Actes JFPC 2012

Réseaux temporels simples étendus Application à la gestion de satellites agiles

Cédric Pralet **Gérard Verfaillie**

ONERA – The French Aerospace Lab, F-31055, Toulouse, France

cedric.pralet@onera.fr gerard.verfaillie@onera.fr

Résumé

Les réseaux temporels simples (STN, Simple Temporal Networks) permettent de représenter une conjonction de contraintes de distance minimale et maximale requises sur certains couples de positions temporelles. Ce papier propose une extension des STN incluant des contraintes temporelles plus générales, pour lesquelles la distance à respecter entre deux positions temporelles x et y n'est plus forcément constante mais peut dépendre des instanciations de x et de y. De telles contraintes sont utiles pour traiter des problèmes dans lesquels le temps de transition entre deux activités peut dépendre de l'instant d'activation de la transition. L'extension proposée est appelée le cadre des "Time-dependent STN" (TSTN). Les propriétés de ce cadre sont étudiées et les techniques de résolution classiques sur les STN sont étendues aux TSTN. Les contributions proposées sont intégrées dans un algorithme de recherche locale utilisé pour la gestion de satellites dits agiles.

1 Motivations

La gestion des aspects temporels est un point clé dans le traitement d'applications issues du domaine de l'ordonnancement ou de la planification. Ces domaines mettent en effet généralement en jeu des contraintes sur les dates de début au plus tôt et de fin au plus tard de certaines activités, des contraintes de précédence ou de non recouvrement entre activités, ou encore des contraintes de distance temporelle requise entre activités. Ces contraintes peuvent dans de nombreux cas être exprimées (1) ou bien comme des contraintes temporelles dites *simples*, de la forme $x - y \in [\alpha, \beta]$, avec x, y des variables correspondant à deux positions temporelles et α, β des constantes ; (2) ou bien comme des contraintes temporelles dites *disjonctives*, correspondant à une disjonction de contraintes temporelles simples. Une conjonction de contraintes temporelles simples peut être représentée sous la forme d'un STN (Simple Temporal Network [6]). Les contraintes temporelles disjonctives peuvent quant à elles être représentées sous la forme d'un DTN (Disjunctive Temporal Network [18]).

Un des intérêts du cadre STN est la complexité polynomiale de certains problèmes [6], notamment : (a) la détermination de la cohérence d'un STN, c'est-à-dire l'existence d'une instanciation des variables temporelles satisfaisant toutes les contraintes, (b) la détermination des dates au plus tôt / au plus tard associées à chaque variable temporelle, utile pour maintenir un ordonnancement avec flexibilité temporelle, et (c) la détermination du *minimal network* [13] associé au STN, donnant les distances temporelles minimale et maximale séparant deux variables temporelles quelconques du STN. Un autre intérêt des STN est qu'ils sont souvent (voire toujours) utilisés comme élément de base dans la résolution des DTN ou de réseaux temporels plus complexes.

Dans ce papier, nous mettons à profit les différentes techniques développées dans le cadre STN pour traiter une application issue du domaine spatial. Cette application, dans laquelle les aspects temporels sont fortement présents, correspond à la gestion de satellites d'observation de la Terre tels que ceux du système *Pléiades*. De tels satellites sont situés en orbite basse, à quelques centaines de kilomètres d'altitude. Ils embarquent un instrument d'observation fixe à bord et sont dits *agiles*, ce qui signifie qu'ils ont la capacité de se mouvoir autour de leurs trois axes de rotation (roulis, tangage, lacet). Cette agilité leur permet de pointer à gauche, à droite, en avant ou en arrière du point au sol à la verticale du satellite. La mission de ces satellites consiste à réaliser des prises de vue de polygones à la surface du globe. Ces polygones sont découpés en bandes devant être balayées par l'instrument d'obser-

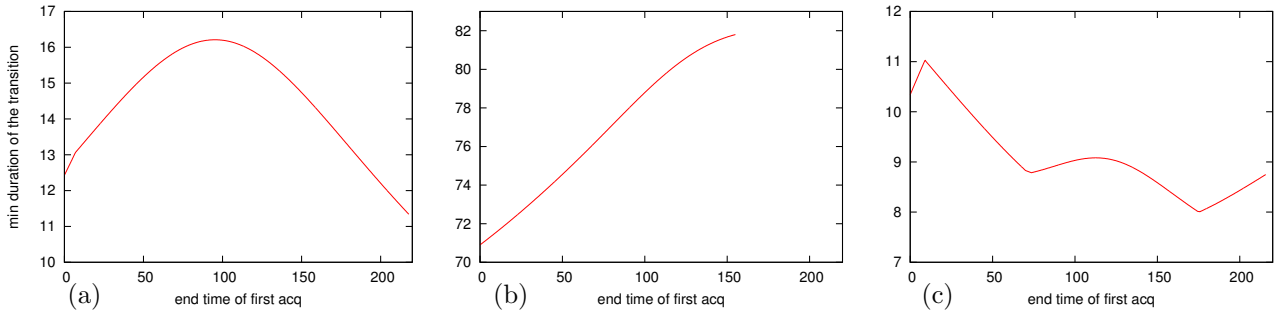


FIGURE 1 – Durées minimales, en secondes, de basculement d'une zone i se terminant au point de latitude-longitude 41°17′48″N-2°5′12″E à une zone j commençant au point de latitude-longitude 42°31′12″N-2°6′15″E, pour différents angles de balayage par rapport à la trace au sol du satellite : (a) balayage de i à 40° et balayage de j à 20° ; (b) balayage de i à 40° et balayage de j à -80° ; (c) balayage de i à 90° et balayage de j à 82°

vation. Le problème d'ordonnancement de prises de vue pour un satellite agile a notamment donné lieu au challenge ROADEF 2003 [16], pour lequel certaines simplifications par rapport au problème réel ont été adoptées, la principale étant que les durées nécessaires au basculement du satellite entre deux prises de vue i et j sont précalculées et considérées comme constantes. Dans ce cas, l'exécutabilité des ordonnancements produits n'est garantie que si les durées constantes choisies sont des majorants des durées réelles.

En pratique, l'hypothèse de temps de transition constants entre deux prises de vue n'est pas vérifiée [12]. Comme illustré à la figure 1, la durée minimale de basculement requise entre la fin d'une prise de vue i et le début d'une prise de vue j dépend de la date à laquelle la prise de vue i se termine. Pour la figure 1(b), l'absence de point sur la courbe après la date $t = 150$ correspond au fait qu'après cette date, la transition de i vers j n'est plus possible. Ces figures montrent la grande variabilité des temps de transition (jusqu'à une dizaine de secondes ici, durée pendant laquelle le satellite parcourt entre 50 et 100km au sol). Elles illustrent également la diversité des schémas d'évolution des temps de transitions minimaux. Ces temps sont calculés par résolution d'un problème d'optimisation de commande continue prenant en compte le mouvement du satellite sur son orbite, le déplacement des points au sol du fait de la rotation de la Terre et des contraintes sur la cinématique du satellite.

Pour toutes ces raisons, il apparaît utile de définir un nouveau cadre permettant de modéliser des transitions entre activités dont la durée dépend de la date d'enclenchement de la transition. Cet aspect se rapproche des travaux effectués autour du *time-dependent scheduling* [5, 7], dans lequel les durées de transition prennent des formes particulières, constantes par morceaux ou linéaires par morceaux, non directement utilisables ici. Nous décrivons tout d'abord le cadre proposé (les Time-dependent STN) ainsi que ses propriétés (sect. 2). Les preuves ne sont pas incluses par manque de place. Des techniques sont ensuite introduites pour calculer les dates de réalisation au plus tôt et au plus tard associées à chaque variable temporelle (sect. 3). Ces techniques sont utilisées au sein d'un algorithme de recherche locale utilisé pour l'ordonnancement d'activités d'un satellite agile (sect. 4).

2 Réseaux temporels simples étendus

Nous rappelons tout d'abord quelques définitions associées aux STN. Dans tout ce qui suit, le domaine d'une variable x est noté $\mathbf{d}(x)$.

2.1 Réseaux temporels simples

Définition 1. *Un réseau temporel simple (STN) est un couple (V, C) avec V un ensemble fini de variables dont le domaine est un intervalle fermé $[l, u] \subset \mathbb{R}$ et C un ensemble fini de contraintes binaires de la forme $x - y \in [\alpha, \beta]$ avec $x, y \in V$, $\alpha \in \mathbb{R} \cup \{-\infty\}$ et $\beta \in \mathbb{R} \cup \{+\infty\}$. Ces contraintes sont appelées des contraintes temporelles simples.*

Une solution d'un STN (V, C) est une instanciation des variables de V qui satisfait toutes les contraintes de C. Un STN est dit cohérent si et seulement si il admet au moins une solution.

Les contraintes unaires $x \in [\alpha, \beta]$, y compris celles définissant les domaines de valeurs possibles des différentes variables, peuvent se reformuler comme des contraintes temporelles simples $x - x_0 \in [\alpha, \beta]$, avec x_0 une variable de domaine $[0, 0]$ jouant le rôle de position temporelle de référence. Par ailleurs, comme $x - y \in [\alpha, \beta]$ équivaut à $(x - y \leq \beta) \wedge (y - x \leq -\alpha)$, il est

possible de se ramener uniquement à des contraintes de la forme $y - x \leq c$ avec c une constante.

Un élément important associé à un STN est son *graphe de distance*. Ce graphe contient un nœud par variable du STN et, pour chaque contrainte $y - x \leq c$ du STN, un arc orienté de x vers y et pondéré par c. Sur la base de ce graphe de distance, il est possible de montrer les résultats suivants [6] :

1. un STN est cohérent si et seulement si il n'existe pas de cycle de longueur négative dans son graphe de distance ;
2. si d_{0i} (resp. d_{i0}) désigne la longueur du plus court chemin dans le graphe de distance du nœud référence étiqueté par x_0 à un nœud étiqueté par une variable temporelle x_i (resp. de x_i à x_0), alors l'ensemble des instanciations cohérentes de x_i correspond à l'intervalle $[-d_{i0}, d_{0i}]$; les plus courts chemins peuvent être calculés pour tout i via l'algorithme de Bellman-Ford ou via des algorithmes s'apparentant à de la propagation de contraintes par arc-cohérence [3, 4, 8, 11] ;
3. si l'on note d_{ij} la longueur du plus court chemin de x_i à x_j dans le graphe de distance et d_{ji} la longueur du plus court chemin de x_j à x_i, alors l'intervalle $[-d_{ji}, d_{ij}]$ représente l'ensemble des distances temporelles possibles entre x_i et x_j ; de tels plus courts chemins peuvent être déterminés pour tous i, j en utilisant l'algorithme de Floyd-Warshall ou des algorithmes basés sur de la propagation par chemin-cohérence [6, 19, 15, 14].

Exemple Nous considérons un problème simplifié d'ordonnancement d'activités d'un satellite agile. Ce problème fait intervenir 3 acquisitions acq_1, acq_2, acq_3 devant être réalisées dans l'ordre $acq_3 \rightarrow acq_1 \rightarrow acq_2$. Pour tout $i \in [1..3]$, on note $Tmin_i$ et $Tmax_i$ les dates de début au plus tôt et de fin au plus tard de acq_i. La durée de acq_i est notée Da_i. Les durées de transition minimales entre la fin de acq_3 et le début de acq_1, et entre la fin de acq_1 et le début de acq_2, sont notées respectivement $Dt_{3,1}$ et $Dt_{1,2}$. Ces durées sont supposées constantes dans cette version simplifiée. Nous considérons également deux fenêtres temporelles $w_1 = [Ts_1, Te_1]$ et $w_2 = [Ts_2, Te_2]$ au cours desquelles un vidage de données vers des stations sol est possible. Le satellite doit vider acq_2 puis acq_3 dans la fenêtre w_1, avant de vider acq_1 dans w_2. La durée de vidage d'une acquisition acq_i est notée Dv_i.

Ce problème peut être modélisé sous la forme d'un STN contenant, pour toute acquisition acq_i ($i \in [1..3]$), les variables temporelles suivantes :

- une date de début de réalisation sa_i et une date de fin de réalisation ea_i, avec comme domaines de valeurs $\mathbf{d}(sa_i) = \mathbf{d}(ea_i) = [Tmin_i, Tmax_i]$;
- une date de début de vidage sv_i et une date de fin de vidage ev_i, avec comme domaines de valeurs $[Ts_1, Te_1]$ pour $i = 2, 3$ et $[Ts_2, Te_2]$ pour $i = 1$.

Les contraintes temporelles simples 1 à 7 sont imposées sur ces variables. Les contraintes 1 et 2 définissent la durée des acquisitions et des vidages. Les contraintes 3 et 4 imposent des temps de transition minimaux entre acquisitions. Les contraintes 5 et 6 assurent le non chevauchement des activités de vidages. La contrainte 7 exprime qu'une acquisition ne peut être vidée qu'après la fin de sa réalisation. La figure 2 donne le graphe de distance associé à cet exemple.

$$\forall i \in [1..3],\ ea_i - sa_i = Da_i \quad (1)$$
$$\forall i \in [1..3],\ ev_i - sv_i = Dv_i \quad (2)$$
$$sa_1 - ea_3 \geq Dt_{3,1} \quad (3)$$
$$sa_2 - ea_1 \geq Dt_{1,2} \quad (4)$$
$$sv_3 - ev_2 \geq 0 \quad (5)$$
$$sv_1 - ev_3 \geq 0 \quad (6)$$
$$\forall i \in [1..3],\ sv_i - ea_i \geq 0 \quad (7)$$

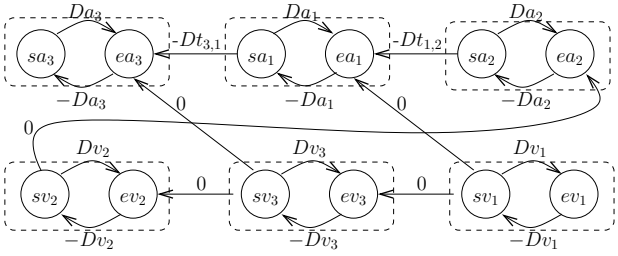


FIGURE 2 – Graphe de distance (la position temporelle de référence x_0 n'est pas représentée)

2.2 Contraintes temporelles t-simples

Nous introduisons une nouvelle classe de contraintes temporelles appelées des *contraintes temporelles t-simples*. Ces dernières sont plus générales que les contraintes temporelles simples et permettent de manipuler des temps de transition entre activités fonctions de la date d'enclenchement des transitions.

Définition 2. *Une* contrainte temporelle t-simple *est un triplet $(x, y, dmin)$ composé de deux variables temporelles x et y, et d'une fonction $dmin : \mathbf{d}(x) \times \mathbf{d}(y) \rightarrow \mathbb{R}$ appelée fonction de distance minimale (fonction non nécessairement continue).*

Une contrainte temporelle t-simple $(x, y, dmin)$ est également notée $y - x \geq dmin(x, y)$. La contrainte est satisfaite par un couple de valeurs $(a, b) \in \mathbf{d}(x) \times \mathbf{d}(y)$ si et seulement si $b - a \geq dmin(a, b)$.

Informellement, $dmin(x, y)$ spécifie une distance temporelle minimale entre l'occurrence de l'événement associé à la variable temporelle x et l'occurrence de l'événement associé à la variable temporelle y.

Pour illustrer l'intérêt d'avoir une distance minimale *dmin* dépendant à la fois de x et de y, considérons l'exemple des satellites agiles. Soit x la variable spécifiant la date de fin d'une acquisition *acq*. L'attitude obtenue en terminant *acq* à la date x est notée $Att(x)$, une attitude du satellite étant définie par une direction de pointage et par les vitesses de rotation suivant chacun des trois axes (roulis, tangage, lacet). Soit y la variable donnant la date de début de l'acquisition acq' à réaliser juste après *acq*. L'attitude nécessaire pour commencer acq' à la date y est notée $Att'(y)$. Soit *tminbas* la fonction (disponible dans notre librairie de calculs d'agilité) telle que $tminbas(att, att')$ donne le temps minimal requis pour basculer d'une attitude *att* à une attitude att'. On parle dans ce cas de *rendez-vous en attitude* de *att* vers att'. Alors, en définissant $dmin(x, y) = tminbas(Att(x), Att'(y))$, la contrainte temporelle t-simple $y - x \geq dmin(x, y)$ exprime que la durée entre la fin de *acq* et le début de acq' doit être supérieure à la durée minimale de basculement de l'attitude $Att(x)$ à l'attitude $Att'(y)$.

Dans certains cas, la fonction $dmin(x, y)$ est indépendante de y. Cette remarque concerne notamment le *time-dependent scheduling* [5, 7], dans lequel la durée de réalisation d'une tâche dépend uniquement de la date de début de cette tâche (contrainte temporelle t-simple "dégénérée", de la forme $y - x \geq dmin(x)$ avec $dmin(x)$ le temps de réalisation de la tâche si cette dernière commence à la date x). Les contraintes temporelles t-simples couvrent également les contraintes temporelles simples $y - x \geq c$, en utilisant une fonction de distance minimale constante $dmin = c$.

Remarquons enfin qu'une contrainte temporelle t-simple se réfère à la durée *minimale* d'une transition. Une telle approche est utilisable pour gérer les satellites agiles sous l'hypothèse (réaliste) que tout rendez-vous en attitude faisable en une durée δ est aussi faisable en une durée $\delta' \geq \delta$. Cette hypothèse de faisabilité d'un "rendez-vous paresseux" n'est pas forcément vérifiée pour tout système physique.

2.3 Propriétés de monotonie

Nous commençons par définir la fonction de retard associée à une contrainte temporelle t-simple.

Définition 3. *La* fonction de retard *associée à une contrainte temporelle t-simple* $ct : (x, y, dmin)$ *est la fonction* $delay_{ct} : \mathbf{d}(x) \times \mathbf{d}(y) \to \mathbb{R}$ *définie par :* $delay_{ct}(a, b) = a + dmin(a, b) - b$.

Informellement, $delay_{ct}(a, b)$ donne le retard obtenu en b si l'on enclenche, à la date a, une transition en temps minimal de x vers y. Ce retard s'exprime comme la différence entre la date minimale de fin de la transition $(a + dmin(a, b))$ et la date de fin de transition requise (b). Un retard strictement négatif correspond à une transition se terminant en avance par rapport à la date limite b. Un retard strictement positif correspond à une violation de la contrainte ct. Un retard nul correspond à une arrivée "pile à l'heure".

La fonction de retard peut satisfaire une propriété de monotonie définie ci-dessous, qui sera utilisée par la suite pour identifier des classes traitables.

Définition 4. *Une contrainte temporelle t-simple* $ct : (x, y, dmin)$ *est dite* à retard monotone *si et seulement si sa fonction de retard* $delay_{ct}(.,.)$ *satisfait :* $\forall a, a' \in \mathbf{d}(x)$, $\forall b, b' \in \mathbf{d}(y)$,

$$(a \leq a') \to (delay_{ct}(a, b) \leq delay_{ct}(a', b)) \quad (8)$$
$$(b \leq b') \to (delay_{ct}(a, b) \geq delay_{ct}(a, b')) \quad (9)$$

La contrainte est dite à retard strictement monotone *si les monotonies sur les deux arguments sont strictes.*

La définition 4 signifie que pour être à retard monotone, une contrainte temporelle t-simple $(x, y, dmin)$ doit vérifier d'une part que plus la transition de x vers y est enclenchée tard, plus le retard est grand à l'arrivée en y, et d'autre part que plus l'on cherche à arriver tôt au rendez-vous (en y), plus le retard est élevé.

Nous définissons maintenant les fonctions d'arrivée au plus tôt et de départ au plus tard associées à une contrainte temporelle t-simple. Par la suite, pour une fonction $F : \mathbb{R} \to \mathbb{R}$ et un intervalle fini fermé I de $\mathbb{R}$, on note (1) $firstNeg(F, I)$ le plus petit $a \in I$ tel que $F(a) \leq 0$ (valeur $+\infty$ si une telle valeur n'existe pas) ; (2) $lastNeg(F, I)$ le plus grand $a \in I$ tel que $F(a) \leq 0$ (valeur $-\infty$ si une telle valeur n'existe pas) [1].

Définition 5. *Les fonctions d'arrivée au plus tôt (earliest arrival) et de départ au plus tard (latest departure) associées à une contrainte temporelle t-simple* $ct : (x, y, dmin)$ *sont les fonctions* $earr_{ct}$ *et* $ldep_{ct}$ *définies respectivement sur* $\mathbf{d}(x)$ *et* $\mathbf{d}(y)$ *et données par :*

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

Informellement, $earr_{ct}(a)$ fournit la plus petite date d'arrivée en y sans retard si la transition est enclenchée à la date a. $ldep(b)$ fournit la date d'enclenchement au plus tard pour une arrivée sans retard en b.

1. Les quantités $firstNeg(F, I)$ et $lastNeg(F, I)$ ne sont mathématiquement pas forcément bien définies si la fonction F présente des discontinuités ; nous utilisons en fait implicitement le fait que le travail se fait en machine avec une précision finie.

Pour une contrainte temporelle simple $y - x \geq c$, il est possible de donner une formulation analytique de *earr* et *ldep*. Dans le cas général, il faut passer par le calcul de *firstNeg*(F, I) et *lastNeg*(F, I), qui constitue un problème d'optimisation en soi. Une méthode itérative pour approximer *firstNeg*$(F, I = [a_1, a_2])$ est donnée à l'algorithme 1. Cette méthode généralise la méthode dite de la *fausse position*, utilisable pour trouver un zéro d'une fonction. Appliquée au cas des contraintes temporelles t-simples, elle consiste, si le point $P_1 = (a_1, F(a_1))$ est à retard positif ($F(a_1) > 0$) et le point $P_2 = (a_2, F(a_2))$ est à retard négatif ($F(a_2) < 0$), à étudier comment se comporte le retard $F(a_3)$ en a_3 avec a_3 l'abscisse du point d'intersection de la droite passant par P_1, P_2 avec l'axe des abscisses. Si $P_3 = (a_3, F(a_3))$ est à retard positif (resp. négatif), la recherche se poursuit en prenant $P_1 = P_3$ (resp. $P_2 = P_3$). Si la contrainte temporelle t-simple considérée est à retard strictement monotone, cette méthode converge vers *firstNeg*(F, I) ; sinon, la méthode peut renvoyer une valeur $a >$ *firstNeg*(F, I), mais dans ce cas a satisfait $F(a) \leq 0$ (à la précision près). La vitesse de convergence en pratique est particulièrement bonne pour la fonction de retard des satellites agiles.

Algorithm 1: Méthode possible de calcul de *firstNeg*(F, I), avec I=$[a_1, a_2]$, maxIter un nombre max d'itérations et prec une précision souhaitée

```
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
```

Sur la base des quantités *earr* et *ldep*, il est possible de définir une seconde propriété de monotonie, plus forte que la monotonie de la fonction de retard (cf. prop. 1) et pouvant être violée en pratique.

Définition 6. *Une contrainte temporelle t-simple ct :* $(x, y, dmin)$ *est dite à décalage monotone si et seulement si elle vérifie,* $\forall a, a' \in \mathbf{d}(x)$, $\forall b, b' \in \mathbf{d}(y)$,

$$(a' \geq a) \rightarrow (earr_{ct}(a') \geq earr_{ct}(a) + (a' - a))$$
$$(b' \geq b) \rightarrow (ldep_{ct}(b') \geq ldep_{ct}(b) + (b' - b))$$

Proposition 1. *Soit ct :* $(x, y, dmin)$ *une contrainte temporelle t-simple. Si ct est à décalage monotone, alors ct est à retard strictement monotone.*

Informellement, une contrainte temporelle t-simple est à décalage monotone si et seulement si d'une part lorsque la date d'enclenchement d'une transition est repoussée, la date d'arrivée au plus tôt est décalée d'autant, et d'autre part lorsque la date de fin de la transition est avancée, la date d'enclenchement au plus tard est décalée d'autant. Dans le cas des satellites agiles, la fonction de distance minimale *dmin* peut présenter des comportements du type de ceux de la figure 1 qui entraînent que les contraintes ne sont pas toujours à décalage monotone.

Les deux propositions suivantes montrent que les propriétés de monotonie sont satisfaites par les contraintes temporelles simples et par plusieurs contraintes temporelles classiquement utilisées dans le time-dependent scheduling (voir [5]).

Proposition 2. *Les contraintes temporelles simples* $y - x \geq c$ *sont à décalage monotone (et donc aussi à retard strictement monotone).*

Proposition 3. *Soit x, y deux variables temporelles correspondant respectivement aux dates de début et de fin d'une tâche. Les résultats de monotonie donnés à la table 1 sont corrects.*

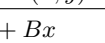
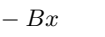
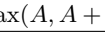
Fonction de distance $dmin(x, y) = dmin(x)$	Forme	*Monotonie*	
		décalage	*retard*
$A + Bx$		oui	oui (strict)
$A - Bx$		non	oui ssi $B \leq 1$ strict ssi $B < 1$
$\max(A, A + B(x - D))$		oui	oui (strict)
A si $x < D$, $A + B$ sinon		oui	oui (strict)
$A - B\min(x, D)$		non	oui ssi $B \leq 1$ strict ssi $B < 1$

TABLE 1 – Monotonie de fonctions de distance utilisées en time-dependent scheduling, avec A, B, D des constantes telles que $A \geq 0$, $B > 0$ et $D > \min(\mathbf{d}(x))$

2.4 Arc-cohérence des contraintes temporelles t-simples

La propriété de retard monotone permet de simplifier l'établissement de l'arc-cohérence.

Proposition 4. *Soit ct :* $(x, y, dmin)$ *une contrainte temporelle t-simple à retard monotone. Etablir l'arc-cohérence aux bornes pour ct est équivalent à établir l'arc-cohérence sur les domaines complets de x et y.*

La raison est qu'en cas de retard monotone, si $\max(\mathbf{d}(x))$ possède un support b sur y, alors b est aussi

un support pour toutes les autres valeurs du domaine de x, et de manière similaire le support de $\min(\mathbf{d}(y))$ sur x est un support pour toutes les valeurs de y.

Proposition 5. *Soit $ct : (x, y, dmin)$ une contrainte temporelle t-simple. L'arc-cohérence aux bornes pour la contrainte ct peut être établie par les modifications de domaine suivantes :*

$$\mathbf{d}(y) \leftarrow \mathbf{d}(y) \cap [earr_{ct}(\min(\mathbf{d}(x))), +\infty[\quad (10)$$
$$\mathbf{d}(x) \leftarrow \mathbf{d}(x) \cap]-\infty, ldep_{ct}(\max(\mathbf{d}(y)))] \quad (11)$$

La règle 10 permet de modifier la date de réalisation au plus tôt associée à y. La règle 11 permet de modifier la date de réalisation au plus tard associée à x. Les modifications de domaines sont telles que les domaines courants $\mathbf{d}(x)$ et $\mathbf{d}(y)$ restent des intervalles fermés.

Proposition 6. *L'arc-cohérence d'une contrainte temporelle t-simple $ct : (x, y, dmin)$ à retard monotone peut être établie par les règles 10 et 11.*

Si l'hypothèse de retard monotone n'est pas satisfaite, l'arc-cohérence aux bornes peut tout de même être établie. Dans ce cas, elle n'implique pas l'arc-cohérence et elle est susceptible de supprimer des valeurs cohérentes des domaines des variables.

2.5 Cadre des TSTN

Sur la base des contraintes temporelles t-simples, il est possible d'introduire un nouveau cadre appelé le cadre des TSTN pour "Time-dependent STN".

Définition 7. *Un TSTN est un couple (V, C) avec V un ensemble fini de variables de domaine $[l, u] \subset \mathbb{R}$ et C un ensemble fini de contraintes temporelles t-simples $(x, y, dmin)$ avec $x, y \in V$.*

Une solution d'un TSTN est une affectation des variables de V qui satisfait toutes les contraintes de C. Un TSTN est dit cohérent s'il admet au moins une solution. Un TSTN est dit à retard monotone (resp. à décalage monotone) si toutes ses contraintes temporelles sont à retard monotone (resp. à décalage monotone).

Exemple Nous reconsidérons l'exemple faisant intervenir trois acquisitions acq_1, acq_2, acq_3. Les durées de transition minimales requises entre acquisitions ne sont cette fois pas considérées comme constantes. Dans le modèle TSTN obtenu, la seule différence est que les contraintes temporelles simples des équations 3 et 4 sont remplacées par les contraintes temporelles t-simples des équations 12 et 13, dans lesquelles pour une acquisition acq_i, $Satt_i(t)$ et $Eatt_i(t)$ donnent respectivement les attitudes requises pour enclencher et finir acq_i à la date t. Le graphe de distance associé à un TSTN peut être défini de manière analogue au graphe de distance associé à un STN (voir la figure 3).

$$sa_1 - ea_3 \geq tminbas(Eatt_3(ea_3), Satt_1(sa_1)) \quad (12)$$
$$sa_2 - ea_1 \geq tminbas(Eatt_1(ea_1), Satt_2(sa_2)) \quad (13)$$

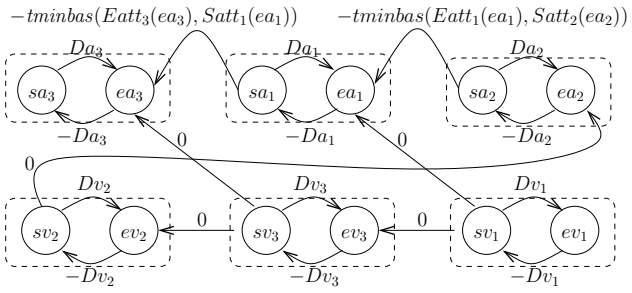


FIGURE 3 – Graphe de distance du TSTN (la référence temporelle x_0 n'est pas représentée)

La proposition suivante étend sur les TSTN un résultat classique sur les STN. Elle est valable que les contraintes soient à retard monotone ou non.

Proposition 7. *Si toutes les contraintes d'un TSTN sont arc-cohérentes aux bornes, alors l'ordonnancement obtenu en fixant des dates au plus tôt (resp. des dates au plus tard) est une solution du TSTN.*

3 Résolution des TSTN

La tâche à laquelle nous nous intéressons est celle de la détermination de la cohérence d'un TSTN et des dates au plus tôt et au plus tard associées à chaque variable temporelle. Les algorithmes proposés reprennent et généralisent des techniques STN existantes.

3.1 Propagation de contraintes

Le premier élément utilisé est un schéma de propagation des bornes min et max des différentes variables temporelles. Cet élément relativement standard s'inspire des approches définies dans [4, 8, 11]. Ces dernières reviennent à maintenir une liste de variables pour lesquelles les contraintes portant sur ces variables doivent être révisées avec, pour chaque variable z de la liste, un maintien de la nature de la (des) révision(s) à effectuer : (a) si z a subi une modification de sa borne min, les bornes min de toutes les variables t liées à z par une contrainte $t - z \geq c$ doivent être révisées ; (b) si z a subi une modification de sa borne max, les bornes max de toutes les variables t liées à z par une contrainte $z - t \geq c$ doivent être révisées.

Par rapport aux approches STN classiques, nous choisissons pour les TSTN une propagation de

contraintes dans laquelle est maintenue une liste contenant non pas des variables mais des contraintes. Cette liste est partitionnée en deux sous-listes, l'une contenant les contraintes à réviser pouvant modifier une borne min (contraintes $y - x \geq dmin(x, y)$ réveillées suite à une modification de $\min x$ et susceptibles de modifier $\min y$), l'autre contenant les contraintes à réviser pouvant modifier une borne max (contraintes $y - x \geq dmin(x, y)$ réveillées suite à une modification de $\max y$ et susceptibles de modifier $\max x$). Par rapport à une version liste de variables, le maintien de listes de contraintes permettra de gérer plus finement certains aspects (voir plus loin).

Enfin, la révision d'une contrainte temporelle t-simple du point de vue des bornes min et/ou max se fait en utilisant les règles de la proposition 6, avec une évaluation de $earr_{ct}$ et $ldep_{ct}$ analytique lorsque cela est possible, par recherche itérative sinon.

3.2 Détection de cycles de longueur négative

Avec des domaines de valeurs bornés, la propagation par arc-cohérence sur les STN est capable de détecter l'incohérence. Cependant, le nombre de révisions de contraintes requises pour arriver à ce résultat est potentiellement prohibitif par rapport à des approches définies dans [3, 4], qui utilisent le fait que l'incohérence d'un STN est équivalente à l'existence d'un cycle de longueur négative dans son graphe de distance.

L'idée de base de ces approches consiste à détecter de tels cycles à la volée en maintenant des *chaînes de propagation*. Ces dernières correspondent à des explications des valeurs courantes des bornes min et max des différentes variables. Une contrainte $y - x \geq c$ est dite active du point de vue des bornes min (resp. des bornes max) si et seulement si la dernière révision de cette contrainte est responsable de la dernière modification de la borne min de y (resp. de la borne max de x). [3] montre que s'il existe un cycle dans le graphe orienté dont les arcs correspondent aux contraintes actives du point de vue des bornes min, alors le STN est incohérent. L'intuition est que si un cycle de propagation $x_1 \to x_2 \to \ldots \to x_n \to x_1$ est détecté, alors la valeur minimale de x_1 modifie la valeur minimale de x_2... qui elle-même modifie la valeur minimale de x_n qui elle-même modifie la valeur minimale de x_1, et donc il est possible de vider le domaine de x_1 en parcourant ce cycle un nombre suffisant de fois. Le même résultat s'applique au graphe orienté contenant un arc par contrainte active du point de vue des bornes max.

Le point bloquant pour adapter ces techniques au cas des contraintes temporelles t-simples est que pour un TSTN dans le cas général, l'existence d'un cycle de propagation n'est pas nécessairement synonyme d'incohérence, comme le montre l'exemple suivant.

Exemple Soit $(V, C) = (\{x, y\}, \{ct_1, ct_2\})$ un TSTN contenant deux variables temporelles de domaines $\mathbf{d}(x) = \mathbf{d}(y) = [0.5, 2]$, et deux contraintes $ct_1 : y - x \leq 0.5$ et $ct_2 : y - x \geq dmin(x, y)$, avec $dmin$ la fonction de distance minimale définie par $dmin(a, b) = 1 - a/2$. La fonction de retard associée à ct_2 est strictement monotone (elle vaut $delay_{ct_2}(a, b) = 1 + a/2 - b$).

L'établissement de l'arc-cohérence pour ct_2 (règle 10) modifie la borne min de y et donne $\mathbf{d}(y) = [1 + 1/4, 2]$. Une propagation sur la contrainte ct_1 modifie alors la borne min de x et donne $\mathbf{d}(x) = [1 - 1/4, 2]$. Le résultat obtenu est un cycle de propagation, puisque la borne min de x a modifié la borne min de y qui elle-même a modifié la borne min de x. Dans le cadre STN classique, l'existence d'un tel cycle implique une incohérence. Dans le cadre TSTN, une telle conclusion est fausse puisque par exemple l'instanciation $x = 1$, $y = 1.5$ est cohérente.

La raison à cela est que dans le cadre TSTN, les réductions de domaines induites par des reparcours d'un cycle de propagation peuvent devenir de plus en plus petites. C'est ce qui se produit sur l'exemple ci-dessus où après n parcours du cycle on a $\mathbf{d}(x) = [1 - 1/2^n, 2]$. D'un point de vue implémentation en machine, la précision finie implique que le parcours des cycles s'arrête à un moment donné, mais potentiellement après un très grand nombre d'itérations.

L'exemple montre également que la monotonie stricte de la fonction de retard ne suffit pas pour conclure à l'incohérence en cas de détection de cycle. Une condition suffisante, satisfaite pour les STN classiques, est donnée ci-dessous. Cette condition assure qu'un cycle ne devient pas "moins négatif" au fur et à mesure de ses reparcours.

Proposition 8. *Si un cycle de propagation est détecté dans un TSTN, alors il suffit que toutes les contraintes temporelles t-simples du cycle soient à décalage monotone pour conclure que le TSTN est incohérent.*

Proposition 9. *En particulier, (1) pour un TSTN à décalage monotone, l'existence d'un cycle de propagation implique l'incohérence du TSTN; (2) pour un TSTN dont le graphe de distance ne contient aucun cycle impliquant une contrainte temporelle t-simple à décalage non monotone, l'existence d'un cycle de propagation implique l'incohérence du TSTN.*

Dans l'application satellite agile qui motive ce travail, la fonction de distance minimale n'est pas monotone, mais par contre le point 2 de la proposition 9 s'applique sur les cas d'étude traités. Déduire une incohérence suite à une détection de cycle est dans ce cas correct. Si aucune des conditions de la proposition 9 ne s'applique, plusieurs options sont envisageables. La

première consiste à ne pas inclure les contraintes temporelles à décalage non monotones dans les chaînes de propagation. La seconde, incorrecte dans le sens où elle peut conduire à des détections d'incohérence à tort, consiste à considérer que le TSTN est incohérent dès qu'un cycle de propagation est détecté.

En termes de complexité, la proposition suivante généralise aux TSTN, et donc aux problèmes de time-dependent scheduling, les résultats de complexité polynomiale disponibles sur les STN.

Proposition 10. *L'algorithme utilisant propagation par arc-cohérence aux bornes et détection de cycle sur un TSTN (V, C) établit l'arc-cohérence aux bornes en $O(|V||C|)$ révisions de contraintes (borne indépendante de la taille des domaines des variables).*

Côté implémentation, la détection de cycles de propagation à la volée s'appuie sur une structure de donnée efficace [1] utilisée pour maintenir des ordres topologiques dans les graphes de propagation des bornes min et max. L'existence de cycles dans ces graphes est en effet équivalente à la non existence d'ordres topologiques des nœuds dans ces mêmes graphes.

3.3 Dépropagation de contraintes

Les méthodes de propagation de contraintes permettent directement de traiter le cas de l'ajout d'une contrainte au TSTN de départ ou le cas du renforcement d'une contrainte. Concernant le retrait ou l'assouplissement d'une contrainte, il est possible de réutiliser directement les stratégies définies dans [11] pour les STN. Ces stratégies permettent de rétablir à moindre coût les bornes min et max des variables temporelles. La technique de base consiste à suivre les chaînes de propagation pour savoir quels domaines de variables réinitialiser et quelles contraintes repropager. Lors du retrait ou d'un assouplissement d'une contrainte $y - x \geq dmin(x, y)$, si cette contrainte est active du point de vue de la borne min de y (resp. de la borne max de x), alors la borne min de y (resp. la borne max de x) est réinitialisée à la valeur qu'elle avait avant toute propagation. Cette réinitialisation peut elle-même déclencher d'autres réinitialisations. Les contraintes du TSTN de la forme $y - z \geq dmin(z, y)$ (resp. $z - x \geq dmin(x, z)$) sont ajoutées à la liste des contraintes à propager du point de vue des bornes min (resp. max) des variables.

La seule différence par rapport à la littérature STN est l'utilisation de listes de contraintes à propager (et non de variables), qui permet de faire une dépropagation légèrement plus fine : sur l'exemple de réinitialisation de la borne min de y, la version classique STN ajoute à une liste des variables à propager toute variable z liée à y par une contrainte $y - z \geq dmin(z, y)$, et ce faisant repropage *in fine* toutes les contraintes de la forme $u - z \geq dmin(z, u)$, même celles avec $u \neq y$.

3.4 Ordonnancement des révisions de contraintes

Nous utilisons une dernière technique dont le but est de minimiser le nombre de révisions de contraintes. Cet objectif est intéressant pour les STN mais l'est d'autant plus pour les TSTN, pour lesquels le coût d'une révision de contrainte peut être significativement plus élevé. L'approche proposée reprend et étend une technique développée pour les STN$^-$ [8], une sous-classe des STN dans laquelle les contraintes temporelles simples doivent pouvoir être mises sous la forme $y - x \geq c$ avec $c \geq 0$. L'idée consiste à construire les composantes fortement connexes du graphe de distance, à les ordonner suivant un ordre topologique, et à utiliser cet ordre topologique pour savoir dans quel ordre propager les contraintes. Nous rappelons tout d'abord la définition de la notion de composante fortement connexe.

Définition 8. *Soit $G = (V, A)$ un graphe orienté, avec V l'ensemble des nœuds et A l'ensemble des arcs. Une composante fortement connexe de G (SCC,* Strong Connected Component*) est un sous-graphe G' maximal de G tel qu'il existe dans G' un chemin de n'importe quel nœud à n'importe quel autre nœud.*

*Le DAG (*Directed Acyclic Graph*) des SCCs de G est le graphe orienté dont les nœuds sont les SCCs de G et tel qu'il existe un arc d'un SCC c_1 vers un SCC c_2 si et seulement si il existe dans G un arc d'un des sommets de c_1 vers un des sommets de c_2.*

Un ordre topologique des SCCs est un ordre $\preceq$ tel que chaque SCC c est placé dans cet ordre strictement après chacun de ses parents c' dans le DAG des SCCs ($c' \prec c$). Pour un nœud x du graphe de départ G, on note $scc(x)$ le SCC qui contient x.

Propager les contraintes temporelles en suivant un ordre topologique des SCCs du graphe de distance revient à utiliser le résultat selon lequel résoudre un problème de plus court chemin dans un graphe acyclique est plus facile que dans un graphe quelconque. Pour appliquer ce résultat, les contraintes à propager sont ordonnées suivant un ordre topologique des SCCs. Plus précisément, concernant la propagation des bornes min, on propage en priorité les contraintes $y - x \geq dmin(x, y)$ avec $scc(y)$ maximal au sens de l'ordre des SCCs, et en cas d'égalité en privilégiant les contraintes telles que $scc(x) \neq scc(y)$, pour repousser au maximum la propagation des contraintes internes à un SCC. Pour la propagation des bornes max, les contraintes sont ordonnées par $scc(x)$ croissant et en cas d'égalité en privilégiant les contraintes telles que $scc(y) \neq scc(x)$. Sur l'exemple de la figure 3, les SCCs sont représentés en pointillés. Un mauvais ordre de

propagation des bornes min consisterait à propager d'abord la contrainte entre sa_2 et ea_1 puis celle entre sa_1 et ea_3. Un bon ordre, cohérent avec l'ordre des SCCs, consisterait à faire l'inverse.

Par rapport à l'utilisation des SCCs faite dans [8] pour les STN$^-$, la méthode proposée est adaptée non seulement aux STN généraux mais aussi aux TSTN. Côté implémentation, pour ne pas recalculer le DAG des SCCs après chaque ajout ou retrait de contrainte, nous utilisons des algorithmes adaptés au maintien dynamique des SCCs [9, 17].

4 Expérimentations

Les techniques proposées précédemment sont implémentées dans un outil d'ordonnancement fonctionnant à base de recherche locale. Les mouvements locaux consistent en des modifications d'un ordonnancement courant. Ils se traduisent par des ajouts ou des retraits de contraintes temporelles t-simples. Le côté recherche locale implique qu'il est rapide de faire/défaire un mouvement local, dans un esprit similaire aux outils Comet [10] et LocalSolver [2]. Une différence par rapport aux outils classiques de recherche locale à base de contraintes est que dans le cas des TSTN, les variables temporelles ne sont pas instanciées à une valeur unique (on maintient pour chacune un intervalle de valeurs). L'outil de résolution STN/TSTN est implémenté en Java. Les résultats sont obtenus sur un processeur Intel i5-520 1.2GHz, 4GBRAM.

Des expérimentations non détaillées ici ont tout d'abord été réalisées sur des STN obtenus à partir de problèmes d'ordonnancement de la SMT-LIB. L'objectif était de valider l'intérêt de l'heuristique de propagation basée sur un ordre topologique des SCCs. Cette heuristique se révèle être une stratégie robuste permettant, sur certains problèmes, de gagner un ou plusieurs ordres de grandeur en termes de nombre de révisions de contraintes. Plus précisément, sur des STN cohérents, cette heuristique est toujours au moins aussi bonne qu'une approche par gestion des contraintes à propager sous forme de liste FIFO ou LIFO. Sur des STN incohérents, propager suivant l'ordre des SCCs n'est par contre pas toujours la stratégie qui établit le plus vite l'incohérence.

Nous détaillons ci-après les expérimentations réalisées sur les TSTN dans le contexte satellites agiles. Le problème considéré est une simple contrainte de non chevauchement sur une séquence ordonnée de n acquisitions $acq_1 \rightarrow \ldots \rightarrow acq_n$, avec n variant entre 5 et 13. Ces acquisitions correspondent à des zones au sol situées entre le nord le l'Espagne et le nord de la France. La contrainte de non chevauchement entre acquisitions s'exprime comme un ensemble de contraintes temporelles t-simples de la forme $s_{i+1} - e_i \geq tminbas(Eatt_i(e_i), Satt_{i+1}(s_{i+1}))$ avec, pour une acquisition j, s_j/e_j les dates de début/fin de cette acquisition, et $Satt_j(t)/Eatt_j(t)$ les attitudes nécessaires pour enclencher/finir j à la date t. Sont ajoutées à cela des contraintes temporelles simples fixant une durée constante pour chaque prise de vue.

Deux méthodes sont comparées : d'un côté une approche TSTN avec prise en compte des temps de transition exacts entre acquisitions, et de l'autre une approche STN avec précalcul de majorants sur les temps de transition. Les ordonnancements obtenus dans les deux cas sont flexibles dans le sens où la propagation sur les STN/TSTN fournit en sortie des variables temporelles dont le domaine élagué n'est en général pas réduit à une seule valeur possible. Le critère étudié est la flexibilité temporelle moyenne, mesurée comme la moyenne sur l'ensemble des variables temporelles x de la différence entre date au plus tôt et date au plus tard pour x. La flexibilité temporelle est importante pour offrir le plus de latitude possible concernant le choix de l'angle de réalisation d'une prise de vue, qui influence la qualité image.

Trois scénarios sont étudiés. Dans le premier scénario, les acquisitions sont des bandes de longueur 80km environ, à observer avec un cap de 0 degré, le cap correspondant à l'angle entre la trace au sol du satellite et la direction de balayage de la zone. La figure 4(a) montre que dans ce cas la flexibilité temporelle obtenue avec les TSTN est à peine meilleure que celle obtenue avec des STN. La raison est que si toutes les prises de vue sont réalisées avec un cap de 0 degré, les temps de transition minimaux entre prises de vue varient peu avec les dates d'enclenchement des transitions.

Dans le deuxième scénario, le cap de balayage des zones passe à 30 degrés. La figure 4(b) montre que la flexibilité temporelle obtenue avec les TSTN est meilleure d'environ 20 secondes en moyenne, et que la différence de flexibilité entre STN et TSTN croît avec le nombre d'acquisitions planifiées.

Dans le troisième et dernier scénario, la longueur des zones à balayer passe à 40km environ et le cap de balayage est choisi aléatoirement pour chaque zone. Dans ce cas, l'approche STN ne permet de planifier que les séquences de longueur $n = 5$ et 6. Elle conclue à une incohérence du problème pour $n \geq 7$. A l'inverse, l'approche TSTN permet de planifier l'ensemble des 13 acquisitions. Une des raisons est que plus les caps de balayage sont distincts, plus les temps de transition minimaux entre prises de vue dépendent des dates d'enclenchement des transitions. La possibilité d'avoir des caps de balayage distincts est importante en pratique. Elle permet est effet, étant donné un polygone au sol, de le découper suivant des bandes dont l'orien-

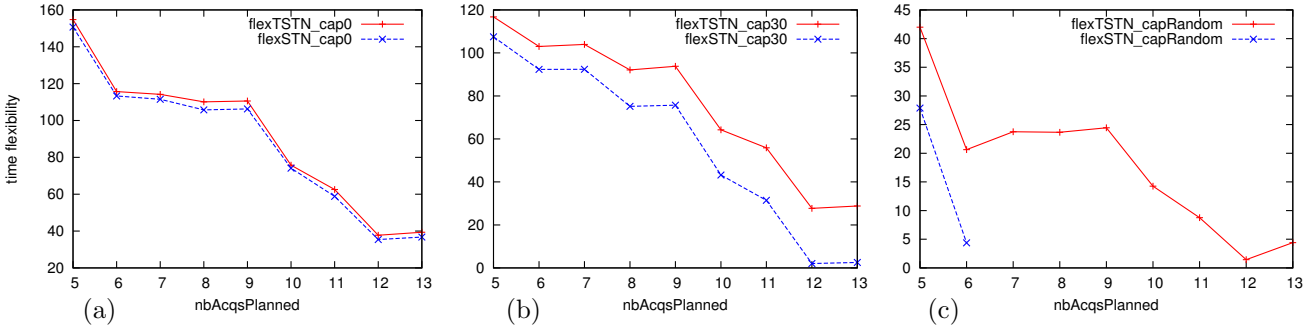


FIGURE 4 – Comparaison des flexibilités temporelles moyennes, en secondes, obtenues avec l'utilisation de majorants précalculés sur les temps de transition (flexSTN) et avec les temps de transitions exacts (flexTSTN)

tation est libre, et donc potentiellement de réduire le nombre de bandes à balayer.

Pour donner un ordre d'idée des temps de calcul, pour 13 acquisitions ajoutées une à une au plan courant, une précision de une seconde sur les dates, et un nombre d'itérations égal à 10^4 pour la convergence des fonctions *firstNeg* et *lastNeg*, l'approche TSTN consomme en moyenne 2ms par ajout. Avec l'approche STN, le temps de calcul est inférieur à 0.1ms par ajout. Pour des précisions de 10^{-1}, 10^{-2}, 10^{-3} et 10^{-4} seconde sur les dates, les temps de calcul avec les TSTN passent respectivement à 3ms, 12ms, 66ms et 182ms par ajout. Une approche type peut consister à étudier d'abord les différents ordonnancements possibles d'un point de vue gros grain puis à préciser les choses en utilisant une précision plus fine.

En conclusion, ce papier a présenté les TSTN, leurs propriétés, des techniques de résolution et une application à la gestion de satellites agiles. Il serait intéressant d'étudier d'autres propriétés de ce cadre et de tester l'approche sur d'autres applications.

Références

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

[2] 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*, 9(3) :299–316, 2011.

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

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

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

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

[7] S. Gawiejnowicz. *Time-Dependent Scheduling*. Springer, 2008.

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

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

[10] P. Van Hentenryck and L. Michel. *Constraint-Based Local Search*. The MIT Press, 2005.

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

[12] 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.

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

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

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

[16] Challenge ROADEF'03. http://challenge.roadef.org.

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

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

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