Noname manuscript No.
(will be inserted by the editor)

Solving Strong Controllability of Temporal Problems with Uncertainty using SMT*

Alessandro Cimatti · Andrea Micheli · Marco Roveri

the date of receipt and acceptance should be inserted later

Abstract Temporal Problems (TPs) represent constraints over the timing of activities, as arising in many applications such as scheduling and temporal planning. A TP with uncertainty (TPU) is characterized by activities with uncontrollable duration. Different classes of TPU are possible, depending on the Boolean structure of the constraints: we have simple (STPU), constraint satisfaction (TCSPU), and disjunctive (DTPU) temporal problems with uncertainty.

In this paper we tackle the problem of strong controllability, i.e. finding an assignment to all the controllable time points, such that the constraints are fulfilled under any possible assignment of uncontrollable time points.

Our approach casts the problem in the framework of Satisfiability Modulo Theory (SMT), where the uncertainty of durations can be modeled by means of universal quantifiers. The use of quantifier elimination techniques leads to quantifier-free encodings, which are in turn solved with efficient SMT solvers.

We obtain the first practical and comprehensive solution for strong controllability. We provide a family of efficient encodings, that are able to exploit the specific structure of the problem. The approach has been implemented, and experimentally evaluated over a large set of benchmarks. The results clearly demonstrate that the proposed approach is feasible, and outperforms the best state-of-the-art competitors, when available.

* This is an extended version of the paper [10] presented at the 18th International Conference on Principles and Practice of Constraint Programming (CP 2012) in Quebec City, Canada. The extensions with respect to the previous version are explained in detail in the cover letter of this submission.

Alessandro Cimatti
E-mail: cimatti@fbk.eu
Fondazione Bruno Kessler — IRST, Via Sommarive 18, 38123 Povo (TN) Italy

Andrea Micheli
E-mail: amicheli@fbk.eu
Fondazione Bruno Kessler — IRST, Via Sommarive 18, 38123 Povo (TN) Italy

Marco Roveri
E-mail: roveri@fbk.eu
Fondazione Bruno Kessler — IRST, Via Sommarive 18, 38123 Povo (TN) Italy

1 Introduction

Many applications require the scheduling of a set of activities over time, subject to constraints of various nature. This is often expressed as a *Temporal Problem* (TP), where each activity is associated with two time points, representing the start time and the end time, subject to constraints. Several kinds of temporal problems have been identified, depending on the nature and structure of the constraints [14]. If the problem is expressible as a pure conjunction of constraints over distances of time points, then we have the so-called *Simple Temporal Problem* (STP). A more complex class is *Temporal Constraint Satisfaction Problem* (TCSP), where the temporal distance between two time points can be constrained to lie in the union of disjoint intervals. When arbitrary Boolean combinations are allowed, we have a *Disjunctive Temporal Problem* (DTP). A temporal problem is said to be *consistent* if there exists an assignment for the time points, such that all the constraints are satisfied [14]. Such an assignment is called a *schedule*, and it corresponds to sequential time-triggered programs, that are often used in control of satellites and rovers.

In many practical cases, however, the durations of some activities are uncontrollable. TPs are thus extended with *uncertainty* in the duration of activities, thus obtaining the classes of STPU, TCSPU and DTPU [36]. Given a temporal problem with uncertainty (TPU), three different problems can be addressed, namely weak, dynamic and strong controllability [36]. *Weak controllability* concerns the existence of a strategy that schedules each activity, as a function of all the uncontrollable durations. The executor is assumed to know the duration of the uncontrollable activities before the execution starts (this property is sometimes known as "clairvoyance"). In *dynamic controllability*, a solution is a strategy, similarly to weak controllability, but each decision is constrained to depend on past events only. In *strong controllability*, we disallow any runtime observation, and we require a fixed schedule for the activities that is independent of the uncertainty. As in the case of consistency, we look for a schedule. However, the schedule only determines the start of all the activities, and the end of the activities that are controllable, and must satisfy the constraints *for all* the durations of the uncontrollable activities. If such a schedule exists, the problem is said to be *strongly controllable* [36].

Strong controllability is an important problem, because it results in a schedule that is satisfactory under all possible uncertainties. Clearly, a strong schedule can yield a longer timespan compared to a dynamic strategy. However, dynamic information may not be available, e.g. due to the lack of sensors. Furthermore, most algorithms for dynamic execution require run-time reasoning [19]. This may be incompatible with some operational settings: for example, in mission-critical systems, validating the run-time reasoner to the required level of assurance may be prohibitively hard. Furthermore, the computational resources available during execution may be too limited for a dynamic approach. Examples of such application domains can be found in production planning and in mission critical robotics, for which strong controllability is a very relevant problem. We also remark that, in the same domains, the expressivity of disjunctive constraints (compared to simple temporal problems) is often necessary [27].

In this paper, we propose a comprehensive and effective approach for solving the strong controllability problem for TPUs in the most general form including arbitrary disjunctions. The approach relies on the Satisfiability Modulo Theory

(SMT) framework [6]. This framework provides representational and reasoning capabilities within decidable fragments of first order logic, where interpretations are constrained to satisfy a specific theory of interest (e.g. linear real arithmetic). Modern SMT solvers are a tight integration of a Boolean SAT solver, that is highly optimized for the *case split* required by the Boolean combination of constraints, with dedicated constraint solvers for the theories of interest. Some SMT solvers provide embedded primitives to handle *quantifiers*, and dedicated techniques for quantifier-elimination are available for some theories of interest. Several effective SMT solvers are available (e.g. MathSAT [7,9], Z3 [26], Yices [15], OpenSMT [8]). SMT solving has had increasing applications in many areas including Answer Set Programming (ASP) [28], Formal Verification [17], and Test Case Generation [18].

We tackle the strong controllability problem of TPUs by reduction to SMT problems, that are then fed into efficient SMT solvers. First, we show how to encode a TPU into the theory of quantified linear real arithmetic (LRA) and, by leveraging the specific nature of the problem, we optimize the encoding by reducing the scope of quantifiers. The resulting formula can be solved by any SMT solver for (quantified) LRA. Second, we present a general reduction procedure from strong controllability to consistency, based on the eager application of quantifier elimination techniques. The resulting formulae can be directly fed into any SMT solver for the quantifier-free LRA. This gives the first general comprehensive solver for strong controllability of TPUs. Third, we generalize the results by Vidal and Fargier [36], originally stated for STPU, to the subclass of (simple-natured) TCSPU. In this way, we avoid the use of expensive general purpose quantifier elimination techniques, with significant performance improvements.

The proposed approach has been implemented in a solver based on state-of-the-art SMT techniques. To the best of our knowledge, this is the first implemented solver for strong controllability of TPUs. We carried out a thorough experimental evaluation, over a large set of benchmarks. We analyze the merits of the various encodings, and demonstrate the overall feasibility of the approach. We also compare the proposed approaches with state-of-the-art algorithms on consistency problems. SMT solvers turned out to be competitive with, and often outperform, the best known dedicated solving techniques. Finally, we compared our approach with the only other algorithm to check strong controllability for DTPU [29]. The results show that the symbolic techniques proposed in this paper can dramatically outperform the enumerative approach in [29].

Structure of the paper. In sections 2 and 3 we present some technical preliminaries and background about SMT. In section 4 we formally define temporal problems, while in section 5 we present several SMT encodings for consistency. Encodings for strong controllability are described in section 6 and an overview of the related work is given in section 7. We report the results of the performed experimental evaluation in section 8, and in section 9 we draw some conclusions and outline directions for future work.

2 Technical Preliminaries

Our setting is standard first order logic [21]. The first-order signature is composed of constants, variables, function symbols, Boolean variables, and predicate symbols. A term is either a constant, a variable, or the application of a function

symbol of arity n to n terms. A theory constraint (also called a theory atom) is the application of a predicate symbol of arity n to n terms. An atom is either a theory constraint or a Boolean variable. A literal is either an atom or its negation. A clause is a finite disjunction of literals. A formula is either true ($\top$), false ($\bot$), a Boolean variable, a theory constraint, the application of a propositional connective ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) of arity n to n formulae, or the application of a quantifier ($\forall$, $\exists$) to an individual variable and a formula. We use $x, y, v, \ldots$ for variables, and $\vec{x}, \vec{y}, \vec{v}, \ldots$ for vectors of individual variables. Terms and formulae are referred to as expressions. Formulae are denoted with $\phi, \psi, \ldots$. Let $\vec{x}$ be a vector of variables, we indicate the i-th variable in the vector with x_i. We write $\phi(x)$ to highlight the fact that x occurs in ϕ, and $\phi(\vec{x})$ to highlight the fact that the free variables of ϕ are variables in $\vec{x}$. We indicate with $Q\vec{x}.\phi(\vec{x})$ the formula $Qx_1.Qx_2.\ldots.Qx_n.\phi(x_1, \ldots, x_n)$, where $Q \in \{\forall, \exists\}$.

Let $\phi(\vec{x}) \doteq \bigwedge_i \phi_i(\vec{x}_i)$ be a conjunction of formulae. We write $\phi(\vec{x})|_{\vec{y}}$ to represent the conjunction of the $\phi_i(\vec{x}_i)$ in which at least one variable of $\vec{y}$ occurs in $\vec{x}_i$.

Substitution is defined in the standard way [21]. We write $\phi[s/v]$ for the substitution of every occurrence of variable v in ϕ with term s. Let $\vec{v}$ be a vector of variables and $\vec{s}$ be a vector of terms, we write $\phi[\vec{s}/\vec{v}]$ for the parallel substitution of every occurrence of v_i in ϕ with s_i.

We use the standard semantic notions of interpretation and satisfiability. We call *satisfying assignment* or *model* of a formula $\phi(\vec{x})$ a total function μ that assigns to each x_i an element of its domain such that the formula $\phi[\mu(\vec{x})/\vec{x}]$ evaluates to $\top$. A formula $\phi(\vec{x})$ is *satisfiable* if and only if it has a satisfying assignment.

The satisfiability of a formula is the problem of finding a satisfying assignment for the formula. This problem is approached in propositional logic with enhancements of the DPLL algorithm [13]: the formula is converted into an equi-satisfiable one in Conjunctive Normal Form (CNF); then, a satisfying assignment is incrementally built, until either all the clauses are satisfied, or a conflict is found, in which case back-jumping takes place (i.e. certain assignments are undone). Keys to efficiency are heuristics for the variable selection, and learning of conflicts [25].

3 Satisfiability Modulo Theories

Given a first-order formula ψ in a decidable background theory T, *Satisfiability Modulo Theory* (SMT) [6] is the problem of deciding whether there exists a satisfying interpretation that satisfies ψ. For example, consider the formula $(x \leq y) \wedge ((x + 3 = z) \vee (z \geq y))$ in the theory of real arithmetic ($x, y, z \in \mathbb{R}$, and the symbols $\leq$, $+$, $=$ and $\geq$ are interpreted in the usual way). The formula is satisfiable and a satisfying assignment is $\{x := 5,\ y := 6,\ z := 8\}$. The theory of real arithmetic interprets 3 as a real number and $+, =, <, >, \leq, \geq$ as the usual mathematical functions and relations.

In this work we concentrate on the theory of linear arithmetic over the real numbers (LRA). A formula in LRA is an arbitrary Boolean combination, a universal ($\forall$) or an existential ($\exists$) quantification, of atoms in the form $\sum_i a_i x_i \bowtie c$ where $\bowtie \in \{>, <, \leq, \geq, \neq, =\}$, every x_i is a real variable and every a_i and c is a real constant. Given two real constants l, u such that $l \leq u$, we denote with $t \in [l, u]$ the formula $l \leq t \wedge t \leq u$. Difference logic (RDL) is the subset of LRA such that

atoms have the form $x_i - x_j \bowtie c$. We denote with QF_LRA and QF_RDL the quantifier-free fragments of LRA and RDL, respectively.

An SMT solver [6] is a decision procedure which solves the satisfiability problem for a formula expressed in a decidable subset of First-Order Logic. The most efficient implementations of SMT solvers use the so-called "lazy approach", where a SAT solver is tightly integrated with a T-solver, that is demanded to decide conjunction of constraints in the theory T. The role of the SAT solver is to enumerate the truth assignments to the Boolean abstraction of the first-order formula. The Boolean abstraction has the same Boolean structure of the first-order formula, but "replaces" the predicates which contain T information with fresh Boolean variables. The Boolean abstraction of $(x \leq y) \wedge ((x+3=z) \vee (z \geq y))$ is $a \wedge (b \vee c)$, where a, b, c are fresh Boolean variables. The T-solver is invoked when the SAT solver finds a satisfying assignment for the Boolean abstraction: the satisfying assignment to Boolean abstraction maps directly to a conjunction of T atoms, which the T-solver can handle. If the conjunction is satisfiable also the original formula is satisfiable. Otherwise the T-solver returns a conflict set which identifies a reason for the unsatisfiability. Then, the negation of the conflict set is learned by the SAT solver in order to prune the search. Examples of solvers based on the "lazy approach" are MathSAT [7,9], Z3 [26], Yices [15] and OpenSMT [8]).

In order to deal with quantifiers in LRA many techniques have been developed and implemented in SMT solvers. Some solvers (e.g. Z3 [26]) natively support quantifiers. However, many SMT solvers cannot deal with them. Several techniques have been developed for removing quantifiers from an LRA formula (e.g. Fourier-Motzkin [32], Loos-Weispfenning [22,23]): they transform an LRA formula into an *equivalent* QF_LRA formula. [1]

Fourier-Motzkin elimination In order to decide the satisfiability of a conjunction of constraints in linear real arithmetic, we can apply the so-called *Fourier-Motzkin elimination* (FME) technique. The method was discovered in 1826 by Fourier and re-discovered by Motzkin in 1936.

Consider the following formula where $\{x_1, ..., x_n\}$ is a set of real variables ($x_i \in \mathbb{R}$) and $a_{i,j}, b_i \in \mathbb{R}$ are real coefficients.

$$\exists x_e . \bigwedge_{i=1}^{m} (\sum_{j=1}^{n} a_{i,j} x_j) \leq b_i$$

The elimination method allows the elimination of the variable x_e and obtain a new system without x_e that is equi-satisfiable with respect to the original one.

The basic principle of the method consists in the projection of the polytope in the x_e dimension. A description of the method is beyond the scope of this paper; the interested reader can consult Schrijver's thorough description [32] and Kessler's efficient implementation [20].

If we are given a general formula $\exists x_e . \phi(\vec{X})$, we can naïvely apply this technique by computing the Disjunctive Normal Form of $\phi(\vec{X})$ and apply the Fourier-Motzkin technique on each disjunct separately, because the existential quantification distributes over the $\vee$. More advanced approaches are also possible [23].

[1] A theory T is said to admit quantifier elimination, if for every quantified formula ϕ in T, there exist a quantifier-free formula ϕ' that is logically equivalent to ϕ. It has been proved that LRA admits quantifier elimination [32].

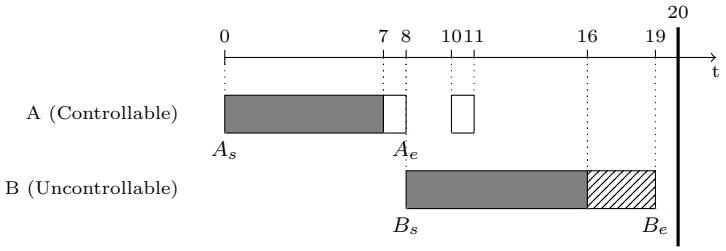

Fig. 1 Schema of a possible temporal situation in the running example. Activities are depicted in time, filled regions are used to indicate the minimal guaranteed duration of an activity, the region in which uncontrollable event B_e can happen is striped, while the region in which A_e can be scheduled is the union of the two white rectangles. The problem deadline is indicated with the solid line at time 20.

Loos-Weispfenning elimination Another approach for quantifier elimination in LRA is the *Loos-Weispfenning* technique [22], named after Rüdiger Loos and Volker Weispfenning. The technique solves the same problem of FME but is based on a completely different mechanism. The idea behind *Loos and Weispfenning* elimination (LWE) approach is that an existentially quantied formula

$$\exists x.\phi(x, \vec{y})$$

with free variables $\vec{y} = y_1, ..., y_n$ can be replaced by a formula $\psi(\vec{y})$

$$\psi(\vec{y}) = \phi(\bar{x}_1, \vec{y}) \vee ... \vee \phi(\bar{x}_m, \vec{y})$$

where $\bar{x}_1, ..., \bar{x}_m$ are expressed as functions of $\vec{y}$. In the case of Loos-Weispfenning elimination the number of produced disjuncts is linear, but the overall complexity bound is still doubly exponential in the number of disjuncts of $\psi(\vec{y})$.

4 Temporal Problems

> *Running Example.* Suppose we have two activities A and B. Activity A has duration of at least 7 units and at most 8 units or at least 10 units and at most 11 units, depending on a controllable decision. Activity B is uncontrollable, meaning that the actual duration is not decidable by the solver, but we can assume that it is at least 8 units and at most 11 units. We require that activity B must start after activity A and both activities must end within 20 units. The situation is depicted in Figure 1.

A Temporal Problem (TP) is a formalism that is used to represent temporal constraints over time-valued variables representing time points. This formalism is rich enough to express Allen's interval algebra [1] and also quantitative constraints over intervals and time points. Two families of TPs have been presented in literature over the years: TP without uncertainty, in which all the time points can be freely assigned by the solver [14,34] and TP with uncertainty (TPU), in which only some of the time points can be assigned by the solver, while the others are intended to be assigned by an adversary [36,29]. As such, TPUs can be seen as a form of game between the solver and an adversarial environment.

Definition 1 A TPU is a tuple (X_c, X_u, C_c, C_f), where $X_c \doteq \{b_1, ..., b_n\}$ is the set of *controllable time points*, $X_u \doteq \{e_1, ..., e_m\}$ $(n \geq m)$ is the set of *uncontrollable time points*, $C_c \doteq \{cc_1, ..., cc_m\}$ is the set of *contingent constraints*, and $C_f \doteq \{cf_1, ..., cf_h\}$ is the set of *free constraints*.

$$cc_i \doteq \bigvee_{j=1}^{E_i}(e_i - b_i) \in [l_{i,j}, u_{i,j}] \qquad cf_i \doteq \bigvee_{j=1}^{D_i}(x_{i,j} - y_{i,j}) \in [l_{i,j}, u_{i,j}]$$

such that: $l_{i,j}, u_{i,j} \in \mathbb{R} \cup \{+\infty, -\infty\}$, $l_{i,j} \leq u_{i,j}$, D_i is the number of disjuncts for the i-th free constraint, E_i is the number of disjuncts for the i-th contingent constraint, $x_{i,j}, y_{i,j} \in X_c \cup X_u$, $x_{i,j} \neq y_{i,j}$,

Intuitively, time points belonging to X_c are time decisions that can be controlled by the solver, while time points in X_u are under the control of the environment. A similar subdivision is imposed on the constraints: free constraints C_f are constraints that the solver is required to fulfill, while contingent constraints (C_c) are the assumptions that the environment will fulfill. As in previous work [36, 29], we consider only contingent constraints that start with a controllable time point. Thus, each uncontrollable time point e_i is linked by exactly one contingent constraint to a controllable time point b_i[2].

Within the framework of TPU, we can express only uncertainty on the duration of activities (and not, for example, uncertainty on whether an activity could occur or not, or on its discrete outcome). Contingent constraints represent the difference in the durations of the uncontrollable activities, while uncontrollable time points represent the uncontrollable ending time of activities. We adopt a continuous model of time, and we explicitly avoid any discretization.

A temporal problem is defined over a set of time points, namely variables representing time instants. A temporal situation such as the one in our running example can be encoded in a temporal problem by using two time points to represent the starting and ending time of each activity. Therefore, in order to model the running example with a temporal problem we need a total of four time points A_s, A_e, B_s and B_e representing the start and end of activity A and B respectively (Figure 1). B_e is the only uncontrollable time point, as we can control the starting and end time of A but we cannot control the duration of B. The only contingent constraint is the constraint on the duration of B. The rest of the problem is composed of requirements that have to be fulfilled in any possible situation, and can be translated in three free constraints. The resulting temporal problem is then (X_c, X_u, C_c, C_f) where $X_c = \{A_s, A_e, B_s\}$, $X_u = \{B_e\}$, $C_c = \{B_e - B_s \in [8, 11]\}$ and $C_f = \{B_e - A_s \in [0, 20], B_s - A_e \in [0, \infty], A_e - A_s \in [7, 8] \vee A_e - A_s \in [10, 11]\}$. Figure 2 shows a graphical visualization of the resulting TPU.

A *TP without uncertainty* is a TPU $(X_c, \emptyset, \emptyset, C_f)$, in which the set of uncontrollable time points is empty (from which it also follows that the set of contingent constraints is empty).

Depending on the generality of the constraints in C_c and C_f, three classes of TPUs are possible [29]. Definition 1 in its general form identifies *Disjunctive Temporal Problem with Uncertainty* (DTPU) [29]. If each constraint contains at most two time points, the resulting problem is a *Temporal Constraint Satisfaction Problem*

[2] This formulation assumes a complete independence between contingent constraints. This means that this formalism cannot express assumptions of interdependence between uncontrollable durations.

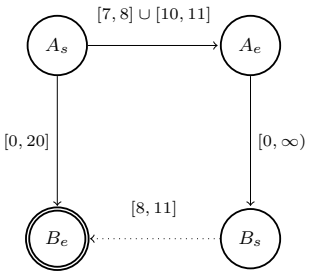

Fig. 2 Graphical representation of the TPU model derived from the running example description. Each node of the graph is a time point, doubly circled nodes are uncontrollable the others are controllable, solid arrows are free constraints and dashed arrows are contingent constraints.

with Uncertainty (TCSPU). If each constraint has exactly one disjunct (i.e. $D_i = 1$ for all i), we obtain a *Simple Temporal Problem with Uncertainty* (STPU). Similarly, we can define the corresponding TP without uncertainty (DTP [34], TCSP, and STP [14]). Following the classification of Peintner et al. [29], we also say that a problem is *simple-natured* if the contingent constraints have no disjunctions ($E_i = 1$ for each i).

We define an assignment to the time points as a total function from time points to real values. Given a TP without uncertainty, checking *consistency* corresponds to deciding the existence of an assignment that fulfills all the constraints of the problem. We call such an assignment a *consistent schedule*, and we say that the TP is *consistent*. Checking the consistency of a TPU (X_c, X_u, C_c, C_f) is defined as checking the consistency of the TP without uncertainty $(X_c \cup X_u, \emptyset, \emptyset, C_c \cup C_f)$. The running example is consistent, and a consistent schedule is for example $\mu = \{A_s = 0, A_e = 11, B_s = 11, B_e = 20\}$. Intuitively, when checking consistency of a TPU, the behavior of the environment is assumed to be "cooperative" with the solver. In this paper, we focus on Strong Controllability (SC) for a TPU, where the environment is adversarial [36]. SC consists in deciding the existence of an assignment to controllable time points that fulfills the free constraints under any assignment of uncontrollable time points that satisfies the contingent constraints. Such an assignment is called a *strong schedule* of the problem. A TPU for which there exists a strong schedule is said to be *strongly controllable*. Consider again the running example, the problem is strongly controllable and a strong schedule is $\mu' = \{A_s = 0, A_e = 8, B_s = 8\}$.

If a TPU is strongly controllable, it is also consistent [36]. However, the converse does not hold in general. Consider for example a variation to the running example in which the deadline is moved from 20 time units to 17 time units. The problem is consistent because a consistent schedule is $\{A_s = 0, A_e = 7.5, B_s = 8, B_e = 16\}$. However, this version of the problem is not strongly controllable because the activity B cannot be started before 7 (that is the minimal duration of activity A that must precede B): since B is uncontrollable with duration in $[8, 11]$, it may be the case that the execution of the activity takes more than 10 time units, thus exceeding the deadline.

5 Encoding of Consistency Problems in SMT

We first focus on the consistency problem, i.e. the case in which there is no uncontrollability. Checking the consistency of a temporal problem amounts to checking whether the conjunction of the constraints admits a model. Therefore, the consistency problem can be reduced to checking the satisfiability of a quantifier-free formula modulo the LRA theory. The temporal problem is consistent if and only if the corresponding SMT formula is satisfiable, and any satisfying assignment for the formula corresponds to a consistent schedule for the problem.

In this work, consistency checking plays the role of backend for strong controllability. We present several SMT encodings, that turn out to have different performance in the solvers, depending on the nature of the constraints. We exploit the characteristics of such encodings to improve the performances of the approach we propose to solve the strong controllability problem.

In the following, we assume that a TP $P = (X_c, \emptyset, \emptyset, C_f)$ is given. The first encoding in SMT of the consistency problem can be directly obtained as follows: for every time point in X_c we introduce a real variable, and we denote with $\vec{X_c}$ the vector of such variables; each constraint in C_f is directly mapped on the corresponding SMT formula; the encoding is the SMT formula shown in Equation 1.

$$\bigwedge_{i=1}^{|C_f|} \bigvee_{j=1}^{D_i} (((x_{i,j} - y_{i,j}) \geq l_{i,j}) \wedge ((x_{i,j} - y_{i,j}) \leq u_{i,j})) \tag{1}$$

Proposition 1 *The temporal problem P is consistent if and only if Equation 1 is satisfiable (and a model of the formula yields a strong schedule for P).*

Proposition 1 is justified by the fact that Equation 1 is the conjunction of the formalization of the problem constraints, and by definition, checking consistency amounts to checking the existence of a model of the constraints. This formalization is such that a consistent schedule can be extracted from a model of the encoding formula by interpreting the value assigned to each SMT variable as a time value of the corresponding time point.
Equation 1 is already a working SMT encoding. It is linear in the size of the original TP, but does not exploit any knowledge on the structure of the problem, and is thus referred to as *naïve encoding*. In particular, we notice that the resulting SMT formula is not in CNF. [3]

In the running example this encoding amounts to checking the satisfiability of the conjunction of all the constraints as follows.

$$\begin{aligned} &(B_e - B_s \geq 8) \wedge (B_e - B_s \leq 11) \wedge \\ &(B_e - A_s \geq 0) \wedge (B_e - A_s \leq 20) \wedge \\ &(B_s - A_e \geq 0) \wedge \\ &((A_e - A_s \geq 7) \wedge (A_e - A_s \leq 8)) \vee ((A_e - A_s \geq 10) \wedge (A_e - A_s \leq 11)) \end{aligned} \tag{2}$$

In the rest of this section we introduce three optimizations: the switch encoding (applicable to any TP), the switch encoding with mutual exclusion and the hole encoding (both for TCSPs only).

[3] Since most efficient SMT solvers work by combining a SAT and a T-solver, a CNF formulation of the problem is an advantage that prevents the solver for computing a (possibly less efficient) CNF by itself.

5.1 Switch Encoding

The *switch encoding* performs a CNF conversion of the formula in Equation 1 by means of a polarity-based CNF labeling conversion [33]. To this extent, we introduce $\sum_{i=0}^{|C_f|} D_i$ Boolean "switch" variables $s_{i,j}$, and the resulting encoding is the one in Equation 3.

$$\begin{aligned}\bigwedge_{i=1}^{|C_f|}((\bigwedge_{j=1}^{D_i}((\neg s_{i,j} \vee ((x_{i,j} - y_{i,j}) \geq l_{i,j})) \wedge \\ (\neg s_{i,j} \vee ((x_{i,j} - y_{i,j}) \leq u_{i,j})))) \wedge (\bigvee_{j=1}^{D_i} s_{i,j}))\end{aligned} \tag{3}$$

Theorem 1 *The temporal problem P is consistent if and only if Equation 3 is satisfiable (and a consistent schedule can be derived from any model of Equation 3).*

Proof. We prove that Equation 3 is equi-satisfiable to Equation 1 and that the model of Equation 3 is always an extension of a model of Equation 1 from which a consistent schedule can be extracted.

First we prove that a model μ of Equation 1 can be extended to a model μ' of Equation 3. For each i, there exists $\bar{\jmath}$ such that $((x_{i,\bar{\jmath}} - y_{i,\bar{\jmath}}) \geq l_{i,\bar{\jmath}}) \wedge ((x_{i,\bar{\jmath}} - y_{i,\bar{\jmath}}) \leq u_{i,\bar{\jmath}})$. Let $\mu' \doteq \mu \cup \{s_{i,\bar{\jmath}} = \top | \forall i\} \cup \{s_{i,j} = \bot | \forall i, \forall j : j \neq \bar{\jmath}\}$. μ' is a model of Equation 3 because (1) $\bigvee_{j=1}^{D_i} s_{i,j}$ is satisfied by $s_{i,\bar{\jmath}}$, and (2) each conjunct is satisfied because in μ', $s_{i,j}$ is false for all $j \neq \bar{\jmath}$ and thus the i-j-conjunct is satisfied and the i-$\bar{\jmath}$-conjunct is such that $x_{i,\bar{\jmath}}$ and $y_{i,\bar{\jmath}}$ fulfill $((x_{i,\bar{\jmath}} - y_{i,\bar{\jmath}}) \geq l_{i,\bar{\jmath}}) \wedge ((x_{i,\bar{\jmath}} - y_{i,\bar{\jmath}}) \leq u_{i,\bar{\jmath}})$.

We now prove that a model μ' of Equation 3 can be reduced to a model μ of Equation 1. Let μ be the restriction of μ' to the $x_{i,j}$ and $y_{i,j}$ variables only. μ' fulfills $\bigvee_{j=1}^{D_i} s_{i,j}$, therefore there is a $\bar{\jmath}$ such that $s_{i,\bar{\jmath}}$ is true in μ'. $x_{i,\bar{\jmath}}$ and $y_{i,\bar{\jmath}}$ are such that $((x_{i,\bar{\jmath}} - y_{i,\bar{\jmath}}) \geq l_{i,\bar{\jmath}}) \wedge ((x_{i,\bar{\jmath}} - y_{i,\bar{\jmath}}) \leq u_{i,\bar{\jmath}})$, therefore, also Equation 1 is satisfied. □

This encoding is also linear in the size of the original TP problem, and it directly produces a CNF formula. We notice that the clauses involving theory atoms are binary; furthermore, if a switch variable is assigned to false, the corresponding clauses are satisfied without any theory reasoning. These factors have a positive impact on the performance of the SMT solver.

In the running example this encoding is as follows.

$$\begin{aligned}&(B_e - B_s \geq 8) \wedge (B_e - B_s \leq 11) \wedge \\ &(B_e - A_s \geq 0) \wedge (B_e - A_s \leq 20) \wedge \\ &(B_s - A_e \geq 0) \wedge \\ &(\neg s_1 \vee (A_e - A_s \geq 7)) \wedge \\ &(\neg s_1 \vee (A_e - A_s \leq 8)) \wedge \\ &(\neg s_2 \vee (A_e - A_s \geq 10)) \wedge \\ &(\neg s_2 \vee (A_e - A_s \leq 11)) \wedge \\ &(s_1 \vee s_2)\end{aligned} \tag{4}$$

5.2 Switch Encoding With Mutual Exclusion

If we focus on the TCSP class, we can exploit the problem structure to further improve our encodings. In particular, we assume that the disjuncts in each con-

straint are mutually exclusive, otherwise two or more disjuncts can be merged together by simply taking the union of the intervals they represent. For example, a constraint $(a-b \in [10,20]) \vee (a-b \in [30,35]) \vee (a-b \in [15,25])$ can be simplified to $(a-b \in [10,25]) \vee (a-b \in [30,35])$ by merging the first and the last disjuncts. Formally, this means that each TCSP constraint is composed of disjuncts of the form $x - y \in [l_j, u_j]$, where x and y are time points, and for all j, $l_j \leq u_j$ and $u_j < l_{j+1}$.

If we use the previous encodings, it is left to the solver (in particular to the theory solver) to discover this mutual exclusion property. We can strengthen the switch encoding by statically adding mutual exclusion constraints of the form $(\neg s_h \vee \neg s_k)$, with $h \neq k$. Adding this information to the encoding is a form of static learning, and it can guide the Boolean search by pruning branches that are unsatisfiable in the theory. The *switch encoding with mutual exclusion* is presented in Equation 5.

$$\begin{aligned} \bigwedge_{i=1}^{|C_f|} (\bigwedge_{j=1}^{D_i} ((\neg s_{i,j} \vee ((x_{i,j} - y_{i,j}) \geq l_{i,j})) \wedge \\ (\neg s_{i,j} \vee ((x_{i,j} - y_{i,j}) \leq u_{i,j}))) \wedge \\ (\bigvee_{j=1}^{D_i} s_{i,j}) \wedge (\bigwedge_{j=1}^{D_i} \bigwedge_{k=j+1}^{D_i} (\neg s_{i,j} \vee \neg s_{i,k}))) \end{aligned} \tag{5}$$

Theorem 2 *If P is a TCSP, P is consistent if and only if Equation 5 is satisfiable (and a model of Equation 5 yields a consistent schedule).*

Proof. We prove that for any TCSP, Equation 5 is logically equivalent to Equation 3.

We highlight that Equation 5 is analogous to Equation 3, but it adds a new constraint in the form $\bigwedge_{j=1}^{D_i} \bigwedge_{k=j+1}^{D_i} (\neg s_{i,j} \vee \neg s_{i,k})$.

Let μ be a model of Equation 3. Since the problem is a TCSP we know that in each constraint the intervals are disjoint. Therefore, for each constraint c_i, there exists exactly one $s_{i,\bar{j}}$ that is true, while all the other $s_{i,j}$ are assigned to false. Clearly, μ is also a model for Equation 5 because the added term $\bigwedge_{j=1}^{D_i} \bigwedge_{k=j+1}^{D_i} (\neg s_{i,j} \vee \neg s_{i,k})$ is satisfied: only $s_{i,\bar{j}}$ is set to true, therefore in each conjunct at least one variable is set to false.

Let μ' be a model of Equation 5. Because of the added term, there exists exactly one $s_{i,\bar{j}}$ that is true, while all the other $s_{i,j}$ are assigned to false. For the same reasoning followed above, μ' is also a model of Equation 3. □

This encoding is in CNF, but its size is quadratic in the size of the TP because of the added term $\bigwedge_{j=1}^{D_i} \bigwedge_{k=j+1}^{D_i} (\neg s_{i,j} \vee \neg s_k)$.

In the running example this encoding is as follows.

$$\begin{aligned} &(B_e - B_s \geq 8) \wedge (B_e - B_s \leq 11) \wedge \\ &(B_e - A_s \geq 0) \wedge (B_e - A_s \leq 20) \wedge \\ &(B_s - A_e \geq 0) \wedge \\ &(\neg s_1 \vee (A_e - A_s \geq 7)) \wedge \\ &(\neg s_1 \vee (A_e - A_s \leq 8)) \wedge \\ &(\neg s_2 \vee (A_e - A_s \geq 10)) \wedge \\ &(\neg s_2 \vee (A_e - A_s \leq 11)) \wedge \\ &(s_1 \vee s_2) \wedge \\ &(\neg s_1 \vee \neg s_2) \end{aligned} \tag{6}$$

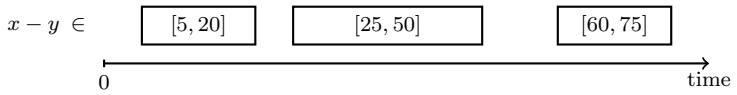

Fig. 3 Example of TCSPU constraint

5.3 Hole Encoding

A different encoding for the TCSP problem class is obtained as follows. For each constraint $c_i = \bigvee_{j=1}^{D_i}(x_i - y_i) \in [l_{i,j}, u_{i,j}]$[4], we constrain $x_i - y_i \in [l_{i,1}, u_{i,D_i}]$, and we exclude the "holes" between intervals, a hole being an open interval $(u_{i,j}, l_{i,j+1})$. The result is the *hole encoding* reported in Equation 7.

$$\begin{aligned}\bigwedge_{i=1}^{|C_f|}(((x_i - y_i) \geq l_{i,1}) \wedge ((x_i - y_i) \leq u_{i,D_i}) \wedge \\ (\bigwedge_{j=1}^{D_i-1}((x_i - y_i) \leq u_{i,j}) \vee ((x_i - y_i) \geq l_{i,(j+1)})))\end{aligned} \tag{7}$$

Theorem 3 *If P is a TCSP, P is consistent if and only if Equation 7 is satisfiable (and a model of Equation 7 yields a consistent schedule for P).*

Proof. Assuming that P is a TCSP, we prove that Equation 7 is equivalent to Equation 1.

We start from a the formula in Equation 1, and we consider a single constraint in isolation: $\bigvee_{j=1}^{D_i}(((x_i - y_i) \geq l_{i,j}) \wedge ((x_i - y_i) \leq u_{i,j}))$. This sub-formula is in Disjunctive Normal Form, and can be transformed in an equivalent exponential-size CNF by applying the distributive rule. We obtain a formula with 2^{D_i} clauses of D_i disjuncts each. Each clause is a permutation obtained by picking a conjunct for each disjunct of the original formula.

One clause in this CNF formula is composed of all the upper bound constraints: $\bigvee_{k=1}^{D_i}((x_i - y_i) \leq u_{i,k})$, this clause can be trivially simplified to $((x_i - y_i) \leq u_{i,D_i})$ as u_{i,D_i} is bigger than any other upper bound. Similarly, the clause $\bigvee_{k=1}^{D_i}((x_i - y_i) \geq l_{i,k})$ becomes $((x_i - y_i) \geq l_{i,1})$.

The remaining clauses contain both upper and lower constraints. Each clause c can be reduced to a binary clause in the form $((x_i - y_i) \geq L_{i,c}) \vee ((x_i - y_i) \leq U_{i,c})$, where $L_{i,c}$ is the minimum of the lower bounds and $U_{i,c}$ is the maximum of the upper bounds. The obtained 2-CNF formula is exponential in the size of the original constraint. For each j, the clause $((x_i - y_i) \leq u_{i,j}) \vee ((x_i - y_i) \geq l_{i,(j+1)})$ subsumes all the binary clauses with bigger upper bound and smaller lower bound.

We can apply this reasoning to all the conjuncts of Equation 1, and we obtain the formulation in Equation 7. Since the two formulations are equivalent, they have the same models, a consistent schedule can be extracted from each model of Equation 7 □

Consider for example Figure 3, depicting the constraint $(x - y \in [5, 20]) \vee (x - y \in [25, 50]) \vee (x - y \in [60, 75])$. The hole encoding of this constraint is $((x - y) \geq 5) \wedge ((x - y) \leq 20 \vee (x - y) \geq 25) \wedge ((x - y) \leq 50 \vee (x - y) \geq 60) \wedge ((x - y) \leq 75)$. This encoding is linear in the size of the original TP, does not introduce any additional variable, and, most importantly, results in a 2-CNF formula. These properties are noteworthy and will be exploited in the following sections.

[4] Note that in TCSP the constraints are binary, i.e. each constraint relates exactly two variables.

In the running example this encoding is as follows.

$$\begin{aligned}
&(B_e - B_s \geq 8) \wedge (B_e - B_s \leq 11) \wedge \\
&(B_e - A_s \geq 0) \wedge (B_e - A_s \leq 20) \wedge \\
&(B_s - A_e \geq 0) \wedge \\
&(A_e - A_s \geq 7) \wedge \\
&((A_e - A_s \leq 8) \vee (A_e - A_s \geq 10)) \wedge \\
&(A_e - A_s \leq 11)
\end{aligned} \tag{8}$$

Finally, we notice that Equation 7 is logically equivalent to Equation 1 (in the applicable case of TCSP), while Equations 3 and 5 are only are equi-satisfiable to it, because of the added switch variables. The solution to the temporal problem is still obtained directly from any satisfying assignment, gathering the values for the variables in $\vec{X}_c$.

6 Encoding of SC Problems in SMT

We now consider the SC problem, in which some time points are not schedulable by the solver, and are considered uncontrollable when looking for a schedule for the controllable time points. We describe the reduction of the SC problem to SMT. We developed a number of encodings that are satisfiable if and only if the temporal problem is strongly controllable, and such that a model of each encoding yields a solution for the original problem.

In the following, we assume that a TPU $P = (X_c, X_u, C_c, C_f)$ is given.

6.1 Encodings into Quantified LRA

As in the previous section, each time point is associated with an SMT variable. The encoding in Equation 9 is a direct logical mapping of the notion of strong controllability; we call this encoding *direct encoding*.

$$\forall \vec{X}_u.(C_c(\vec{X}_c, \vec{X}_u) \rightarrow C_f(\vec{X}_c, \vec{X}_u)) \tag{9}$$

Proposition 2 *The TPU P is strongly controllable if and only if Equation 9 is satisfiable (and a model of Equation 9 yields a strong schedule for P).*

Proposition 2 is directly obtained by formalizing the definition of strong controllability. Intuitively, Equation 9 is satisfiable if and only if there exists an assignment to the controllable variables X_c such that, for all assignments to the uncontrollable variables X_u satisfying the contingent constraints C_c, the free constraints C_f are also satisfied[5]. In the above formula, the controllable variables are implicitly existentially quantified. In case of satisfiability, the SMT solver returns a satisfying assignment to the controllable variables that is exactly a strong schedule.

[5] Here we assume that the contingent constraints are not contradictory, otherwise the implication will be automatically true. However, the non-contradiction of contingent constraints is true by construction in our definition of temporal problem, as no relationship between different contingent constraints is possible.

In the running example, this encoding is as follows.

$$\begin{aligned}&\forall B_e.(((B_e - B_s \geq 8) \wedge (B_e - B_s \leq 11)) \rightarrow \\ &\quad ((B_e - A_s \geq 0) \wedge (B_e - A_s \leq 20) \wedge (B_s - A_e \geq 0) \wedge \\ &\qquad (((A_e - A_s \geq 7) \wedge (A_e - A_s \leq 8)) \vee ((A_e - A_s \geq 10) \wedge (A_e - A_s \leq 11)))))\end{aligned} \tag{10}$$

In order to enable further simplifications, we notice that contingent constraints depend both on controllable and uncontrollable time points, and we re-code the problem as follows. We rewrite each uncontrollable time point e_i in terms of the time difference with its starting time point b_i by means of an uncontrollable offset variable y_i. For every contingent constraint $cc_i = \bigvee_{j=1}^{E_i}(e_i - b_i) \in [l_{i,j}, u_{i,j}]$, let $y_i \in \mathbb{R}$ be the uncontrollable offset variable associated to e_i such that: $0 \leq y_i \leq (u_{i,E_i} - l_{i,1})$ and $\bigwedge_{j=E_i}^{2}((y_i \leq u_{i,E_i} - l_{i,j}) \vee (y_i \geq u_{i,E_i} - u_{i,j-1}))$. Note that this rewriting is actually an hole-encoding of the i-th contingent constraint[6]. The rewriting is such that $e_i = b_i + u_{i,E_i} - y_i$.

For example, the contingent constraint $B_e - B_s \in [8, 11]$ of our running example can be rewritten as $(y_1 \geq 0) \wedge (y_1 \leq 3)$ and the occurrences of B_e in free constraints are replaced by $(B_s + 11 - y_1)$.

Intuitively, y_i represents the offset with respect to the maximum duration, and can be used to rewrite all the constraints involving e_i in terms of b_i and y_i only. We formalize this rewriting as a function ρ such that $\rho(e_i) \doteq b_i + u_{i,E_i} - y_i$. With a small abuse of notation, we denote with $\rho(\vec{X}_u)$ the vector of formulae obtained by the application of ρ to all the elements of X_u. To simplify the notation, we also introduce the vector $\vec{Y}_u$ that is the vector of uncontrollable offset variables $(y_1, ..., y_m)$. Thanks to the redefinition of each e_i in terms of y_i, the rewriting of the contingent constraints depends on $\vec{Y}_u$ only.

Let $\Gamma(\vec{Y}_u)$ be the formula representing the conjunction of all the contingent constraints after the recoding, and $\Psi(\vec{X}_c, \vec{Y}_u)$ be the conjunction of all the free constraints rewritten in terms of $\vec{X}_c$ and $\vec{Y}_u$.

$$\begin{aligned}\Gamma(\vec{Y}_u) &\doteq \bigwedge_{i=1}^{m}((y_k \in [0, u_{i,E_i} - l_{i,1}]) \wedge \\ &\qquad (\bigwedge_{j=E_i}^{2}((y_i \leq u_{i,E_i} - l_{i,j}) \vee (y_i \geq u_{i,E_i} - u_{i,j-1})))) \\ \Psi(\vec{X}_c, \vec{Y}_u) &\doteq \bigwedge_{c \in C_f} c[\rho(\vec{X}_u)](\vec{X}_c, \vec{Y}_u)/\vec{X}_u]\end{aligned}$$

In this setting, the strong controllability problem consists in finding a value for $\vec{X}_c$ that satisfies the free constraints $\Psi(\vec{X}_c, \vec{Y}_u)$ under any possible value of $\vec{Y}_u$ that satisfies $\Gamma(\vec{Y}_u)$.

The SC encoding in Equation 9 can be recoded as an LRA formula in the free variables $\vec{X}_c$ as follows.

$$\forall \vec{Y}_u.(\Gamma(\vec{Y}_u) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u)) \tag{11}$$

Theorem 4 *The TPU P is strongly controllable if and only if Equation 11 is satisfiable (and a strong schedule can be extracted from any of its models).*

[6] This is possible because the contingent constraints are restricted to be binary.

Proof. Equation 11 is obtained by rewriting Equation 9 with the offsets, we prove that it is equivalent to Equation 9. We show this property by refutation.

Suppose μ is a model of Equation 9 but it is not a model of Equation 11. Then, there exist a $\vec{Y}_u$ such that $\Gamma(\vec{Y}_u)$ holds but $\Psi(\mu, \vec{Y}_u)$ does not hold. Let $\vec{X}_u \doteq \rho^{-1}(\vec{Y}_u)$. By definition of ρ, $C_f(\vec{X}_c, \vec{X}_u)$ does not hold and $C_c(\vec{X}_c, \vec{X}_u)$ holds. But this is absurd because this makes Equation 9 unsatisfiable.

Similarly we can show that any model of Equation 11 is also a model of Equation 9. □

We call this encoding *Offset Encoding*. This formulation corresponds to a quantified SMT problem in LRA, and still requires a solver that supports quantified formulae, but the part of the encoding representing the contingent constraint is now dependent on $\vec{Y}_u$ only.

In the running example, this encoding is as follows.

$$\begin{aligned}&\forall y_1.(((y_1 \geq 0) \wedge (y_1 \leq 3)) \rightarrow \\ &\quad((B_s + 11 - y_1 - A_s \geq 0) \wedge (B_s + 11 - y_1 - A_s \leq 20) \wedge (B_s - A_e \geq 0) \wedge \\ &\quad(((A_e - A_s \geq 7) \wedge (A_e - A_s \leq 8)) \vee ((A_e - A_s \geq 10) \wedge (A_e - A_s \leq 11)))))\end{aligned} \tag{12}$$

The main problem in the previous encodings is the scope of the universal quantifier. Since the computational cost of quantification is very high, we can rewrite the offset encoding in Equation 11 in order to obtain a possibly more efficient encoding. Let us assume that $\Psi(\vec{X}_c, \vec{Y}_u)$ is written as a conjunction of H clauses $\psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})$, where $X_{c_h} \subseteq X_c$ and $Y_{u_h} \subseteq Y_u$ are the variables used in the clause ψ_h. This assumption can be easily satisfied by converting $\Psi(\vec{X}_c, \vec{Y}_u)$ in CNF using any consistency encoding we presented in the previous section [7].

$$\Psi(\vec{X}_c, \vec{Y}_u) = \textstyle\bigwedge_{h=1}^{H} \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})$$

We have that $\bigwedge_h \forall \vec{Y}_u.(\neg\Gamma(\vec{Y}_u) \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h}))$ can be equivalently rewritten to $\bigwedge_h \forall \vec{Y}_{u_h}.(\neg\Gamma(\vec{Y}_u)|_{Y_{u_h}} \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h}))$, and we obtain the following *distributed encoding*.

$$\textstyle\bigwedge_h \forall \vec{Y}_{u_h}.(\neg\Gamma(\vec{Y}_u)|_{Y_{u_h}} \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})) \tag{13}$$

Theorem 5 *If the TPU P is consistent, it is strongly controllable if and only if Equation 13 is satisfiable (and each model yields a strong schedule).*

Proof. We show that Equation 13 is equivalent to Equation 11.
We start from Equation 11. and we rewrite it as follows.
$\forall \vec{Y}_u.(\Gamma(\vec{Y}_u) \rightarrow \Psi(\vec{X}_c, \vec{Y}_u))$
$\Leftrightarrow \forall \vec{Y}_u.(\neg\Gamma(\vec{Y}_u) \vee \Psi(\vec{X}_c, \vec{Y}_u))$

We can now replace $\Psi(\vec{X}_c, \vec{Y}_u)$ with its CNF formulation and distribute the disjunction over the big conjunction.
$\Leftrightarrow \forall \vec{Y}_u.(\neg\Gamma(\vec{Y}_u) \vee \bigwedge_{h=1}^{H} \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h}))$
$\Leftrightarrow \forall \vec{Y}_u. \bigwedge_{h=1}^{H} (\neg\Gamma(\vec{Y}_u) \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h}))$

[7] If the used encoding introduces additional variables, those are existentially quantified and extend the model of Equation 9 by preserving the satisfiability and the strong schedules encoded in the models.

By distribution of $\forall$ over $\wedge$ we obtain the following formulation.
$\Leftrightarrow \bigwedge_{h=1}^{H} \forall \vec{Y}_u.(\neg \Gamma(\vec{Y}_u) \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h}))$

Let $\vec{Y}_{u_k} \doteq \vec{Y}_u \setminus \vec{Y}_{u_h}$ be the variables of $\vec{Y}_u$ not appearing in the h-th clause. The clauses of $\Gamma(\vec{Y}_u)$ can be split in two parts depending on the offset variables they constrain, because every clause contains exactly one offset variable by construction.
$\Leftrightarrow \bigwedge_{h=1}^{H} \forall \vec{Y}_u.(\neg \Gamma(\vec{Y}_u)|_{Y_{u_h}} \vee \neg \Gamma(\vec{Y}_u)|_{Y_{u_k}} \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h}))$

The universal quantification $\forall \vec{Y}_u$ can be split in $\forall \vec{Y}_{u_k}.\forall \vec{Y}_{u_h}$.
$\Leftrightarrow \bigwedge_{h=1}^{H} \forall \vec{Y}_{u_k}.(\neg \Gamma(\vec{Y}_u)|_{Y_{u_k}} \vee \forall \vec{Y}_{u_h}.(\neg \Gamma(\vec{Y}_u)|_{Y_{u_h}} \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})))$

Since $\Gamma(\vec{Y})$ is assumed to be non-contradictory, $\Gamma(\vec{Y}_u)|_{Y_{u_k}}$ cannot be false for every $\vec{Y}_{u_k}$. Therefore, $\neg \Gamma(\vec{Y}_u)|_{Y_{u_k}}$ reduces to $\bot$. We can then remove this disjunct and the relative quantification become useless.
$\Leftrightarrow \bigwedge_{h=1}^{H} \forall \vec{Y}_{u_k}.(\forall \vec{Y}_{u_h}.(\neg \Gamma(\vec{Y}_u)|_{Y_{u_h}} \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})))$
$\Leftrightarrow \bigwedge_{h=1}^{H} \forall \vec{Y}_{u_h}.(\neg \Gamma(\vec{Y}_u)|_{Y_{u_h}} \vee \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h}))$
This is exactly the formulation of Equation 13. □

The size of the produced (quantified) formula is linear with respect to the original TPU. This encoding still requires a solver that supports quantified formulae, and contains as many quantifiers as clauses. However, each quantification is now restricted to the offset variables $Y_{u_h} \subseteq Y_u$ occurring in each clause ψ_h. This encoding also limits the scope of the universal quantifiers, which turns out to be beneficial in practice. Intuitively, this is related to the fact that a number of quantifier eliminations in LRA on smaller formulae may be much cheaper than a single, monolithic quantifier elimination over a large formula.

If we use the hole encoding to obtain the CNF formula for the free constraints, the running example formulation of this encoding is as follows.

$$\begin{aligned} &(\forall y_1.((y_1 < 0) \vee (y_1 > 3) \vee (B_s + 11 - y_1 - A_s \geq 0))) \wedge \\ &(\forall y_1.((y_1 < 0) \vee (y_1 > 3) \vee (B_s + 11 - y_1 - A_s \leq 20))) \wedge \\ &(B_s - A_e \geq 0) \wedge (A_e - A_s \geq 7) \wedge \\ &((A_e - A_s \leq 8) \vee (A_e - A_s \geq 10)) \wedge (A_e - A_s \leq 11) \end{aligned} \tag{14}$$

6.2 Encodings into Quantifier-free LRA

In order to exploit solvers that do not support quantifiers, we propose an encoding of strong controllability into a quantifier-free SMT (LRA) formula. This is obtained by resorting to an external procedure for quantifier elimination.

We rewrite Equation 13 as $\bigwedge_h \neg(\exists \vec{Y}_{u_h}.(\Gamma(\vec{Y}_u)|_{Y_{u_h}} \wedge \neg \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})))$, in order to apply a procedure for the elimination of existential quantifiers (e.g. Fourier-Motzkin [32]). In the following we refer to each conjunct after quantifier elimination as $\psi_h^{\Gamma}(\vec{X}_{c_h})$ ($\psi_h^{\Gamma}(\vec{X}_{c_h})$ is then a quantifier-free formula).

$$\psi_h^{\Gamma}(\vec{X}_{c_h}) \leftrightarrow \neg(\exists \vec{Y}_{u_h}.(\Gamma(\vec{Y}_u)|_{Y_{u_h}} \wedge \neg \psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})))$$

The resulting encoding, reported in Equation 15, is called *eager for-all elimination encoding*.

$$\bigwedge_h \psi_h^{\Gamma}(\vec{X}_{c_h}) \tag{15}$$

Theorem 6 *The TPU P is strongly controllable if and only if Equation 13 is satisfiable (and a strong schedule can be extracted from a model of Equation 13).*

Proof. Equation 15 derives from Equation 13 by resolving the universal quantifier using a quantifier elimination procedure. Since the elimination procedure builds equivalent formulae, Equation 15 and Equation 13 are logically equivalent. □

In the running example, this encoding is as follows.

$$\begin{aligned}&(\neg\exists y_1.((y_1 \geq 0) \wedge (y_1 \leq 3) \wedge (B_s + 11 - y_1 - A_s < 0))) \wedge \\ &(\neg\exists y_1.((y_1 \geq 0) \wedge (y_1 \leq 3) \wedge (B_s + 11 - y_1 - A_s > 20))) \wedge \\ &(B_s - A_e \geq 0) \wedge (A_e - A_s \geq 7) \wedge \\ &((A_e - A_s \leq 8) \vee (A_e - A_s \geq 10)) \wedge (A_e - A_s \leq 11)\end{aligned} \tag{16}$$

For the simple-natured TCSPU class, it is not necessary to apply a general purpose quantifier elimination procedure. Given the specific nature of the constraints and the limitation to convex contingent constraints, only few cases are possible, and for each of them we use a pattern-based encoding, that in essence precomputes the result of quantifier elimination. This result can be thought of as generalizing to simple-natured TCSPUs the result proposed by Fargier and Vidal [36] for the case of STPU. We start from the distributed encoding of Equation 13, where the each (sub)clause ψ_h is generated by the hole encoding. We treat each clause as a separate existential quantification problem, and provide static results for each case. The final result is logically equivalent to the corresponding $\psi_h^\Gamma(\vec{X}_{c_h})$ in Equation 15.

Each clause under analysis results from the encoding of a free constraint in the TCSPU over variables v and w, with D intervals. Let t be $v - w$. The encoding results in two unit clauses ($t \geq l_1$ and $t \leq u_D$), and in $D - 1$ binary clauses in the form $(t \leq u_i) \vee (t \geq l_{i+1})$.

The static elimination procedure must deal with eight possible cases, depending on v and w being controllable or uncontrollable[8]. The eight possible clause patterns are shown in Table 1. For unit clauses, we proceed as in the work by Fargier and Vidal [36]: the first four rows of Table 1 report these results.

The rest of the table present the results for the disjunctive binary clauses. The static quantification is possible by knowing that the contingent constraints are in the shape $e_i - b_i \in [L_i, U_i]$ and thus each possible free constraint clause can be parametrized and resolved upfront.

During the encoding of a given problem, we can now generate the set of clauses using the hole encoding, search in the table which is the applicable pattern and instantiate the resulting $\psi_h^\Gamma(\vec{X}_{c_h})$ (The quantifier free formula that is equivalent to $\neg(\exists\vec{Y}_{u_h}.(\Gamma(\vec{Y}_u)|_{Y_{u_h}} \wedge \neg\psi_h(\vec{X}_{c_h}, \vec{Y}_{u_h})))$) that form a set of clauses to be conjoined to obtain a sound and complete SMT encoding for strong controllability.

In the running example, Equation 16 can be equivalently transformed using this technique as follows.

$$\begin{aligned}&(B_s - A_s \leq 9) \wedge (A_s - B_s \leq 8) \wedge (B_s - A_e \geq 0) \wedge (A_e - A_s \geq 7) \wedge \\ &((A_e - A_s \leq 8) \vee (A_e - A_s \geq 10)) \wedge (A_e - A_s \leq 11)\end{aligned} \tag{17}$$

[8] The possible cases are actually sixteen but $v - w \geq k$ can be rewritten as $w - v \leq -k$, thus halving the possibilities.

Clause pattern	**Quantification Result** ($\psi_h^{\Gamma}(\vec{X}_{c_h})$)
$(b_i - b_j) \geq k$	$(b_i - b_j) \geq k$
$(e_i - b_j) \geq k$	$(b_i - b_j) \geq k - L_i$
$(b_i - e_j) \geq k$	$(b_i - b_j) \geq k + U_j$
$(e_i - e_j) \geq k$	$(b_i - b_j) \geq k - L_i + U_j$
$(b_i - b_j) \leq k_1 \vee$ $(b_i - b_j) \geq k_2$	$(b_i - b_j) \leq k_1 \vee (b_i - b_j) \geq k_2$
$(e_i - b_j) \leq k_1 \vee$ $(e_i - b_j) \geq k_2$	$((b_i + L_i - b_j > k_1) \vee (b_i + U_i - b_j \leq k_1)) \wedge$ $((b_i + L_i - b_j < k_1) \vee (b_i + L_i - b_j \geq k_2))$
$(b_i - e_j) \leq k_1 \vee$ $(b_i - e_j) \geq k_2$	$((b_i - b_j - L_j < k_2) \vee (b_i - b_j - U_j \geq k_2)) \wedge$ $((b_i - b_j - L_j > k_2) \vee (b_i - b_j - L_j \leq k_1))$
$(e_i - e_j) \leq k_1 \vee$ $(e_i - e_j) \geq k_2$	$((b_i + U_i - b_j - U_j > k_1) \vee (b_i + U_i - b_j - L_j \leq k_1)) \wedge$ $((b_i + U_i - b_j - U_j < k_1) \vee (b_i + L_i - b_j - U_j \geq k_1)) \wedge$ $((b_i + L_i - b_j - L_j < k_2) \vee (b_i + L_i - b_j - U_j \geq k_2)) \wedge$ $((b_i + L_i - b_j - L_j > k_2) \vee (b_i + L_i - b_j - L_j \leq k_2))$

Table 1 Static quantification for simple-natured TCSPUs. For each clause pattern coming from an hole-encoding of free constraints, the corresponding $\psi_h^{\Gamma}(\vec{X}_{c_h})$ is presented, assuming that if e_i is an uncontrollable time point, b_i is its corresponding controllable time point that relates to it with the i-th contingent constraint $e_i - b_i \in [L_i, U_i]$.

The constraint $(B_s - A_s \leq 9)$ comes from the $B_e - A_s \leq 20$ clause using the third rule, while $(A_s - B_s \leq 8)$ is derived from $B_e - A_s \geq 0$ using the second rule.

In order to explain the intuition of the rules, let us show, as an example, why the constraint $(B_s - A_s \leq 9)$ comes from the $B_e - A_s \leq 20$. By rewriting the constraint $B_e - A_s \leq 20$, we get $B_s + 11 - y_1 - A_s \leq 20$. This inequality must hold for any $y_1 \in [0, 3]$, because y_1 is uncontrollable. A fortiori, it must hold for $y_1 = 0$, that is the worst case for an upper bound constraint[9]. Therefore, we obtain $B_s + 11 - A_s \leq 20$, that is $B_s - A_s \leq 9$.

The construction described above can be used in Equation 15. This specialized quantification technique results in a 2-CNF formula that has linear size in the original TCSPU. This is because the size of the hole encoding is linear, and for each clause, we statically resolve the quantification by creating at most four new binary clauses. This encoding spares the computational cost of quantifier elimination and produces a highly optimized QF_LRA formula.

7 Related Work

The notion of strong controllability was introduced by Fargier and Vidal in their seminal paper [36]. The problem is defined for the limited case of STPU. The authors identify a very efficient, static quantification technique to represent the solution space of an STPU in form of an STP. This technique is at the core of their STPU procedure (hereafter referred to as FargierVidal). Compared to [36], we propose a comprehensive solution and an implementation for the general cases of TCSPU and DTPU. Furthermore, we generalize the static quantification techniques proposed in [36] to the case of simple-natured TCSPUs.

[9] Recall that setting $y_1 = 0$ means assuming the duration of activity B to be its maximum, namely 11.

The only previous work tackling cases of strong controllability beyond STPU is [29]. This work, in the following referred to as PVYS[10], is discussed in detail in the following section. A comparison with possible approaches based on polyhedra is presented in section 7.2. Other related papers are discussed in section 7.3.

7.1 The PVYS algorithm

The first (and only) technique to solve the strong controllability problem for the DTPU problem class is proposed in [29]. The PVYS algorithm can be described in terms of two nested enumerations. At the highest level, it explicitly enumerates every possible way (hereafter refereed to as *contingent choice*) to satisfy the contingent constraints. Intuitively, a contingent choice corresponds to picking, for each contingent constraint, one disjunct. For each contingent choice, PVYS obtains a *simple-natured* DTPU. In turn, the solution space (i.e. the set of strong schedules) of each simple-natured DTPU is represented as a DTP. The DTPs thus obtained are intersected, and result into a DTP that represents the solution space for the original DTPU. The innermost enumeration is used to convert each simple-natured DTPU, associated with the contingent choice μ_c, to the corresponding DTP. More specifically, PVYS explicitly enumerates every possible *free choice*, i.e. every possible way to satisfy each free constraint. Each free choice μ_f, in combination with μ_c, yields an STPU, which can be efficiently reduced to an STP by FargierVidal. All the STPs are then combined, by disjunction, into the DTP representing the solution space for the contingent choice μ_c.

Consider the following example, with $C_c \doteq \{cc_1, cc_2, cc_3\}$ and $C_f \doteq \{cf_1, cf_2, cf_3\}$.

$$cc_1 \doteq (e_1 - b_1 \in [10, 20]) \vee (e_1 - b_1 \in [30, 40]) \vee (\mathbf{e_1 - b_1 \in [70, 80]})$$
$$cc_2 \doteq (\mathbf{e_2 - b_2 \in [5, 8]})$$
$$cc_3 \doteq (e_3 - b_3 \in [1, 5]) \vee (\mathbf{e_3 - b_3 \in [10, 15]})$$

$$cf_1 \doteq (\mathbf{x_1 - x_2 \in [10, 30]}) \vee (x_1 - x_3 \in [3, 4])$$
$$cf_2 \doteq (\mathbf{x_3 - x_5 \in [10, \infty)})$$
$$cf_3 \doteq (x_2 - x_4 \in [20, 20]) \vee (\mathbf{x_1 - x_4 \in [10, 10]})$$

A possible contingent choice of C_c is shown in blue, and a free choice is highlighted in red. A logical characterization of the algorithm is given below. Given a DTPU (X_c, X_u, C_c, C_u), the algorithm computes the final DTP as follows.

$$\bigwedge_{\mu_c \in \textsc{Choice}(C_c)} \bigvee_{\mu_f \in \textsc{Choice}(C_f)} \textsc{FargierVidal}(X_c, X_u, \mu_c, \mu_f)$$

where the two calls to Choice are used to produce the free and contingent choices, and embody the two enumerations. Intuitively, the external conjunction iterates over the "blue" choices, while the internal disjunction iterates over the "red" ones.

The pseudo-code of PVYS, reported in Algorithm 1, has several optimizations with respect to the high-level view proposed above. The top-level function

[10] We use PVYS after the surnames of the authors.

Algorithm 1 PVYS algorithm (taken from Peintner et al. [29])

```
1: procedure DTPU-SC(A, A_C, C_S, C_C, C_E)
2:     S ← ∅
3:     if C_S = ∅ then
4:         G ← MinimalNetwork(A)
5:         S ← ALL-PATHS-SC(A, A_C, C_C, C_E, G)
6:     else
7:         C_i ← SelectVariable(C_S)
8:         C'_S ← C_S − {C_i}
9:         for c_ij ∈ Disjuncts(C_i) do
10:            A'_C ← A_C ∪ c_ij
11:            A' ← A ∧ SCTransform(A'_C, c_ij)
12:            if IsConsistent(A') then
13:                S ← DTPU-SC(A', A'_C, C'_S, C_C, C_E)
14:                if S ≠ ∅ then
15:                    return S
16:    return S

1: procedure ALL-PATHS-SC(A, A_C, C_C, C_E, G)
2:     if C_C = ∅ then
3:         G ← G ∧ SATISFY-Ce(A, A_C, C_E)
4:     else
5:         C_i ← SelectVariable(C_C)
6:         C'_C ← C_C − {C_i}
7:         for c_ij ∈ Disjuncts(C_i) do
8:             A'_C ← A_C ∪ c_ij
9:             A' ← A ∧ SCTransform(A'_C, c_ij)
10:            if IsConsistent(A') then
11:                G ← ALL-PATHS-SC(A', A'_C, C'_C, C_E, G)
12:                if G = ∅ then
13:                    return ∅
14:            else
15:                return ∅
16:    return G

1: procedure SATISFY-Ce(A, A_C, C_E)
2:     H = ∅
3:     if C_E = ∅ then
4:         H ← MinimalNetwork(A)
5:     else
6:         C_i ← SelectVariable(C_E)
7:         C'_E ← C_E − {C_i}
8:         for c_ij ∈ Disjuncts(C_i) do
9:             A' ← A ∧ SCTransform(A'_C, c_ij)
10:            if IsConsistent(A') then
11:                H ← H ∨ SATISFY-Ce(A', A'_C, C'_E)
12:    return H
```

DTPU-SC considers the problem constraints divided into three sets: C_S contains the simple constraints (i.e. constraints not containing disjunctions), and the disjunctive constraints that are defined over controllable variables only; C_C contains the disjunctive contingent constraints; C_E contains the other disjunctive free constraints. The procedure makes use of four additional data structures: the STP A, the STPU A_c, and the DTPs G and H.

In procedure DTPU-SC, the algorithm selects a combination of one disjunct for each constraint in C_S and accumulates the result in A. The constraints are rewritten one by one, using the approach by Fargier and Vidal: the function SC-

Transform takes a constraint and a STPU, and rewrites that constraint with respect to the STPU eliminating uncontrollable time points.

For each combination of disjuncts, the function ALL-PATHS-SC checks if the choice of free disjuncts satisfies all the contingent constraints by considering each possible combination of contingent disjuncts separately. For each combination, it accumulates the rewriting in the STP A and invokes the function SATISFYCe that computes a DTP with all the possible solutions for the remaining free constraints. All the DTPs are accumulated by conjunction in G until either G becomes empty, meaning that there exist no solution that works for all the contingent disjunct combinations, or it contains at least on solution that is compatible with all the combinations and is returned. The algorithm terminates when a solution is found, or when all the combinations of C_S have been explored. The intermediate checks for consistency (via the function IsConsistent) are used for early-termination.

The key difference between our approach and PVYS is in the nature of enumerations. PVYS explicitly enumerates the choices over the free and contingent constraints. This may be costly if many disjunctive constraints are present in the problem: in fact, in the worst case, all the possible combinations of disjuncts must be analyzed. In our approach, the enumerations are carried out symbolically, and relying on the SMT infrastructure for efficiency. In this way, we inherit effective splitting heuristics, learning, and backjumping. Moreover, the early pruning mechanism in the SMT solver is able to draw conclusions from "partial" choices, where a disjunct is not (yet) chosen for each clause [6]. Finally, we use more powerful quantifier elimination techniques, that are able to deal with DTPUs at once.

7.2 Polyhedra-based approach

The ideas presented in this paper are based on LRA formulae, and are made practical by leveraging SMT solvers. In principle, given the geometric interpretation of LRA, the problem could be addressed by other means. In fact, each conjunction of LRA atoms is a Non-Necessarily Closed Convex Polyhedron (NNC-Polyhedron), and each formula over LRA can be seen as the (non-convex) union of finitely-many NNC-Polyhedra. In this parallelism, conjunction corresponds to intersection, disjunction to union, negation to complement, and existential quantification to projection.

Many libraries for manipulating NNC-Polyhedra are available (e.g. [3,37]), and could be used as a backend for the problems described here instead of SMT solvers. In an early stage of this research, we also explored this possibility, experimenting with the Parma Polyhedra Library [3], one of the most efficient libraries available. Unfortunately, the results we obtained were dramatically in favor of the SMT approach. We could identify various reasons for this lack of scalability. On the one side, the explicit manipulation of polyhedra disjunctions may be very costly. On the other, using polyhedra-based solvers, we are computing the entire solution space, while the SMT based approaches are only looking for one solution. Further discussion of these techniques is out of the scope of this paper.

7.3 Other related work

We mention two other forms of controllability for TPUs: weak controllability (WC) and dynamic controllability (DC). A TPU is said to be WC if, for every possible evolution of the uncontrollable environment, there exists an allocation to the controllable time points that fulfills the free constraints of the problem. This notion is much weaker than SC, because the allocation strategy for the controllable time points is allowed to depend on the allocation of the uncontrollable time points. In this setting, the solver is assumed to be "clairvoyant" and is able to decide its moves based on the past and also the future moves of the opponent. In their seminal paper, Vidal and Fargier [36] also address the WC problem for the STPU class. Algorithms for deciding WC for TCSPU and DTPU are provided by Venable et al. [35]. The use of SMT techniques to deal with weak controllability has been recently investigated [11], addressing both the decision and the strategy extraction problems (i.e. the problem of checking if a TPU is WC, and the problem of building a strategy for the solver). The work presented in this paper tackles a radically different problem. An important difference between SC and WC is the shape of the solution: while in SC a solution is a static assignment to controllable time points, in WC the strategy requires conditional structures to be expressed. Thus, the use of SMT techniques is also substantially different from what is done here.

DC is similar to WC, but the choices of the scheduler can be based on past environment decisions only. As pointed out by Fargier and Vidal [36], if a problem is SC then it is also DC and if it is DC then it is also WC, but the implication chain is not reversible. In Morris et al. [24], the authors focus on deciding DC for the STPU problem class, while Venable et al. [35] extended the result for TCSPUs. However, no effective solutions to DC exists for the DTPU problem class.

As far as the consistency problem is concerned, the state-of-the-art for STP is the work by Planken et al. [30], which presents an efficient algorithm for computing all-pairs shortest paths in a directed graph with possibly negative real weights. The use of SMT techniques to solve the consistency problem has been explored in [2], where the TSAT++ tool is presented. TSAT++ can be seen as a specialized SMT solver for DTP problems. The work does not deal with strong controllability, and is limited to consistency for temporal problems. The performance of TSAT++ relative to more modern SMT solvers is analyzed in the next section, on consistency of temporal problems.

8 Experimental Evaluation

In this section, we experimentally evaluate our approach. We describe our implementation (section 8.1), the experimental set-up (section 8.2), and the results for consistency and for strong controllability (sections 8.3 and 8.4). Finally, in section 8.5 we evaluate (our implementation of) the PVYS algorithm [29].

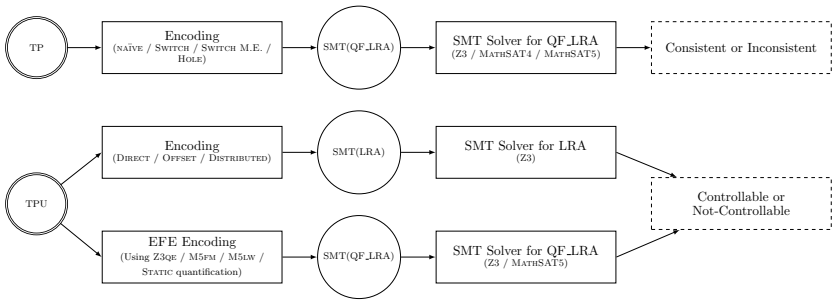

Fig. 4 Graphical representation of the developed toolchain.

8.1 Implementation

We developed a tool that automatically encodes the various classes of temporal problems as SMT problems. The tool, depicted in Figure 4, can deal with consistency problems, by generating SMT (QF_LRA) encodings (upper row), that can then be solved by MathSAT4 [7], MathSAT5 [9] and Z3 [26].

For strong controllability problems, the tool has two flows. First, it implements the three encodings to quantified SMT (LRA), that are then solved by Z3. Second, it generates quantifier-free SMT (QF_LRA) encodings, by applying eager quantifier elimination techniques. The quantifier elimination procedure in the eager for-all elimination encoding is carried out by calling one of the following procedures: the internal formula simplifier of Z3 [26] (denoted EFE Z3qe); Fourier-Motzkin quantifier elimination (EFE M5fm), built on top of MathSAT5 [9]; and the Loos-Weispfenning (EFE M5lw) procedure, also built on MathSAT5. The resulting encodings are solved using Z3 and MathSAT5. Given that the encodings are written in SMT-LIB2 [5] language, it would be straightforward to use any modern SMT solver as a backend[11]. However, our purpose is to assess the encodings we propose, and not to compare the various SMT solvers. Z3 can be seen as a representative for solvers that support quantified theories, and MathSAT as representative for quantifier-free solvers. We expect other solvers (e.g. Yices [15], OpenSMT [8]) to exhibit a similar behavior. (See [4] for a recent summary on the performances of current state-of-the-art solvers.)

8.2 Experimental set-up

We used a set of randomly-generated benchmarks. Consistency problems are generated using the random instance generator presented in [2]. Strong controllability problems are generated by means of an extension of the same generator, where uncertainty is randomly introduced: each constraint generated by the consistency problem generator is turned in a contingent constraint with a given probability, and its destination node is considered as uncontrollable. The benchmark set contains 2108 simple-natured instances for each problem class in TP without uncertainty (STP, TCSP and DTP), and 1054 instances for each TPU class (STPU, TCSPU and DTPU). We used random instance generators because they are typically used in literature (e.g. [2]), and because they can be easily scaled to stress the solvers.

[11] In fact, the tool can also generate the benchmarks also in SMT-LIB1 [31] format.

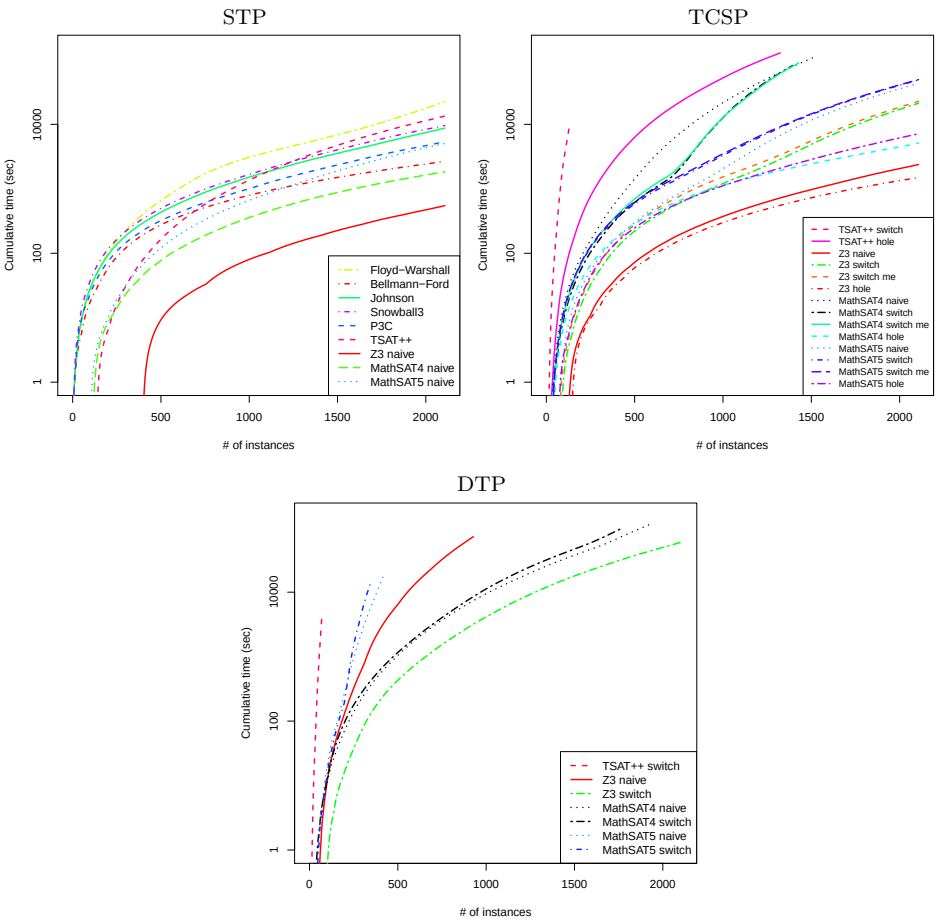

Fig. 5 Results for consistency experimental evaluation of STP (left-top), TCSP (right-top), and DTP (bottom).

We performed all our experiments on a machine running Scientific Linux 6.0, equipped with two quad-core Xeon processors @ 2.70GHz. We considered a memory limit of 2GB and a time-out of 300 seconds. The benchmarks and the results are available from `https://es.fbk.eu/people/roveri/tests/constraints2013`.

8.3 Results for Consistency

For consistency problems, we analyzed the performance of the various SMT solvers on the various encodings. We also compared our tool chain with the other available solvers for TP without uncertainty, namely the Snowball3 [30] tool, that implements many algorithms for the case of STP (i.e Floyd-Warshall, Bellman-Ford, Johnson and Snowball3), and TSAT++ [2], that is able to solve STP, TCSP and DTP problems.

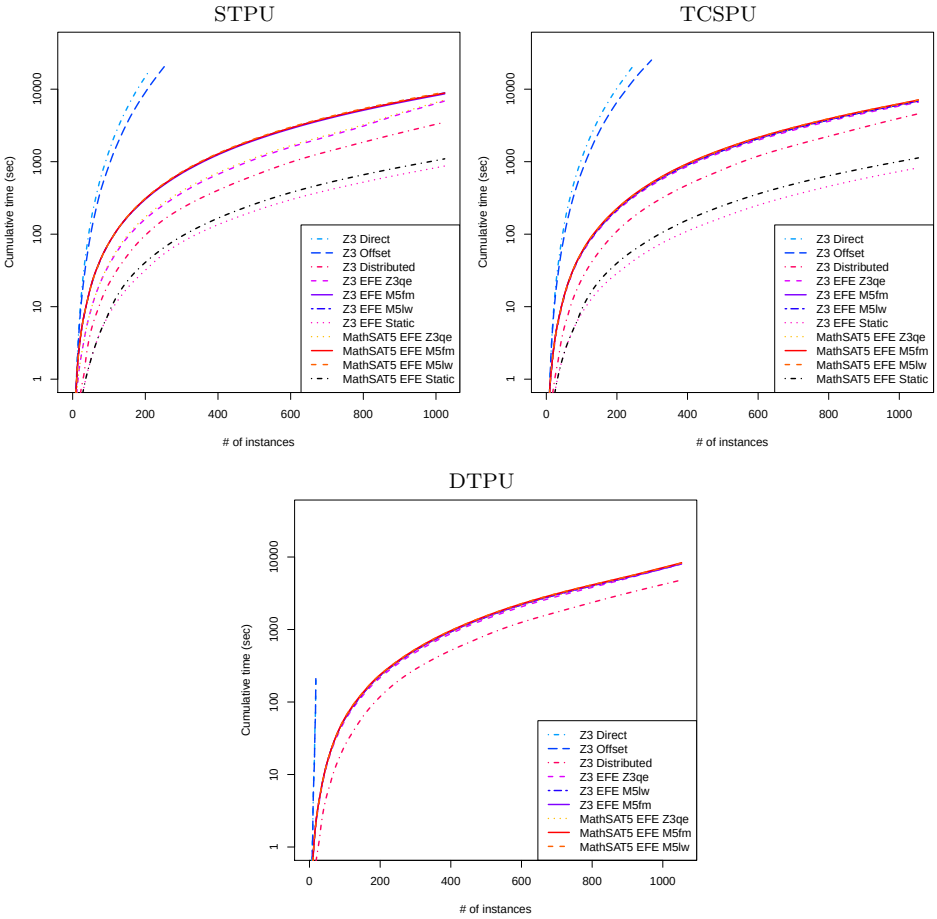

Fig. 6 Results for strong controllability experimental evaluation of STPU (left-top), TCSPU (right-top) and DTPU (bottom).

The results for consistency problems are reported in Figure 5. The cactus plot reports the number of solved instances in the horizontal axis and the cumulative time for each approach in logarithmic scale on the vertical axis. For example, MathSAT4 takes about 100 seconds to solve the easiest 500 STP instances. For STP problems, we compared the naïve encoding with various algorithms available in the SnowBall3 [30] tool, and with TSAT++ [2]. (In the case of STP, the other encodings coincide with the naïve encoding.) In TCSP and DTP, we tested all the applicable encodings with all the SMT solvers under analysis and with TSAT++. The plots show that the SMT approach is competitive with dedicated techniques. MathSAT4 implements a dedicated algorithm for the theory of difference logic [12], and is thus faster than MathSAT5, that uses a general purpose algorithm for LRA [16]. All solvers perform better on problems with Hole encoding. This encoding produces a formula that has just one real variable for every time point and has at most two literals per clause: this simplifies the SMT search procedure by augmenting the number of unit propagations, and by reducing the

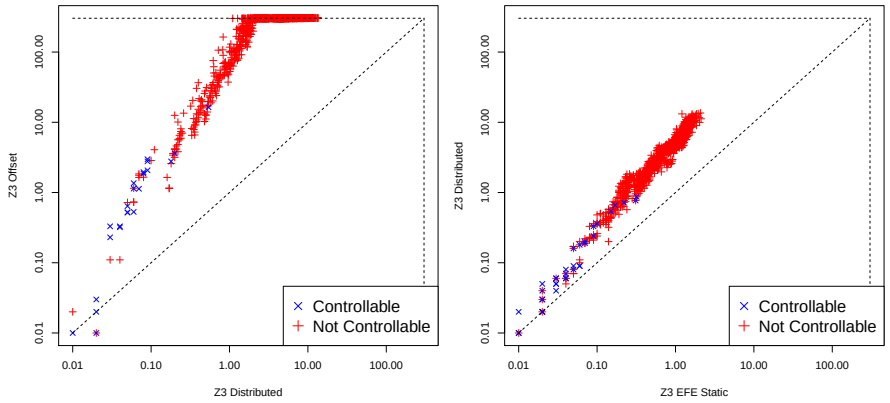

Fig. 7 Scatter plots of TCSPU solving time benchmarks obtained using the Z3 solver. The left plot shows the comparison of Offset and Distributed encodings, while the right plot contrasts the Distributed and the Static Eager For-all Elimination encodings. Controllable instances are marked with blue × signs, while not controllable instances are marked with red + signs.

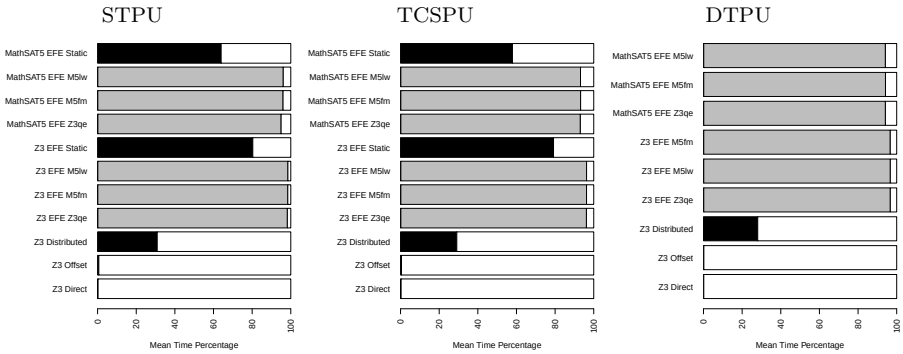

Fig. 8 Breakdown of computation time for the analyzed encodings for different classes of TPU: encoding time percentage (in black); quantifier elimination time percentage (in gray); solving time percentage (in white). In Eager For-all Elimination Static encodings the static quantification and the encoding time are considered together.

size of the search space. TSAT++ is outperformed by the other (more modern) SMT solvers.

8.4 Results for Strong Controllability

To the best of our knowledge, there are no available solvers for strong controllability problems. Thus, we evaluated the different approaches we presented, to highlight the difference in performance and the respective merits. The results for strong controllability are reported in Figure 6. We plotted in logarithmic scale the cumulative time in seconds to solve the considered set of benchmarks. Differently from the consistency case, the total time includes the encoding time, which may be significant in the case of quantifier-free encodings.

The plots show that the Offset and Direct encodings quickly reach the resource limits, and are unable to solve all the instances. The behavior of the Distributed encoding is slightly better than the eager for-all elimination approaches. The difference can be explained in purely technological terms: the quantifier elimination modules are called via pipe in our implementation, while Z3, on the Distributed encoding, performs quantifier elimination "in-memory".

We notice that the static quantification techniques (EFE Static), when applicable (i.e. for STPU and simple-natured TCSPU), yield a substantial improvement in performance: the expensive quantifier elimination step is avoided altogether.

In Figure 7, we report the scatter plots obtained comparing the performance of the Offset and Distributed encodings, and the Distributed and the EFE Static encodings using the Z3 solver. The plots show how the performances are affected by the encoding, in fact the Offset encoding is unable to solve the most complex instances. Moreover, we see that instances that are harder to solve are the ones that are not strongly controllable for all the tested encodings. In order to assess the real gap between the Offset and the Distributed encodings, we isolated three TCSPU benchmarks in which the Offset encoding timed out, and we tested them without time limits. Two benchmarks were solved in 1356.8 and 26353.2 seconds, while the third one was still running after 33249.8 seconds. This shows that the logical rewriting performed in the Distributed encoding yields a very significant performance improvement; in fact, the same benchmarks are solved by the Distributed encoding in 1.6, 3.46, and 10.92 seconds respectively. In turn, the static encoding yields a further speed-up (to 1.5, 3.02 and 9.12 seconds).

We also plotted the distribution of time consumption between encoding time, quantifier elimination and solving time (Figure 8). The plots highlight the fact that the quantification is the major issue in solving TPUs. The plot shows that the encoding time is absolutely negligible when quantifier elimination is applied, in fact the black part of the diagram is hardly visible. In EFE Static encodings we could not distinguish the quantifier elimination time from the encoding time because the elimination is performed together with the encoding. In approaches where the quantification is demanded to the solver, the vast majority of time is in the SMT solving, while in eager for-all elimination approaches the quantifier elimination dominates the solving time. The relatively high encoding time of Distributed encoding is mainly due to the time needed to printout the big output file in SMT-LIB format.

8.5 Comparison with PVYS

In this section we compare our approach with the PVYS algorithm [29]. To the best of our knowledge, no implementation is available. Therefore we implemented our own version of PVYS. The tool is written in Python, and exploits the MathSAT SMT solver to check the consistency of DTPs. The PVYS pseudo-code reported in [29] includes steps to compute the minimal network, and to check the consistency of DTPs constructed by the algorithm, but gives no details on how to push and intersect constraints. Thus, we implemented the same operations using the SMT technology.

The tool[12] has been implemented in two variants. The first one directly follows the original pseudo-code; the second one exploits the incrementality feature of the MathSAT SMT solver, to gain more efficiency: instead of checking the consistency of each problem separately, it reuses information derived from previous checks whenever possible.

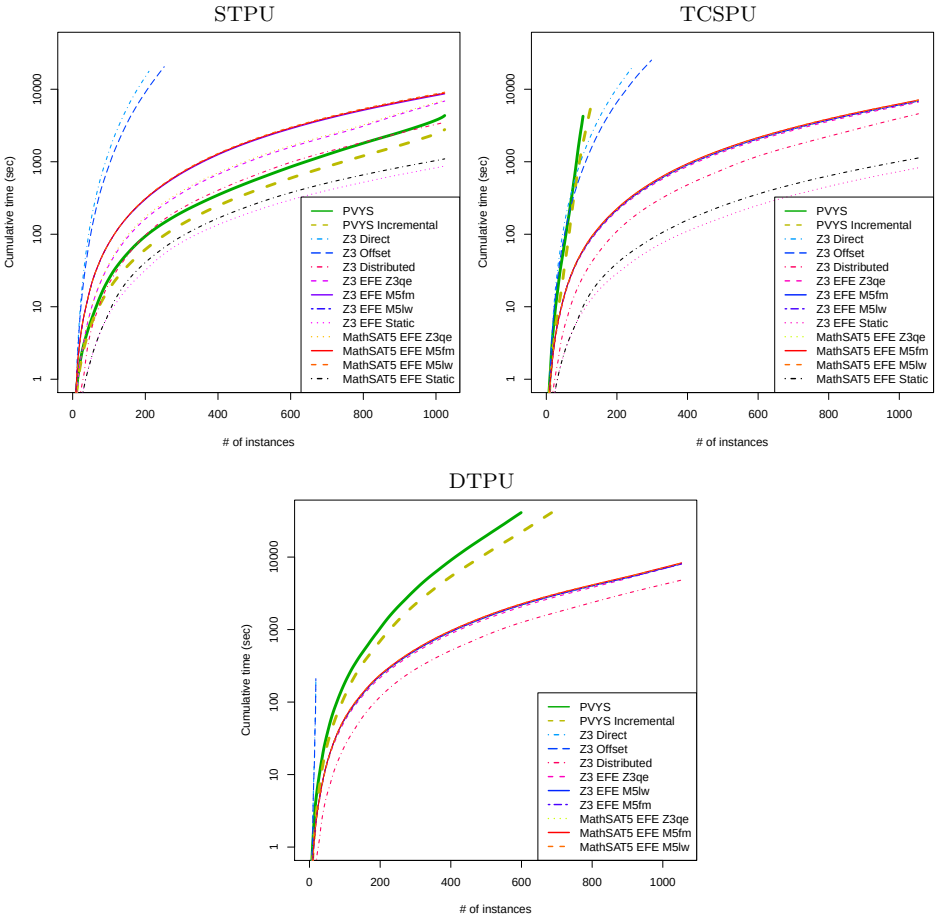

Fig. 9 Results for strong controllability using the PVYS implementation for the different classes of problems: STPU (left-top), TCSPU (right-top) and DTPU (bottom).

In Figure 9, we overlay the results achieved by PVYS in the same experimental conditions of Figure 6. In the STPU problem class, the results of PVYS are comparable to the SMT-based approaches. This is expected, because the implementation of PVYS uses the same SMT-based calls to FargierVidal. In the disjunctive cases, PVYS performs dramatically worse than the SMT-based approaches, due to the enumerative treatment of disjunctions. Finally, we notice that incrementality improves the performance to some extent.

[12] The tool is available for download from `https://es.fbk.eu/people/roveri/tests/constraints2013`.

9 Conclusions

In this paper, we presented a comprehensive approach to strong controllability for temporal problems with uncertainty. We considered the most complete class of problems, namely the DTPU class. We formalized the problem in the SMT framework by means of a number of encodings for the general case, and we also developed specialized approaches for the TCSPU subclass of problems. In our work we leveraged logic-based quantifier-elimination techniques to deal with temporal uncertainty. Our experiments demonstrate the scalability of the approach, based on the use of efficient SMT solvers.

In the future, we will investigate the problem of searching schedules that optimize a given cost function, and the addition of constraints over resources associated to activities. In addition, we will study the possibility of extending the DTPU formalism to model interdependence between contingent constraints. In fact, it is currently impossible to model assumptions (contingent constraints) involving different uncontrollable time-points. We believe that this extension can be trivially handled by our encodings as it amounts to additional constraints in $\Gamma(\vec{Y}_u)$. Finally, within the SMT-based framework, we will investigate the case of dynamic controllability.

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. Communication of the ACM **26**(11), 832–843 (1983)
2. Armando, A., Castellini, C., Giunchiglia, E.: SAT-Based Procedures for Temporal Reasoning. In: S. Biundo, M. Fox (eds.) European Conference on Planning - ECP, *LNCS*, vol. 1809, pp. 97–108. Springer (1999)
3. Bagnara, R., Hill, P.M., Zaffanella, E.: The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Science of Computer Programming **72**(1–2), 3–21 (2008)
4. Barrett, C., Deters, M., Moura, L., Oliveras, A., Stump, A.: 6 Years of SMT-COMP. Journal of Automated Reasoning **50**, 243–277 (2013)
5. Barrett, C., Stump, A., Tinelli, C., Boehme, S., Cok, D., Deharbe, D., Dutertre, B., Fontaine, P., Ganesh, V., Griggio, A., Grundy, J., Jackson, P., Oliveras, A., Krsti, S., Moskal, M., Moura, L.D., Sebastiani, R., Cok, T.D., Hoenicke, J.: The smt-lib standard: Version 2.0. Tech. rep. (2010)
6. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Satisfiability, pp. 825–885. IOS Press (2009)
7. Bruttomesso, R., Cimatti, A., Franzén, A., Griggio, A., Sebastiani, R.: The MathSAT 4 SMT solver. In: A. Gupta, S. Malik (eds.) Computer Aided Verification - CAV, *LNCS*, vol. 5123, pp. 299–303. Springer (2008)
8. Bruttomesso, R., Pek, E., Sharygina, N., Tsitovich, A.: The OpenSMT Solver. In: J. Esparza, R. Majumdar (eds.) Tools and Algorithms for the Construction and Analysis of Systems - TACAS, *LNCS*, vol. 6015, pp. 150–153. Springer (2010)
9. Cimatti, A., Griggio, A., Schaafsma, B.J., Sebastiani, R.: The MathSAT5 SMT solver. In: Tools and Algorithms for the Construction and Analysis of Systems - TACAS (2013)
10. Cimatti, A., Micheli, A., Roveri, M.: Solving temporal problems using smt: Strong controllability. In: CP, pp. 248–264 (2012)
11. Cimatti, A., Micheli, A., Roveri, M.: Solving Temporal Problems using SMT: Weak Controllability. In: J. Hoffmann, B. Selman (eds.) American Association for Artificial Intelligence - AAAI. AAAI Press (2012)
12. Cotton, S., Maler, O.: Fast and flexible difference constraint propagation for dpll(t). In: A. Biere, C.P. Gomes (eds.) Theory and Applications of Satisfiability Testing - SAT, *LNCS*, vol. 4121, pp. 170–183. Springer (2006)

13. Davis, M., Logemann, G., Loveland, D.W.: A machine program for theorem-proving. Communications of ACM **5**(7), 394–397 (1962)
14. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artif. Intell. **49**(1-3), 61–95 (1991)
15. Dutertre, B., de Moura, L.: The Yices SMT solver. Tool paper at `http://yices.csl.sri.com/tool-paper.pdf` (2006)
16. Dutertre, B., de Moura, L.M.: A Fast Linear-Arithmetic Solver for DPLL(T). In: T. Ball, R.B. Jones (eds.) Computer Aided Verification - CAV, *LNCS*, vol. 4144, pp. 81–94. Springer (2006)
17. Franzén, A., Cimatti, A., Nadel, A., Sebastiani, R., Shalev, J.: Applying SMT in symbolic execution of microcode. In: R. Bloem, N. Sharygina (eds.) Formal Methods in Computer-Aided Design - FMCAD, pp. 121–128. IEEE (2010)
18. Godefroid, P., Levin, M.Y., Molnar, D.A.: Automated whitebox fuzz testing. In: Network and Distributed System Security Symposium - NDSS. The Internet Society (2008)
19. Hunsberger, L.: A fast incremental algorithm for managing the execution of dynamically controllable temporal networks. In: TIME, pp. 121–128 (2010)
20. Keßler, C.W.: Parallel fourier-motzkin elimination. In: Euro-Par, Vol. II, pp. 66–71 (1996)
21. Kleene, S.: Mathematical Logic. J. Wiley & Sons (1967)
22. Loos, R., Weispfenning, V.: Applying linear quantifier elimination. Computer Journal **36**(5), 450–462 (1993)
23. Monniaux, D.: A Quantifier Elimination Algorithm for Linear Real Arithmetic. In: I. Cervesato, H. Veith, A. Voronkov (eds.) Logic for Programming, Artificial Intelligence, and Reasoning - LPAR, *LNCS*, vol. 5330, pp. 243–257. Springer (2008)
24. Morris, P.H., Muscettola, N., Vidal, T.: Dynamic control of plans with temporal uncertainty. In: B. Nebel (ed.) International Joint Conference on Artificial Intelligence - IJCAI, pp. 494–502. Morgan Kaufmann (2001)
25. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an Efficient SAT Solver. In: Design Automation Conference - DAC, pp. 530–535. ACM Press, New York, NY, USA (2001)
26. de Moura, L.M., Bjørner, N.: Z3: An Efficient SMT Solver. In: C.R. Ramakrishnan, J. Rehof (eds.) Tools and Algorithms for the Construction and Analysis of Systems - TACAS, *LNCS*, vol. 4963, pp. 337–340. Springer (2008)
27. Muscettola, N., Nayak, P.P., Pell, B., Williams, B.C.: Remote agent: To boldly go where no ai system has gone before. Artificial Intelligence **103**(1-2), 5–47 (1998)
28. Niemelä, I.: Integrating Answer Set Programming and Satisfiability Modulo Theories. In: E. Erdem, F. Lin, T. Schaub (eds.) Logic Programming and Nonmonotonic Reasoning, 10th International Conference - LPNMR, *LNCS*, vol. 5753, p. 3. Springer (2009)
29. Peintner, B., Venable, K.B., Yorke-Smith, N.: Strong controllability of disjunctive temporal problems with uncertainty. In: C. Bessiere (ed.) Principles and Practice of Constraint Programming - CP, *LNCS*, vol. 4741, pp. 856–863. Springer (2007)
30. Planken, L., de Weerdt, M., van der Krogt, R.: Computing all-pairs shortest paths by leveraging low treewidth. Journal of Artificial Intelligence Research (JAIR) **43**, 353–388 (2012)
31. Ranise, S., Loria, Tinelli, C.: The smt-lib standard: Version 1.2. Tech. rep. (2006)
32. Schrijver, A.: Theory of Linear and Integer Programming. J. Wiley & Sons (1998)
33. de la Tour, T.: Minimizing the number of clauses by renaming. In: M. Stickel (ed.) Conference on Automated Deduction - CADE, *LNCS*, vol. 449, pp. 558–572. Springer (1990)
34. Tsamardinos, I., Pollack, M.E.: Efficient solution techniques for disjunctive temporal reasoning problems. Artificial Intelligence **151**(12), 43 – 89 (2003)
35. Venable, K.B., Volpato, M., Peintner, B., Yorke-Smith, N.: Weak and dynamic controllability of temporal problems with disjunctions and uncertainty. In: Workshop on Constraint Satisfaction Techniques for Planning & Scheduling, pp. 50–59 (2010)
36. Vidal, T., Fargier, H.: Handling contingency in temporal constraint networks: from consistency to controllabilities. Journal of Experimental Theoretical Artificial Intelligence **11**(1), 23–45 (1999)
37. Wilde, D.K.: A library for doing polyhedral operations. Tech. rep. (1993)