous numeric effects); and $dur_a$ is a set of duration constraints. An instantaneous action $a$ is a tuple $\langle pre_a, \mathit{eff}_a\rangle$, where $pre_a$ is a set of pre-conditions and $\mathit{eff}_a$ is the set of action effects, partitioned in $\mathit{eff}^+_a$ (positive effects), $\mathit{eff}^-_a$ (negative effects) and $\mathit{eff}^{num}_a$ (numeric effects).*

In the usual PDDL 2.1 setting, a plan is defined as a set of actions associated with a starting time and a duration. We define this kind of plans as time-triggered plans.

**Definition 2** *A **time-triggered plan** $\pi$ for a planning problem $\mathcal{P} \doteq \langle P, V, A, I, G\rangle$ is a set of tuples $\langle t, a, d\rangle$, with $t \in \mathbb{R}_{\geq 0}$, $a \in A$ and $d \in \mathbb{R}_{>0}$ iff $a$ is a durative action.*

For the sake of brevity, we omit the formal definition of validity for such a plan, which can be found in (Fox and Long 2003). Here, it suffices to remind oneself that a plan is valid if by simulating the system controlled by the plan, all the prescribed actions are applicable (all their conditions are satisfied at the time the action is executed) and the goal is reached after the last action terminates.

We define an STN plan as a constraint network of time points indicating the starting or the ending of actions. Note that the STN plan contains all the information of, and is strictly more general than a time-triggered plan. Moreover, that it is not necessary to first find a time-triggered plan in order to generate an STN plan.

**Definition 3** *An **STN plan** $\pi$ for $\mathcal{P} \doteq \langle P, V, A, I, G\rangle$ is a tuple $\langle T, C\rangle$, where $T$ is the set of time points $\{z\} \cup \{t^s_{da}, t^e_{da} \mid da \text{ is a durative action instance}\} \cup \{t_a \mid a \text{ is an instantaneous action instance}\}$ and $C$ is a set of constraints in the form $t_i - t_j \leq b$ with $t_i, t_j \in T$, and $b \in \mathbb{R}$.*

Finally, we can define the validity of an STN plan by considering the set of all possible time-triggered plans that are compatible with the STN specification. If all such plans are valid, we say that the STN plan is valid.

**Definition 4** *Given an STN plan $\pi \doteq \langle T, C\rangle$ and an assignment $\mu : T \rightarrow \mathbb{R}$ s.t. $\mu(z) = 0$, the **induced time-triggered plan** by $\mu$ is the time-triggered plan $tt(\mu) \doteq \{\langle \mu(t^s_{da}), da, \mu(t^e_{da}) - \mu(t^s_{da})\rangle \mid da \text{ is a durative action}\} \cup \{\langle \mu(t_a), a, 0\rangle \mid a \text{ is an instantaneous action}\}$.*

**Definition 5** *An STN plan $\pi \doteq \langle T, C\rangle$ for $\mathcal{P}$ is **valid** if for each assignment $\mu : T \rightarrow \mathbb{R}$ s.t. $\mu(z) = 0$ and for all $t_i - t_j \leq b \in C$ $\mu(t_i) - \mu(t_j) \leq b$, the time-triggered plan $tt(\mu)$ is valid for $\mathcal{P}$.*

The first problem we consider is checking the validity of an STN plan: we call this problem *STN Plan Validation*.

**Running example.** Consider for example an exploration robot, initially in location $S$, that is tasked to collect some data from location $D$ and then reach location $T$ to transmit the data to a control station. The robot is equipped with a battery that is initially 100% full. Suppose that the robot can reach location $D$ in a minimum of 60 minutes at full speed and must be there before 100 minutes to get the data. Moreover, the time needed for the journey from $D$ to $T$ takes a minimum of 120 minutes and the robot must transmit the data no later than 200 minutes after getting the data. For this example, we assume that the moving durations between locations are independent one another and that the battery is drained at a constant rate of 0.4% per minute of traveling (in a more realistic model the battery draining model would be much more complex and also data acquisition and transmission would require energy and time: we disregard these details for the sake of simplicity). A time-triggered plan to achieve the objective in this example problem is the following: $\pi_{tt} \doteq \{\langle 0, \text{“go from } S \text{ to } D\text{”}, 60\rangle, \langle 60.1, \text{“go from } D \text{ to } T\text{”}, 120\rangle\}$. This plan is valid: one can simulate the plan in the formal model reaching the goal without completely draining the battery (in fact, at the end of the plan, the battery would be still 28% full) and respecting all time requirements. The plan is also optimal with respect to the make-span (assuming a PDDL 2.1 $\epsilon$-separation of 0.1 time units): it achieves the goal in the minimum possible time. A possible STN plan for this problem, instead, is $\pi_{STN}$ depicted in Figure 1. Also this plan is valid, because every execution

$$\begin{tikzcd}
z \arrow[r, "{[0, 0]}"] & t^s_{SD} \arrow[r, "{[60, 80]}"] & t^e_{SD} \arrow[r, "{[0, 0]}"] & t^s_{DT} \arrow[r, "{[120, 150]}"] & t^e_{DT}
\end{tikzcd}$$

Figure 1: The $\pi_{STN}$ example STN plan: nodes are time points and edges are temporal constraints. A label $[l, u]$ of an edge from $x$ to $y$ indicates the constraint $y-x \leq u \wedge x-y \leq -l$. The STN plan reads as follows: go from $S$ to $D$ (labeled as $SD$) at time 0 and arrive there not before 60 minutes and no later that 80 minutes, then immediately drive to $T$ (labeled as $DT$) arriving no earlier than 120 minutes after leaving $D$ and no later than 150 minutes after leaving $D$.

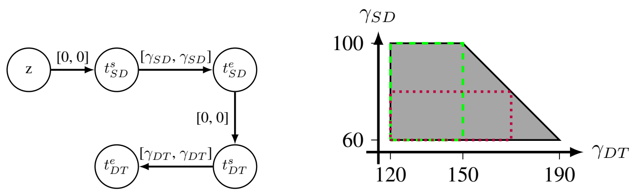

Figure 2: (Left) A parametrization of $\pi_{STN}$. (Right) The robustness envelope for the parametrized $\pi_{STN}$: any parameter assignment within the gray-filled envelope yields a valid plan. The two dashed rectangles show two possible parameter decouplings of this envelope (described in Section 5).

that satisfies the plan constraints will achieve the goal without draining the battery. We highlight that this example STN plan is totally-ordered for the sake of simplicity, but all our techniques can deal with any STN as per Def 3.

**Synthesis of robustness envelopes.** One important research question concerning STN plans is how to characterize the situations in which the plan is guaranteed to work. On the same line, checking whether a given STN plan is maximally "flexible" or if we can relax the constraint bounds while keeping the STN structure to allow the plan to work in more diverse situations, is also crucial.

Ideally, one may want to synthesize the weakest possible bounds in the STN constraints that still guarantee the STN plan validity for the given planning problem. To this end, we propose to introduce real-valued symbolic parameters in the formulation of an STN plan and a technique that is able to synthesize all the possible values for such parameters that keep the plan valid. Consider again the $\pi_{STN}$ plan and suppose that we are interested in maximizing the possible durations of the two moving actions. We introduce two parameters $\gamma_{SD}$ and $\gamma_{DT}$ in the plan, and use them to constrain the duration of the movement from $S$ to $D$ and from $D$ to $T$, respectively. Given this parametrization of the plan, the problem we are after consists in finding all the values for the two parameters such that an STN plan instantiated with these values is valid. The parametrized STN plan and the *robustness envelope* resulting from the synthesis are shown in Figure 2. With this parametrization idea, we can synthesize the robustness envelope of an STN plan by creating a fresh parameter for some (possibly all) edges of interest. The case in Figure 2 is an example of this parametrization: the temporal distance between the two moving actions is fixed to 0 while we parametrized the duration of the two plan actions.

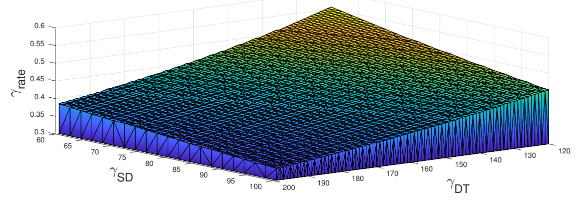

Figure 3: A 3D visualization of the robustness envelope for the example problem.

We further extend and generalize this idea by allowing parameters to occur also in the planning problem specification: in fact, the same approaches will be used to synthesize a robustness envelope for problem parameters, formally characterizing all the possible parameter values in the problem definition for which a given plan (that could be manually or automatically generated) is guaranteed to work. This problem is extremely important to evaluate plans before putting them in operation, allowing a formal analysis of the situations for which the plan is "robust".

Consider again our running example, if we add a parameter $\gamma_{rate}$ indicating the rate of battery consumption per minute (in the original formulation $\gamma_{rate}$ was fixed to $0.4$), we could synthesize all the draining rates for which $\pi_{STN}$ is valid. The robustness envelope in this case is $\gamma_{rate} \in [0, \frac{10}{23}]$, so we are sure that even if the battery were drained faster than expected, the plan execution would still succeed.

Finally, we can combine these two ideas by synthesizing at the same time parameters for the plan and the planning problem so that we obtain a robustness envelope indicating a class of planning problems for which an STN plan exists.

In our example, if we consider the set of parameters $\gamma_{rate}$, $\gamma_{SD}$ and $\gamma_{DT}$ at the same time, we obtain the envelope depicted in Figure 3. Note that the robustness envelope shown in Figure 2 is just a projection of this envelope for $\gamma_{rate} = 0.4$.

In order to uniformly formalize the robust envelope synthesis problem, we only need to introduce a planning model comprising parameters in either the problem or the plan specifications.

**Definition 6** *A* ***parametrized planning problem*** $\mathcal{P}_\Gamma$ *is a tuple* $\langle \Gamma, \mathcal{P} \rangle$*, where* $\Gamma$ *is a finite set of real-valued parameters* $\{\gamma_1, \cdots, \gamma_n\}$ *and* $\mathcal{P}$ *is a planning problem in which conditions, effects, goals and initial states can contain parameters.*

Intuitively, we are introducing a set of symbols $\Gamma$ that can be used in expressions where real-typed constants are usually allowed. In this way, the user can define the quantities that are of interest for the synthesis. We also allow the use of parameters in the plan specification by generalizing the definition of STN plans.

**Definition 7** *A* ***parametrized STN plan*** $\pi_\Gamma$ *for a parametrized planning problem* $\mathcal{P}_\Gamma \doteq \langle \Gamma, \mathcal{P} \rangle$ *is a tuple* $\langle T, C, C_\Gamma \rangle$*, where* $\langle T, C \rangle$ *is an STN plan for* $\mathcal{P}$ *and* $C_\Gamma$ *is a set of constraints in the form* $t_i - t_j \leq \gamma_i$ *with* $t_i \in T$*,*

*$t_j \in T$, and $\gamma_i \in \Gamma$.*

For our purposes, we are interested in finding values (i.e. assignments) to the parameters that when substituted in the problem and the plan specifications yield a valid plan as per Definition 5.

**Definition 8** *Given a parametrized planning problem $\mathcal{P}_\Gamma$, a parametrized STN plan $\pi_\Gamma$ for $\mathcal{P}_\Gamma$ and an assignment $\mu : \Gamma \rightarrow \mathbb{R}$ of real values to all the parameters in $\Gamma$, we define the **parameter-assigned planning problem** $\mathcal{P}_\Gamma(\mu)$ and the **parameter-assigned STN plan** $\pi_\Gamma(\mu)$ as the planning problem and plan where any parameter $\gamma_i$ is substituted with its assigned value $\mu(\gamma_i)$.*

At this point, we can formally define the objective of our synthesis: the envelope of all the valid assignments to parameters that make a parametrized STN plan valid.

**Definition 9** *Given a parametrized STN plan $\pi_\Gamma$ for a parametrized planning problem $\mathcal{P}_\Gamma$, the **robustness envelope** is a relation $\Lambda \subseteq \mathbb{R}^N$ where $N \doteq |\Gamma|$, such that for all the assignments $\mu : \Gamma \rightarrow \mathbb{R}$ in which $\mu(\gamma_i) = v_i$ with $\langle v_1, \cdots, v_N \rangle \in \Lambda$, $\pi_\Gamma(\mu)$ is a valid plan for $\mathcal{P}_\Gamma(\mu)$.*

## 4 Encoding in SMT

We can now present our encoding in the SMT framework: we will use the same encoding in different ways to tackle all the problems discussed in the previous section. The basic idea is to encode all the time-triggered plan executions in the planning problem as an SMT formula.

We start by considering a planning problem $\mathcal{P} \doteq \langle P, V, A, I, G \rangle$ without parameters and an STN plan $\pi \doteq \langle T, C \rangle$. In the following, let $T_z$ be the set $T \setminus \{z\}$ and let $H$ be the cardinality of $T_z$. In order to encode the execution of $\mathcal{P}$ controlled by $\pi$, we need to model $H$ "time points", each corresponding to the starting or ending of an action (or to a timed-initial-literal[1]). This is somehow similar to a SAT-Plan (Kautz and Selman 1992) encoding, but here we have no need to increase the bound since the plan $\pi$ fixes the exact length. The variables used in the encoding (indicated with $Var^\pi$) are listed below.

- $p_i$ with Boolean type, for each $p \in P$ and each $i \in [1, H]$;
- $f_i$ with Real type, for each $f \in V$ and each $i \in [1, H]$;
- $t_i$ with Real type, for each $i \in [1, H]$;
- $pos_t$ with Real type, for each $t \in T_z$;
- $val_t$ with Real type, for each $t \in T_z$;

In addition, we define the expressions (not variables) $p_0 \doteq I(p)$ for each $p \in P$, $f_0 \doteq I(f)$ for each $f \in V$, $t_0 \doteq 0$, $pos_z \doteq 0$ and $val_z \doteq 0$.

We subdivide our encoding in three SMT expressions: indicated as $enc^\pi_{tn}$, $enc^\pi_{\mathit{eff}}$ and $enc^\pi_{proofs}$. The formula $enc^\pi_{tn}$ encodes the temporal constraints imposed by $\pi$ limiting the possible orderings of time points. The formula $enc^\pi_{\mathit{eff}}$ encodes the effects of each time point on the variables $f_i$, while $enc^\pi_{proofs}$ encodes the validity properties of the plan, namely that the conditions of each executed action are satisfied, that the goal is reached, and that the $\epsilon$-separation constraint imposed by PDDL 2.1 is respected. We are designing our encoding in such a way that $\pi$ is a valid plan for $\mathcal{P}$ if $enc^\pi_{tn} \wedge enc^\pi_{\mathit{eff}}$ is satisfiable and $enc^\pi_{tn} \wedge enc^\pi_{\mathit{eff}} \rightarrow enc^\pi_{proofs}$ is a valid formula. The first check ensures that the formula is not-trivially-unrealizable due to inconsistencies in the STN, while the second validates the plan, checking that each execution allowed by the plan is valid and reaches the goal.

The formula $enc^\pi_{tn}$ encodes the temporal network and its constraints, it is defined as the conjunction of the following constraints.

- $val_t \geq 0$ for each $t \in T_z$;
- $\bigvee_{i=1}^{H} pos_t = i$ for each $t \in T_z$;
- $\bigwedge_{i=1}^{H} (pos_t = i \rightarrow t_i = val_t)$ for each $t \in T_z$;
- $\bigwedge_{o \in T_z, o \neq t} (val_t > val_o \rightarrow pos_t > pos_o)$ for each $t \in T_z$;
- $\bigwedge_{o \in T_z, o \neq t} pos_o \neq pos_t$ for each $t \in T_z$;
- $val_{t_i} - val_{t_j} \leq b$ for each $t_i - t_j \leq b \in C$.

The formula $enc^\pi_{\mathit{eff}}$ encodes the effects at each time point: $enc^\pi_{\mathit{eff}} \doteq \bigwedge_{t \in T_z} \bigwedge_{i=1}^{H} (pos_t = i \rightarrow \eta(t, i))$, where $\eta(t, i)$ encodes the effects of time point $t$ at step $i$. $\eta(t, i)$ is the conjunction of the following constraints:

- $p_i$ if $p \in \mathit{eff}_{add}$; $\neg(p_i)$ if $p \in \mathit{eff}_{del}$;
- $f_i = x$ with $f := x \in \mathit{eff}_{num}$;
- $f_i = f_{i-1} + \iota(f, i) + x$ with $f {+}{=}\, x \in \mathit{eff}_{num}$;
- $f_i = f_{i-1} + \iota(f, i) - x$ with $f {-}{=}\, x \in \mathit{eff}_{num}$;
- $p_i = p_{i-1}$ for $p \in P$ if $p \notin \mathit{eff}_{add} \cup \mathit{eff}_{del}$;
- $f_i = f_{i-1} + \iota(f, i)$ for each $f \in V$ if $f \notin \mathit{eff}_{num}$.

The sets of effects $\mathit{eff}_{add}$, $\mathit{eff}_{del}$, $\mathit{eff}_{num}$ depend on the kind of action associated with the time point $t$. If $t = t_a$ being $a$ an instantaneous action, $\mathit{eff}_{add} = \mathit{eff}^{+}_{a}$, $\mathit{eff}_{del} = \mathit{eff}^{-}_{a}$, $\mathit{eff}_{num} = \mathit{eff}^{num}_{a}$; if $t = t^s_{da}$ being $da$ a durative action, $\mathit{eff}_{add} = \mathit{eff}^{+}_{\vdash da}$, $\mathit{eff}_{del} = \mathit{eff}^{-}_{\vdash da}$, $\mathit{eff}_{num} = \mathit{eff}^{num}_{\vdash da}$; otherwise, if $t = t^e_{da}$ being $da$ a durative action, $\mathit{eff}_{add} = \mathit{eff}^{+}_{\dashv da}$, $\mathit{eff}_{del} = \mathit{eff}^{-}_{\dashv ad}$, $\mathit{eff}_{num} = \mathit{eff}^{num}_{\dashv da}$. The term $\iota(f, i)$ encodes the increment of value for fluent $f$ between steps $i - 1$ and $i$ due to continuous change and it is defined as $\sum_{da \in A} ((pos_{t^s_{da}} < i \wedge pos_{t^e_{da}} \geq i)\,?\, \mathit{eff}^{num}_{\leftrightarrow da|f}(t_i - val_{t^s_{da}}) - \mathit{eff}^{num}_{\leftrightarrow da|f}(t_{i-1} - val_{t^s_{da}}) : 0)$[2], where $\mathit{eff}^{num}_{\leftrightarrow da|f}(x)$ indicates the continuous effect of durative action $da$ for fluent $f$ where the time parameter (`#t`) is substituted with $x$.

Finally, the formula $enc^\pi_{proofs}$ constituting the right-hand side of the encoding captures the proof-obligations for the plan. We indicate the trivial translation of a PDDL expression $e$ into an SMT formula using variables at time $i$ as $[\![e]\!]_i$: for example, a PDDL expression `(and (p) (= (f) 5))` is translated as $p_i \wedge f_i = 5.0$. $enc^\pi_{proofs}$ is the conjunction of the following formulae.

- Goal is achieved: $\bigwedge_{g \in G} [\![g]\!]_H$.
- Durative action conditions are satisfied. For each durative action $da$:
  - $\bigwedge_{i=1}^{H} pos_{t^s_{da}} = i \rightarrow \bigwedge_{cpre_{\vdash da}} [\![c]\!]_i$;

[1]For the sake of brevity, we did not formally include timed-initial-literals (TILs) in our formulation. However, these can be handled as instantaneous actions having the TIL as an effect that are artificially added to the STN plan and constrained to happen at the TIL time. With our technique it is therefore possible to synthesize the set of possible timings for a TIL that make a plan valid.

[2]The syntax $\phi\ ?\ term_1\ :\ term_2$ indicates the if-then-else term (Kim, Somenzi, and Jin 2009). The semantics is analogous to the if-then-else expression available in programming languages.

- $\bigwedge_{i=1}^{H} pos_{t^e_{da}} = i \rightarrow \bigwedge_{c \in pre_{\dashv da}} [\![c]\!]_i$;
- $\bigwedge_{i=1}^{H} (pos_{t^e_{da}} > i \wedge pos_{t^s_{da}} \leq i) \rightarrow \bigwedge_{c \in pre_{\leftrightarrow da}} [\![c]\!]_i$.

- Instantaneous action conditions are satisfied: $\bigwedge_{i=1}^{H} pos_{t_a} = i \rightarrow \bigwedge_{c \in pre_a} [\![c]\!]_i$ if $a$ is an instantaneous action.
- Duration conditions are respected: $t^e_{da} - t^s_{da} \bowtie b$ for each durative action $da$ and for each constraint $duration \bowtie b \in dur_{da}$.
- Over-all invariants are respected: $\bigwedge_{i=1}^{H} (pos_{t^e_{da}} > i \wedge pos_{t^s_{da}} \leq i) \rightarrow \forall \hat{t}. 0 < \hat{t} < (t_{i+1} - t_i) \rightarrow CC(da, i)$ if $da$ is a durative action.
- $\epsilon$-separation: $val_x - val_y >= \epsilon \vee val_x - val_y <= -\epsilon$ for each pair of interfering time points $\langle x, y \rangle$.

The formula $CC(da, i)$ where $da$ is a durative action is defined as $\bigwedge_{c \in pre_{\leftrightarrow da}} [\![c]\!]_i [f_i \rightarrow \phi(f, i) \mid f \in V]$, where $\phi(f, i)$ is $\sum_{da \in A} ((pos_{t^s_{da}} \leq i \wedge pos_{t^e_{da}} > i) ? (\mathit{eff}^{\,num}_{\leftrightarrow da|f}(\hat{t} + t_i - val_{t^s_{da}}) - \mathit{eff}^{\,num}_{\leftrightarrow da|f}(t_i - val_{t^s_{da}})) : 0)$.

**Theorem 1** *Given a planning problem $\mathcal{P}$ and an STN plan $\pi$ for $\mathcal{P}$, $\pi$ is valid if and only if $enc^{\pi}_{tn} \wedge enc^{\pi}_{\mathit{eff}}$ is satisfiable and $enc^{\pi}_{tn} \wedge enc^{\pi}_{\mathit{eff}} \rightarrow enc^{\pi}_{proofs}$ valid.*

At this point, we can exploit the same encoding to address the other three synthesis problems. We can symbolically express the parameter envelope by adding one real-valued SMT variable for each parameter and using them in our encoding in place of the constant values used for validation. Formally, we indicate with $enc^{\pi_\Gamma}_{tn}$, $enc^{\pi_\Gamma}_{\mathit{eff}}$ and $enc^{\pi_\Gamma}_{proofs}$ the SMT formulae obtained by applying the encoding described above to a parametrized planning problem and plan. The resulting formulae are defined on the same set of variables of the validation encoding with the addition of the real-typed variables $\bar{\Gamma} \doteq \{par_i \mid \gamma_i \in \Gamma\}$. Whenever in an expression or effect we encounter a parameter $\gamma_i$, we encode it using the SMT variable $par_i$. In this way, we are left with a formula defined over the variables $Var^{\pi} \cup \bar{\Gamma}$. So, by using appropriate quantifications, we can express (and compute via quantifier elimination) the assignments to parameters that make the plan valid. In particular, we can compute the robustness envelope as the formula $\rho(\bar{\Gamma})$ defined as:

$$\exists \bar{X}.(enc^{\pi_\Gamma}_{tn} \wedge enc^{\pi_\Gamma}_{\mathit{eff}}) \wedge \forall \bar{X}.((enc^{\pi_\Gamma}_{tn} \wedge enc^{\pi_\Gamma}_{\mathit{eff}}) \rightarrow enc^{\pi_\Gamma}_{proofs}).$$

The models of $\rho(\bar{\Gamma})$ are all and only the parameter values that make the plan valid for the problem. This formulation reflects the plan validity check described above. The existential check ensures that we limit ourselves to parameter assignments that do not trivially violate the consistency of the execution by making the STN unsatisfiable or that make two effects clash at the same time. The universal check, ensures that under any possible execution of the system, the goals, action conditions, resource constraints and $\epsilon$-separation constraints are satisfied. We can use quantifier-elimination techniques to compute a closed-form formulation of $\rho(\bar{\Gamma})$. The resulting formula is effectively a specification of the robustness envelope as per Def 9: each model corresponds to an element of $\Lambda$.

## 5 Parameter Decoupling

A parameter envelope can be an arbitrary set of points in the space of the possible parameter values. This makes it difficult to represent it compactly and also to reason efficiently on the parameter values. In particular, if we want to use it during the execution of the plan to grant the maximum freedom of choice, we need an executor that is able to reason on such a representation and to extrapolate the possible values for a particular quantity being controlled. For this reason it might be convenient to give up some of the valid parameter values in the plan to obtain a simpler envelope that makes reasoning easier. Naturally, in this case we need to under-approximate the envelope because we want to retain the guarantee that any assignment we choose from such an envelope yields a valid plan execution. We propose an under-approximation strategy that reduces the (possibly very complex) parameter envelope to a set of closed intervals over the real numbers, one for each parameter, with the guarantee that by picking any parameter assignment where each parameter value is chosen form the corresponding interval, we get an assignment yielding a valid plan. We call such a simplified envelope "decoupled" because the range of values of each parameter becomes independent of all the others.

**Definition 10** *A **decoupled robustness envelope** for a parametrized STN plan $\pi_\Gamma$ for $\mathcal{P}_\Gamma$ is a function $\theta : \Gamma \rightarrow \mathbb{R} \times \mathbb{R}$ such that for all the assignments $\mu : \Gamma \rightarrow \mathbb{R}$ in which $\mu(\gamma_i) \in [l, h]$ with $\langle l, h \rangle \doteq \theta(\gamma_i)$, $\pi_\Gamma(\mu)$ is valid for $\mathcal{P}_\Gamma(\mu)$.*

We can compute a decoupled robustness envelope from the formula $\rho(\Gamma)$ as follows. First, we define two sets of lower-bound $LB \doteq \{lb_i \mid \gamma_i \in \Gamma\}$ and upper-bound $UB \doteq \{ub_i \mid \gamma_i \in \Gamma\}$ real-valued SMT variables.

Then, we encode all the possible decoupled robustness envelopes as the formula $RE(LB, UB)$ defined as: $(\bigwedge_{\gamma_i \in \Gamma} lb_i \leq ub_i) \wedge \forall \bar{\Gamma}.((\bigwedge_{\gamma_i \in \Gamma} lb_i \leq par_i \leq ub_i) \rightarrow \rho(\bar{\Gamma}))$.

Any model $\mu$ of $RE(LB, UB)$ corresponds to a decoupled robustness envelope $\theta$ that assigns to each $\gamma_i$ the pair of real numbers $\langle \mu(lb_i), \mu(ub_i) \rangle$. In this space, we are interested in fixing one specific decoupled robustness envelope that maximizes some user-defined criterion. In fact, there is no single criterion that yields an absolutely best decoupled robustness envelope. See for example the two decouplings highlighted in green and purple in Figure 2: both are maximal in the sense that we cannot add possible values to one parameter without losing some valid values for another, and are thus incomparable. An example of a possible objective for this maximization is the sum of the length of the intervals for each parameter. This can be formulated as the following optimization problem on the formula $RE(LB, UB)$.

$$maximize \quad \sum_{\gamma_i \in \Gamma} (ub_i - lb_i) \quad s.t. \quad RE(LB, UB)$$

In the running example, the result of this maximization would be the region highlighted in green in Figure 2: $\gamma_{SD} \in [60, 100]$ and $\gamma_{DT} \in [120, 150]$, yielding a total sum of 70 minutes. In other cases, we might be interested in widening as much as possible the intervals of some parameters sacrificing those of some of the others, therefore, we can weight the sum to be maximized. For example, if we are only interested in widening the interval for $\gamma_{DT}$, we can compute the decoupled envelope $\gamma_{SD} = 60$ and $\gamma_{DT} \in [120, 190]$. These optimization problems can be practically and efficiently solved using the Optimization Modulo Theories (Sebastiani and Tomasi 2015) framework.

## 6 Related Work

In this paper we make two contributions. First we deal with the validation of STN plans and the computation of their robustness envelopes. Our synthesis approach supports parameters in both the plan and the problem. This second case is completely new for the planning literature: to the best of our knowledge no other approach is able to formally compute the space of domain variations (in terms of parameters values) that guarantee plan validity. Instead, broadening the applicability of plans to handle the run-time contingencies is not a new idea, but this paper addresses this issue for a very expressive language (we support the full PDDL 2.1 language with continuous change) and from a formal standpoint.

Plan "flexibility", robustness and their implications in executability of plans have been studied in the context of constraint-based planning and scheduling. The concept of "envelope" has been introduced in (Muscettola 2002), where the author deals with the scheduling of a temporal network with piecewise-constant resources. This work has been extended in (Frank and Morris 2007) to deal with continuous linear resources. Both these works focus on synthesizing a flexible execution given a temporal network subject to constraints on continuous resources. This paper generalizes these ideas allowing to directly reason on the planning domain instead of a generated temporal network. Essentially, we keep into account the planning model for which the STN is a plan in addition to the scheduling constraints. Moreover, these works are limited to purely-temporal flexibility: their focus is on finding when to start and terminate actions in order to respect the resources; here instead we can also reason on the resources themselves by synthesizing the possible initial values or the consumption rates that guarantee the successful execution of a plan.

Policella et al. (2004) propose a technique that is similar in spirit to ours: we both want to generalize the applicability of a given STN plan. However, our synthesis considers the planning problem for which the STN encodes a plan, yielding an extremely non-convex problem due to the presence of predicates and fluents that interfere in the timing decisions. Essentially we are considering all the possible reorderings (Bäckström 1998) of a plan instead of the partial orders considered by Policella et al..

Similarly, Do and Kambhampati (2003) derive "order-constrained" plans from time-triggered temporal plans. Order-constrained plans are a sub-case of STN plans in which only non-metric precedence constraints are allowed between the end of an action and the start of another. The goal of Do and Kambhampati is to allow the reordering of actions during execution, hence augmenting the possible plan executions. In our paper, we focus on widening the temporal applicability for metric (non only order-constrained) STNs: we do not change the causal links in the STN plan, but we allow to reason on continuous resources and complex temporal constraints.

Concerning the validation of STN plans, Cesta et al. (2010) propose a reduction from these plans to timed automata model-checking. Here, we encode the STN validity by exploiting the expressiveness of the SMT framework that we then use for our synthesis objectives. Our encoding is conceptually similar to a Bounded Model Checking (Biere et al. 2003), but differently from Cesta et al. we consider continuous resources and exploit the length and structure of the plan, avoiding the need for full-blown model-checking.

In timeline-based planning, it is customary to generate plans in the form of STNs. Planners such as IxTeT (Ghallab and Laruelle 1994), EUROPA (Frank and Jónsson 2003) or APSI (Cesta et al. 2009) can produce correct-by-construction STN plans. PLATINUM (Umbrico et al. 2018) propose a framework for planning with resources able to generate flexible plans. In this work, we do not focus on the issue of plan generation: we want to perform the synthesis of a robustness envelope (for both time and resources) a-posteriori. (Mayer, Orlandini, and Umbrico 2016) and (Gigante et al. 2017) formalize the validity of flexible plans for timeline planning models; here we address action-based languages, proposing a synthesis technique for computing robustness envelopes for both plan and problem parameters.

Nilsson et al. (Nilsson, Kvarnström, and Doherty 2018) consider temporal uncertainty in their planning model synthesizing plans that are guaranteed to achieve the goal under any possible modeled contingency. We aim at a similar goal taking a radically different path: starting from a plan that is valid for a specific contingent choice, we synthesize the space of contingencies for which it is guaranteed to work. Moreover, we consider the parametrization of the domain that corresponds to synthesizing the contingent resource variations that keep the plan execution valid.

Fox, Howey, and Long deal with the problem of measuring the robustness of plans by statistically computing the maximum disturbance in the timing of activities that maintain the plan valid with a certain probability (Fox, Howey, and Long 2006). The work in (Fritz and McIlraith 2009) elaborates on the same idea, proposing a method to compute the robustness analytically via regression for purely sequential plans; moreover, a technique to generate plans maximizing the robustness is presented. In this paper, we are able to do the same kind of reasoning and beyond: we are not limited to disturbances in the duration of actions and we consider a wider class of plans (STN plans instead of PDDL timed plans). In addition, our reasoning is exact instead of statistical. Finally, our technique allows the generation of the robustness envelope keeping the plan formally valid, so we do not produce a simple measure of the plan like the maximal disturbance, but we can analyze the inter-dependencies of the actions composing the plan.

## 7 Experiments

In this section, we present an experimental evaluation aimed at showing the immediate applicability of the technique: we prove that the validation of STN plans is effectively applicable to various domains taking very reasonable computational resources, and that the robustness envelope synthesis problem, while not scaling to huge plan and domains yet, can be solved by directly employing existing solvers and tools and shows promising results.

All the techniques presented in the paper have been implemented in a tool that takes as input a PDDL2.1 problem and domain together with an STN plan. The tool is implemented in Python, and uses the PySMT library (Gario and Micheli 2015) for SMT solving and quantification. We use the "Virtual Substitution" (Loos and Weispfenning 1993) quantifier elimination technique for LRA provided by the MATHSAT5 (Cimatti et al. 2013) SMT solver to perform the synthesis, and the Z3 (de Moura and Bjørner 2008) solver to perform the validation of STN plans and to solve the optimization problem arising from parameter decoupling. Currently, the tool is limited to LRA formulae, and is thus unable to handle problems in which parameters are multiplied with non-constants. This is not a limitation of the technique itself, but a technical limitation of the tool due to the library being used. The implementation of the tool and all benchmarks are available online[3].

**STN Plan Validation.** As a case-study, we used three domains: Autonomous Underwater Vehicle (AUV) (Buksz et al. 2018), the Solar Rover (Piotrowski et al. 2016), and the linear generator from the PDDL+ benchmarks (Cashmore et al. 2016). Using these domains, we created several problem instances. We varied the size of problem for each domain by varying the number of missions in the AUV domain, the number of required batteries in the Solar Rover domain, and the number of required required refuel actions in the linear generator domain. The total benchmark set consists of 58 problem instances of varying size.

We generated time-triggered plans for each problem instance using the planner SMTPLAN (Cashmore et al. 2016) for the linear generator and POPF (Coles et al. 2010) for the others. STN plans were then generated by relaxing the duration of each action in the time-triggered plan. For each plan, 10 STNs were generated, allowing the duration $d$ of each action to lie between $(d - d * 0.01 * v)$ and $(d + d * 0.01 * v)$ for $v = 0 \cdots 9$. This resulted in 580 STNs, with a duration variation up to $18\%$ of the action duration specified in the original domain model. These STN plans are not necessarily valid for the original domain, unless $v = 0$, in which case the STN represents the original time-triggered plan.

To examine the efficiency of our SMT encoding for STN plan validation, we validated all the STN plans against the problem using our tool. The run-time for validating each STN plan and the validation results are shown in Figure 4. The plot shows that validation time gracefully increases with the size of the problem, and length of the plan. Also, we note that proving invalidity does not take noticeably different time than proving validity.

**Robustness Envelope Synthesis.** For the synthesis problem, we parametrized each domain by expressing constants as problem parameters: the time needed for a full recharge and the speed of the navigation actions in the AUV domain, the required charge for communication in the Solar Rover, and refuel rate in the linear generator domain. In addition, we parametrized the duration of increasing numbers of actions in the STN plans. We applied our encoding to synthesize the robustness envelope for all the domains and plans; we also computed the decoupled robustness envelope (using the the sum of the size of the intervals for each parameter as objective function). Tables 1 and 2 show the run-time for computing the decoupled envelopes as the size of the problem and the number of parameters increase, respectively.

First we report that the decoupling phase took negligible time in all the cases: all the time is consumed in the quantifier elimination needed to synthesize the robustness envelope. The tables show that the length of the plan has a strong impact on the runtime, while the number of parameters is less detrimental. This is due to the "BMC nature" of our en-

[3]http://es.fbk.eu/people/amicheli/resources/aaai2019

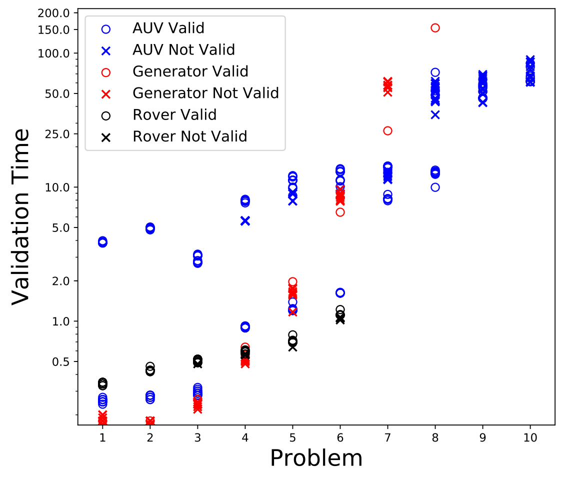

Figure 4: Times to validate plans.

| Problem | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| AUV | 9.8 | 16.4 | 25.6 | 21.7 | 33.9 | 60 |
| Generator | 0.31 | 0.28 | 0.46 | 1.12 | 23.1 | Time Out |
| Solar Rover | 0.75 | 1.03 | 1.39 | 1.64 | 2.25 | 3.45 |

Table 1: Times (sec) taken to synthesis robustness envelopes for increasing problem sizes.

| Problem | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| AUV_#1 | 1.7 | 0.78 | 0.97 | 3.14 | 51.15 | TO |
| AUV_#2 | 2.92 | 1.05 | 1.32 | 7.41 | 94.84 | TO |
| AUV_#3 | 5.1 | 1.2 | 1.82 | 9.87 | 107.17 | TO |
| AUV_#4 | 7.06 | 1.2 | 2.04 | 16.36 | 89.1 | TO |
| Gen_#1 | 11.14 | 59.91 | 542.3 | 6350.3 | TO | TO |
| Gen_#2 | 14.13 | 72.76 | 615.22 | TO | TO | TO |
| Gen_#3 | 375.4 | 422.55 | 1130.43 | TO | TO | TO |
| Gen_#4 | TO | TO | TO | TO | TO | TO |
| Rover_#1 | 1.59 | 2.32 | 3.83 | 5.55 | 5.28 | 8.47 |
| Rover_#2 | 2.69 | 4.52 | 5.14 | 5.62 | 8.32 | 13.02 |
| Rover_#3 | 6.49 | 6.67 | 9.07 | 7.98 | 11.55 | 19.7 |
| Rover_#4 | 8.0 | 32.72 | 22.16 | 12.52 | 67.6 | 29.55 |

Table 2: Times (sec) taken to synthesize robustness envelopes for increasing numbers of parameters, in problems of increasing complexity. For each domain, *#X* means X parameters are considered, TO means time out after 30 mins.

coding that keeps a copy of each variable for each step of the plan, making the formula significantly larger for longer plans, similar behaviors are observed also in SAT/SMT-based planners. The number of parameters is also a source of complexity, but by pushing the quantification and exploiting the simplifications offered by the SMT solvers we can scale gracefully. Interestingly, the behavior of the "Rover_#4" instance shows that there is no direct correlation between number of parameters and the solving times in all the cases: if the robustness envelope turns out to be simpler or the search inside the quantifier elimination algorithm quickly finds good models, synthesizing regions in for a higher number of dimensions could take lees time.

## 8 Conclusion and Future Work

This paper addresses two important problems concerning STN plans in planning domains with durative actions and continuous change. First, we propose a formal approach to check the validity of a given STN plan and to compute its robustness envelope. Second, we generalize this approach to computing the robustness envelope for a planning problem for which a (possibly parametric) STN plan remains valid.

In this framework, all the plan executions are analyzed not only with respect to the domain used at planning time, but also with respect to variants of the domain. This allows a formal impact analysis of variations in the domain (e.g. resource consumptions, action durations). The approach exploits the expressiveness of the SMT framework, and is made practical by the effectiveness of SMT solvers.

This paper is a first step towards the adoption of formal techniques to generalize and study the applicability of plans and to analyze the planning problem specifications. In the future, we will investigate how to adapt SMT-based techniques to increase the scalability and to support more expressive (e.g. non-linear) dynamics. Finally, we plan to study the theoretical connections between our techniques and the works in non-deterministic temporal planning, such as those concerning temporal controllability.

## References

Bäckström, C. 1998. Computational aspects of reordering plans. *Journal of Artificial Intelligence Research* 9:99–137.

Barrett, C. W.; Sebastiani, R.; Seshia, S. A.; and Tinelli, C. 2009. Satisfiability modulo theories. In *Handbook of Satisfiability*. IOS Press. 825–885.

Biere, A.; Cimatti, A.; Clarke, E. M.; Strichman, O.; and Zhu, Y. 2003. Bounded model checking. *Advances in Computers* 58:117–148.

Buksz, D.; Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; and Ridder, B. 2018. Strategic-tactical planning for autonomous underwater vehicles over long horizons. In *IROS*.

Cashmore, M.; Fox, M.; Long, D.; and Magazzeni, D. 2016. A compilation of the full PDDL+ language into SMT. In *ICAPS*.

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

Cesta, A.; Finzi, A.; Fratini, S.; Orlandini, A.; and Tronci, E. 2010. Validation and verification issues in a timeline-based planning system. *Knowledge Engineering Review* 25(3):299–318.

Cimatti, A.; Griggio, A.; Schaafsma, B. J.; and Sebastiani, R. 2013. The MathSAT5 SMT solver. In *TACAS*, 93–107.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *ICAPS*, 42–49.

de Moura, L., and Bjørner, N. 2008. Z3: An efficient SMT solver. In *TACAS*, 337–340.

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

Do, M. B., and Kambhampati, S. 2003. Improving temporal flexibility of position constrained metric temporal plans. In *ICAPS*, 42–51.

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.

Fox, M.; Howey, R.; and Long, D. 2006. Exploration of the robustness of plans. In *AAAI*, 834–839.

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

Frank, J., and Morris, P. H. 2007. Bounding the resource availability of activities with linear resource impact. In *ICAPS*, 136–143.

Fritz, C., and McIlraith, S. A. 2009. Computing robust plans in continuous domains. In *ICAPS*.

Gario, M., and Micheli, A. 2015. pySMT: a solver-agnostic library for fast prototyping of SMT-based algorithms. In *SAT - SMT Workshop*.

Ghallab, M., and Laruelle, H. 1994. Representation and control in IxTeT, a temporal planner. In *AIPS*, 61–67.

Gigante, N.; Montanari, A.; Mayer, M. C.; and Orlandini, A. 2017. Complexity of timeline-based planning. In *ICAPS*, 116–124.

Kautz, H. A., and Selman, B. 1992. Planning as satisfiability. In *ECAI*, 359–363.

Kim, H.; Somenzi, F.; and Jin, H. 2009. Efficient Term-ITE conversion for satisfiability modulo theories. In *SAT*, 195–208.

Loos, R., and Weispfenning, V. 1993. Applying linear quantifier elimination. *Computer Journal* 36(5):450–462.

Mayer, M. C.; Orlandini, A.; and Umbrico, A. 2016. Planning and execution with flexible timelines: a formal account. *Acta Informatica* 53(6-8):649–680.

Monniaux, D. 2008. A quantifier elimination algorithm for linear real arithmetic. In *LPAR*, 243–257.

Muscettola, N.; Morris, P. H.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *KR*, 444–452.

Muscettola, N. 2002. Computing the envelope for stepwise-constant resource allocations. In *CP*, 139–154.

Nilsson, M.; Kvarnström, J.; and Doherty, P. 2018. Planning with temporal uncertainty, resources and non-linear control parameters. In *ICAPS*, 180–189.

Piotrowski, W. M.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic planning for PDDL+ domains. In *IJCAI*, 3213–3219.

Policella, N.; Smith, S. F.; Cesta, A.; and Oddi, A. 2004. Generating robust schedules through temporal flexibility. In *ICAPS*, 209–218.

Sebastiani, R., and Tomasi, S. 2015. Optimization modulo theories with linear rational costs. *ACM Transactions on Computational Logic* 16(2):12:1–12:43.

Umbrico, A.; Cesta, A.; Mayer, M. C.; and Orlandini, A. 2018. Integrating resource management and timeline-based planning. In *ICAPS*, 264–272.