TIME-DEPENDENT SIMPLE TEMPORAL NETWORKS: PROPERTIES AND ALGORITHMS [*]

CÉDRIC PRALET[1] AND GÉRARD VERFAILLIE[1]

Abstract. Simple Temporal Networks (STN) allow conjunctions of minimum and maximum distance constraints between pairs of temporal positions to be represented. This paper introduces an extension of STN called Time-dependent STN (TSTN), which covers temporal constraints for which the minimum and maximum distances required between two temporal positions x and y are not necessarily constant but may depend on the assignments of x and y. Such constraints are useful to model problems in which the duration of an activity may depend on its starting time, or problems in which the transition time required between two activities may depend on the time at which the transition is triggered. Properties of the new framework are analyzed, and standard STN solving techniques are extended to TSTN. The contributions are applied to the management of temporal constraints for so-called *agile* Earth observation satellites.

Keywords: Temporal constraints, time-dependent scheduling, constraint propagation, agile satellites

1. MOTIVATIONS

Managing temporal aspects is crucial when solving planning and scheduling problems. Indeed, the latter generally involve constraints on the earliest start times and latest end times of activities, precedence constraints between activities, no-overlapping constraints over sets of activities, or constraints over the minimum

and maximum temporal distance between activities. In many cases, these constraints can be expressed as *simple* temporal constraints, written as $x - y \in [\alpha, \beta]$ with x, y two variables corresponding to temporal positions and α, β two constants. Such simple temporal constraints can be represented using the STN framework (Simple Temporal Networks [9]). This framework is appealing in practice due to the polynomial complexity of important operations such as determining the consistency of an STN or computing the earliest/latest times associated with each temporal variable of an STN, which is useful to maintain a schedule offering temporal flexibility. Another feature of STN is that they are often used as a basic element when solving more complex temporal problems such as DTN (Disjunctive Temporal Networks [26]).

In this paper, we propose an extension of the STN framework and of STN algorithms. This extension has been motivated by an application from the space domain. The latter corresponds to the management of Earth observation satellites such as those of the *Pleiades* system whose first satellite was launched in December 2011 (see http://smsc.cnes.fr/PLEIADES/). Such satellites are moving around the Earth on a circular, quasi-polar, low-altitude orbit (several hundreds of kilometers). They are equipped with an optical observation instrument which is body-mounted on the satellite. They are said to be *agile*, because they have the capacity, while moving on their orbit, to move very quickly around their gravity center along the three axes (roll, pitch, and yaw), thanks to gyroscopic actuators and to their attitude control system. This agility allow them to acquire via scanning any strip at the Earth surface, in any direction, on the right, on the left, in front of, or behind of the so-called *nadir*, that is the Earth point which is at any time at the vertical of the satellite. Acquiring a given strip requires at any time a particular configuration of the satellite, called an *attitude*, defined by a pointing direction and by a speed of the pointing movement on each of the three axes. Moreover, agility allows satellites to move quickly from the end of the acquisition of a strip to the beginning of the acquisition of the following one.

In the agile satellite context, contrary to the simplified version of the 2003 ROADEF Challenge [6], the minimum transition time required by an attitude movement between the end of an acquisition i and the start of an acquisition j is not constant and depends on the precise time at which acquisition i ends [17]. This is schematically illustrated by the 2-dimension view of Fig. 1.
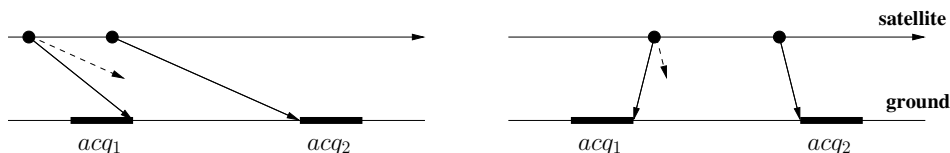

FIGURE 1. How the attitude movement to be performed and thus the minimum transition time between acquisitions may depend on the time at which the first acquisition ends.

In fact, transition times may vary of about ten seconds on the examples provided in Fig. 2. For Fig. 2(b), there is no point after $t = 150$ because no transition from i to j is possible after that time. Fig. 2 also shows how diverse minimum transition times evolution schemes can be. Minimum transition times are computed by solving a continuous command optimization problem which takes into account the movement of the satellite on its orbit, the movement of points on the ground due to the rotation of Earth, and kinematic constraints restricting the possible attitude movements of the satellite (pointing direction, speed and acceleration of the pointing movement).

Logistics problems, where the estimated time to go from one location to another one depends on the traffic congestion and thus on the starting time, is another
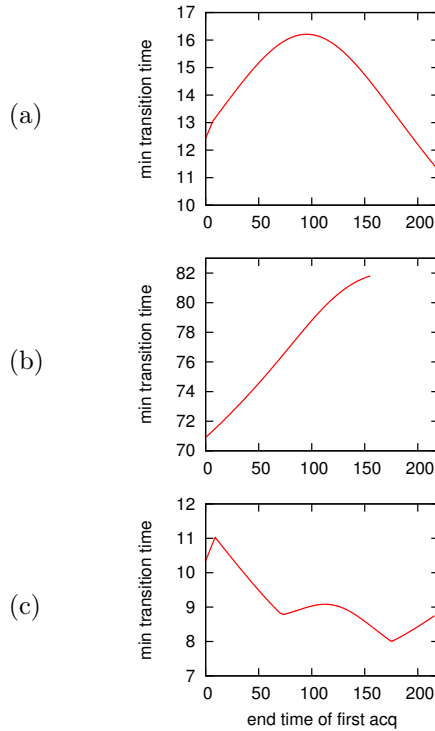

FIGURE 2. Minimum transition time, in seconds, from the acquisition of a strip i ending at point of latitude-longitude $41°17'48''$N-$2°5'12''$E to the acquisition of a strip j starting at point of latitude-longitude $42°31'12''$N-$2°6'15''$E, for different scanning angles with regard to the trace of the satellite on the ground: (a) scan of i at $40°$ and scan of j at $20°$; (b) scan of i at $40°$ and scan of j at $-80°$; (c) scan of i at $90°$ and scan of j at $82°$

example of application in which the transition time may depend on the time at which the transition is triggered.

Similar aspects are taken into account by works on so-called *time-dependent scheduling* [7,12], where activity durations may depend on activity starting times. However, the particular forms (piecewise constant or piecewise linear) of activity durations as functions of activity starting times, that are considered in these works, do not apply to the agile satellite context.

This motivates the need for a new modeling framework able to handle problems in which the minimum transition time between two activities may depend on the precise time at which the transition is triggered. The framework proposed, called Time-dependent STN (TSTN), is first introduced (Sect. 2). Techniques are then defined for computing the earliest and latest times associated with each temporal variable (Sect. 3 to 5). These techniques are used for scheduling activities of an agile satellite, in the context of a local search algorithm (Sect. 6). Proofs can be skipped without loss of continuity. Parts of the work described in this paper were published in [22] and [23].

2. Towards Time-dependent STN

2.1. Simple Temporal Networks (STN)

We first recall some definitions associated with STN. In the following, given a variable x, $\mathbf{D}(x)$ denotes its initial domain of values and $\mathbf{d}(x) \subseteq \mathbf{D}(x)$ denotes its current domain of values.

Definition 2.1. An STN is a pair (V, C) with V a finite set of continuous variables whose initial domain is a closed interval $I \subseteq \mathbb{R}$, and C a finite set of binary constraints of the form $x - y \in [\alpha, \beta]$ with $x, y \in V$, $\alpha \in \mathbb{R} \cup \{-\infty\}$, and $\beta \in \mathbb{R} \cup \{+\infty\}$.[1] Such constraints are called *simple temporal constraints*. A solution to an STN (V, C) is an assignment of all variables in V satisfying all constraints in C. An STN is consistent iff it has at least one solution.

Unary constraints $x \in [\alpha, \beta]$, including those defining the initial domains of possible values of variables, can be formulated as simple temporal constraints $x - x_0 \in [\alpha, \beta]$, with x_0 a special variable of domain $[0, 0]$ playing the role of a temporal reference. Moreover, as $x - y \in [\alpha, \beta]$ is equivalent to $(x - y \leq \beta) \wedge (y - x \leq -\alpha)$, it is possible to use only constraints of the form $y - x \leq c$ with c some constant.

An important element associated with an STN is its *distance graph*. This graph contains one node per variable of the STN and, for each constraint $y - x \leq c$ of the STN, one arc from x to y weighted by c. Based on this distance graph, the following results can be established [9] (some of these results are similar to earlier work on PERT and critical path analysis):

---

[1] $x - y \in [\alpha, \beta]$ generalizes the following situations: (a) $x - y \leq \beta$ (case $\alpha = -\infty$); (b) $x - y \geq \alpha$ (case $\beta = +\infty$); (c) $\alpha \leq x - y \leq \beta$ (case $\alpha, \beta \in \mathbb{R}$ and $\alpha < \beta$); (d) $x - y = \gamma$ (case $\gamma = \alpha = \beta$).

(1) an STN is consistent iff its distance graph has no cycle of negative length;
(2) if d_{0i} (resp. d_{i0}) denotes the length of the shortest path in the distance graph from the reference node labeled by x_0 to a node labeled by variable x_i (resp. from x_i to x_0), then interval $[-d_{i0}, d_{0i}]$ gives the set of consistent assignments of x_i; the shortest paths can be computed for every i using Bellman-Ford's algorithm [1, 11] or arc-consistency filtering [4, 5, 13, 25];
(3) if d_{ij} (resp. d_{ji}) denotes the length of the shortest path from x_i to x_j (resp. x_j to x_i) in the distance graph, then interval $[-d_{ji}, d_{ij}]$ corresponds to the set of all possible temporal distances between x_i and x_j; shortest paths can be computed for every i, j using Floyd-Warshall's algorithm [10, 28] or path-consistency filtering [9, 20, 21, 29], which produces the *minimal network* of the STN [18].

Example 2.2. Let us consider a simplified satellite scheduling problem. This problem involves 3 acquisitions acq_1, acq_2, acq_3 to be realized in order $acq_3 \rightarrow acq_1 \rightarrow acq_2$. For every $i \in [1..3]$, $Tmin_i$ and $Tmax_i$ denote the earliest start time and latest end time of acq_i, and Da_i denotes the duration of acq_i. The minimum durations of the transitions between the end of acq_3 and the start of acq_1, and between the end of acq_1 and the start of acq_2, are denoted $Dt_{3,1}$ and $Dt_{1,2}$ respectively. These durations are considered as constant in this first simplified version. We also consider two temporal windows $w_1 = [Ts_1, Te_1]$, $w_2 = [Ts_2, Te_2]$ during which data download to ground stations is possible. The satellite must download acq_2 followed by acq_3 in window w_1, before downloading acq_1 in window w_2. For every $i \in [1..3]$, Dd_i denotes the duration taken by the download of acq_i.

This problem can be modeled as an STN containing, for every acquisition acq_i ($i \in [1..3]$), (a) two variables sa_i and ea_i denoting respectively the start time and end time of the acquisition, with domains of values $\mathbf{D}(sa_i) = \mathbf{D}(ea_i) = [Tmin_i, Tmax_i]$; (b) two variables sd_i and ed_i, denoting respectively the start time and end time of the download of the acquisition, with domains of values $[Ts_1, Te_1]$ for $i = 2, 3$ and $[Ts_2, Te_2]$ for $i = 1$.

Simple temporal constraints in Eq. 1 to 4 are imposed over these variables. Eq. 1 defines the duration of acquisitions and data downloads. Eq. 2 imposes minimum transition times between acquisitions. Eq. 3 enforces no-overlap between downloads. Eq. 4 expresses that an acquisition can start being downloaded only after its realization. Fig. 3 gives the distance graph of the obtained STN.

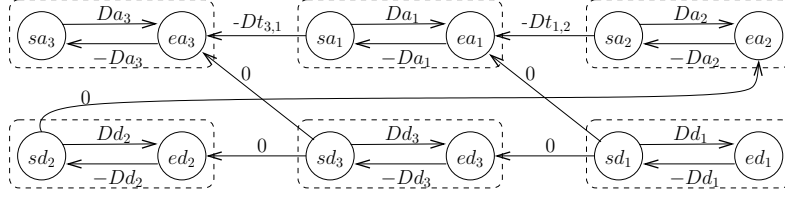$$\forall i \in [1..3], (ea_i - sa_i = Da_i) \wedge (ed_i - sd_i = Dd_i) \tag{1}$$

$$(sa_1 - ea_3 \geq Dt_{3,1}) \wedge (sa_2 - ea_1 \geq Dt_{1,2}) \tag{2}$$

$$(sd_3 - ed_2 \geq 0) \wedge (sd_1 - ed_3 \geq 0) \tag{3}$$

$$\forall i \in [1..3], sd_i - ea_i \geq 0 \tag{4}$$

2.2. T-Simple Temporal Constraints and TSTN

We now introduce a new class of temporal constraints which can be used to model transitions whose minimum duration depends on the precise time at which

FIGURE 3. Distance graph (reference temporal position x_0 is not represented)

the transition is triggered. These constraints are called *t-simple temporal constraints* for "time-dependent"-simple temporal constraints.

Definition 2.3. A *t-simple temporal constraint* is a triple $(x, y, dmin)$ composed of two temporal variables x and y, and of one function $dmin : \mathbf{D}(x) \times \mathbf{D}(y) \to \mathbb{R}$ called *minimum distance function* (function not necessarily continuous). A t-simple temporal constraint $(x, y, dmin)$ is also written as $y - x \geq dmin(x, y)$. The constraint is satisfied by $(a, b) \in \mathbf{D}(x) \times \mathbf{D}(y)$ iff $b - a \geq dmin(a, b)$.

Informally, $dmin(x, y)$ specifies a minimum temporal distance between the events associated with temporal variables x and y respectively. Note that any binary continuous constraint $f(x, y) \leq 0$ over continuous variables x and y can be expressed as a t-simple temporal constraint by taking $dmin(x, y) = y - x + f(x, y)$. Conversely, any t-simple temporal constraint $y - x \geq dmin(x, y)$ can be put in the form $f(x, y) \leq 0$ by taking $f(x, y) = x + dmin(x, y) - y$. This paper considers the "$y - x \geq dmin(x, y)$" version instead of the "$f(x, y) \leq 0$" one in order to make the parallel with STN and time-dependent scheduling more explicit. But all results given in the paper can be applied to $f(x, y) \leq 0$ constraints as well. Function f defined by $f(x, y) = x + dmin(x, y) - y$ will also be particularly important in the following (see the notion of *delay-function* introduced in Def. 3.1).

To illustrate why having a minimum distance function $dmin$ depending on both x and y is useful, let us consider the example of agile satellites. Let x be a variable representing the end time of an acquisition acq. Let $Att(x)$ denote the attitude obtained when finishing acq at time x. Let y be a variable representing the start time of an acquisition acq', to be performed just after acq. Let $Att'(y)$ denote the attitude required for starting acq' at time y. Let $minAttTransTime$ be the function (available in our agile satellite library) such that $minAttTransTime(att, att')$ gives the minimum transition time required by a satellite maneuver to move from attitude att to attitude att'. Then, t-simple temporal constraint $y - x \geq dmin(x, y)$ with $dmin(x, y) = minAttTransTime(Att(x), Att'(y))$ expresses that the duration between the end of acq and the start of acq' must be greater than the minimum duration required to move from attitude $Att(x)$ to attitude $Att'(y)$.

In some cases, function $dmin(x, y)$ does not depend on y. This concerns *time-dependent scheduling* [7, 12], for which the duration of an activity only depends on its start time (t-simple temporal constraint $y - x \geq dmin(x)$ with $dmin(x)$ the activity duration when starting it at time x). T-simple temporal constraints

also cover simple temporal constraints $y - x \geq c$, by using a constant minimum distance function $dmin = c$. They also cover constraints of maximum temporal distance between two temporal variables $y - x \leq dmax(x, y)$, since the latter can be rewritten as $x - y \geq dmin(y, x)$ with $dmin(y, x) = -dmax(x, y)$.

Note that a t-simple temporal constraint only refers to the *minimum* duration of a transition. Such an approach can be used for handling agile satellites under the (realistic) assumption that any maneuver which can be made in duration δ is also feasible in duration $\delta' \geq \delta$. This assumption of feasibility of a "lazy maneuver" is not necessarily satisfied by every physical system.

On this basis of t-simple temporal constraints, a new framework called TSTN for Time-dependent STN can be introduced.

Definition 2.4. A TSTN is a pair (V, C) with V a finite set of continuous variables of domain $[l, u] \subset \mathbb{R}$, and C a finite set of t-simple temporal constraints $(x, y, dmin)$ with $x, y \in V$. A solution to a TSTN is an assignment of variables in V that satisfies all constraints in C. A TSTN is said to be consistent iff it admits at least one solution.

Example 2.5. Let us reconsider the example involving three acquisitions (acq_1, acq_2, acq_3), and remove the unrealistic assumption of constant minimum transition durations between acquisitions. In the TSTN model obtained, the only difference when compared to the initial STN model is that the simple temporal constraints of Eq. 2 are replaced by the t-simple temporal constraints given in Eq. 5 and 6, in which given an acquisition acq_i, $Satt_i(t)$ and $Eatt_i(t)$ respectively denote the attitudes required at the start and at the end of acq_i if this start/end occurs at time t.

$$sa_1 - ea_3 \geq minAttTransTime(Eatt_3(ea_3), Satt_1(sa_1)) \tag{5}$$
$$sa_2 - ea_1 \geq minAttTransTime(Eatt_1(ea_1), Satt_2(sa_2)) \tag{6}$$

The definition of the distance graph associated with a TSTN is similar to the definition of the distance graph associated with an STN (see Fig. 4).
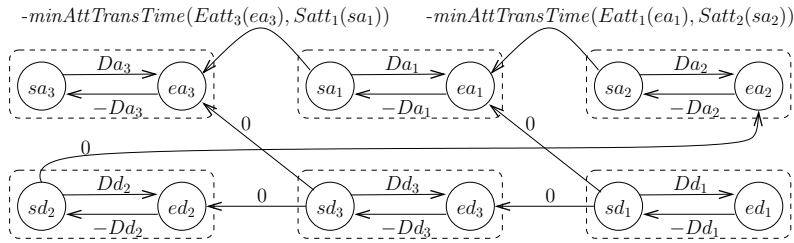

FIGURE 4. TSTN distance graph (temporal reference x_0 is not represented)

3. ARC-CONSISTENCY OF T-SIMPLE TEMPORAL CONSTRAINTS

A first important notion for establishing arc-consistency is the *delay function*.

Definition 3.1. The *delay function* associated with a t-simple temporal constraint $ct : (x, y, dmin)$ is function $delay_{ct} : \mathbf{D}(x) \times \mathbf{D}(y) \to \mathbb{R}$ defined by $delay_{ct}(a, b) = a + dmin(a, b) - b$.

Informally, $delay_{ct}(a, b)$ is the delay obtained in b if a transition in minimum time from x to y is triggered at time a. This delay corresponds to the difference between the minimum arrival time associated with the transition $(a + dmin(a, b))$ and the required arrival time (b). A strictly negative delay corresponds to a transition ending before deadline b. A strictly positive delay corresponds to a violation of constraint ct. A null delay corresponds to an arrival right on time.

Definition 3.2. A t-simple temporal constraint $ct : (x, y, dmin)$ is said to be *delay-monotonic* iff its delay function $delay_{ct}(., .)$ satisfies the conditions below:

$$\forall a, a' \in \mathbf{D}(x), \forall b \in \mathbf{D}(y), \quad (a \leq a') \to (delay_{ct}(a, b) \leq delay_{ct}(a', b))$$
$$\forall a \in \mathbf{D}(x), \forall b, b' \in \mathbf{D}(y), \quad (b \leq b') \to (delay_{ct}(a, b) \geq delay_{ct}(a, b'))$$

Definition 3.2 means that for being delay-monotonic, a t-simple temporal constraint $(x, y, dmin)$ must verify that on one hand the later the transition is triggered in x, the greater the delay in y, and on the other hand the earlier the transition must end in y, the greater the delay. When monotonicities over the two arguments are strict, we speak of a strictly delay-monotonic constraint. A TSTN is said to be delay-monotonic iff it contains only delay-monotonic constraints. Delay-monotonicity can be related to the notion of monotonic constraints, defined for instance in [15]. One slight difference is that in TSTN, domains considered are continuous, and we speak of delay-monotonicity essentially to make the interpretation of this property more explicit in the context of temporal constraints. As shown in Prop. 3.3 and 3.4, simple temporal constraints are delay-monotonic, as well as standard distance functions considered in time-dependent scheduling [7].

Proposition 3.3. *Simple temporal constraints $ct : y - x \geq c$ are strictly delay-monotonic.*

Proof. If $a < a'$, then $delay_{ct}(a, b) - delay_{ct}(a', b) = a - a' < 0$. If $b < b'$, then $delay_{ct}(a, b) - delay_{ct}(a, b') = b' - b > 0$. □

Proposition 3.4. *Let x, y be two temporal variables corresponding to the start time and end time of an activity respectively. Results of Table 1 hold.*

Proof. If $dmin(x, y) = dmin(x)$ decreases at some step, then delay-monotonicity holds if the decrease slope is ≥ -1. Indeed, in this case, if $a \leq a'$, then $delay_{ct}(a, b) - delay_{ct}(a', b) = (a - a') + (dmin(a) - dmin(a')) \leq 0$. Moreover, if $b \leq b'$, then $delay_{ct}(a, b) - delay_{ct}(a, b') = b' - b \geq 0$. Strict delay-monotonicity holds when the decrease slope of $dmin(x)$ is always > -1. □
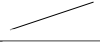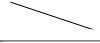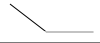
Distance $dmin(x,y) = dmin(x)$	form	delay-monotonic
$A + Bx$	/	yes (strict)
$A - Bx$	\	yes iff $B \leq 1$ (strict iff $B < 1$)
$\max(A, A + B(x - D))$	_/	yes (strict)
A if $x < D, A + B$ otherwise	‾‾	yes (strict)
$A - B\min(x, D)$	_	yes iff $B \leq 1$ (strict iff $B < 1$)

TABLE 1. Delay-monotonicity of some minimum distance functions used in time-dependent scheduling, with x a variable whose domain is not reduced to a singleton, and A, B, D constants such that $A \geq 0$, $B > 0$, and $D > \min(\mathbf{D}(x))$

We now introduce the functions of earliest arrival time and latest departure time associated with a t-simple temporal constraint. In the following, given a function $F : \mathbb{R} \to \mathbb{R}$ and a closed interval $I \subset \mathbb{R}$, we denote by (1) $firstNeg(F, I)$ the smallest $a \in I$ such that $F(a) \leq 0$ (value $+\infty$ if such a value does not exist); (2) $lastNeg(F, I)$ the greatest $a \in I$ such that $F(a) \leq 0$ (value $-\infty$ if such a value does not exist).[2]

Definition 3.5. The functions of earliest arrival time and latest departure time associated with a t-simple temporal constraint $ct : (x, y, dmin)$ are functions $earr_{ct}$ and $ldep_{ct}$, defined over $\mathbf{D}(x)$ and $\mathbf{D}(y)$ respectively by:

$$\forall a \in \mathbf{D}(x),\ earr_{ct}(a) = firstNeg(delay_{ct}(a, .), \mathbf{d}(y))$$
$$\forall b \in \mathbf{D}(y),\ ldep_{ct}(b) = lastNeg(delay_{ct}(., b), \mathbf{d}(x))$$

Informally, quantity $earr_{ct}(a)$ gives the smallest arrival time in y without delay if the transition from x is triggered at time a, and quantity $ldep_{ct}(b)$ gives the latest triggering time of the transition in x for an arrival in b without delay.

Prop. 3.6 shows that the earliest arrival and latest departure functions help establishing bound arc-consistency.

Proposition 3.6. *Bound arc-consistency for a t-simple temporal constraint $ct :$ $(x, y, dmin)$ can be enforced using the following domain modification rules:*

$$\mathbf{d}(y) \leftarrow \mathbf{d}(y) \cap [earr_{ct}(\min(\mathbf{d}(x))), +\infty[\tag{7}$$
$$\mathbf{d}(x) \leftarrow \mathbf{d}(x) \cap] - \infty, ldep_{ct}(\max(\mathbf{d}(y)))] \tag{8}$$

Proof. By definition of $earr_{ct}$, quantity $earr_{ct}(\min(\mathbf{d}(x)))$ used in Rule 7 equals either $+\infty$ or a value in $\mathbf{d}(y)$. If it equals $+\infty$, the domain of y becomes empty.

---

[2]Quantities $firstNeg(F, I)$ and $lastNeg(F, I)$ are mathematically not necessarily well-defined if function F has discontinuities; we implicitly use the fact that all operations are done on computers with finite precision.

Otherwise, $earr_{ct}(\min(\mathbf{d}(x))) \in \mathbf{d}(y)$ is the min value of y after the application of Rule 7. Moreover, $delay_{ct}(\min(\mathbf{d}(x)), earr_{ct}(\min(\mathbf{d}(x)))) \le 0$ by definition of $earr_{ct}$, which proves that the min bound of x and the min bound of y after the application of Rule 7 support each other.

By definition of $ldep_{ct}$, quantity $ldep_{ct}(\max(\mathbf{d}(y)))$ used in Rule 8 equals either $-\infty$ or a value in $\mathbf{d}(x)$. If it equals $-\infty$, the domain of x becomes empty. Otherwise, $ldep_{ct}(\max(\mathbf{d}(y))) \in \mathbf{d}(x)$ is the max value of x after the application of Rule 8. Moreover, $delay_{ct}(ldep_{ct}(\max(\mathbf{d}(y))), \max(\mathbf{d}(y))) \le 0$ by definition of $ldep_{ct}$, which proves that the max bound of x after the application of Rule 8 and the max bound of y support each other. $\qquad\square$

Rule 7 updates the earliest time associated with y. Rule 8 updates the latest time associated with x. These domain modification rules are such that current domains $\mathbf{d}(x)$ and $\mathbf{d}(y)$ remain closed intervals. Prop. 3.7 below establishes the equivalence between bound arc-consistency and arc-consistency for delay-monotonic constraints.

Proposition 3.7. *Let $ct : (x, y, dmin)$ be a t-simple temporal constraint with monotonic delay. Establishing bound arc-consistency for ct using Rules 7 and 8 is equivalent to establishing arc-consistency over the whole domains of x and y.*

Proof. Let x^-, x^+, y^-, y^+ denote the min/max bounds of x and y before application of the rules. Let $b \in [y^-, y^+]$. If $b < earr_{ct}(x^-)$, then b has no support over x for ct because $\forall a \in [x^-, x^+]$, $delay_{ct}(a, b) \ge delay_{ct}(x^-, b) > 0$ (by delay-monotonicity and by definition of $earr_{ct}(x^-)$). Conversely, if $b \ge earr_{ct}(x^-)$, then $delay_{ct}(x^-, b) \le delay_{ct}(x^-, earr_{ct}(x^-)) \le 0$, hence b is supported by x^-. Therefore, y-values pruned by Rule 7 are those that have no support over x.

Let $a \in [x^-, x^+]$. If $a > ldep_{ct}(y^+)$, then a has no support over y for ct because $\forall b \in [y^-, y^+]$, $delay_{ct}(a, b) \ge delay_{ct}(a, y^+) > 0$ (by delay-monotonicity and by definition of $ldep_{ct}(y^+)$). Conversely, if $a \le ldep_{ct}(y^+)$, then $delay_{ct}(a, y^+) \le delay_{ct}(ldep_{ct}(y^+), y^+) \le 0$, hence a is supported by y^+. Therefore, x-values pruned by Rule 8 are those that have no support over y. $\qquad\square$

When delay-monotonicity is violated, Rules 7-8 can be applied but they do not necessarily establish arc-consistency.

Concerning the way *earr* and *ldep* can be computed in practice, for a simple temporal constraint $ct : y - x \ge c$, an analytic formulation of *earr* and *ldep* can be given. Rules 7-8 then correspond to the standard propagation rules associated with distance constraints in constraint programming. Analytic formulations can also be derived when distance functions of Table 1 are used.

However, in the general case, in order to compute *earr* and *ldep*, quantities $firstNeg(F, I)$ and $lastNeg(F, I)$ must be evaluated. Such an evaluation is an optimization problem in itself. An iterative method for approximating $firstNeg(F, I = [a_1, a_2])$ is illustrated in Fig. 5 and formally defined in Algorithm 1. This method corresponds to the standard *false position method*, used to find a zero of an

arbitrary function. Applied to the case of t-simple temporal constraints, the method works as follows. If leftmost point $P_1 = (a_1, F(a_1))$ has a negative delay ($F(a_1) \leq 0$), then a_1 is directly returned. Otherwise, if rightmost point $P_2 = (a_2, F(a_2))$ has a strictly positive delay ($F(a_2) > 0$), then $+\infty$ is returned. Otherwise, points P_1 and P_2 have opposite delay-signs ($F(a_1) > 0$ and $F(a_2) \leq 0$), and the method computes delay $F(a_3)$ in a_3, the x-value of the intersection between segment (P_1, P_2) and the x-axis. See Fig. 5(a) for an illustration. If the delay in $P_3 = (a_3, F(a_3))$ is negative, then the mechanism is applied again by taking $P_2 = P_3$, as from Fig. 5(a) to Fig. 5(b). If the delay in P_3 is positive, then the mechanism is applied again by taking $P_1 = P_3$, as from Fig. 5(b) to Fig. 5(c). The procedure stops when value $F(a_3)$ is less than a given precision, as it may be the case after the computation of $F(a_3)$ in Fig. 5(c).

If the t-simple temporal constraint considered has a strictly monotonic delay, the convergence to $firstNeg(F, I)$ is ensured; otherwise, the method may return a value $a > firstNeg(F, I)$, but in this case a still satisfies $F(a) \leq 0$ (with a given precision). It can be observed in practice that the convergence is particularly fast for the delay function associated with agile satellites.
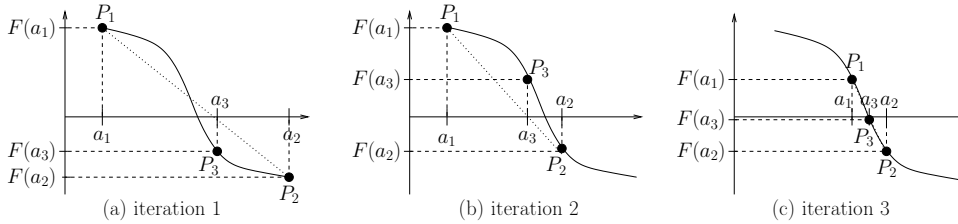

(a) iteration 1   (b) iteration 2   (c) iteration 3

FIGURE 5. Interpolation procedure used for computing $earr_{ct}$

Algorithm 1: Possible way of computing $firstNeg(F, I)$, with I=$[a_1, a_2]$, maxIter a maximum number of iterations, and prec a desired precision

---

1 $firstNeg(F, [a_1, a_2], \text{maxIter}, \text{prec})$
2 begin
3 $\quad f_1 \leftarrow F(a_1);$ **if** $f_1 \leq 0$ **then return** a_1
4 $\quad f_2 \leftarrow F(a_2);$ **if** $f_2 > 0$ **then return** $+\infty$
5 $\quad$ **for** $i = 1$ *to maxIter* **do**
6 $\quad\quad a_3 = (f_1 * a_2 - f_2 * a_1)/(f_1 - f_2)$
7 $\quad\quad f_3 = F(a_3)$
8 $\quad\quad$ **if** $|f_3| < prec$ **then return** a_3
9 $\quad\quad$ **else if** $f_3 > 0$ **then** $(a_1, f_1) \leftarrow (a_3, f_3)$
10 $\quad\quad$ **else** $(a_2, f_2) \leftarrow (a_3, f_3)$
11 $\quad$ **return** a_2

4. GLOBAL CONSISTENCY FOR TSTN

Prop. 4.1 generalizes an STN result to TSTN and shows why maintaining bound arc-consistency is useful.

Proposition 4.1. *If all constraints of a TSTN are made bound arc-consistent using Rules 7-8, then the schedule which assigns to each variable its earliest (resp. latest) possible time is a solution of the TSTN.*

Proof. Let $ct : (x, y, dmin)$ be a constraint of the TSTN. As shown in the proof of Prop. 3.6, the min bounds of x and y after application of Rules 7-8 form a consistent pair of values for ct, as well as their max bounds. Therefore, all min bounds are consistent with each other, as well as all max bounds. $\square$

Another interesting aspect concerns the global consistency of non-extremal values in the domains. For STN, it is known that any value in the domain of a variable after arc-consistency enforcing can be extended to a solution. Prop. 4.2 shows that this result does not hold for TSTN in general, even if delay-monotonicity holds.

Proposition 4.2. *Consider a TSTN made bound arc-consistent using Rules 7-8. Let x be a variable of the TSTN and let a be a value in $\mathbf{d}(x)$ distinct from the min and max bounds of x. Then, there does not necessarily exist a solution of the TSTN in which x is assigned to value a, that is value a is not necessarily globally consistent. The result holds even for delay-monotonic TSTN.*

Proof. Consider a TSTN containing two variables x, y of domain $[0, 2]$ and two t-simple temporal constraints $ct_1 : x - y \geq 0$ and $ct_2 : y - x \geq dmin(x, y)$, with $dmin(x, y) = x/2$ if $x \leq 1$, $1 - x/2$ otherwise. See Fig. 6(a) for a representation of $dmin(x, y)$, which is independent from y in this case. Domains are already bound arc-consistent on this example, since value 0 (resp. 2) of x and value 0 (resp. 2) of y support each other for ct_1 and ct_2.

Consider value 1 in $\mathbf{d}(x)$. For this value to be globally consistent, we must find a value b for y such that $1 - b \geq 0$ (due to ct_1) and $b - 1 \geq dmin(1, b)$ (due to ct_2), that is such that $b \leq 1$ and $b \geq 1.5$, which is impossible. This proves that assignment $x = 1$ cannot be extended to a solution of the TSTN. See Fig. 6(b) for an illustration of the regions of acceptable (x,y)-values for each constraint, showing that value 1 of x has no y-support common to both ct_1 and ct_2. Moreover, all constraints considered here are delay-monotonic (for ct_2, $x + dmin(x, y) - y$ is a non decreasing function of x, as shown in Fig. 6(c)).

$\square$

Prop. 4.3 gives a sufficient condition guaranteeing that every value remaining in the domain of a variable after arc-consistency enforcing is globally consistent.

Proposition 4.3. *Consider a delay-monotonic TSTN made bound arc-consistent using Rules 7-8. If all cycles of the distance graph involve only simple temporal constraints (and not t-simple ones), then for every variable x and every value $a \in \mathbf{d}(x)$, assignment $x = a$ can be extended to a solution of the TSTN.*
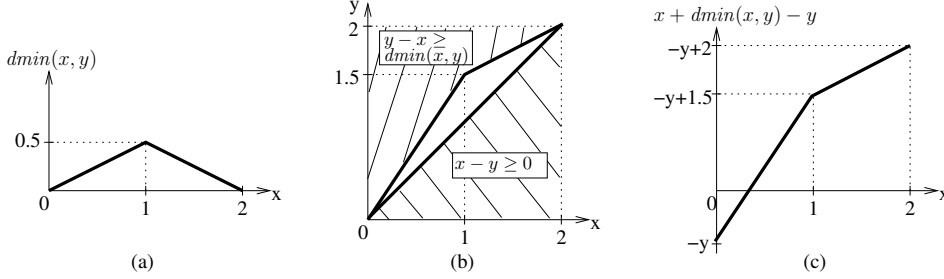
FIGURE 6. Counter-example for global consistency

Proof. The proof is based on the decomposition of the distance graph into Strongly Connected Components (SCCs). See Definition 5.8 page 20 for the definition of SCCs. Let x be a variable of the delay-monotonic TSTN made bound arc-consistent using Rules 7-8. Consider a value $a \in \mathbf{d}(x)$. Let $scc(x)$ denote the strongly connected component containing x. Let $Desc(x)$ (resp. $Ndesc(x)$) denote the set of variables belonging to SCCs that are descendant (resp. non-descendant) of $scc(x)$ in the DAG of SCCs of the distance graph.

- By assigning each variable $y \in Desc(x)$ to its earliest possible time (*i.e.* its min value, denoted y^-), all t-simple temporal constraints holding only over variables in $Desc(x)$ are satisfied (by Prop. 4.1).
- By assigning each variable $y \in Ndesc(x) \setminus scc(x)$ to its latest possible time (*i.e.* its max value, denoted y^+), all t-simple temporal constraints holding only over variables in $Ndesc(x) \setminus scc(x)$ are satisfied (by Prop. 4.1 again).
- Results on standard STN (Corollary 3.4 in [9]) ensure that $x = a$ can be extended to an assignment of variables in $scc(x)$ that satisfies all simple temporal constraints holding only over variables in $scc(x)$.
- The only constraints not checked yet involve one variable z in $scc(x)$, assigned to value b, and one variable $y \notin scc(x)$. These constraints are:
  - either constraints of the form $ct : z - y \geq dmin(y, z)$ with $y \in Desc(x)$; in this case, value y^- assigned to y and value b assigned to z satisfy ct because $delay_{ct}(y^-, b) \leq delay_{ct}(y^-, z^-) \leq 0$ (first inequality obtained by delay-monotonicity and second one by Prop. 4.1);
  - or constraints of the form $ct : y - z \geq dmin(z, y)$ with $y \in Ndesc(x) \setminus scc(x)$; in this case, value y^+ assigned to y and value b assigned to z satisfy ct because $delay_{ct}(b, y^+) \leq delay_{ct}(z^+, y^+) \leq 0$ (first inequality obtained by delay-monotonicity and second one by Prop. 4.1).

As a result, the assignment built satisfies all constraints of the TSTN, which proves that $x = a$ can be extended to a solution.   □

Prop. 4.3 can be applied to the example of Fig. 4, for which cycles only contain simple temporal constraints. It entails that if the t-simple temporal constraints used are delay-monotonic, then for any acquisition index $i \in [1..3]$ and for any

value a remaining in domain $\mathbf{d}(sa_i)$, there exists a consistent schedule in which acquisition acq_i starts at time a.

5. Solving TSTN

The problem considered hereafter is to determine the consistency of a TSTN and to compute the earliest and latest possible times associated with each temporal variable. We also consider a context in which temporal constraints can be successively added and removed from the problem. This dynamic aspect is useful for instance when using local search for solving scheduling problems. In this kind of search, local moves are used for modifying a current schedule. They may correspond to additions and removals of activities, which are translated into additions and removals of temporal constraints. The different techniques used, which generalize existing STN resolution techniques, are successively presented.

5.1. Constraint Propagation

We first use constraint propagation for computing min and max bounds of temporal variables. This standard method is inspired by approaches defined in [5, 13, 25]. The latter correspond to maintaining a list of variables for which constraints holding over these variables must be revised with, for each variable z in the list, the nature of the revision(s) to be performed: (a) if z had its min bound updated, then the min bound of every variable t linked to z by a constraint $t - z \geq c$ must be revised; (b) if z had its max bound updated, then the max bound of every variable t linked to z by a constraint $z - t \geq c$ must be revised.

Compared to standard STN approaches, we choose for TSTN a constraint propagation scheme in which a list containing constraints to be revised is maintained, instead of a list containing variables. This list is partitioned into two sub-lists, the first one containing constraints to be revised which may modify a min bound (constraints $y - x \geq dmin(x,y)$ awoken following a modification of $\min x$, which may modify $\min y$), and the second one containing constraints to be revised which may modify a max bound (constraints $y - x \geq dmin(x,y)$ awoken following a modification of $\max y$, which may modify $\max x$). Compared to the version maintaining lists of variables, maintaining lists of constraints allows some aspects to be more finely handled (see below for more details). The idea of distinguishing modifications of min bounds and max bounds of variables for propagation is present in many solvers, including those based on constraint logic programming for finite domains [8].

Last, a t-simple temporal constraint is revised using Rules 7 and 8 of Prop. 3.6.

5.2. Negative Cycle Detection

With bounded domains of values, the establishment of arc-consistency for STN is able to detect inconsistency. However, the number of constraint revisions required for deriving inconsistency may be prohibitive compared to STN approaches

defined in [4, 5], which use the fact that STN inconsistency is equivalent to the existence of a cycle of negative length in the distance graph.

The basic idea of these existing STN approaches consists in detecting such negative cycles on the fly by maintaining so-called *propagation chains*. The latter can be seen as explanations for the current min and max bounds of the different variables. A constraint $y - x \geq c$ is said to be active with regard to min bounds (resp. max bounds) if and only if the last revision of this constraint is responsible for the last modification of the min of y (resp. the max of x). It is shown in [4] that if there exists a cycle in the directed graph where an arc is associated with each active constraint with regard to min bounds, then the STN is inconsistent. The intuition is that if a propagation cycle $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n \rightarrow x_1$ is detected for min bounds, then this means that the min value of x_1 modified the min value of x_2... which modified the min value of x_n which modified the min value of x_1. By traversing this propagation cycle a sufficient number of times, the domain of x_1 can be entirely pruned. The same result holds for the directed graph containing one arc per active constraint with regard to max bounds.

These results cannot however be directly reused for t-simple temporal constraints, because for TSTN in general, the existence of a propagation cycle does not necessarily imply inconsistency, as shown in the examples below.

Example 5.1. Let $dmin$ be the minimum distance function defined by $dmin(a, b) = (1 - a)/2$. Let $(V, C) = (\{x, y\}, \{ct_1 : x - y \geq 0, ct_2 : y - x \geq dmin(x, y)\})$ be a TSTN containing two temporal variables of initial domain $[0, 1]$ and two constraints. Constraints ct_1 and ct_2 can also be written as $x \geq y$ and $y \geq (1 + x)/2$ respectively. The delay functions associated with ct_1 and ct_2 are strictly monotonic (for ct_2, it equals $delay_{ct_2}(a, b) = a + dmin(a, b) - b = (1 + a)/2 - b$).

Propagating ct_2 using Rule 7 updates the min of y and gives $\mathbf{d}(y) = [1/2, 1]$. Propagating ct_1 using the same rule then updates the min of x and gives $\mathbf{d}(x) = [1/2, 1]$. The result obtained is a cycle of propagation since the min value of x modified the min of y which itself modified the min of x. On standard STN, the existence of such a cycle means inconsistency. Such a conclusion is wrong for the TSTN considered, because for instance assignment $x = 1$, $y = 1$ is consistent.

Example 5.2. The same phenomenon may happen when propagating max values of variables. For example, consider a TSTN involving two variables x and y of initial domain $[0..1]$, and two temporal constraints $ct_1 : x - y \geq 0$ and $ct_2 : y - x \geq dmin(x, y)$ with $dmin(x, y) = x$. Constraints ct_1 and ct_2 can also be written as $y \leq x$ and $x \leq y/2$ respectively. They are both strictly delay-monotonic (for ct_2, $delay_{ct_2}(a, b) = a + dmin(a, b) - b = 2a - b$).

Propagating ct_2 using Rule 8 updates the max of x and gives $\mathbf{d}(x) = [0, 1/2]$. Propagating ct_1 using the same rule then updates the max of y and gives $\mathbf{d}(y) = [0, 1/2]$. The result obtained is a cycle of propagation since the max value of y modified the max of x which itself modified the max of y. On standard STN, the existence of such a cycle means inconsistency. Such a conclusion is wrong for the TSTN considered, because for instance assignment $x = 0$, $y = 0$ is consistent.

On the two examples provided, the existence of a propagation cycle does not imply inconsistency. The reason is that in TSTN, domain reductions obtained by traversing cycles again and again may become smaller and smaller and consequently may not necessarily prune all values in the domains. This is what happens here: after n traversals of the propagation cycle between x and y, we get $\mathbf{d}(x) = [1 - 1/2^n, 1]$ in Example 5.1 and $\mathbf{d}(y) = [0, 1/2^n]$ in Example 5.2. The finite computer precision implies that cycle traversals stop at some step, but potentially only after many iterations.

In the following, we provide sufficient conditions for inferring inconsistency in case of propagation cycle detection. Examples 5.1 and 5.2 show that strict monotonicity of the delay function does not suffice. The conditions we propose are directly based on monotonicity properties of minimum duration functions $dmin$. They guarantee that a propagation cycle does not become "less negative" when traversed again and again.

Definition 5.3. A t-simple temporal constraint $ct : (x, y, dmin)$ is said to have a non-decreasing duration (resp. non-increasing duration) iff function $dmin$ is non-decreasing (resp. non-increasing) over its two arguments.

Sufficient conditions for inferring inconsistency in case of detection of a propagation cycle are given in Prop. 5.5 and 5.6.[3] These conditions are derived from the basic result given in Prop. 5.4. The latter shows that a non-decreasing duration implies that when the start time of a constraining transition is shifted forward, then the earliest arrival time is shifted forward by at least the same amount. On the other hand, a non-increasing duration implies that when the arrival time of a constraining transition is shifted backward, then the latest departure time is shifted backward by at least the same amount. See Fig. 7 for an illustration.
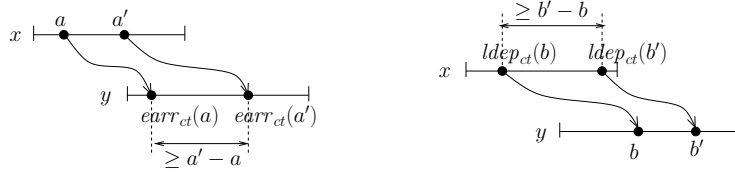

FIGURE 7. Influence of duration-monotonicity over earliest arrival and latest departure times (left: non-decreasing duration; right: non-increasing duration)

Proposition 5.4. *Let $ct : (x, y, dmin)$ be a delay-monotonic constraint.*
- *If ct has a non-decreasing duration, then for all $a, a' \in \mathbf{D}(x)$ such that $a \leq a'$, $(earr_{ct}(a) \neq \min(\mathbf{D}(y))) \rightarrow (earr_{ct}(a') \geq earr_{ct}(a) + (a' - a))$.*
- *If ct has a non-increasing duration, then for all $b, b' \in \mathbf{D}(y)$ such that $b \leq b'$, $(ldep_{ct}(b') \neq \max(\mathbf{D}(x))) \rightarrow (ldep_{ct}(b) \leq ldep_{ct}(b') - (b' - b))$.*

---

[3]In [23] and [22], a sufficient condition called *shift-monotonicity* was proposed. However, this condition does not actually suffice (mistake in one of the proof given in [22]).

Proof. Let us consider a delay-monotonic t-simple temporal constraint with a non-decreasing duration. Let $a, a' \in \mathbf{D}(x)$ such that $a \leq a'$. Assume that $earr_{ct}(a) \neq \min(\mathbf{D}(y))$. Then, it is possible to show that for every $\epsilon \in]0, earr_{ct}(a) - \min(\mathbf{D}(y))]$, $delay_{ct}(a', earr_{ct}(a) + (a' - a) - \epsilon) > 0$. Indeed,

$$delay_{ct}(a', earr_{ct}(a) + (a' - a) - \epsilon)$$
$$= a' + dmin(a', earr_{ct}(a) + (a' - a) - \epsilon) - (earr_{ct}(a) + (a' - a) - \epsilon)$$
$$= a + dmin(a', earr_{ct}(a) + (a' - a) - \epsilon) - (earr_{ct}(a) - \epsilon)$$
$$\geq a + dmin(a, earr_{ct}(a) - \epsilon) - (earr_{ct}(a) - \epsilon) \text{ (by non-decreasing duration)}$$

This proves that $delay_{ct}(a', earr_{ct}(a) + (a' - a) - \epsilon) \geq delay_{ct}(a, earr_{ct}(a) - \epsilon) > 0$ (strict inequality obtained by definition of $earr_{ct}(a)$). As set $]0, earr_{ct}(a) - \min(\mathbf{D}(y))]$ is not empty and as ct is delay-monotonic, this entails that $earr_{ct}(a') \geq earr_{ct}(a) + (a' - a)$.

Conversely, consider a delay-monotonic t-simple temporal constraint with a non-increasing duration. Let $b, b' \in \mathbf{D}(x)$ such that $b \leq b'$. Assume that $ldep_{ct}(b') \neq \max(\mathbf{D}(x))$. Then, it is possible to show that for every $\epsilon \in]0, \max(\mathbf{D}(x)) - ldep_{ct}(b')]$, $delay_{ct}(ldep_{ct}(b') - (b' - b) + \epsilon, b) > 0$. Indeed,

$$delay_{ct}(ldep_{ct}(b') - (b' - b) + \epsilon, b)$$
$$= ldep_{ct}(b') - (b' - b) + \epsilon + dmin(ldep_{ct}(b') - (b' - b) + \epsilon, b) - b$$
$$= ldep_{ct}(b') + \epsilon + dmin(ldep_{ct}(b') - (b' - b) + \epsilon, b) - b'$$
$$\geq ldep_{ct}(b') + \epsilon + dmin(ldep_{ct}(b') + \epsilon, b') - b' \text{ (by non-increasing duration)}$$

This proves that $delay_{ct}(ldep_{ct}(b') - (b' - b) + \epsilon, b) \geq delay_{ct}(ldep_{ct}(b') + \epsilon, b') > 0$ (strict inequality obtained by definition of $ldep_{ct}(b')$). As set $]0, \max(\mathbf{D}(x)) - ldep_{ct}(b')]$ is not empty and as ct is delay-monotonic, this entails that $ldep_{ct}(b) \leq ldep_{ct}(b') - (b' - b)$. □

Proposition 5.5. *Consider a TSTN and a propagation cycle over min values (resp. max values). If all constraints involved in the cycle are delay-monotonic and have a non-decreasing duration (resp. a non-increasing duration), then the TSTN is inconsistent.*

Proof. Assume that propagation cycle $x_1 \to x_2 \to \ldots \to x_n \to x_1$ is detected for min bounds, following the revision of a constraint linking x_n and x_1. Let $\delta > 0$ be the increase in the min bound of x_1 following this last constraint revision. Non-decreasing durations guarantee that if the cycle is traversed again, the min bounds of $x_2, \ldots, x_n$ will be increased again by at least δ, thanks to Prop 5.4. After a sufficient number of cycle traversals, the domain of one variable of the cycle becomes empty. The proof is similar for a propagation cycle over max values. □

Proposition 5.6. *For a TSTN such that all cycles of the distance graph involve only simple temporal constraints (and not t-simple ones), the existence of a propagation cycle implies inconsistency.*

Proof. Consider a TSTN such that all cycles of the distance graph involve only simple temporal constraints. Assume that a propagation cycle is detected. As a propagation cycle necessarily corresponds to a cycle in the distance graph, the propagation cycle detected contains only simple temporal constraints. As these constraints both have a non-decreasing and non-increasing duration, Prop. 5.5 allows inconsistency to be inferred.   □

In the agile satellite application which motivates this work, the minimum distance functions used are not necessarily non-decreasing or non-increasing, as can be seen in Fig. 2, but Prop. 5.6 applies for case studies considered. Note that checking the satisfaction of the condition given in Prop. 5.6 is easy (linear in the number of variables and constraints).

The results provided can also be applied to time-dependent scheduling. Among minimum duration functions given in Table 1, the ones in rows 1, 3, and 4 are non-decreasing, and the detection of a propagation cycle for min values implies inconsistency; the ones in rows 2 and 5 are non-increasing, and the detection of a propagation cycle for max values implies inconsistency.

If the sufficient condition given in Prop. 5.6 is not satisfied, several options can be considered:

(1) first, it is possible not to consider constraints with a decreasing duration (resp. increasing duration) in propagation chains for min values (resp. for max values); this approach is correct but may lose time in propagation cycles;

(2) second, it is possible not to consider constraints with a decreasing duration (resp. increasing duration) in propagation chains for min values (resp. for max values), as in the first point, but to stop propagating constraints when some time-limit or some precision is reached; the price to pay is then that inconsistency may not be detected;

(3) a third option is to consider a TSTN as inconsistent as soon as a propagation cycle is detected, even if it contains constraints whose minimum distance functions do not have the right theoretical properties; this may be incorrect in the sense that it may wrongly conclude to inconsistency.

In terms of implementation, we perform on the fly detection of propagation cycles based on an efficient data structure introduced in [2]. The latter is used for maintaining a topological order of nodes in the graphs of propagation of min and max bounds. When no topological order exists, the graph contains a cycle.

5.3. Complexity

Prop. 5.7 generalizes polynomial complexity results available on STN to TSTN, and therefore to time-dependent scheduling. It gives conditions allowing bound arc-consistency to be established using a polynomial number of constraint revisions. Let us emphasize that it does not give conditions allowing such a consistency property to be established in *polytime*. The distinction between number of revisions and time is due to the fact that very few assumptions are made on the

kind of *dmin* functions considered. In particular, *dmin* functions can be hard to compute.

Proposition 5.7. *Given a TSTN (V, C), if the existence of a propagation cycle implies inconsistency, then the algorithm using Rules 7-8 for propagation, plus a FIFO ordering on the propagation queue and a propagation cycle detection, establishes bound arc-consistency in $O(|V||C|)$ constraint revisions (bound independent of the size of the variable domains).*

Proof. Similar to the result stating that the number of arc revisions in the Bellman-Ford's FIFO label-correcting algorithm is $O(|V||C|)$.   □

5.4. Constraint Depropagation for Dynamic TSTN

Constraint propagation techniques are directly able to handle constraint addition or constraint strengthening. As for constraint removal or constraint weakening, constraint depropagation strategies defined in [25] for STN can be directly reused. These strategies allow min and max bounds of temporal variables to be recomputed at minimum cost. They avoid reinitializing all variable domains and repropagating all constraints from scratch when a constraint is removed or weakened. The basic idea is to use propagation chains in order to determine which variable domains must be reinitialized and which constraints need to be revised. More precisely, when a constraint $y - x \geq dmin(x, y)$ is removed or weakened, if this constraint is active with regard to the min bound of y (resp. the max bound of x), then the min bound of y (resp. the max bound of x) is reinitialized to the value it had before any propagation. This reinitialization may trigger other reinitializations. TSTN constraints of the form $y - z \geq dmin(z, y)$ (resp. $z - x \geq dmin(x, z)$) are then added to the list of constraints to be revised from the point of view of min bounds (resp. max bounds).

The only difference when compared to standard STN techniques is the use of lists of constraints to be revised instead of lists of variables. This allows constraint depropagation to be slightly less costly: on the example of reinitialization of the min bound of y, the standard STN version would add to the list of variables to be propagated every variable z linked to y by some constraint $y - z \geq dmin(z, y)$, and doing so would repropagate in the end all constraints of the form $u - z \geq dmin(z, u)$, even those with $u \neq y$.

5.5. Constraint Revision Ordering

A last technique is used for minimizing the number of constraint revisions. This can be particularly useful for TSTN, for which revising one constraint can be significantly more costly than for STN. The proposed approach extends a technique developed for STN$^-$ [13], a sub-class of STN in which every constraint must be rewritable as $y - x \geq c$ with $c \geq 0$. The idea consists in building the strongly connected components of the distance graph, in ordering them in topological order, and in using this order to determine which constraint to propagate first. We first recall definitions concerning strongly connected components.

Definition 5.8. Let $G = (V, A)$ be a directed graph with V the set of nodes and A the set of arcs. A *Strongly Connected Component* (SCC) of G is a maximum sub-graph G' of G such that there exists in G' a path from every node to every other node.

The DAG (*Directed Acyclic Graph*) of SCCs of G is the directed graph whose nodes are the SCCs of G and which contains an arc from SCC c_1 to SCC c_2 iff there exists in G an arc from one of the nodes of c_1 to one of the nodes of c_2.

A topological order of SCCs is an order $\preceq$ where each SCC c is put strictly after each of its parents c' in the DAG of SCCs ($c' \prec c$). Given a node x in graph G, $scc(x)$ denotes the unique SCC of G that contains x.

Propagating temporal constraints following a topological order of SCCs in the distance graph boils down to using the fact that solving shortest path problems is easier for acyclic graphs than for arbitrary graphs. To apply this result, constraints to be propagated are ordered according to a topological order of SCCs. More precisely, concerning the propagation of min bounds, we propagate first constraints $y - x \geq dmin(x, y)$ such that $scc(y)$ is maximum in the order of SCCs and, in case of equality, we propagate first constraints such that $scc(x) \neq scc(y)$, to postpone as much as possible the propagation of "internal" constraints in an SCC. To break remaining ties, a FIFO ordering strategy is used. Concerning the propagation of max bounds, constraints are ordered by increasing $scc(x)$ and, in case of equality, we propagate first constraints such that $scc(y) \neq scc(x)$, and break remaining ties using a FIFO ordering strategy. In the example of Fig. 4, SCCs are represented as dotted boxes. A bad propagation order for min bounds would consist in propagating first the constraint between sa_2 and ea_1, and then the constraint between sa_1 and ea_3. A good order, consistent with the order of SCCs, would consist in using the opposite strategy.

Compared to the way SCCs are used in [13] for STN$^-$, the method we propose is adapted not only to general STN, but also to TSTN. In terms of implementation, in order to avoid recomputing the DAG of SCCs from scratch after each constraint addition or removal, we use recent algorithms proposed for maintaining SCCs in a dynamic graph [14, 24].

5.6. Specific Management of Unary Constraints

In the constraint propagation mechanism used, unary constraints $x \leq c$ and $x \geq c$ are actually revised first because their revision is easy. We give another argument in favor of handling such constraints separately.

Consider n temporal variables $x_1, \ldots, x_n$ of domain $[0, 1]$, n unary domain constraints $\forall i \in [1..n], x_i \in [0, 1]$, and $n - 1$ t-simple temporal constraints $\forall i \in [1..n - 1], x_{i+1} - x_i \geq 1/n$. The associated distance graph without unary constraints is given in Fig. 8(a). It contains one SCC per variable x_i. Following the order of SCCs, once unary constraints are revised, min bounds of variables are computed in $n - 1$ constraint revisions (one left-right traversal of the chain defined by SCCs), as well as max bounds (one right-left traversal of the chain defined by SCCs).

If a global reference temporal position x_0 of domain $[0, 0]$ is introduced, unary domain constraints $x_i \geq 0$ and $x_i \leq 1$ are transformed into $x_i - x_0 \geq 0$ and $x_0 - x_i \geq -1$ respectively. The distance graph obtained is given in Fig. 8(b). It contains a unique SCC. With a FIFO management of the propagation queue, it can be shown that in the worst case, a quadratic number of constraint revisions is performed to compute min and max bounds of variables (propagation in the order opposite to the order defined by SCCs of Fig. 8(a)). By handling unary constraints specifically, we avoid considering them in the distance graph, and doing so they do not hide the real structure of the problem.
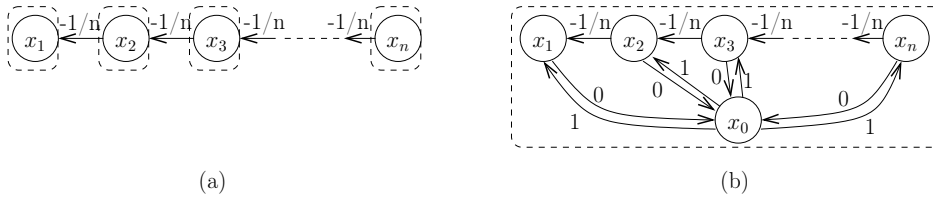

(a)   (b)

FIGURE 8. Distance graph without (a) and with (b) a reference temporal position x_0, and associated SCCs (dotted boxes)

6. EXPERIMENTS

All techniques presented in Section 5 (constraint propagation, propagation cycle detection, constraint depropagation, SCC ordering, specific management of unary constraints) are integrated and simultaneously used in a scheduling tool based on local search. The local search aspect entails that doing/undoing a local move is fast, similarly to constraint-based local search tools Comet [16] and LocalSolver [3]. Our STN/TSTN solver is implemented in Java. Results are obtained on an Intel i5-520 1.2GHz, 4GBRAM.

Experiments not detailed here were first performed on STN obtained from scheduling problems of the SMT-LIB. The objective was to evaluate the propagation heuristics based on a topological ordering of the SCCs. This heuristics appears to be a robust strategy, which significantly decreases the number of constraint revisions on some problems. More precisely, for consistent STN, the SCC heuristics is always at least as good as a pure FIFO heuristics, but for inconsistent STN, it is not always the fastest strategy for proving inconsistency.

We detail below experiments realized on TSTN in the context of agile satellites. Even if the TSTN approach is not deployed and actually used for satellites, the scenario considered is realistic in terms of acquisitions to be performed. More precisely, the problem considered here is a simple no overlapping constraint over an ordered sequence of n acquisitions $acq_1 \rightarrow \ldots \rightarrow acq_n$, with n varying between 5 and 13. These acquisitions correspond to ground strips located between the north of Spain and the north of France. The no-overlapping constraint between

acquisitions can be written as a set of t-simple temporal constraints of the form $s_{i+1} - e_i \geq minAttTransTime(Eatt_i(e_i), Satt_{i+1}(s_{i+1}))$ with, for an acquisition j, s_j/e_j the start/end time of this acquisition, and $Satt_j(t)/Eatt_j(t)$ the attitudes required to start/end j at time t. In addition, simple temporal constraints are used to define the constant duration of each acquisition.

Two methods are compared: (1) a TSTN approach in which exact transition times between acquisitions are used, and (2) an STN approach in which upper bounds on transition times are pre-computed, by sampling on the different possible start times of the transitions. The schedule obtained in both cases is flexible in the sense that the domains of values after propagation over STN/TSTN are generally not reduced to singletons. The criterion considered for comparing the two approaches is the mean temporal flexibility $mtf = \frac{1}{|V|} \sum_{x \in V} (max(x) - min(x))$, measured as the mean, over all temporal variables $x \in V$, of the difference between the latest and earliest possible times associated with x. Such a flexibility is important in practice to offer as much freedom as possible concerning the choice of an angle of acquisition of ground strips, which influences image quality.

Three scenarios are considered. In the first one, acquisitions correspond to strips of length about 80km, to be observed with a scanning direction of 0 degrees (angle between the trace of the satellite on the ground and the direction in which the strip must be scanned). Fig. 9(a) shows that in this case, the temporal flexibility obtained with TSTN only slightly improves on the flexibility obtained with STN. The reason is that if all acquisitions are realized with a scanning direction of 0 degrees, the minimum transition times between acquisitions are almost independent of the triggering time of transitions: they are only time-dependent when the rotation around the pitch axis is the most constraining from a temporal point of view, compared to the rotation around the roll axis.

In the second scenario, the scanning direction becomes 30 degrees. Fig. 9(b) shows that the temporal flexibility obtained with TSTN is better than with STN (improvement of about 20 seconds in flexibility), and that the flexibility gap between STN and TSTN increases with the number of scheduled acquisitions.

In the third and last scenario, the length of the strips considered becomes approximately 40km, and the scanning direction is chosen at random for each strip. In this case, Fig. 9(c) shows that the STN approach only allows sequences of length 5 and 6 to be scheduled. It concludes to an inconsistency of the problem for $n \geq 7$. On the other hand, the TSTN approach schedules all 13 acquisitions considered. One reason explaining these results is that the more distinct the scanning directions are, the more the minimum transition times between acquisitions depend on the triggering time of the transitions. The possibility to have distinct scanning directions is important in practice. It indeed allows acquisitions defined as polygons to be split into strips whose orientation can be freely chosen, which can reduce the number of strips to be scanned.

To give an idea of computation times, for 13 acquisitions added one by one to the current schedule, a precision of one second on dates, and a maximum number of iterations equal to 10^4 for computing $firstNeg$ and $lastNeg$, the TSTN approach takes about 2ms per acquisition addition. With STN, the computation time is
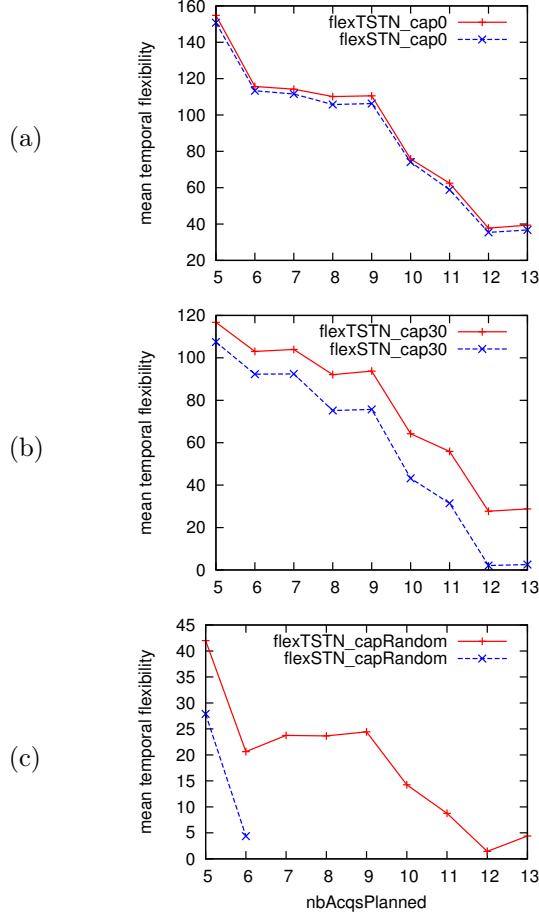
FIGURE 9. Comparison of temporal flexibilities, in seconds, obtained with precomputed upper bound on transition times (flexSTN) and with exact transition times (flexTSTN)

less than 0.1ms per addition. For precisions of 10^{-1}, 10^{-2}, and 10^{-3} second on dates, computation times with TSTN respectively become 3ms, 12ms, and 66ms per addition. A typical technique can consist in first searching for schedules with a fast coarse-grained approach, before using a finer precision.

In practice, satellite acquisitions may have to be planned over a full-day period, and plans obtained may contain several hundreds of acquisitions. The latter are chosen from a larger set of candidate acquisitions submitted by end-users. Experiments on such real test case scenarios remain to be done.

7. Conclusion

This paper introduced TSTN, their properties, resolution techniques, and their application to agile satellites. It showed that some standard STN properties can be generalized to TSTN, whereas others do not, especially concerning global consistency issues or detection of propagation cycles. Compared to time-dependent scheduling, one specificity of TSTN is that they use a minimum distance function $dmin(x, y)$ depending on both the start time of the transition (x) and the end time of the transition (y). This can be useful in any problem in which transitions must be made between the tracking of two moving targets whose trajectories are completely known in advance. For agile satellites, targets correspond to points at the Earth surface, and point trajectories can be fully determined thanks to laws of orbital mechanics. Models previously proposed in time-dependent scheduling do not cover such problems.

For future work directions, it would be interesting to combine TSTN with optimization. Indeed, in the space domain and in other domains as well, the quality of an acquisition usually depends on the angle between the pointing direction of the observation instrument and the area to be acquired. For optical observation satellites, the optimal angle corresponds to an acquisition at the vertical of the satellite (at the *nadir*). For other kinds of satellites, the optimal angle may correspond to an acquisition at the *astronomical horizon*, *i.e.* when the satellite pointing direction is perpendicular to the zenith-nadir axis for the target. In any case, in order to handle such problems and assign temporal variables to good quality values, it would be interesting for instance to try and reuse works combining STN and linear objective functions [19]. Moreover, in a context where the memory size of an acquisition is not fully known in advance, especially due to non-deterministic ratio for image compression, the duration of data downloads is actually uncertain. Techniques for facing this uncertainty, such as those available on STN with uncertainties (STNU [27]), should therefore also be designed.

References

[1] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[2] M. Bender, R. Cole, E. Demaine, M. Farach-Colton, and J. Zito. Two simplified algorithms for maintaining order in a list. In *Proc. of ESA-02*, pages 152–164, 2002.

[3] T. Benoist, B. Estellon, F. Gardi, R. Megel, and K. Nouioua. LocalSolver 1.x: a black-box local-search solver for 0-1 programming. *4OR: A Quarterly Journal of Operations Research*, 9(3):299–316, 2011.

[4] R. Cervoni, A. Cesta, and A. Oddi. Managing dynamic temporal constraint networks. In *Proc. of AIPS-94*, pages 13–18, 1994.

[5] A. Cesta and A. Oddi. Gaining efficiency and flexibility in the simple temporal problem. In *Proc. of TIME-96*, pages 45–50, 1996.

[6] Challenge ROADEF-03. Handling the mission of Earth observation satellites, 2003. http://challenge.roadef.org/2003/fr/.

[7] T. Cheng, Q. Ding, and B. Lin. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research*, 152:1–13, 2004.

[8] P. Codognet and D. Diaz. Compiling constraints in clp(FD). *Journal of Logic Programming*, 27(3):185–226, 1996.

[9] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[10] R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[11] L. R. Ford and D. R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[12] S. Gawiejnowicz. *Time-dependent scheduling*. Springer, 2008.

[13] A. Gerevini, A. Perini, and F. Ricci. Incremental algorithms for managing temporal constraints. In *Proc. of ICTAI-96*, pages 360–365, 1996.

[14] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. Tarjan. Incremental cycle detection, topological ordering, and strong component maintenance. *ACM Transactions on Algorithms*, 8(1), 2012.

[15] P. Van Hentenryck, Y. Deville, and C. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57(2-3):291–321, 1992.

[16] P. Van Hentenryck and L. Michel. *Constraint-based local search*. The MIT Press, 2005.

[17] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6:367–381, 2002.

[18] U. Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Sciences*, 7(2):95–132, 1974.

[19] P. Morris, R. Morris, L. Khatib, S. Ramakrishnan, and A. Bachmann. Strategies for global optimization of temporal preferences. In *Proc. of CP-04*, pages 408–422. Springer, 2004.

[20] L. Planken, M. de Weerdt, and R. van der Krogt. P3C: a new algorithm for the simple temporal problem. In *Proc. of ICAPS-08*, pages 256–263, 2008.

[21] L. Planken, M. de Weerdt, and N. Yorke-Smith. Incrementally solving STNs by enforcing partial path consistency. In *Proc. of ICAPS-10*, pages 129–136, 2010.

[22] C. Pralet and G. Verfaillie. Réseaux temporels simples étendus. Application à la gestion de satellites agiles. In *Proc. of JFPC-12*, pages 264–273, 2012.

[23] C. Pralet and G. Verfaillie. Time-dependent simple temporal networks. In *Proc. of CP-12*, pages 608–623, 2012.

[24] L. Roditty and U. Zwick. Improved dynamic reachability algorithms for directed graphs. *SIAM Journal on Computing*, 37(5):1455–1471, 2008.

[25] I. Shu, R. Effinger, and B. Williams. Enabling fast flexible planning through incremental temporal reasoning with conflict extraction. In *Proc. of ICAPS-05*, pages 252–261, 2005.

[26] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120:81–117, 2000.

[27] T. Vidal and H. Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence*, 11(1):23–45, 1999.

[28] T. Warshall. A theorem on Boolean matrices. *Journal of the ACM*, 9:11–12, 1962.

[29] L. Xu and B. Choueiry. A new efficient algorithm for solving the simple temporal problem. In *Proc. of TIME-ICTL-03*, pages 210–220, 2003.