Faster and Better Simple Temporal Problems

Dario Ostuni[1], Alice Raffaele[2], Romeo Rizzi[1], Matteo Zavatteri[1]*

[1]University of Verona, Department of Computer Science, Strada Le Grazie 15, 37134 Verona (Italy)
[2]University of Trento, Department of Mathematics, Via Sommarive 14, 38123 Povo (Italy)
{dario.ostuni, romeo.rizzi, matteo.zavatteri}@univr.it, alice.raffaele@unitn.it

Abstract

In this paper we give a structural characterization and extend the *tractability frontier* of the Simple Temporal Problem (STP) by defining the class of the *Extended Simple Temporal Problem* (ESTP), which augments STP with strict inequalities and monotone Boolean formulae on inequations (i.e., formulae involving the operations of conjunction, disjunction and parenthesization). A polynomial-time algorithm is provided to solve ESTP, faster than previous state-of-the-art algorithms for other extensions of STP that had been considered in the literature, all encompassed by ESTP. We show the practical competitiveness of our approach through a proof-of-concept implementation and an experimental evaluation involving also state-of-the-art SMT solvers.

Introduction and Related Work

The *Temporal Constraint Satisfaction Problem* (TCSP), originally introduced by (Dechter, Meiri, and Pearl 1991), takes as input a finite set of real variables and a finite set of constraints. Two kinds of constraints are allowed: unary and binary; both can take a disjunctive form. A unary constraint has the form $x \in [\ell_1, u_1] \vee \cdots \vee [\ell_n, u_n]$, where x is a variable and $\ell_i, u_i \in \mathbb{R} \cup \{-\infty, +\infty\}$, $-\infty \leq \ell_i \leq u_i \leq \infty$. A binary constraint has the form $y - x \in [\ell_1, u_1] \vee \cdots \vee [\ell_n, u_n]$, where x and y are variables and, again, $\ell_i, u_i \in \mathbb{R} \cup \{-\infty, +\infty\}$, $-\infty \leq \ell_i \leq u_i \leq \infty$. Regardless of the type of constraint, the corresponding involved intervals are disjoint. Note that each unary constraint $x \in [\ell_1, u_1] \vee \cdots \vee [\ell_n, u_n]$ can be seen as a binary one $x - z \in [\ell_1, u_1] \vee \cdots \vee [\ell_n, u_n]$ where z is an extra variable on which we impose the unary constraint $z \in [0, 0]$. As such, one single unary constraint suffices. Or, we can even relax that one and then consider the solution obtained by subtracting the value of z from that of every other variable. Indeed, the solution space of a TCSP without unary constraints is closed under rigid shifting. We remark that binary constraints are disjunctions over the same pair of variables (this restriction is overcome in (Stergiou and Koubarakis 2000) with the proposal of disjunctive temporal networks). TCSP asks the following question: *does there exist an assignment of real values to the variables such that all constraints are satisfied?* TCSP is NP-complete (Dechter, Meiri, and Pearl 1991).

The *Simple Temporal Problem* (STP) is the fragment of TCSP whose set of constraints consists of atoms of the form $y - x \in [\ell, u]$ only. Historically, STP allows for two representations: the interval based one (as we just discussed above) or a set of inequalities $y - x \leq k$ where x, y are variables and $k \in \mathbb{R}$. While these are equivalent ($y - x \leq k$ can be written as $y - x \in [-\infty, k]$), the second representation is more elementary as $y - x \in [\ell, u]$ amounts to $(y - x \leq u) \wedge (x - y \leq -\ell)$. STP is in P (Dechter, Meiri, and Pearl 1991).

The set of constraints of an STP can also be generalized to include strict inequalities $y - x < k$ and inequations $y - x \neq k$. Of course, any strict inequality $y - x < k$ can be seen as a non-strict inequality $y - x \leq k$ plus the inequation $y - x \neq k$. Inequations are useful to model temporal plans in which, for example, we require two events not to happen at the same time. Koubarakis augmented STP with disjunctions on inequations $(y_1 - x_1 \neq k_1) \vee \cdots \vee (y_n - x_n \neq k_n)$ in (Koubarakis 1992). He proved tractability of this extension by providing an algorithm that runs in $\mathcal{O}(||C||^4)$, where C is the set of constraints and $||C||$ is the number of atoms $y - x \bowtie k$, $\bowtie \in \{\leq, \neq\}$ appearing in C. As far as we know, this was still the bound for this problem.

Gerevini and Cristani defined STP$^{\neq}$, which is a restriction of (Koubarakis 1992) allowing only simple inequations and not disjunctions of these, resulting in the possibility of handling both sets of strict-inequalities and inequations (Gerevini and Cristani 1997). They provided an algorithm to solve STP$^{\neq}$ that runs in $\mathcal{O}(n^3 + h)$, where n is the number of variables and h is the number of inequations (or $\mathcal{O}(n^3)$ if there are only strict and non-strict inequalities). As far as we know, the result in (Gerevini and Cristani 1997) was the state of the art for STP$^{\neq}$, still used by more recent works (e.g., (Broxvall 2002; Montanari et al. 2012; Cooper, Maris, and Régnier 2013; Carbonnel and Cooper 2015)).

For more details about temporal reasoning, the interested reader is referred to (Vila 1994; Schwalb and Vila 1998; Bartak, Morris, and Venable 2014).

Satisfiability Modulo Theory (SMT) (Barrett et al. 2009) can address generalizations of the problems mentioned above by relying on the quantifier-free fragment of the theory of real difference logic (QF_RDL). QF_RDL allows for

*Corresponding author

arbitrary Boolean combinations of atoms $y - x \bowtie k$, where x, y are real variables, $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$, and $k \in \mathbb{Q}$.

In this logic, strict inequalities $y - x < k$ are usually turned into non-strict inequalities $y - x \leq k - \varepsilon$, with ε treated either numerically (Armando et al. 2005) or symbolically as an infinitesimal parameter (Dutertre and de Moura 2006). As far as we know, the latter is the state of the art on difference arithmetic constraints (Dutertre and de Moura 2006; de Moura, Dutertre, and Shankar 2007).

Note that the core algorithms to solve the problems mentioned above are typically based on tuned versions of shortest path algorithms or the simplex method, e.g., (Dutertre 2014). It is thus interesting and reasonable, e.g., when developing a more efficient procedure to solve STP with strict inequalities, to also make a comparison with SMT solvers supporting QF_RDL.

Organization and Contribution

We start by giving background on STP. Then, we define the *Extended Simple Temporal Problem* (ESTP), in which we allow for inequalities, strict inequalities and monotone Boolean formulae on inequations. We give a characterization of ESTP. Specifically, we define verification criteria to detect inconsistency or decide consistency of any instance of ESTP. These allow us to provide a strongly polynomial-time algorithm to solve ESTP, which is faster than current state-of-the-art algorithms for previously studied extended classes of STP. We discuss a proof-of-concept implementation by comparing several variants of shortest paths algorithms. Moreover, we encode our model as a QF_RDL formula since it is well-known that the satisfiability of conjunctions of atoms $y - x \bowtie k$ for $\bowtie \in \{\leq, <\}$ is tractable. We evaluate our performance against the state-of-the-art SMT solvers that competed in the *15th International Satisfiability Modulo Theories Competition* — SMT-COMP 2020 — for QF_RDL (Model Validation Track and Single Query Track)[1]. Computational results confirm the practical competitiveness of our approach, which can solve instances close to 3 million variables within 100 seconds. Finally, we draw conclusions and discuss future work.

Background

In this section we sum up essential information on STP and how to efficiently solve it through a reduction to the *Single Source Shortest Paths* problem (SSSP) (Cormen et al. 2009).

Problem 1 (STP)**.** Given

- a finite set X of real variables, and
- a finite set C of inequalities of the form $y - x \leq k$ with $y, x \in X$ and $k \in \mathbb{R}$,

does there exist a mapping $\delta\colon X \mapsto \mathbb{R}$ satisfying all constraints in C?

An instance of STP is *consistent* if such a solution δ exists; *inconsistent* otherwise. We assume that any instance of STP appearing in this paper has no dominated inequalities, where an inequality $y - x \leq k$ dominates an inequality $y - x \leq k'$ if $k < k'$.

[1] https://smt-comp.github.io/2020/index.html

Example 1. Let $S = (X, C)$ be the following instance of STP: $X := \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$,

$$C := \{x_2 - x_1 \leq -2.2, x_3 - x_2 \leq -3.5, x_1 - x_3 \leq 5.7,\\ x_4 - x_5 \leq 2, x_5 - x_7 \leq 1, x_7 - x_6 \leq 6,\\ x_6 - x_4 \leq -9, x_6 - x_3 \leq -3.3, x_2 - x_4 \leq -2\}.$$

Most of the algorithms that solve instances of STP rely on their corresponding directed weighted graph representation.

Definition 1 (Directed weighted graph)**.** A *directed graph* is a pair (V, A) where V is a set of nodes and $A \subseteq V \times V$ is a set of *arcs* (or, equivalently, *directed edges*). A *directed weighted graph* is a triple (V, A, w) where (V, A) is a directed graph and $w\colon A \mapsto \mathbb{R}$ is a weight function assigning a real value to each arc.

We write $P_{x,y} := \langle (x, x_1), \ldots, (x_n, y) \rangle$ for the *path* (sequence of arcs) going from x to y through a sequence of distinct nodes. A *cycle* is a path $P_{x,y}$ where $x = y$. In case of directed weighted graphs, we write $w(P_{x,y})$ for the weight of the path (i.e., the sum of the weights of the arcs in the sequence). A path or a cycle $P_{x,y}$ in a weighted graph is *negative* if $w(P_{x,y}) < 0$.

Definition 2 (Constraint graph)**.** Let $S = (X, C)$ be an instance of STP. The corresponding constraint graph of S is the directed weighted graph $G_S = (X, A, w)$ where:

- $A := \{(x, y) \mid y - x \leq k \in C\}$, and
- $w(x, y) := k$ for each $y - x \leq k \in C$.

Figure 1 shows the constraint graph of Example 1.

Theorem 1 ((Dechter, Meiri, and Pearl 1991))**.** An instance of STP is consistent iff G_S has no negative-weight cycle.

Deciding whether an instance of STP $S = (X, C)$ is consistent can be done by computing a *potential function* for G_S, i.e., a function $\pi\colon X \mapsto \mathbb{R}$ such that $\pi(y) - \pi(x) \leq w(x, y)$ for each $(x, y) \in A$. This feasible potential is a possible solution δ and can be computed by a run of any SSSP algorithm. Since we deal with temporal problems, when unary constraints are not involved, it is also reasonable to seek a non-negative π; a guarantee that, unfortunately, we do not get by just computing a feasible potential. However, this is not an issue because potential functions are *shift-invariant* (Cormen et al. 2009). That is, for any feasible potential function π and any $\sigma \in \mathbb{R}$, it holds that the function defined as $\pi'(x) := \pi(x) + \sigma$ for each $x \in X$ is still a feasible potential function. Indeed, for any $(x, y) \in A$:

$$\begin{aligned} &\pi(y) - \pi(x) \leq w(x, y)\\ &\pi(y) - \pi(x) + \sigma \leq w(x, y) + \sigma \qquad (\text{for any } \sigma \in \mathbb{R})\\ &(\pi(y) + \sigma) - (\pi(x) + \sigma) \leq w(x, y)\\ &\pi'(y) - \pi'(x) \leq w(x, y). \end{aligned}$$

Example 1 is consistent. Indeed, a potential function π for its constraint graph is shown in Figure 1. Thus, a solution is $\delta(x_1) := \pi(x_1) = 0$, $\delta(x_2) := \pi(x_2) = -2.2$, $\delta(x_3) := \pi(x_3) = -5.7$, $\delta(x_4) := \pi(x_4) = 0$, $\delta(x_5) := \pi(x_5) = -2$, $\delta(x_6) := \pi(x_6) = -9$, and $\delta(x_7) : \pi(x_7) = -3$. In case we desire δ to be non-negative, we define $\delta(x) := \pi(x) + \sigma$ for each $x \in X$, where $\sigma := |\min_{x \in X} \pi(x)| = 9$.

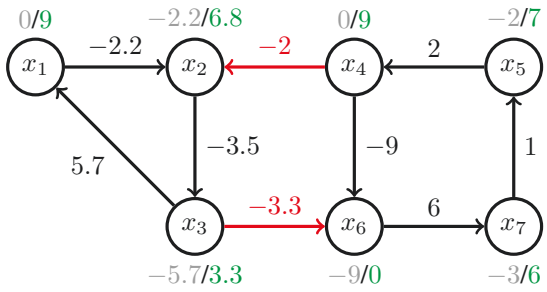


Figure 1: Constraint graph of Example 1 (ignoring arc colors) and Example 2 (considering red arcs too). Near the nodes, there are two labels, $\pi/\pi+9$: π (in gray) is the value of the feasible potential taking x_1 as a source, whereas $\pi+9$ (in green) is the corresponding rigidly shifted value. We do not draw formulae on inequations to keep the figure clear.

Extended Simple Temporal Problem

In (Koubarakis 1992), STP was extended to also handle disjunctions on inequations. The definition of *Extended STP* (ESTP) generalizes (Koubarakis 1992) by allowing arbitrary monotone Boolean formulae on inequations, rather than only disjunctions.

Problem 2 (ESTP)**.** Given

- a finite set X of real variables,
- a finite set $C_{\leq}$ of inequalities of the form $y - x \leq k$ with $y, x \in X$ and $k \in \mathbb{R}$,
- a finite set $C_{<}$ of strict inequalities of the form $y - x < k$ with $y, x \in X$ and $k \in \mathbb{R}$, and
- a finite set $C_{\neq}$ of monotone Boolean formulae on inequations generated by the grammar $F ::= y - x \neq k \mid F \wedge F \mid F \vee F \mid (F)$ with $y, x \in X$ and $k \in \mathbb{R}$,

does there exist a mapping $\delta \colon X \mapsto \mathbb{R}$ satisfying all constraints in $C_{\leq}$, $C_{<}$ and $C_{\neq}$?

Similarly to STP, an instance of ESTP is *consistent* if such a solution δ exists; *inconsistent* otherwise.

Let $S = (X, C_{\leq}, C_{<}, C_{\neq})$ be any instance of ESTP. We write $y - x \bowtie k$ for a generic atom appearing in some constraint, where $\bowtie \in \{\leq, <, \neq\}$. We denote by $\mathcal{A}_S$ the set of all atoms appearing in S and, similarly, we denote by $\mathcal{A}_F := \{y - x \neq k \mid y - x \neq k \in F\} \subseteq \mathcal{A}_S$ the set of atoms of any $F \in C_{\neq}$.

Moreover, we just write *inequality* when we do not care if it is strict or not, and we still assume that any instance of ESTP appearing in this paper has no dominated inequalities, where, this time, an inequality $y - x \bowtie k$ dominates an inequality $y - x \bowtie' k'$ if either $k < k'$, or $\bowtie$ is $<$ and $\bowtie'$ is $\leq$ and $k = k'$.

Example 2. Let $S = (X, C_{\leq}, C_{<}, C_{\neq})$ be the following instance of ESTP: $X := \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$,

$$C_{\leq} := \{x_2 - x_1 \leq -2.2, x_3 - x_2 \leq -3.5, x_1 - x_3 \leq 5.7, \\ x_4 - x_5 \leq 2, x_5 - x_7 \leq 1, x_7 - x_6 \leq 6, x_6 - x_4 \leq -9\}$$
$$C_{<} := \{x_6 - x_3 < -3.3, x_2 - x_4 < -2\}$$
$$C_{\neq} := \{x_6 - x_1 \neq -9 \wedge (x_4 - x_7 \neq 3 \vee x_6 - x_5 \neq -7.14)\}.$$

One could fairly wonder whether $C_{\leq}$ and $C_{\neq}$ together subsume $C_{<}$. It is true; indeed, any strict inequality of the form $y - x < k$ can be rewritten as a classic inequality $y - x \leq k$ plus an inequation $y - x \neq k$. The reason we also support strict inequalities is because we want to explore their structure without masquerading them as the composition of different parts.

We now discuss three independent conditions to identify inconsistent instances of ESTP. It will later turn out that any instance of ESTP not affected by any of these conditions is consistent. The notion of relaxation of an ESTP instance helps to state these conditions.

Definition 3 (Relaxation of an ESTP instance)**.** Let $S = (X, C_{\leq}, C_{<}, C_{\neq})$ be an instance of ESTP. The relaxation of S is the following instance of STP $R = (X_R, C_R)$, where $C_R := C_{\leq} \cup \{y - x \leq k \mid y - x < k \in C_{<}\}$.

In other words, the relaxation of S keeps all non-strict inequalities and turns each strict inequality into a non-strict inequality. Note that, by construction, R cannot have dominated inequalities either. The relaxation of the instance in Example 2 is $R = (X_R, C_R)$, where X_R is the same of Example 2 and C_R is exactly the set C of Example 1.

Lemma 1. Let S be an instance of ESTP and let G_R be the constraint graph of its relaxation. If G_R has a *negative-weight cycle*, then S is inconsistent.

Proof. A solution for S would also be a solution for R. This cannot exist if G_R has a negative-weight cycle (by Theorem 1). □

Figure 1 does not contain any negative-weight cycle.

Definition 4 (Arc color)**.** Let $S = (X, C_{\leq}, C_{<}, C_{\neq})$ be an instance of ESTP and $G_R = (X, A, w)$ be the constraint graph of its relaxation. We say that an arc (x, y) in G_R is *red* if S contains the strict inequality $y - x < w(x, y)$. A path $P_{x,y}$ is *red* if it contains a red arc.

That is, given an instance of ESTP S, the red arcs in the constraint graph of its relaxation G_R identify exactly strict inequalities of S. In Figure 1, (x_3, x_6) and (x_4, x_2) are red since $\{x_6 - x_3 < -3.3, x_2 - x_4 < -2\} \subseteq C_{<}$ in Example 2. Red arcs allow for the definition of a condition to discover other inconsistent instances of ESTP. To the best of our knowledge, this condition has never been identified before.

Lemma 2. Let S be an instance of ESTP and let G_R be the constraint graph of its relaxation. If G_R has a *zero-weight cycle containing a red arc*, then S is inconsistent.

Proof. Assume that S admits a solution δ and let $c := \langle (x_1, x_2), \ldots, (x_e, x_{e+1} = x_1) \rangle$ be a zero-weight cycle in G_R where (x_1, x_2) is a red arc. Then $\delta(x_2) - \delta(x_1) < w(x_1, x_2)$ and $\delta(x_{i+1}) - \delta(x_i) \leq w(x_i, x_{i+1})$, for every $i = 2, \ldots, e$. Summing up these inequalities (one for every arc in c), the valuations of δ on the e nodes simplify away and we end up with the contradiction $0 < 0$. □

Figure 1 contains two zero-weight cycles: $c_1 := \langle (x_1, x_2), (x_2, x_3), (x_3, x1) \rangle$ and $c_2 := \langle (x_4, x_6), (x_6, x_7), (x_7, x_5), (x_5, x4) \rangle$. Anyway, none of them contains red arcs.

We now provide the third (and last) condition to identify inconsistent instances of ESTP.

Definition 5 (Hopeless formula)**.** Let $S = (X, C_{\leq}, C_{<}, C_{\neq})$ be an instance of ESTP and G_R be the constraint graph of its relaxation. A monotone formula $F \in C_{\neq}$ is said *hopeless* if F is **false** under any Boolean evaluation that assigns **false** to all $y - x \neq k \in \mathcal{A}_F$ for which G_R contains the two non-red weighted paths $w(P_{x,y}) = k$ and $w(P_{y,x}) = -k$.

Lemma 3. Let S be an instance of ESTP. If S *has a hopeless formula*, then S is inconsistent.

Proof. Let $S = (X, C_{\leq}, C_{<}, C_{\neq})$ be an instance of ESTP and F be a hopeless formula. Let $\mathcal{A}'_F$ be the set of atoms $y - x \neq k$ of F for which G_R contains the two non-red weighted paths $w(P_{x,y}) = k$ and $w(P_{y,x}) = -k$. Suppose that S is consistent and let δ be a solution for it. Then, δ must satisfy F. However, for each atom in $\mathcal{A}'_F$ we have that $\delta(y) - \delta(x) = k$ because both $\delta(y) - \delta(x) \leq k$ and $\delta(x) - \delta(y) \leq -k$ hold. Therefore, none of these atoms can be satisfied by δ. But then, by definition of hopeless, δ does not satisfy F either because, even if δ satisfies all remaining atoms — and that is exactly why this check can be done in polynomial time — F is still **false** under such an interpretation (contradiction). □

Consider the unique formula on inequations in Example 2: $x_4 - x_7 \neq 3$ is the unique atom for which Figure 1 contains the two non-red weighted paths $w(\langle (x_7, x_5), (x_5, x_4) \rangle) = 3$ and $w(\langle (x_4, x_6), (x_6, x_7) \rangle) = -3$. Thus, any possible solution δ (if any exists) will not satisfy that atom. Anyway, that formula is not hopeless since its remaining atoms could still make it **true**.

To sum up, an inconsistency certificate for an instance S of ESTP is either a negative-weight cycle in G_R or a zero-weight cycle containing a red arc in G_R or a hopeless formula in S. We are finally in position to define consistency.

Theorem 2. An ESTP instance is consistent iff none of the conditions of Lemma 1, Lemma 2 and Lemma 3 applies.

Proof. Let $S = (X, C_{\leq}, C_{<}, C_{\neq})$ be an instance of ESTP.

($\Rightarrow$) If S is consistent, then none of the conditions of Lemma 1, Lemma 2 and Lemma 3 applies. In fact, if any did, S would be inconsistent.

($\Leftarrow$) If none of the conditions of Lemma 1, Lemma 2 and Lemma 3 applies, then S is consistent. To prove it, we build a mapping $\delta \colon X \mapsto \mathbb{R}$ as follows:

1. Let π be a feasible potential for the constraint graph G_R of the relaxation of S.
2. Let $G'_R := (X, A, w')$ be the reweighted graph obtained from G_R where $w'(x, y) := w(x, y) + \pi(x) - \pi(y)$ for each $(x, y) \in A$.
3. Let $A_0 := \{(x, y) \mid (x, y) \in A, w'(x, y) = 0\}$ be the subset of zero-weight arcs of G'_R and let $D_0 := (X, A_0)$ be the corresponding directed acyclic graph (DAG). Note that a cycle in D_0 is a zero-weight cycle in both G_R and G'_R, since the reweighting operation maintains shortest-paths and cycle weights.
4. Let $\mathcal{P} := \{X_1, \ldots, X_{|\mathcal{P}|}\}$ be the partition of X coinciding with the set of strongly connected components (SCCs) of D_0. Note that each SCC is a rigid component. That is, a set of nodes such that, once the value of one of these nodes is fixed, so are the values of all the other nodes in the same component.
5. Let $D_{\mathcal{P}} := (\mathcal{P}, A_{\mathcal{P}})$ be the DAG of the SCCs of D where $A_{\mathcal{P}} := \{(X_i, X_j) \mid X_i, X_j \in \mathcal{P}, X_i \neq X_j, \exists (x, y) \in A_0, x \in X_i, y \in X_j\}$.
6. Let $\varepsilon := \dfrac{\min\left(1, \min\limits_{\substack{y - x \bowtie k \in \mathcal{A}_S, \\ |k + \pi(x) - \pi(y)| > 0}} |k + \pi(x) - \pi(y)|\right)}{|X|}$.
7. Let $T := \langle \overline{X}_1, \ldots, \overline{X}_{|\mathcal{P}|} \rangle$ be any topological sort of $D_{\mathcal{P}}$.
8. Define δ as follows:

$$\begin{aligned}
&\delta(x) := \pi(x) + 0\varepsilon && \text{for each } x \in \overline{X}_{|\mathcal{P}|} \\
&\delta(x) := \pi(x) + 1\varepsilon && \text{for each } x \in \overline{X}_{|\mathcal{P}|-1} \\
&\ldots && \ldots \\
&\delta(x) := \pi(x) + (|\mathcal{P}| - 1)\varepsilon && \text{for each } x \in \overline{X}_1.
\end{aligned}$$

We claim that δ is a solution for S. Note that we do not need to check atomic constraints involving variables in the same SCC. Indeed, for each $y - x \bowtie k \in \mathcal{A}_S$ with x, y in the i-th SCC, by letting $\varepsilon_i := (|\mathcal{P}| - i)\varepsilon$, it holds that:

$$\begin{aligned}
\pi(y) - \pi(x) \bowtie k &\Leftrightarrow \pi(y) - \pi(x) + \varepsilon_i \bowtie k + \varepsilon_i \\
&\Leftrightarrow (\pi(y) + \varepsilon_i) - (\pi(x) + \varepsilon_i) \bowtie k \\
&\Leftrightarrow \delta(y) - \delta(x) \bowtie k.
\end{aligned}$$

Notice that, due to Lemma 1 and Lemma 2, if $\bowtie$ is $\leq$ or $<$, then the atom always holds. Instead, if $\bowtie$ is $\neq$, then the atom might also not be satisfied; in such a case it must belong to some formula F which has some other atom satisfied (otherwise Lemma 3 would have applied).

Therefore, what we are really left to prove is that any atom $y - x \bowtie k \in \mathcal{A}_S$, with y and x belonging to two *different* SCCs, is satisfied by δ. Let X_i and X_j be two different SCCs with $i < j$ in the topological sort T. Let $\varepsilon_i := (|\mathcal{P}| - i)\varepsilon$ and $\varepsilon_j := (|\mathcal{P}| - j)\varepsilon$ be the distances of δ from π for each $x \in X_i$ and for each $y \in X_j$, respectively.

For each type of atom (i.e., for each $\bowtie$), we consider its two possible orientations: the right one from X_i to X_j and the left one from X_j to X_i. We rely on the fact that $\varepsilon_j < \varepsilon_i$, since $i < j$. We start by considering the right oriented atoms, i.e., those of the form $y - x \bowtie k$.

Case $y - x \leq k$**.** We highlight that $\pi(y) - \pi(x) \leq k$ holds. If we substitute $\delta(y) - \pi(y)$ for ε_j and $\delta(x) - \pi(x)$ for ε_i, we obtain $\delta(y) - \pi(y) < \delta(x) - \pi(x)$ and thus $\delta(y) - \delta(x) < \pi(y) - \pi(x) \leq k$.

Case $y - x < k$**.** Same of **Case** $y - x \leq k$.

Case $y - x \neq k$**.** We have two cases:

- If $\pi(y) - \pi(x) = k$, then, by substituting $\delta(y) - \pi(y)$ for ε_j and $\delta(x) - \pi(x)$ for ε_i, we obtain $\delta(y) - \pi(y) < \delta(x) - \pi(x)$, implying $\delta(y) - \delta(x) < \pi(y) - \pi(x) = k$.

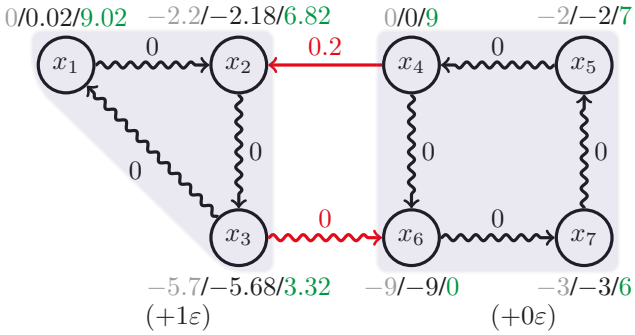


Figure 2: Graphical construction of the solution δ for the instance in Example 2. Node labels have the form $\pi/\delta/\delta+9$, where π (in gray) is the feasible potential, δ (in black) is the solution and $\delta + 9$ (in green) is the rigidly shifted value.

- Otherwise $\pi(y)-\pi(x) \neq k$; since $|k+\pi(x)-\pi(y)| > 0$ and $\varepsilon_j-\varepsilon_i < 0$, it holds that $\varepsilon_j-\varepsilon_i < |k+\pi(x)-\pi(y)|$. By substituting $\delta(y) - \pi(y)$ for ε_j and $\delta(x) - \pi(x)$ for ε_i, we get $\delta(y)-\pi(y)-\delta(x)+\pi(x) < |k+\pi(x)-\pi(y)|$. By solving the modulus, we get that:
  - If $k + \pi(x) - \pi(y) > 0$, then $\delta(y) - \delta(x) < k$.
  - Otherwise $k + \pi(x) - \pi(y) < 0$, and then $\delta(y) - \delta(x) < -k + 2(\pi(y) - \pi(x)) \neq k$.

Then, we consider the left oriented atoms, i.e., $x - y \bowtie k$.

Case $x - y \leq k$**.** We highlight that $\pi(x) - \pi(y) < k$. In fact, if $\pi(x) - \pi(y)$ were equal to k, there would exist a zero-weight arc (y, x) in the reweighted graph G'_R and therefore an arc (X_j, X_i) in the DAG of the SCCs $D_{\mathcal{P}}$, contradicting the topological sort T. Since $k + \pi(y) - \pi(x) > 0$ and $\varepsilon_j - \varepsilon_i < 0$, it holds that $\varepsilon_j - \varepsilon_i < k + \pi(y)-\pi(x)$. By substituting $\delta(x)-\pi(x)$ for ε_i and $\delta(y)-\pi(y)$ for ε_j, we get $\delta(x)-\pi(x)-\delta(y)+\pi(y) < k+\pi(y)-\pi(x)$ which implies $\delta(x) - \delta(y) < k$.

Case $x - y < k$**.** Same of **Case** $x - y \leq k$ (once again $\pi(x) - \pi(y) = k$ cannot hold, otherwise the topological sort T would be contradicted).

Case $x - y \neq k$**.** We have two cases:

- If $\pi(x) - \pi(y) = k$, then by substituting $\delta(x) - \pi(x)$ for ε_i and $\delta(y) - \pi(y)$ for ε_j, we obtain $\delta(y) - \pi(y) < \delta(x) - \pi(x)$ implying $\delta(x) - \delta(y) > \pi(x) - \pi(y) = k$.
- Otherwise $\pi(x)-\pi(y) \neq k$; since $|k+\pi(y)-\pi(x)| > 0$ and $\varepsilon_j-\varepsilon_i < 0$, it holds that $\varepsilon_j-\varepsilon_i < |k+\pi(y)-\pi(x)|$. By substituting $\delta(y) - \pi(y)$ for ε_i and $\delta(x) - \pi(x)$ for ε_j we get $\delta(y)-\pi(y)-\delta(x)+\pi(x) < |k+\pi(y)-\pi(x)|$. By solving the modulus, we get that:
  - If $k + \pi(y) - \pi(x) > 0$, then $\delta(y) - \delta(x) < k + 2(\pi(y)-\pi(x))$ and thus $\delta(x)-\delta(y) > -k+2(\pi(x)-\pi(y)) \neq k$.
  - Otherwise $k - \pi(y) + \pi(x) < 0$, then $\delta(y) - \delta(x) < -k$ and thus $\delta(x) - \delta(y) > k$.

□

Algorithm 1: ESTP solver

Input: An instance $S = (X, C_{\leq}, C_{<}, C_{\neq})$ of ESTP.
Output: A solution δ if S is consistent; inconsistent otherwise.

```
if Lemma 1 applies then return inconsistent;
if Lemma 2 applies then return inconsistent;
if Lemma 3 applies then return inconsistent;
Compute δ as discussed in the proof of Theorem 2;
return δ;
```

Our ESTP solver is summarized in Algorithm 1.

For the instance S in Example 2, Figure 1 shows a feasible potential π for its relaxation. Figure 2 represents the reweighted graph of that in Figure 1 (considering all appearing arcs). The same Figure also shows the remaining DAGs (ignoring all weights). Specifically, D_0 is the DAG consisting of all snake-pattern edges (with its SCCs highlighted in gray), whereas $D_{\mathcal{P}}$ is the DAG having as nodes the sets $X_1 := \{x_1, x_2, x_3\}$ and $X_2 := \{x_4, x_5, x_6, x_7\}$ and as unique arc the snake-pattern red one. We consider the topological sort $T := \langle X_1, X_2 \rangle$. Regarding all atoms of S, the smallest $|k + \pi(x) - \pi(y)| > 0$ is due to the atom $x_6 - x_5 \neq -7.14$ in the formula on inequations and its value is 0.14. Therefore, $\varepsilon := 0.02$. We compute the mapping δ as follows: we set $\delta(x) := \pi(x) + 0\varepsilon$ for each $x \in X_2$ and we set $\delta(x) := \pi(x) + 1\varepsilon$ for each $x \in X_1$.

The reader can check that $\delta(x_1) := \pi(x_1) + 1\varepsilon = 0.02$, $\delta(x_2) := \pi(x_2) + 1\varepsilon = -2.18$, $\delta(x_3) := \pi(x_3) + 1\varepsilon = -5.68$, $\delta(x_4) := \pi(x_4) + 0\varepsilon = 0$, $\delta(x_5) := \pi(x_5) + 0\varepsilon = -2$, $\delta(x_6) := \pi(x_6)+0\varepsilon = -9$, and $\delta(x_7) := \pi(x_7)+0\varepsilon = -3$ is a solution. In case, δ can be rigidly shifted by 9 to make it non-negative.

Theorem 3. Let $S = (X, C_{\leq}, C_{<}, C_{\neq})$ be an instance of ESTP. Let $n := |X|$, $m := |C_{\leq}| + |C_{<}|$ and $h := \sum_{F \in C_{\neq}} |\mathcal{A}_F|$. Then, Algorithm 1 runs in time $\mathcal{O}(nm + h)$.

Proof. We analyze the steps needed by Algorithm 1. It takes $\mathcal{O}(nm)$ to compute a feasible potential of G_R or prove that none exists (i.e., checking Lemma 1). This can be done, for example, by a run of the Bellman-Ford algorithm. Reweighting G_R to obtain G'_R costs $\mathcal{O}(m)$ and the computation of D_0 is $\mathcal{O}(n + m)$. Finding the SCCs of D_0, along with computing a correspondence allowing each node to retrieve its SCC, takes $\mathcal{O}(n + m)$ (Gabow 2000). Checking Lemma 2 corresponds to verify whether there exists any red arc in some SCC. That is, for each $y - x < k$ we just need to check if the SCCs of y and x are the same (because any cycle in that SCC including that arc has weight zero). Thus, this step takes $\mathcal{O}(m)$. Likewise, checking Lemma 3 takes $\mathcal{O}(h)$ because we need to evaluate each $F \in C_{\neq}$ under the interpretation that sets to **false** all $y - x \neq k \in \mathcal{A}_F$ *if and only if* y and x are in the same SCC and $\pi(y)-\pi(x) = k$. Computing $D_{\mathcal{P}}$ takes $\mathcal{O}(n+m)$. To compute ε, scanning all atoms of S takes $\mathcal{O}(m+h)$. Finding a topological sort takes $\mathcal{O}(n+m)$ and finally computing a solution δ from the potential π takes $\mathcal{O}(n)$ (also when we make δ non-negative by rigid shifting). Overall, the complexity is $\mathcal{O}(nm + h)$. □

When dealing with strict inequalities only, then $h = 0$ and the bound boils down to $\mathcal{O}(nm)$. Notice that this is the time that Bellman-Ford needs to solve even a simple instance of STP. This is faster than (Gerevini and Cristani 1997) as their algorithm is based on Floyd-Warshall, whose complexity is $\mathcal{O}(n^3) \geq \mathcal{O}(nm)$ regardless of the density of the graph.

Experimental Evaluation

In this section we provide a proof-of-concept implementation of our approach, comparing several variants of shortest paths algorithms. Then, we encode our model as a QF_RDL formula since it is well-known that satisfiability of conjunctions of atoms $y - x \bowtie k$ for $\bowtie \in \{\leq, <\}$ is tractable. Therefore, this experimental evaluation offers a fair comparison[2].

As seen in Theorem 3, computing a feasible potential function π (or proving that none exists checking Lemma 1) is the most expensive step and costs $\mathcal{O}(mn)$. This is a *strongly-polynomial* bound, that does not depend in any way on the magnitude of the numbers in the input. Actually, these could be rather clumsy objects that we could prefer to let an oracle to manage. Still, the problem would be solved in at most $\mathcal{O}(mn)$ calls to the oracle.

Although asymptotically optimal, Bellman-Ford with negative-cycle detection (from now on, BFTO) can be outperformed by other algorithms with the same time complexity but still better in practice: (Goldberg and Radzik 1993) combined with (Goldberg 1995) admissible-graph search negative-cycle detection algorithm (from now on, GORC); Bellman-Ford-Moore with (Tarjan 1981) subtree disassembly negative-cycle detection algorithm (from now on, BFCT). The interested reader is referred to (Bellman 1958; Ford Jr 1956; Moore 1959; Cormen et al. 2009) and to (Cherkassky and Goldberg 1999) for more details about the Bellman-Ford algorithm and for a survey about negative-cycle detection algorithms, respectively.

Another nontrivial step of our approach is the computation of SCCs; for that, we rely on (Gabow 2000), the fastest SCC algorithm according to (Alshomrani and Iqbal 2012).

As a result, we have implemented three different versions of our approach using BFTO, GORC and BFCT, respectively. To get an idea about their performance, we have also implemented (Dutertre and de Moura 2006)'s approach to handle strict inequalities through the symbolic computation of ε values. Again, we consider the three variants offered by BFTO, GORC and BFCT, since also (Dutertre and de Moura 2006) executes a non-modified SSSP algorithm combined with a negative-cycle detection procedure. Moreover, we included all SMT solvers that competed in the *15th International Satisfiability Modulo Theories Competition* — SMT-COMP 2020 — for QF_RDL (Model Validation Track and Single Query Track). These were: Yices2 v2.6.2 (Dutertre 2014), CVC4 v1.8 (Barrett et al. 2011), MathSAT5 v5.6.3 (Cimatti et al. 2013), z3 v4.8.8 (De Moura and Bjørner 2008), veriT v2016 (Bouton et al. 2009) and SMTInterpol v2.5 (Christ, Hoenicke, and Nutz 2012).

To encode an instance $S = (X, C_{\leq}, C_{<}, C_{\neq})$ of ESTP, we simply convert it into the QF_RDL formula:

$$\bigwedge_{y-x\leq k\in C_{\leq}} y - x \leq k \quad \bigwedge_{y-x<k\in C_{<}} y - x < k \quad \bigwedge_{F\in C_{\neq}} (F).$$

Current experiments involve conjunctions of strict and non-strict inequalities only, as this work is the first to prove tractability of monotone Boolean formulae on inequations. Therefore, we excluded formulae on inequations as the algorithms facing them in QF_RDL are designed for NP-complete problems.

We have defined four classes of instances, including *hard* instances to provide various degrees of hardness and require maximal work from all approaches, thus highlighting their performance. We consider hard all consistent instances with deep shortest-paths trees and all inconsistent instances with few and long negative cycles. All classes of instances contain an Hamiltonian zero-weight cycle, thus ensuring hardness for consistent instances. Moreover, all instances contain random edges, in proportion 8 to 1 with respect to the number of nodes, which do not form any negative cycle. Here follows a description of the four classes:

- **H000** are consistent instances with no further additions;
- **H001** are inconsistent instances with a single negative cycle with length equal to the 1% of the number of nodes;
- **H025** are inconsistent instances with a single negative cycle with length equal to the 25% of the number of nodes;
- **H100** are inconsistent instances with a single Hamiltonian negative cycle.

Starting with number of nodes $n = 32$, for each class 8 instances are generated and solved by each variant, computing the average of the running times. At the end of each iteration, n is multiplied by $\sqrt{2}$ and the process is repeated, until the execution takes more than 100 seconds. Table 1 specifies the hardware and software of the machine we used.

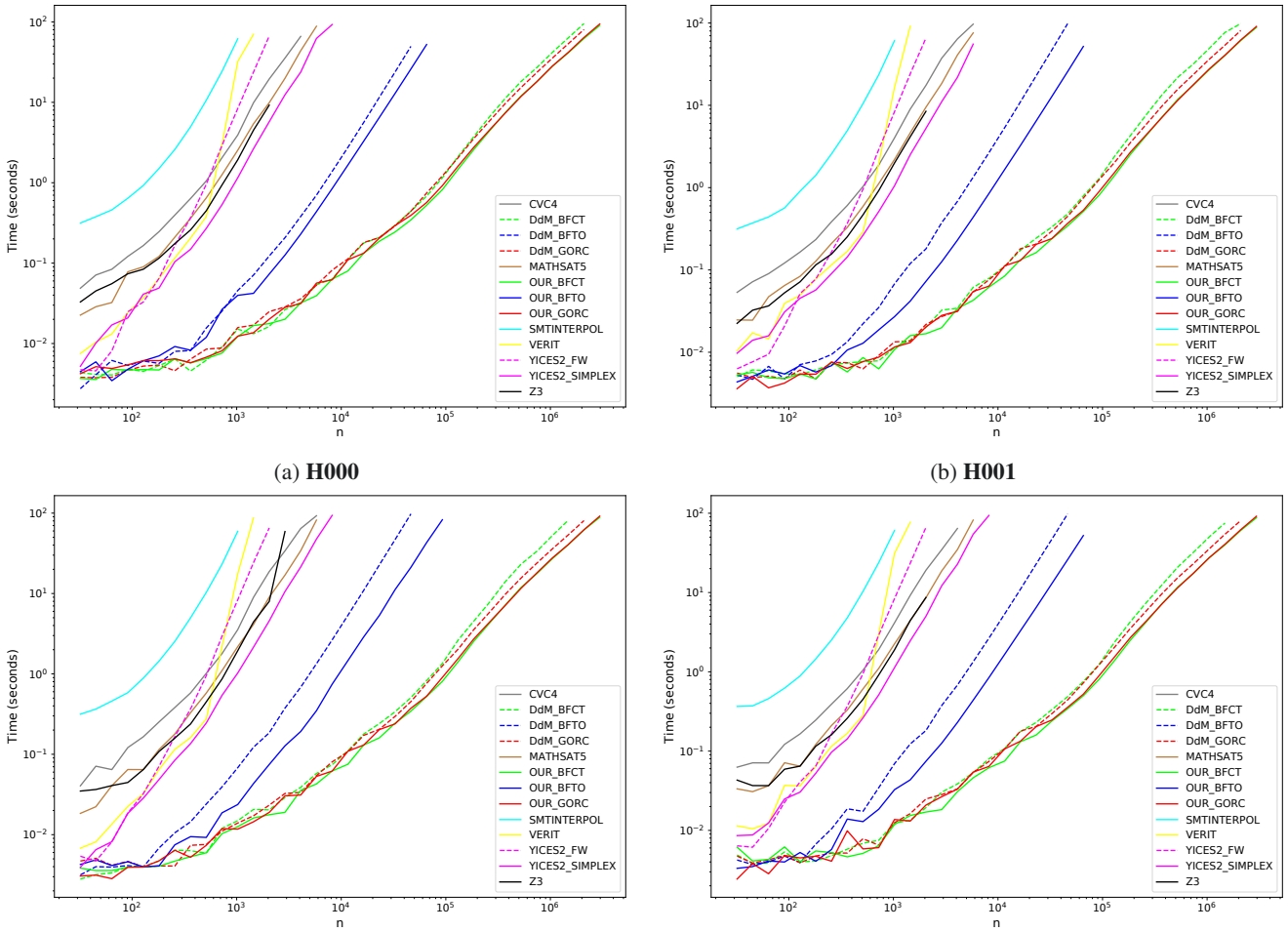
CPU	Intel Core i7-6700K @ 4.00GHz
RAM	64 GB DDR4 @ 2133MHz
SSD	Samsung SSD 850 EVO 1TB
OS	Arch Linux – kernel 5.8.5-zen
CXXFLAGS	-O3 -march=native -std=c++17

Table 1: Benchmark environment.

Benchmark results for **H000**, **H001**, **H025** and **H100** instance classes are shown in Figure 3. The prefix `OUR_` refers to our approach, whereas the prefix `DdM_` refers to that in (Dutertre and de Moura 2006). Yices2 has been tested twice with the two available solvers: simplex (`_SIMPLEX`) and Floyd-Warshall (`_FW`) (Dutertre 2014).

We can see that our approach, in both variants `OUR_BFCT` and `OUR_GORC`, is always faster on any input class. Regarding `OUR_` and `DdM_` implementations, this is to be expected, since (Dutertre and de Moura 2006)'s symbolic approach suffers from the overhead of doing the symbolic ε computations, whereas our approach, being dominated by the SSSP and negative-cycle detection step, does not incur in any overhead and the time margin gained is enough to cover the remaining steps. Also towards SMT solvers, `OUR_BFCT` and

[2]https://github.com/CALIPSO-UniVR/estp-aaai2021



(a) **H000** (b) **H001**

(c) **H025** (d) **H100**

Figure 3: Benchmark results. All plots are in logarithmic scale.

OUR_GORC demonstrate their competitiveness, being able to solve instances up to $n = 2965821$ within 100 seconds. Instead YICES2_SIMPLEX, which proved to be the fastest among the SMT solvers we considered, fails to solve instances with more than 8192 (**H000**, **H025** and **H100**) and 5793 (**H001**) variables within 100 seconds.

Conclusions and Future Work

We defined the Extended Simple Temporal Problem (ESTP) by generalizing the class of constraints studied by Koubarakis in (Koubarakis 1992) to consider monotone Boolean formulae on inequations. We analyzed the structure of the resulting problem. Given an instance of ESTP, we provided three conditions to detect inconsistency: the first condition looks for negative-weight cycles in the constraint graph of the relaxation of the instance; the second one searches for zero-weight cycles containing red arcs (i.e., strict inequalities) in the same graph; the third condition looks for hopeless formulae on inequations. An instance is consistent iff none of these three applies. Checking these conditions results in an $\mathcal{O}(nm + h)$ algorithm to solve any ESTP instance, where n is the number of variables, m is the number of inequalities and h is the number of inequations appearing in all monotone Boolean formulae. We carried out a fully-reproducible experimental evaluation focused on instances of ESTP which currently considers strict and non-strict inequalities. This was motivated by the possibility of making a comparison with several variants of SSSP algorithms and also state-of-the-art SMT solvers supporting QF_RDL, in which satisfiability of conjunctions of inequalities is known to be tractable. Overall, our approach can handle instances close to 3 million variables within 100 seconds, while all SMT solvers hit this timeout on much smaller instances (at best 8192 variables). Our results suggest to employ BFCT or GORC as core solving algorithms.

As future work, we plan to extend our experimental evaluation to also consider monotone Boolean formulae on inequations by providing a reference dataset.

Acknowledgments

This work was partially supported by MIUR, Project *Italian Outstanding Departments, 2018-2022*, and by INdAM,

GNCS 2020, Project *Strategic Reasoning and Automated Synthesis of Multi-Agent Systems*.

References

Alshomrani, S.; and Iqbal, G. 2012. Analysis of strongly connected components (SCC) using dynamic graph representation. *International Journal of Computer Science Issues (IJCSI)* 9(4).

Armando, A.; Castellini, C.; Giunchiglia, E.; and Maratea, M. 2005. The SAT-based Approach to Separation Logic. *Journal of Automated Reasoning* 35(1): 237–263.

Barrett, C. W.; Conway, C. L.; Deters, M.; Hadarean, L.; Jovanovic, D.; King, T.; Reynolds, A.; and Tinelli, C. 2011. CVC4. In *CAV 2011*, volume 6806 of *Lecture Notes in Computer Science*, 171–177. Springer.

Barrett, C. W.; Sebastiani, R.; Seshia, S. A.; and Tinelli, C. 2009. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, 825–885. IOS Press.

Bartak, R.; Morris, R. A.; and Venable, K. B. 2014. *An Introduction to Constraint-Based Temporal Reasoning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Bellman, R. 1958. On a routing problem. *Quarterly of applied mathematics* 16(1): 87–90.

Bouton, T.; Caminha B. de Oliveira, D.; Déharbe, D.; and Fontaine, P. 2009. veriT: An Open, Trustable and Efficient SMT-Solver. In *Automated Deduction – CADE-22*, 151–156. Springer Berlin Heidelberg.

Broxvall, M. 2002. A Method for Metric Temporal Reasoning. In *AAAI/IAAI 2002*, 513–518.

Carbonnel, C.; and Cooper, M. 2015. Tractability in Constraint Satisfaction Problems: A Survey. *Constraints An Int. J.* 21(2): 115–144.

Cherkassky, B. V.; and Goldberg, A. V. 1999. Negative-cycle detection algorithms. *Mathematical Programming* 85(2).

Christ, J.; Hoenicke, J.; and Nutz, A. 2012. SMTInterpol: An Interpolating SMT Solver. In Donaldson, A. F.; and Parker, D., eds., *SPIN 2012*, volume 7385 of *Lecture Notes in Computer Science*, 248–254. Springer.

Cimatti, A.; Griggio, A.; Schaafsma, B.; and Sebastiani, R. 2013. The MathSAT5 SMT Solver. In Piterman, N.; and Smolka, S., eds., *Proceedings of TACAS*, volume 7795 of *LNCS*. Springer.

Cooper, M. C.; Maris, F.; and Régnier, P. 2013. Relaxation of Temporal Planning Problems. In *TIME 2013*, 37–44. IEEE Computer Society.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

De Moura, L.; and Bjørner, N. 2008. Z3: An Efficient SMT Solver. In *TACAS 2008*, TACAS'08/ETAPS'08, 337–340. Berlin, Heidelberg: Springer-Verlag.

de Moura, L.; Dutertre, B.; and Shankar, N. 2007. A Tutorial on Satisfiability Modulo Theories. In *CAV '07*, volume 4590 of *LNCS*, 20–36. Springer-Verlag.

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

Dutertre, B. 2014. Yices 2.2. In *CAV 2014*, volume 8559 of *Lecture Notes in Computer Science*, 737–744. Springer.

Dutertre, B.; and de Moura, L. 2006. A Fast Linear-Arithmetic Solver for DPLL(T). In *Computer Aided Verification*, 81–94. Springer Berlin Heidelberg.

Ford Jr, L. R. 1956. Network flow theory. Technical report, Rand Corp Santa Monica Ca.

Gabow, H. N. 2000. Path-based depth-first search for strong and biconnected components. *Information Processing Letters* 74(3): 107 – 114.

Gerevini, A.; and Cristani, M. 1997. On Finding a Solution in Temporal Constraint Satisfaction Problems. In *IJCAI '97*, 1460–1465. Morgan Kaufmann.

Goldberg, A. V. 1995. Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing* 24(3): 494–504.

Goldberg, A. V.; and Radzik, T. 1993. A heuristic improvement of the Bellman-Ford algorithm. *Applied Mathematics Letters* 6(3): 3 – 6.

Koubarakis, M. 1992. Dense Time and Temporal Constraints with !=. In *(KR'92)*, 24–35. Morgan Kaufmann.

Montanari, A.; Navarrete, I.; Sciavicco, G.; and Tonon, A. 2012. A tractable formalism for combining rectangular cardinal relations with metric constraints. In *ICAART 2012*, volume 1, 154–163. SciTePress.

Moore, E. F. 1959. The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959*, 285–292.

Schwalb, E.; and Vila, L. 1998. Temporal Constraints: A Survey. *Constraints* 3(2): 129–149.

Stergiou, K.; and Koubarakis, M. 2000. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence* 120(1): 81–117.

Tarjan, R. E. 1981. Shortest paths. *Tech reports* .

Vila, L. 1994. A Survey on Temporal Reasoning in Artificial Intelligence. *AI Commun.* 7(1): 4–28.