Time-dependent Simple Temporal Networks.

C. Pralet, G. Verfaillie

Time-dependent Simple Temporal Networks

Cédric Pralet, Gérard Verfaillie

ONERA – The French Aerospace Lab, F-31055, Toulouse, France
{cedric.pralet,gerard.verfaillie}@onera.fr

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 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" 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 [1]). 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 [2]).

In this paper, we propose an extension of the STN framework and of STN algorithms. This extension is illustrated on an application from the space domain. The latter corresponds to the management of Earth observation satellites such as those of the *Pleiades* system. Such satellites are moving around the Earth on a circular, low-altitude orbit (several hundreds of kilometers). They are said to be *agile*, which means that they have the capacity to move around the three axes (roll, pitch, and yaw). This agility allows them to point to the right, left, in

front of, or behind of the Earth point at the vertical of the satellite at each time (Nadir). The mission of these satellites is to perform acquisitions of polygons at the Earth surface. These polygons are split into strips which must be scanned using an observation instrument fixed on the satellite. Scanning 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 on each of the three axes.

In the agile satellite context, contrary to the simplified version of the 2003 ROADEF Challenge [3], the minimum transition time taken by a maneuver 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 the first acquisition ends [4]. Transition times may vary of about ten seconds on the examples provided in Fig. 1, duration during which the satellite covers between 50 and 100 kilometers on the ground. Fig. 1 also shows how diverse minimum transition times evolution schemes can be. They are obtained 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 moves of the satellite.

This context motivates the need for a new modeling framework for problems in which the minimum transition time between two activities can depend on the precise time at which the transition is triggered. This aspect is close to work on *time-dependent scheduling* [5, 6], where transition times take particular forms, piecewise constant or piecewise linear (these forms cannot be directly reused here). It also appears in applications such as congestion-aware logistics, in which traveling times depend on the hour of the day, due to traffic. The framework proposed, called Time-dependent STN, is first introduced (Sect. 2). Techniques are then defined for computing the earliest and latest times associated with each temporal variable (Sect. 3 and 4). These techniques are used for scheduling activities of an agile satellite, in the context of a local search algorithm (Sect. 5).

2 Towards Time-dependent STN

2.1 Simple Temporal Networks (STN)

We first recall some definitions associated with STN. In the following, the domain of values of a variable x is denoted $\mathbf{d}(x)$.

Definition 1. *An STN is a pair (V, C) with V a finite set of continuous variables whose domain is a closed interval $[l, u] \subset \mathbb{R}$, 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 $\beta \in \mathbb{R} \cup \{+\infty\}$. 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 domains of possible values of variables, can be formulated as simple temporal constraints $x - x_0 \in [\alpha, \beta]$, with x_0 a variable of domain $[0, 0]$ playing the role of a temporal reference.

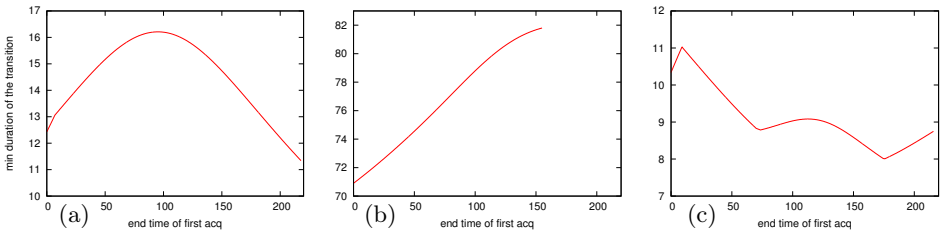

Fig. 1. Minimum durations, in seconds, for a satellite maneuver from a strip i ending at point of latitude-longitude 41°17′48″N-2°5′12″E to a strip j starting at point of latitude-longitude 42°31′12″N-2°6′15″E, for different scanning angles with regard to the trace of the satellite on the ground: (a) 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°

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 [1] (some of these results are similar to earlier work on PERT and critical path analysis):

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 temporal 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 or arc-consistency filtering [7–10];
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 or path-consistency filtering [1, 11–13], which produces the *minimal network* of the STN [14].

Example 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. The duration taken by the download of acq_i is denoted Dd_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. 2 gives the distance graph of the obtained STN.

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

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

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

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

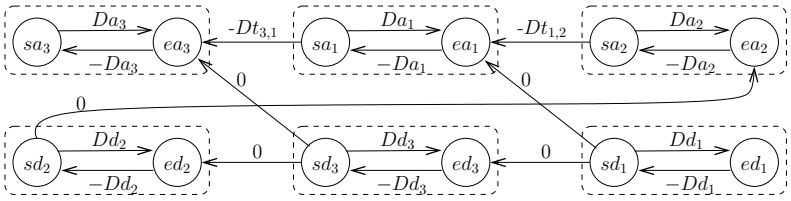

Fig. 2. Distance graph (reference temporal position x_0 is not represented)

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

Definition 2. *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.

To illustrate why having a minimum distance function *dmin* depending on both x and y is useful, 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* [5, 6], for which the processing time of a task only depends on the start time of this task (t-simple temporal constraint $y - x \geq dmin(x)$ with $dmin(x)$ the processing time of the task when this task starts 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 3. *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 Let us reconsider the example involving 3 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 with the initial STN model is that 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. 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. 3).

$$sa_1 - ea_3 \geq minAttTransTime(Eatt_3(ea_3), Satt_1(sa_1)) \quad (5)$$

$$sa_2 - ea_1 \geq minAttTransTime(Eatt_1(ea_1), Satt_2(sa_2)) \quad (6)$$

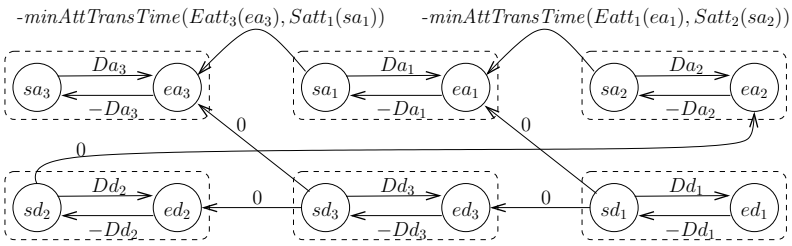

Fig. 3. TSTN distance graph (temporal reference x_0 is not represented)

3 Arc-Consistency of t-Simple Temporal Constraints

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

Definition 4. *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 5. *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:*

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

Definition 5 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 t-simple temporal constraint. The notion of delay-monotonicity can be related to the notion of monotonic constraints, defined for instance in [15]. One difference is that in TSTN, domains considered are continuous.

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).[1]

Definition 6. *The functions of earliest arrival time and latest departure time associated with a t-simple temporal constraint* $ct : (x, y, dmin)$ *are functions denoted* $earr_{ct}$ *and* $ldep_{ct}$ *and 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, $earr_{ct}(a)$ gives the smallest arrival time in y without delay if the transition from x is triggered at time a. $ldep_{ct}(b)$ gives the latest triggering time of the transition in x for an arrival in b without delay.

Prop. 1 shows that these two functions help establishing bound arc-consistency.

Proposition 1. *Bound arc-consistency for a t-simple temporal constraint* $ct : (x, y, dmin)$ *can be enforced using the following domain modification rules:*

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

Proof. Assume that $earr_{ct}(\min(\mathbf{d}(x))) \neq +\infty$ and $ldep_{ct}(\max(\mathbf{d}(y))) \neq -\infty$. By definition of $earr_{ct}$ and $ldep_{ct}$, we then have $delay(\min(\mathbf{d}(x)), earr_{ct}(\min(\mathbf{d}(x)))) \leq 0$ and $delay(ldep_{ct}(\max(\mathbf{d}(y))), \max(\mathbf{d}(y))) \leq 0$. Hence min and max bounds of x and y all have a support after application of Rules 7-8 if domains obtained are not empty.

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. 2 below establishes the equivalence between bound arc-consistency and arc-consistency for delay-monotonic constraints.

Proposition 2. *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 $\forall a \in [x^-, x^+]$, $delay_{ct}(a, b) \geq delay_{ct}(x^-, b) > 0$ (by delay-monotonicity and by definition of $earr_{ct}(x^-)$). Conversely, if $b \geq earr_{ct}(x^-)$, then $delay_{ct}(x^-, b) \leq delay_{ct}(x^-, earr_{ct}(x^-)) \leq 0$, hence b is supported by x^-. Therefore, y-values pruned by Rule 7 are those that have no support over x. Similarly, it can be shown that x-values pruned by Rule 8 are those that have no support over y.

[1] 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.

When delay-monotonicity is violated, Rules 7-8 can be applied but they do not necessarily establish arc-consistency. Prop. 3 generalizes a STN result to TSTN and shows why maintaining bound arc-consistency is useful.

Proposition 3. *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. 1, 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.

Concerning the way *earr* and *ldep* can be computed in practice, for simple temporal constraints $y - x \geq c$, an analytic formulation of *earr* and *ldep* can be given. However, in the general case, $\mathit{firstNeg}(F, I)$ and $\mathit{lastNeg}(F, I)$ must be computed, which corresponds to an optimization problem in itself. An iterative method for approximating $\mathit{firstNeg}(F, I = [a_1, a_2])$ is given in Algorithm 1. This method generalizes the *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. If the delay in $P_3 = (a_3, F(a_3))$ is positive (resp. negative), then the mechanism is applied again by taking $P_1 = P_3$ (resp. $P_2 = P_3$). If the t-simple temporal constraint considered has a strictly monotonic delay, the convergence to $\mathit{firstNeg}(F, I)$ is ensured; otherwise, the method may return a value $a > \mathit{firstNeg}(F, I)$, but in this case a still satisfies $F(a) \leq 0$ (with a given precision). It can be observed in practice that the convergence speed is particularly good for the delay function associated with agile satellites.

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

```
1  firstNeg(F, [a1, a2], maxIter, prec)
2  begin
3      f1 ← F(a1); if f1 ≤ 0 then return a1
4      f2 ← F(a2); if f2 > 0 then return +∞
5      for i = 1 to maxIter do
6          a3 = (f1 * a2 − f2 * a1)/(f1 − f2)
7          f3 = F(a3)
8          if |f3| < prec then return a3
9          else if f3 > 0 then (a1, f1) ← (a3, f3)
10         else (a2, f2) ← (a3, f3)
11     return a2
```

4 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.

4.1 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 [8–10]. The latter correspond to maintaining a list of variables for which constraints holding over these variables must be revised with, for each variable z of 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 (more details below).

Last, a t-simple temporal constraint is revised using Rules 7 and 8 of Prop. 1.

4.2 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 [7, 8], 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 [7] 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, since for TSTN in general, the existence of a propagation cycle does not necessarily imply inconsistency, as shown in the example below.

Example Let *dmin* be the minimum distance function defined by $dmin(a,b) = 1 - a/2$. Let $(V,C) = (\{x,y\}, \{ct_1 : x - y \geq -0.5, ct_2 : y - x \geq dmin(x,y)\})$ be a TSTN containing two temporal variables of domains $\mathbf{d}(x) = \mathbf{d}(y) = [0.5, 2]$ and two constraints. 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 $\mathbf{d}(y) = [1 + 1/4, 2]$. Propagating ct_1 using the same rule then updates the min of x and gives $\mathbf{d}(x) = [1 - 1/4, 2]$. 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. In the context of STN, the existence of such a cycle means inconsistency. In the context of TSTN, such a conclusion does not always hold because for instance assignment $x = 1$, $y = 1.5$ is consistent.

The reason is that in TSTN, domain reductions obtained by traversing cycles again and again may become smaller and smaller. This is what happens here, where we get $\mathbf{d}(x) = [1 - 1/2^n, 2]$ after n traversals of the propagation cycle between x and y. The finite computer precision implies that cycle traversals stop at some step, but potentially only after many iterations.

The example also shows that the strict monotonicity of the delay function does not suffice for deriving inconsistency in case of cycle detection. A sufficient condition satisfied for standard STN is given in Prop. 4. This condition ensures that a cycle does not become "less negative" when traversed again and again.

Definition 7. *A t-simple temporal constraint $ct : (x, y, dmin)$ is said to be shift-monotonic iff it satisfies:*

$$\forall a, a' \in \mathbf{d}(x),\ \forall b \in \mathbf{d}(y),\ (a \leq a') \rightarrow (delay_{ct}(a', b) \geq delay_{ct}(a, b) + (a' - a))$$
$$\forall a \in \mathbf{d}(x),\ \forall b, b' \in \mathbf{d}(y),\ (b \leq b') \rightarrow (delay_{ct}(a, b) \geq delay_{ct}(a, b') + (b' - b))$$

Informally, shift-monotonicity means that on one hand, when the start time of a transition is shifted forward, the arrival time is shifted forward by at least the same amount, and on the other hand when the arrival time of the transition is shifted backward, the delay is increase by at least the same amount.

Proposition 4. *If a propagation cycle involving only shift-monotonic constraints is detected in a TSTN, then the TSTN is inconsistent.*

Proposition 5. *In particular, (1) for TSTN containing only shift-monotonic constraints, the existence of a propagation cycle implies inconsistency; (2) for TSTN whose distance graph does not contain cycles involving non shift-monotonic constraints, the existence of a propagation cycle implies inconsistency.*

Proof. For Prop. 4, assume that propagation cycle $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n \rightarrow x_1$ 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. It can be shown that shift-monotonicity implies that $(a \leq a') \rightarrow (earr_{ct}(a') \geq earr_{ct}(a) + (a' - a))$. Therefore, if the cycle is traversed again, the min bounds of $x_2, \ldots, x_n$ will be increased again by at least δ. After a sufficient number of cycle traversals, the domain of one variable of the cycle becomes empty. Prop. 5 is a direct consequence of Prop. 4.

In the agile satellite application which motivates this work, the minimum distance functions used are not necessarily shift-monotonic, as can be seen in Fig. 1, but point 2 of Prop. 5 applies for case studies considered. Inferring inconsistency due to propagation cycle detection is correct in this case. Checking the satisfaction of the condition given in point 2 of Prop. 5 is easy (linear in the number of variables and constraints).

If none of the sufficient conditions given in Prop. 5 is satisfied, several options can be considered. The first one consists in not considering non shift-monotonic constraints in propagation chains; this approach is correct but may lose time in propagation cycles. The second option consists in considering a TSTN as inconsistent as soon as a propagation cycle is detected, even if it contains non shift-monotonic constraints; this may be incorrect in the sense that it may wrongly conclude to inconsistency. A possible trade-off is to keep the first option but to stop propagating constraints when some time-limit or some precision is reached.

In terms of complexity, Prop. 6 below generalizes polynomial complexity results available on STN to TSTN, and therefore to time-dependent scheduling.

Proposition 6. *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 plus 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 $O(|V||C|)$.

In terms of implementation, we perform on the fly detection of propagation cycles based on an efficient data structure introduced in [16]. 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.

Prop. 8 and 9 show that the two monotonicity properties considered in this paper (delay- and shift-monotonicity) are satisfied by simple temporal constraints and by several constraints used in time-dependent scheduling (see [5]).

Proposition 7. *Shift-monotonicity implies strict delay-monotonicity.*

Proposition 8. *Simple temporal constraints $y - x \geq c$ are shift-monotonic (and therefore also strictly delay-monotonic).*

Proposition 9. *Let x, y be two temporal variables corresponding to the start time and end time of a task respectively. Monotonicity results of Table 1 hold.*

Proof. Prop. 7 is straightforward. Prop. 8 holds because if *dmin* is constant, then $delay_{ct}(a,b) - delay_{ct}(a',b) = a - a'$ and $delay_{ct}(a,b) - delay_{ct}(a,b') = b' - b$. For Prop. 9, some intermediate results can be used: (a) if $dmin(x,y) = dmin(x)$, shift-monotonicity holds iff $dmin(x)$ is a non-decreasing function; (b) if $dmin(x)$ decreases at some step, then delay-monotonicity holds provided that the decrease slope is ≥ -1.

Distance $dmin(x,y) = dmin(x)$	**form**	**shift-monotonic**	**delay-monotonic**
$A + Bx$		yes	yes (strict)
$A - Bx$		no	yes iff $B \leq 1$ (strict iff $B<1$)
$\max(A, A + B(x - D))$		yes	yes (strict)
A if $x < D$, $A + B$ otherwise		yes	yes (strict)
$A - B\min(x, D)$		no	yes iff $B \leq 1$ (strict iff $B<1$)

Table 1. Monotonicity of some 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 $D > \min(\mathbf{d}(x))$

4.3 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 [10] 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 a 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 $u \neq y$.

4.4 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$^-$[9], 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 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 of 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. 3, 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 [9] 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 [17, 18].

5 Experiments

All techniques presented in Section 4 (constraint propagation, propagation cycle detection, constraint depropagation, SCC ordering) 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 [19] and LocalSolver [20]. 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. 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 earliest and latest 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. 4(a) shows that in this case, the temporal flexibility obtained with TSTN only slightly improves 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 considered are almost independent of the precise triggering time of transitions: they are only time-dependent when the rotation on the pitch axis is the most constraining from a temporal point of view, compared to the rotation on the roll axis.

In the second scenario, the scanning direction becomes 30 degrees. Fig. 4(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 acquisitions planned.

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. 4(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 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.

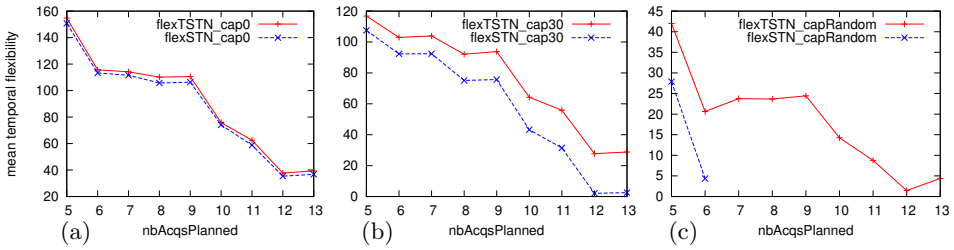

Fig. 4. Comparison of temporal flexibilities, in seconds, obtained with precomputed upper bound on transition times (flexSTN) and with exact transition times (flexTSTN)

As a conclusion, this paper introduced TSTN, their properties, resolution techniques, and their application to agile satellites. It would be interesting to extend other features of STN to TSTN, e.g. concerning *decomposability* issues [1], and to test TSTN on other applications, e.g. from the logistics domain.

References

1. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artificial Intelligence **49** (1991) 61–95
2. Stergiou, K., Koubarakis, M.: Backtracking algorithms for disjunctions of temporal constraints. Artificial Intelligence **120** (2000) 81–117
3. Challenge ROADEF-03: Handling the mission of Earth observation satellites (2003) http://challenge.roadef.org/2003/fr/.
4. Lemaître, M., Verfaillie, G., Jouhaud, F., Lachiver, J.M., Bataille, N.: Selecting and scheduling observations of agile satellites. Aerospace Science and Technology **6** (2002) 367–381
5. Cheng, T., Ding, Q., Lin, B.: A concise survey of scheduling with time-dependent processing times. European Journal of Operational Research **152** (2004) 1–13
6. Gawiejnowicz, S.: Time-dependent scheduling. Springer (2008)
7. Cervoni, R., Cesta, A., Oddi, A.: Managing dynamic temporal constraint networks. In: Proc. of AIPS-94. (1994) 13–18
8. Cesta, A., Oddi, A.: Gaining efficiency and flexibility in the simple temporal problem. In: Proc. of TIME-96. (1996) 45–50
9. Gerevini, A., Perini, A., Ricci, F.: Incremental algorithms for managing temporal constraints. In: Proc. of ICTAI-96. (1996) 360–365
10. Shu, I., Effinger, R., Williams, B.: Enabling fast flexible planning through incremental temporal reasoning with conflict extraction. In: Proc. of ICAPS-05. (2005) 252–261
11. Xu, L., Choueiry, B.: A new efficient algorithm for solving the simple temporal problem. In: Proc. of TIME-ICTL-03. (2003) 210–220
12. Planken, L., de Weerdt, M., van der Krogt, R.: P3C: a new algorithm for the simple temporal problem. In: Proc. of ICAPS-08. (2008) 256–263
13. Planken, L., de Weerdt, M., Yorke-Smith, N.: Incrementally solving STNs by enforcing partial path consistency. In: Proc. of ICAPS-10. (2010) 129–136
14. Montanari, U.: Networks of constraints: fundamental properties and applications to picture processing. Information Sciences **7**(2) (1974) 95–132
15. Hentenryck, P.V., Deville, Y., Teng, C.: A generic arc-consistency algorithm and its specializations. Artificial Intelligence **57**(2-3) (1992) 291–321
16. Bender, M., Cole, R., Demaine, E., Farach-Colton, M., Zito, J.: Two simplified algorithms for maintaining order in a list. In: Proc. of ESA-02. (2002) 152–164
17. Haeupler, B., Kavitha, T., Mathew, R., Sen, S., Tarjan, R.: Incremental cycle detection, topological ordering, and strong component maintenance. ACM Transactions on Algorithms **8**(1) (2012)
18. Roditty, L., Zwick, U.: Improved dynamic reachability algorithms for directed graphs. SIAM Journal on Computing **37**(5) (2008) 1455–1471
19. Hentenryck, P.V., Michel, L.: Constraint-based local search. The MIT Press (2005)
20. Benoist, T., Estellon, B., Gardi, F., Megel, R., Nouioua, K.: Localsolver 1.x: a black-box local-search solver for 0-1 programming. 4OR: A Quarterly Journal of Operations Research **9**(3) (2011) 299–316