

How to Model Planning and Scheduling Problems using Timelines

G rard Verfaillie and C dric Pralet

ONERA, Toulouse, France

{Gerard.Verfaillie, Cedric Pralet}@onera.fr

Abstract

The CNT framework (*Constraint Network on Timelines* (Verfaillie, Pralet, and Lema tre 2008)) has been designed to model discrete event dynamic systems and the properties one knows, one wants to verify, or one wants to enforce on them. In this paper, after a reminder about the CNT framework, we show its modeling power, first on two generic problems (the planning problem in the STRIPS framework and the resource-constrained project scheduling problem), then on two specific problems coming from the aerospace domain (the mission management problems for a team of unmanned air vehicles and for an Earth watching and observing satellite).

The CNT framework

The CNT framework (*Constraint Network on Timelines* (Verfaillie, Pralet, and Lema tre 2008)) is a generic constraint-based modeling framework for discrete event dynamic systems. More precisely, the CNT framework allows any discrete event dynamic system (including its dynamics and the properties one wants to verify or to enforce on it) to be modeled as a kind of *dynamic CSP* (*Constraint Satisfaction Problem* (Rossi, Beek, and Walsh 2006; Mittal and Falkenhainer 1990), also referred to as a *conditional CSP*, where the set of variables and constraints is not a priori fixed, but depends on the assignment of special variables called *dimension* variables.

The CNT framework is built on the usual assumption of discrete *steps* at which *events* or *instantaneous changes* occur. Its basic components are *attributes* which may be of three types: *time*, *state*, or *event*. With each attribute, is associated, in addition to its *type*, a *domain* of possible values, which may be discrete or continuous, finite or infinite. The *time* attributes allow the temporal positions of the successive steps to be modeled. The *state* attributes allow the values taken by the permanent features of the system at the successive steps, such as for example a resource level, to be modeled. Finally, the *event* attributes allow the values taken by the instantaneous phenomena in the system at the successive steps, such as for example subsystem failures, to be modeled. The only difference between state and event attributes is the assumption that, between two successive

steps, a state attribute keeps the value it took at the first of both steps (assumption of a piecewise constant evolution) whereas an event attribute takes no value (or a default value which models the absence of event). See Figures 1 and 2.

A *timeline* puts together completely synchronized attributes, which share the same sequence of steps with the same temporal positions. With each timeline, are associated (i) a variable referred to as *dimension* variable which models the number of steps in the timeline, (ii) a finite set of *state* and *event* attributes, and (iii) at most one *time* attribute (none if the temporal positions of the successive steps are not relevant). The domain of the dimension variable may be finite or infinite. With each timeline, each step and each attribute in this timeline, is associated a variable referred to as *attribute* variable which models the value of this attribute at this step. Various timelines allow concurrent evolutions, possibly partially synchronized, to be modeled.

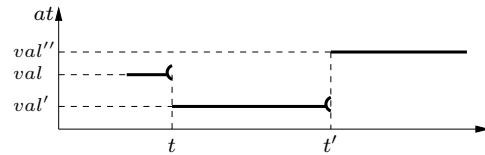


Figure 1: Evolution of a state attribute.

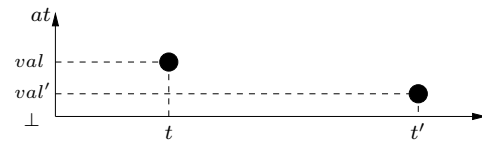


Figure 2: Evolution of an event attribute.

A constraint network on timelines (CNT) is defined by (i) a set of *timelines* which defines itself a set of *variables* (one dimension variable per timeline and one attribute variable per attribute and per step in the associated timeline) and (ii) a set of *constraints* on these variables (with no limitation on the involved variables and on the kind of constraints on these variables). However, differently from classical CSP, the set of variables and constraints associated with a CNT is not a priori fixed. It depends on the assignment of the dimension

variables. The result is a form of *dynamic CSP* (Mittal and Falkenhainer 1990), also referred to as *conditional CSP*.

More formally, we adopt the following definitions:

Definition 1 An attribute at is a pair $\langle ty(at), do(at) \rangle$, where $ty(at)$ is the type of at with three possible values (*state*, *event*, or *time*) and $do(at)$ is the domain of values of at .

If $ty(at) = \textit{state}$, there is no assumption on $do(at)$: it may be discrete or continuous, finite or infinite. If $ty(at) = \textit{event}$, there is no assumption too, except the presence of a special value \perp which models the absence of event. Finally, if $ty(at) = \textit{time}$, then $do(at) \subseteq \mathbb{R}$, but $do(at)$ may be also discrete or continuous, finite or infinite.

Definition 2 A timeline tl is a pair $\langle ats(tl), ns(tl) \rangle$, where $ats(tl)$ is the finite set of attributes associated with tl and $ns(tl)$ is a variable which represents the number of steps in tl .

We make the assumption that an attribute at belongs to one timeline and only one, denoted $tl(at)$, and that a timeline tl involves at most one attribute of type *time*, denoted $ti(tl)$, if it exists. For an attribute at of any type, we denote $ns(at) = ns(tl(at))$ the dimension variable of the timeline at which it belongs. Similarly, for an attribute at of type *state* or *event*, we denote $ti(at) = ti(tl(at))$ the attribute of type *time* of the timeline at which it belongs, if such an attribute exists. We make also the assumption that the domain of values $do(ns(tl))$ of the dimension variable of a timeline tl is any subset of \mathbb{N} , finite or infinite. An infinite domain allows unbounded evolutions to be modeled. The successive steps in a timeline tl are numbered from 1 to $ns(tl)$. With each timeline tl , each attribute $at \in ats(tl)$, and each step $i \mid 1 \leq i \leq ns(tl)$, is associated an attribute variable at_i whose domain of values is $do(at)$. Moreover, for an attribute ti of type *time*, we make the assumption that $\forall i \mid 2 \leq i \leq ns(tl), ti_{i-1} \leq ti_i$ (successive steps are temporally totally ordered) and, for an attribute at of type *state* or *event*, we make the assumption that $\forall i \mid 2 \leq i \leq ns(tl), (ti(at)_{i-1} = ti(at)_i) \rightarrow (at_{i-1} = at_i)$ (if two successive steps are temporally identical, they are identical over all the attributes).

If we consider now a finite set tls of timelines, we can classify the variables associated with tls into *dimension* variables ($d\text{-vars}(tls)$; one per timeline) and *attribute* variables ($a\text{-vars}(tls)$; for each timeline $tl \in tls$, one per attribute and per step in tl). We can also classify them into *mandatory* (or *static*) variables ($m\text{-vars}(tls)$) and *optional* (or *dynamic*) variables ($o\text{-vars}(tls)$). Mandatory variables are the dimension variables and the attribute variables that are associated with mandatory steps in tls . Some steps may be indeed mandatory due to the minimum values in the domains of the dimension variables in tls . If, for example, $do(ns(tl)) = [3.. + \infty]^1$, the timeline tl involves at least three steps and the attribute variables associated with each

attribute $at \in ats(tl)$ and each step $i \mid 1 \leq i \leq 3$ are mandatory. Optional variables are the attribute variables that are not mandatory. It is important to emphasize that the set of optional variables may be infinite, if the domain of some dimension variables is unbounded. However, each assignment A of the dimension variables in tls defines a finite set of optional variables, we denote $o\text{-vars}(tls, A)$.

Definition 3 An assignment A of a finite set tls of timelines is an assignment of all mandatory variables in tls ($m\text{-vars}(tls)$) and of the optional variables that result from the assignment of the dimension variables in tls ($o\text{-vars}(tls, A[d\text{-vars}(tls)])$), if $A[d\text{-vars}(tls)]$ denotes the restriction of A to $d\text{-vars}(tls)$.

Definition 4 A mandatory (or static) constraint c on a finite set tls of timelines is a classical CSP constraint i.e., a pair $\langle sco(c), rel(c) \rangle$, where $sco(c)$ (the scope of c) is a (finite) subset of the mandatory variables in tls ($sco(c) \subseteq m\text{-vars}(tls)$) and $rel(c)$ (the relation associated with c) is any explicit or implicit representation of the allowed combinations of values for the variables in $sco(c)$.

Definition 5 An optional (or dynamic) constraint c on a finite set tls of timelines is a pair $\langle tls(c), fct(c) \rangle$, where $tls(c)$ is a (finite) subset of tls ($tls(c) \subseteq tls$) and $fct(c)$ is a function which associates with each assignment A of the dimension variables in $tls(c)$ a finite set of classical CSP constraints whose scope is, for each one, a (finite) subset of $m\text{-vars}(tls) \cup o\text{-vars}(tls(c), A)$: the set of the mandatory variables in tls and of the optional variables that result from the assignment A of the dimension variables in $tls(c)$.

Definition 6 A constraint network cnt on timelines is a pair $\langle tls(cnt), cs(cnt) \rangle$, where $tls(cnt)$ is finite set of timelines and $cs(cnt)$ is a finite set of mandatory or optional constraints on $tls(cnt)$.

Definition 7 An assignment of a constraint network cnt on timelines is an assignment of $tls(cnt)$. It is consistent if and only if all the constraints in $cs(cnt)$ are satisfied.

Definition 8 A constraint network cnt on timelines is consistent if and only if there exists a consistent assignment of it.

To illustrate these definitions, let us consider a toy example where we are given a set Ls of locations and a robot which starts from a given location $Li \in Ls$ at a given time Ti with a given level of energy Ei , must arrive at another given location $Lg \in Ls$ by a given deadline Tg with a minimum level of energy Eg , and can move from any location in $l_1 \in Ls$ to any other one $l_2 \in Ls$, with each move taking some time $Du[l_1, l_2]$ and consuming some amount of energy $Co[l_1, l_2]$. Durations and consumptions are infinite (equal to large enough numbers) when no direct move is possible from l_1 to l_2 .

To model this problem, we consider one timeline tl where each step is associated with the arrival of the robot at a location and three attributes: an attribute t of type $ty(t) = \textit{time}$ and domain $do(t) = [Ti..Tg]$ which represents the current time, an attribute l of type $ty(l) = \textit{state}$ and domain $do(l) = Ls$ which represents the current location, and an

¹We denote $[a..b]$ the set of integers greater than or equal to integer a and less than or equal to integer b and $[a.. + \infty]$ the set of integers greater than or equal to integer a .

attribute e of type $ty(e) = state$ and domain $do(e) = [Eg..Ei]$ which represents the current level of energy. Thus, we have $ats(tl) = \{t, l, e\}$. Moreover, because the minimum number of steps is equal to 2 (direct move from L_i to L_g) and the maximum is equal to $|L_s|$ (all the locations used), we have $do(ns(tl)) = [2..|L_s|]$.

To specify the initial state, we can use the three following unary mandatory constraints c_1 , c_2 , and c_3 , respectively on t_1 , l_1 , and e_1 :

$$t_1 = Ti \quad (1)$$

$$l_1 = Li \quad (2)$$

$$e_1 = Ei \quad (3)$$

Then, to specify state transitions, we can use the two following optional constraints c_4 and c_5 , with the assumption that energy is consumed at the end of each move:

$$\forall i \in [2..ns(tl)], t_i = t_{i-1} + Du[l_{i-1}, l_i] \quad (4)$$

$$\forall i \in [2..ns(tl)], e_i = e_{i-1} - Co[l_{i-1}, l_i] \quad (5)$$

If we consider c_4 , we have $tls(c_4) = \{tl\}$ and a function $fact(c_4)$ which associates with any assignment NS of the dimension variable $ns(tl)$ the set $\{t_i = t_{i-1} + Du[l_{i-1}, l_i] \mid i \in [2..NS]\}$ of classical CSP constraints, each one connecting variables t_{i-1} , l_{i-1} , t_i , and l_i .

To specify the goal state, we can use the following optional constraint c_6 :

$$l_{ns(tl)} = Lg \quad (6)$$

We have $tls(c_6) = \{tl\}$ and a function $fact(c_6)$ which associates with any assignment NS of the dimension variable $ns(tl)$ the unary classical CSP constraint $l_{NS} = Lg$ on l_{NS} . The deadline Tg and the minimum level of energy Eg are enforced via the domains of values of attributes t and e .

Finally, if we want to enforce that the robot does not use twice the same location, because this would be useless, we can use the following optional constraint c_7 :

$$alldifferent(\{l_i \mid i \in [1..ns(tl)]\}) \quad (7)$$

We have $tls(c_7) = \{tl\}$ and a function $fact(c_7)$ which associates with any assignment NS of the dimension variable $ns(tl)$ the global CSP constraint $alldifferent$ on the set $\{l_i \mid i \in [1..NS]\}$ of variables. Let us emphasize that, due to this latter constraint, the resulting model is not Markovian.

Let us assume that $Ls = \{A, B, C, D\}$, $Li = A$, $Lg = D$, $Ti = 0$, $Tg = 20$, $Ei = 10$, $Eg = 2$, and:

$$Du = \begin{bmatrix} 0 & 5 & 12 & +\infty \\ 5 & 0 & 8 & 17 \\ 12 & 8 & 0 & 5 \\ +\infty & 17 & 5 & 0 \end{bmatrix}$$

$$Co = \begin{bmatrix} 0 & 2 & 7 & +\infty \\ 2 & 0 & 4 & 6 \\ 7 & 4 & 0 & 2 \\ +\infty & 6 & 2 & 0 \end{bmatrix}$$

The CNT associated with this data is consistent and a consistent assignment, with $ns(tl) = 4$, is shown in table 3.

step index i	1	2	3	4
time t	0	5	13	18
location l	A	B	C	D
energy e	10	8	4	2

Figure 3: Consistent CNT assignment.

In fact, the genericity of the CNT framework allows it to subsume many classical frameworks used to model discrete event dynamic systems, such as automata, synchronized products of automata, timed automata, and Petri nets, as well as request on them, such as for example state reachability. See (Verfaillie, Pralet, and Lemaître 2008) for the proofs.

In this paper, we focus on planning and scheduling problems. Before showing how mission management problems for a team of unmanned air vehicles or an Earth watching and observing satellite can be cast into the CNT framework, we start with two well known generic problems: the planning problem in the STRIPS framework and the resource-constrained project scheduling problem (RCPSP).

In this paper, we focus on modeling issues and say nothing about algorithmic ones. This is because we think that modeling is a key question which has not been answered satisfactorily yet and remains one of the main obstacles to the development and the use of planning and scheduling technologies. However, for first CNT solving algorithms and experimental results, see (Pralet and Verfaillie 2008b).

Modeling the planning problem in the STRIPS framework

A planning problem in the STRIPS framework (Ghalab, Nau, and Traverso 2004) is defined by a quadruple $\langle F, A, I, G \rangle$, where:

1. F is a finite set of *fluents* whose values are boolean;
2. A is a finite set of *actions*, with each action $a \in A$ defined by a triple $\langle p_a, e_a^+, e_a^- \rangle \mid p_a, e_a^+, e_a^- \subseteq F$, where p_a , e_a^+ , and e_a^- define respectively the *preconditions*, the *positive effects*, and the *negative effects* of a ;
3. $I \subseteq F$ defines the *initial state*;
4. G is a finite set of logical conditions on F which define the set of *goal states*.

To model such a problem, we can use one timeline tl with $ats(tl) = \{s_f \mid f \in F\} \cup \{a\}$ (one attribute s_f for each fluent $f \in F$, to which we add an attribute a to represent the current action), $do(ns(tl)) = [1..+\infty]$ (at least one step associated with the initial state), $\forall f \in F$, $ty(s_f) = state$, $do(s_f) = \{0, 1\}$ (boolean attributes of type *state*), $ty(a) = event$, $do(a) = A \cup \{\perp\}$ (attribute of type *event*), and the following constraints:

Initial state:

$$\forall f \in I, \quad s_{f,1} = 1 \quad (8)$$

$$\forall f \in F \setminus I, \quad s_{f,1} = 0 \quad (9)$$

No initial action and one action at each step from the second one:

$$a_1 = \perp \quad (10)$$

$$\forall i \in [2..ns(tl)], \quad a_i \neq \perp \quad (11)$$

Action preconditions:

$$\begin{aligned} \forall f \in F, \forall i \in [2..ns(tl)], \\ (f \in p_{a_i}) \rightarrow (s_{f,i-1} = 1) \end{aligned} \quad (12)$$

Positive, negative, and null action effects, all assumed to be instantaneous:

$$\begin{aligned} \forall f \in F, \forall i \in [2..ns(tl)], \\ (f \in e_{a_i}^+) \rightarrow (s_{f,i} = 1) \end{aligned} \quad (13)$$

$$(f \in e_{a_i}^-) \rightarrow (s_{f,i} = 0) \quad (14)$$

$$(f \in F \setminus (e_{a_i}^- \cup e_{a_i}^+)) \rightarrow (s_{f,i} = s_{f,i-1}) \quad (15)$$

Satisfaction of the goal conditions, with $S_i = \{s_{f,i} \mid f \in F\}$:

$$\forall g \in G, g(S_{ns(tl)}) = true \quad (16)$$

We can observe that constraints c_8 to c_{10} are mandatory, whereas constraints c_{11} to c_{16} are optional. Let us also recall the semantics of an optional constraint such as c_{11} which associates with any assignment NS of the dimension variable $ns(tl)$ the set $\{a_i \neq \perp \mid i \in [2..NS]\}$ of classical unary CSP constraints and the semantics of another optional constraint such as c_{16} which, for each logical condition g , associates with any assignment NS of the dimension variable $ns(tl)$ the CSP constraint $g(S_{NS}) = true$.

Modeling the resource-constrained project scheduling problem

A RCPSP (Resource-Constrained Project Scheduling Problem (Baptiste, Pape, and Nuijten 2001)) is defined by a tuple $\langle n, m, D, Co, Ca, Pr, Tmax \rangle$, where:

1. n is the number of *tasks*;
2. m is the number of *resources*;
3. D is a table which associates a *duration* D_t with each task $t \in [1..n]$;
4. Co is a table which associates a *resource consumption* $Co_{t,r}$ with each task $t \in [1..n]$ and each resource $r \in [1..m]$;
5. Ca is a table which associates a *capacity* Ca_r with each resource $r \in [1..m]$: at any time, for each resource r , the sum of the consumptions of r by all the active tasks shall not exceed Ca_r ;
6. Pr is a table which associates with each pair of tasks $t, t' \in [1..n]$ a boolean $Pr_{t,t'}$ equal to 1 if and only if t shall *precede* t' ;
7. $Tmax$ is the *maximum duration* of the project: all the tasks shall end by $Tmax$.

To model a RCPSP, we can use n timelines: a timeline tl_t for each task $t \in [1..n]$, with two attributes per timeline: an attribute ac_t which represents the fact that task t is active

or not and an attribute ti_t which represents the current time i.e, $\forall t \in [1..n], ats(tl_t) = \{ac_t, ti_t\}, do(ns(tl_t)) = [3..3]$ (three steps per timeline associated with time 0 and with the activity starting and ending times of t), $ty(ac_t) = state$, $do(ac_t) = \{\top, \perp\}$ (attribute of type *state*, with two possible values respectively associated with the active state and the inactive one), $ty(ti_t) = time$, $do(ti_t) = [0..Tmax]$ (attribute of type *time* taking its value between 0 and $Tmax$) and the following constraints:

For each task, first step at time 0:

$$\forall t \in [1..n], \quad ti_{t,1} = 0 \quad (17)$$

Activity of each task at steps 1, 2, and 3 (if a task t starts at time 0 ($ti_{t,1} = ti_{t,2} = 0$), we have $ac_{t,1} = ac_{t,2} = \top$ because of the assumption that, if two successive steps are temporally identical, they are identical over all the attributes):

$$\forall t \in [1..n], \quad (ti_{t,2} > ti_{t,1}) \rightarrow (ac_{t,1} = \perp) \quad (18)$$

$$ac_{t,2} = \top \quad (19)$$

$$ac_{t,3} = \perp \quad (20)$$

Duration of each task:

$$\forall t \in [1..n], \quad ti_{t,3} - ti_{t,2} = D_t \quad (21)$$

Precedences between tasks:

$$\forall t, t' \in [1..n] \mid (Pr_{t,t'} = 1), \quad ti_{t,3} \leq ti_{t',2} \quad (22)$$

Resource capacities (when starting each task, for each resource r , the sum of the consumptions of all the active tasks does not exceed the capacity of r):

$$\forall r \in [1..m], \quad \forall t \in [1..n],$$

$$\left(\sum_{t'=1}^n (val(ac_{t'}, tl_t, 2) = \top) \cdot Co_{t',r} \right) \leq Ca_r \quad (23)$$

We can observe that all the variables and constraints are mandatory (all the dimension variables are fixed). In constraint c_{23} , $val(ac_{t'}, tl_t, 2)$ is an expression, function of the problem variables, which refers to the value of the attribute $ac_{t'}$ at step 2 in the timeline tl_t , which differs from the timeline $tl_{t'}$ at which $ac_{t'}$ belongs. Generally speaking, if at is an attribute of type *state* in the timeline $tl(at)$, if tl is another timeline different from $tl(at)$, if tl and $tl(at)$ have both an attribute of type *time*, and if both these attributes can be compared, let be $j = \max(j' \in [1..ns(at)] \mid (ti(at))_{j'} \leq (ti(tl))_i)$: j is the last step in $tl(at)$ that occurs before or just at $(ti(tl))_i$; if j does not exist, $val(at, tl, i)$ is not defined; otherwise, $val(at, tl, i) = at_j$, because of the assumption that, between two successive steps, an attribute of type *state* keeps the value it took at the first of both steps (see Figure 4). We make also the assumption, usual in the CSP framework, that a constraint such as $val(ac_{t'}, tl_t, 2) = \top$ is equivalent to a boolean which is equal to 1 if the constraint is satisfied and 0 otherwise.

In fact, attributes ac_t are not absolutely necessary to model a RCPSP which could be modeled by using only attributes ti_t of type *time*. However, their use may allow RCPSP extensions (with for example various ways of performing a task with various associated durations and resource consumptions) to be more easily modeled.

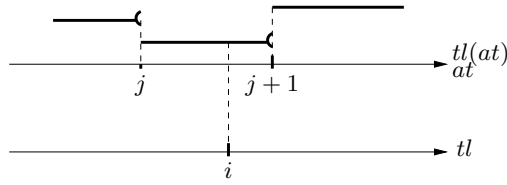


Figure 4: Value of an attribute at of type $state$ at a step i in a timeline tl , different from $tl(at)$.

Modeling the mission management problem for a team of unmanned air vehicles

The problem we consider is inspired from the international competition of small unmanned air vehicles (Micro Air Vehicle Conference Competition; see for example <http://www.mav07.org/>). Its data is the following:

1. a maximum *mission duration* MD ;
2. a *takeoff* and *landing area* HO for all the vehicles ;
3. a set AI of *areas*, each one containing a target to be *identified*; to identify a target, it is necessary to have a camera and to perform an eight over the associated area;
4. a set AL of *areas*, each one containing a target to be *localized*; to localize a target, it is necessary to have a camera and to scan the associated area;
5. a set AT of *areas*, each one containing a target to be *touched*; to touch a target, it is necessary to have at least a marble and to drop it;
6. a number NV of *vehicles*;
7. a table CA with, for each vehicle v , a boolean equal to 1 if and only if v embeds a *camera*;
8. a table NM giving the number of *marbles* initially available in each vehicle;
9. a table SP giving the *speed* of each vehicle;
10. a table DU giving, for each vehicle v and each action a among actions $TOFF$, $LAND$, $EIGHT$, $SCAN$, and $DROP$ (takeoff, landing, eight, scan, and drop) the expected *duration* of a when performed by v ;
11. a table DI giving, for each pair a, a' of areas, including HO , the *distance* from a to a' ;
12. a *minimum duration* $DUMIN$ between two takeoffs or landings of different vehicles at HO .

Let be $A = AI \cup AL \cup AT \cup \{HO\}$ the set of areas, assumed to be all different. Let be also $AC = \{TOFF, LAND, EIGHT, SCAN, DROP, GOTO, NOTHING\}$ (takeoff, landing, eight, scan, drop, goto, and nothing) the set of possible actions.

To model this problem, we can use NV timelines: one timeline tl_v for each vehicle $v \in [1..NV]$, with five attributes per timeline: an attribute fl_v to represent the fact that v is *flying* or not, an attribute at_v to represent its current *position* (area), an attribute nm_v to represent the current number of available *marbles*, an attribute

ac_v to represent the current *action*, and an attribute ti_v to represent the current time i.e., $\forall v \in V, ats(tl_v) = \{fl_v, at_v, nm_v, ac_v, ti_v\}$, $do(ns(tl_v)) = [1..2 \cdot |A| + 3]$ (at least one step associated with the initial state and at most $2 \cdot |A| + 3$ ones associated with the case where all the areas are handled by the same vehicle), $ty(fl_v) = state$, $do(fl_v) = \{\top, \perp\}$ (attribute of type $state$, with two possible values associated with in flight and on the ground), $ty(at_v) = state$, $do(at_v) = A$ (attribute of type $state$, with the set A of areas as a domain), $ty(nm_v) = state$, $do(nm_v) = [0..NM_v]$ (attribute of type $state$, with the set $[0..NM_v]$ as a domain), $ty(ac_v) = state$, $do(ac_v) = AC$ (attribute of type $state$, with the set AC of possible actions as a domain), $ty(ti_v) = time$, $do(ti_v) = [0, MD]$ (attribute of type $time$, with the set $[0..MD]$ as a domain), and the following constraints, which assume that action effects occur at the end of actions:

Initial state of each vehicle:

$$\forall v \in [1..NV], \quad fl_{v,1} = \perp \quad (24)$$

$$at_{v,1} = HO \quad (25)$$

$$nm_{v,1} = NM_v \quad (26)$$

$$ac_{v,1} \in \{TOFF, NOT\} \quad (27)$$

$$ti_{v,1} = 0 \quad (28)$$

Final state of each vehicle:

$$\forall v \in [1..NV], \quad fl_{v,ns(tl_v)} = \perp \quad (29)$$

$$at_{v,ns(tl_v)} = HO \quad (30)$$

$$ac_{v,ns(tl_v)} = NOT \quad (31)$$

Conditions and effects of a takeoff:

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v) - 1], \quad (32)$$

$$(ac_{v,i} = TOFF) \rightarrow$$

$$((fl_{v,i} = \perp) \wedge$$

$$(at_{v,i} = HO) \wedge$$

$$(fl_{v,i+1} = \top) \wedge$$

$$(ti_{v,i+1} - ti_{v,i} = DU_{v,TOFF}))$$

Conditions and effects of a landing:

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v) - 1], \quad (33)$$

$$(ac_{v,i} = LAND) \rightarrow$$

$$((fl_{v,i} = \top) \wedge$$

$$(at_{v,i} = HO) \wedge$$

$$(fl_{v,i+1} = \perp) \wedge$$

$$(ti_{v,i+1} - ti_{v,i} = DU_{v,LAND}))$$

Conditions and effects of an eight:

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v) - 1], \quad (34)$$

$$(ac_{v,i} = EIGHT) \rightarrow$$

$$((CA_v = 1) \wedge$$

$$(fl_{v,i} = \top) \wedge$$

$$(at_{v,i} \in AI) \wedge$$

$$(ti_{v,i+1} - ti_{v,i} = DU_{v,EIGHT}))$$

Conditions and effects of a scan:

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v) - 1], \quad (35)$$

$$(ac_{v,i} = SCAN) \rightarrow$$

$$((CA_v = 1) \wedge$$

$$(fl_{v,i} = \top) \wedge$$

$$(at_{v,i} \in AL) \wedge$$

$$(ti_{v,i+1} - ti_{v,i} = DU_{v,SCAN}))$$

Conditions and effects of a drop:

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v) - 1], \quad (36)$$

$$(ac_{v,i} = DROP) \rightarrow$$

$$((fl_{v,i} = \top) \wedge$$

$$(at_{v,i} \in AT) \wedge$$

$$(nm_{v,i} - nm_{v,i+1} = 1) \wedge$$

$$(ti_{v,i+1} - ti_{v,i} = DU_{v,DROP}))$$

Conditions and effects of a move:

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v) - 1], \quad (37)$$

$$(ac_{v,i} = GOTO) \rightarrow$$

$$((fl_{v,i} = \top) \wedge$$

$$(at_{v,i+1} \neq at_{v,i}) \wedge$$

$$(ti_{v,i+1} - ti_{v,i} = DI_{at_{v,i},at_{v,i+1}}/SP_v))$$

Conditions of a null action:

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v)], \quad (38)$$

$$(ac_{v,i} = NOT) \rightarrow$$

$$((fl_{v,i} = \perp) \wedge$$

$$(at_{v,i} = HO))$$

Conditions of change for the attributes fl , at , and nm :

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v) - 1],$$

$$(fl_{v,i+1} \neq fl_{v,i}) \rightarrow$$

$$(ac_{v,i} \in \{TOFF, LAND\}) \quad (39)$$

$$(at_{v,i+1} \neq at_{v,i}) \rightarrow$$

$$(ac_{v,i} = GOTO) \quad (40)$$

$$(nm_{v,i+1} \neq nm_{v,i}) \rightarrow$$

$$(ac_{v,i} = DROP) \quad (41)$$

Each area shall be handled once and only once:

$$\forall a \in AI, \quad (42)$$

$$\sum_{v=1}^{NV} \sum_{i=1}^{ns(tl_v)} ((at_{v,i} = a) \wedge (ac_{v,i} = EIGHT)) = 1$$

$$\forall a \in AL, \quad (43)$$

$$\sum_{v=1}^{NV} \sum_{i=1}^{ns(tl_v)} ((at_{v,i} = a) \wedge (ac_{v,i} = SCAN)) = 1$$

$$\forall a \in AT, \quad (44)$$

$$\sum_{v=1}^{NV} \sum_{i=1}^{ns(tl_v)} ((at_{v,i} = a) \wedge (ac_{v,i} = DROP)) = 1$$

Minimum duration between takeoffs and landings:

$$\forall v, v' \in [1..NV] \mid v < v', \quad (45)$$

$$\forall i \in [1..ns(tl_v)], \forall i' \in [1..ns(tl_{v'})],$$

$$((ac_{v,i} \in \{TOFF, LAND\})$$

$$\wedge (ac_{v',i'} \in \{TOFF, LAND\})) \rightarrow$$

$$((ti_{v',i'} \geq ti_{v,i+1} + DUMIN) \vee$$

$$(ti_{v,i} \geq ti_{v',i'+1} + DUMIN))$$

Different successive actions:

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v) - 1], \quad (46)$$

$$ac_{v,i+1} \neq ac_{v,i}$$

No landing followed by a takeoff:

$$\forall v \in [1..NV], \quad \forall i \in [1..ns(tl_v) - 1], \quad (47)$$

$$(ac_{v,i} = LAND) \rightarrow$$

$$(ac_{v,i+1} \neq TOFF)$$

Null actions only at the beginning or at the end:

$$\forall v \in [1..NV], \quad \forall i \in [2..ns(tl_v) - 1], \quad (48)$$

$$ac_{v,i} \neq NOT$$

Such a modeling is driven by two principles used by SAT or CSP encodings of planning problems (Kautz and Selman 1992): (i) the presence of an action implies its preconditions and effects (effective changes in attribute values; see constraints c_{32} to c_{38}) and (ii) any effective change in an attribute value is justified by an action (see constraints c_{39} to c_{41}).

Resource constraints such as the fact that a drop cannot be triggered when the number of available marbles is null is enforced by the domain of attributes nm_v . The same way, the maximum mission duration is enforced by the domain of attributes ti_v .

Modeling the mission management problem for an Earth watching and observing satellite

The problem we consider now is a simplified version of the problem of management of the observations and data downloads by an autonomous Earth watching and observing satellite. A description of the whole problem, a timeline-based modeling, and approximate solving algorithms are presented in (Pralet and Verfaillie 2008a). The data of the simplified version is the following:

1. a *planning horizon* defined by its *starting* and *ending times* STA and END ;
2. *satellite features* such as the maximum available *memory* MM_{max} , the minimum and maximum levels of *energy* EN_{min} and EN_{max} , the *power* P_{sun} delivered by the solar panels during day periods, the *powers* P_{sat} , P_{ob} , and P_{dl} consumed respectively by the platform, by any observation action, and by any download action, and the *rotation speed* MS of the sight mirror;
3. an *initial state* defined by the initial available *memory* MM_0 , the initial *energy* level EN_0 , and the initial *orientation* OR_0 of the sight mirror;

4. a set OB of *observations* to be performed and downloaded with, for each observation $o \in OB$, the required *memory* SZ_o , the observation and download durations DO_o and DDO_o , the number NO_o of possible *observation windows* over the planning horizon and, for each window $k \in [1..NO_o]$, its *starting time* $SO_{o,k}$ and the required *orientation* $OR_{o,k}$ of the sight mirror;
5. a number ND of *download windows* over the planning horizon with, for each window $k \in [1..ND]$, its *starting time* SD_k and its *duration* DD_k ;
6. a number NE of *eclipse windows* for the satellite over the planning horizon with, for each window $k \in [1..NE]$, its *starting time* SE_k and its *duration* DE_k .

To model this problem, we can first use usual CSP variables which may be seen as timelines with only one attribute and one step.

1. for each observation $o \in OB$, its *observation window* $no_o \in [0..NO_o]$ and its *download window* $nd_o \in [0..ND]$ (values 0 allow the absence of observation or of download to be modeled);
2. for each observation $o \in OB$, its *observation starting and ending times* so_o and $eo_o \in [STA..END]$;
3. for each download window $k \in [1..ND]$, its *download starting and ending times* sd_k and $ed_k \in [SD_k..SD_k + DD_k]$.

Then, we use only one timeline tl which models the evolution of the state of the satellite, with six attributes: three attributes ob , dl , and ec which respectively represent the fact that the satellite is *observing* or not, *downloading* or not, and in an *eclipse* period or not, two attributes mm and en which respectively represent the current level of available *energy* and *memory*, and an attribute ti which represents the current *time* i.e., $ats(tl) = \{ob, dl, ec, mm, en, ti\}$, $do(ns(tl)) = [2..NSmax]$, with $NSmax = 2 \cdot (NO + ND + NE) + 2$ (at least two steps associated with times STA and END and at most $NSmax$ steps associated with the case where all the observations are performed, all the download windows are used, and all the window starting and ending times are different), $ty(ob) = ty(dl) = ty(ec) = state$, $do(ob) = do(dl) = do(ec) = \{0, 1\}$ (attributes of type *state*, with two possible values respectively associated with the fact that the satellite is observing or not, downloading or not, and in an eclipse period or not), $ty(mm) = ty(en) = state$, $do(mm) = [0..MMmax]$, $do(en) = [ENmin..ENmax]$ (attribute of type *state*), $ty(ti) = time$, $do(ti) = [STA..END]$ (attribute of type *time*), and the following constraints:

Starting and ending times of an observation: the starting time is the starting time of the chosen window; the ending time is the starting time plus the duration; if the observation is not performed, then starting and ending times are arbitrarily set to STA :

$$\forall o \in OB, \quad (no_o \neq 0) \rightarrow (so_o = SO_{o,no_o}) \quad (49)$$

$$(no_o \neq 0) \rightarrow (eo_o = so_o + DO_o) \quad (50)$$

$$(no_o = 0) \rightarrow (so_o = eo_o = STA) \quad (51)$$

Starting and ending times of a data download: the ending time is the starting time plus the duration of all the downloads performed in this window; if no download is performed, then starting and ending times are arbitrarily set to the window starting time:

$$\forall k \in [1..ND],$$

$$ed_k = sd_k + \sum_{o \in OB} (nd_o = k) \cdot DDO_o \quad (52)$$

$$(ed_k = sd_k) \rightarrow (ed_k = sd_k = SD_k) \quad (53)$$

Constraints between observations and data downloads: if an observation is not performed, then it is not downloaded; else, it is downloaded after the end of the observation:

$$\forall o \in OB, \quad (no_o = 0) \rightarrow (nd_o = 0) \quad (54)$$

$$(no_o \neq 0) \rightarrow (eo_o \leq sd_{nd_o}) \quad (55)$$

Constraints between observations: if two observation windows are conflicting and if the first one is performed, the second one cannot be performed; moreover, if an observation window is in conflict with the initial mirror orientation, it cannot be performed:

$$\begin{aligned} & \forall o, o' \in OB, \forall k \in [1..NO_o], \forall k' \in [1..NO_{o'}] \mid \\ & ((o \neq o') \wedge (SO_{o,k} \leq SO_{o',k'})) \\ & (SO_{o',k'} < SO_{o,k} + DO_o + \frac{|OR_{o,k} - OR_{o',k'}|}{MS}) \\ & (no_o = k) \rightarrow (no_{o'} \neq k') \end{aligned} \quad (56)$$

$$\begin{aligned} & \forall o \in OB, \forall k \in [1..NO_o] \mid \\ & SO_{o,k} < STA + \frac{|OR_{o,k} - OR_0|}{MS}, \\ & no_o \neq k \end{aligned} \quad (57)$$

Initial satellite state:

$$mm_1 = MM_0 \quad (58)$$

$$en_1 = EN_0 \quad (59)$$

$$ti_1 = STA \quad (60)$$

Observing, downloading, and eclipse states:

$$\begin{aligned} & \forall i \in [1..ns(tl)], \\ & (ob_i = 1) \leftrightarrow \forall o \in OB (so_o \leq ti_i < eo_o) \end{aligned} \quad (61)$$

$$(dl_i = 1) \leftrightarrow \forall_{k=1}^{ND} (sd_k \leq ti_i < ed_k) \quad (62)$$

$$(ec_i = 1) \leftrightarrow \forall_{k=1}^{NE} (SE_k \leq ti_i < SE_k + DE_k) \quad (63)$$

Memory evolution: the available memory is equal to the memory previously available, plus what is produced by the downloads that end and minus what is consumed by the observations that start (observations are assumed to consume memory when they start and downloads are assumed to produce memory when they end):

$$\begin{aligned} & \forall i \in [1..ns(tl) - 1], \\ & mm_{i+1} = mm_i + \\ & \sum_{o \in OB} ((ti_{i+1} = ed_{nd_o}) - (ti_{i+1} = so_o)) \cdot SZ_o \end{aligned} \quad (64)$$

Energy evolution: the available energy is equal to the energy previously available, plus what has been produced by the solar panels if the satellite was not in an eclipse period and minus what has been consumed by the platform, by observation if the satellite was observing, and by download if it was downloading, with a maximum of $ENmax$:

$$\begin{aligned} \forall i \in [1..ns(tl) - 1], \\ en_{i+1} = \min(ENmax, en_i + (ti_{i+1} - ti_i) \cdot \\ ((1 - ec_i) \cdot P_{sun} - ob_i \cdot P_{ob} - dl_i \cdot P_{dl} - P_{sat})) \end{aligned} \quad (65)$$

Successive steps and times: let $TO = \cup_{o \in OB | so_o \neq eo_o} \{so_o, eo_o\}$, $TD = \cup_{k \in [1..ND] | sd_k \neq ed_k} \{sd_k, ed_k\}$, and $TE = \cup_{k \in [1..NE]} \{SE_k, SE_k + DE_k\}$ be respectively the sets of effective observation, download, and eclipse starting and ending times; let $T = TO \cup TD \cup TE \cup \{STA, END\}$ and let ST be the set T sorted using an increasing order; the number of steps is equal to $|T|$ and, at each step i , the current time is equal to ST_i :

$$union(TO, TD, TE, \{STA, END\}, T) \quad (66)$$

$$ns(tl) = |T| \quad (67)$$

$$sort(T, ST) \quad (68)$$

$$\forall i \in [1..ns(tl)], ti_i = ST_i \quad (69)$$

$union$ is assumed to be a global constraint which enforces that T be the union (without any duplication) of TO , TD , TE , and $\{STA, END\}$ and $sort$ is assumed to be another global constraint which enforces that ST be an increasingly ordered permutation of T (see the *Global Constraint Catalog*, <http://www.emn.fr/x-info/sdemasse/gccat/>).

Memory and energy limitations are enforced by the domains of attributes mm et en .

Lessons and research directions

Contrarily to classical frameworks used in the planning and scheduling community, such as STRIPS, RCPS, or PDDL (Fox and Long 2003), which are all built around the notion of *action*, the CNT framework is a more basic modeling framework, built around the notions of *attribute* and *step*. This is its strength, but may be its weakness too. Indeed, the modeling exercises presented in this paper show that the CNT framework allows complex dynamic phenomena such as concurrent interdependent evolutions to be finely modeled. But, they show also that the modeling task remains a complex human task where the modeler must cope with various modeling alternatives which cannot be easily a priori assessed and where the risks of omission or error are not negligible.

Our current work focuses on algorithmic issues. First, we have already modeled and solved the two problems presented in this paper, coming from the aerospace domain, using standard Constraint Programming (CP) tools, such as *OPL Studio* (<http://www.ilog.com/products/oplstudio/>) and *Comet* (<http://www.comet-online.org/>). See for example (Pralet and Verfaillie 2008a). This was possible because, in both these problems, the domains of the dimension variables are bounded and because a variable number of

steps can easily be managed using *null* steps. Then, we are currently working on algorithms able to manage dimension variables with unbounded domains. See (Pralet and Verfaillie 2008b) for generic CNT solving algorithms implemented on top of the CP *Choco* solver (<http://choco-solver.net/>).

Future works should allow the CNT framework to be extended to state variable evolutions which would be more complex than piecewise constant ones, to optimization via local utility functions (soft constraints (Bistarelli et al. 1999)), and to sequential decision-making under uncertainty via local plausibility functions (soft constraints of another type (Pralet, Verfaillie, and Schiex 2007)).

References

- [Baptiste, Pape, and Nuijten 2001] Baptiste, P.; Pape, C. L.; and Nuijten, W. 2001. *Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers.
- [Bistarelli et al. 1999] Bistarelli, S.; Montanari, U.; Rossi, F.; Schiex, T.; Verfaillie, G.; and Fargier, H. 1999. Semiring-Based CSPs and Valued CSPs: Frameworks, Properties and Comparison. *Constraints* 4(3):199–240.
- [Fox and Long 2003] Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20:61–124.
- [Ghallab, Nau, and Traverso 2004] Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.
- [Kautz and Selman 1992] Kautz, H., and Selman, B. 1992. Planning as Satisfiability. In *Proc. of the 10th European Conference on Artificial Intelligence (ECAI-92)*, 359–363.
- [Mittal and Falkenhainer 1990] Mittal, S., and Falkenhainer, B. 1990. Dynamic Constraint Satisfaction Problems. In *Proc. of the 8th National Conference on Artificial Intelligence (AAAI-90)*, 25–32.
- [Pralet and Verfaillie 2008a] Pralet, C., and Verfaillie, G. 2008a. Decision upon Observations and Data Downloads by an Autonomous Earth Surveillance Satellite. In *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-08)*.
- [Pralet and Verfaillie 2008b] Pralet, C., and Verfaillie, G. 2008b. Using Constraint Networks on Timelines to Model and Solve Planning and Scheduling Problems. In *Proc. of the 18th International Conference on Artificial Intelligence Planning and Scheduling (ICAPS-08)*. To appear.
- [Pralet, Verfaillie, and Schiex 2007] Pralet, C.; Verfaillie, G.; and Schiex, T. 2007. An Algebraic Graphical Model for Decision with Uncertainties, Feasibilities, and Utilities. *Journal of Artificial Intelligence Research* 29:421–489.
- [Rossi, Beek, and Walsh 2006] Rossi, R.; Beek, P. V.; and Walsh, T., eds. 2006. *Handbook of Constraint Programming*. Elsevier.
- [Verfaillie, Pralet, and Lemaître 2008] Verfaillie, G.; Pralet, C.; and Lemaître, M. 2008. Constraint-based Modeling of Discrete Event Dynamic Systems. *Journal of Intelligent Manufacturing, Special Issue on "Planning, Scheduling, and Constraint Satisfaction"*. To appear.