The Knowledge Engineering Review, Vol. 25:3, 319–336. © Cambridge University Press, 2010
doi:10.1017/S0269888910000172

How to model planning and scheduling problems using constraint networks on timelines

GÉRARD VERFAILLIE, CÉDRIC PRALET and MICHEL LEMAÎTRE

ONERA, 2 av. Édouard Belin, BP 74025, F-31055 Toulouse Cedex 4, France;
e-mail: Gerard.Verfaillie@onera.fr, Cedric.Pralet@onera.fr, Michel.Lemaitre@onera.fr

Abstract

The CNT framework (*Constraint Network on Timelines*) 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 article, after a reminder about the CNT framework, we show its modeling power and its ability to support various modeling styles, coming from the planning, scheduling, and constraint programming communities. We do that by producing and comparing various models of two mission management problems in the aerospace domain: management of a team of unmanned air vehicles and of an Earth observing satellite.

1 Introduction

The CNT framework (*Constraint Network on Timelines* (Verfaillie *et al.*, 2008)) is a generic constraint-based modeling framework for discrete event dynamic systems, that is, for systems that are submitted to *instantaneous events* or *changes* which occur at discrete *steps*. As its name suggests, the basic ingredients of the CNT framework are *timelines* and *constraints* on timelines.

Timelines allow the way the attributes of a system evolve with time to be modeled. With each timeline x is associated a special variable, called *horizon* variable, which represents the possibly unknown number of steps in timeline x. With each timeline x and each of its steps i are associated two variables, which represent, respectively, the *temporal position* of step i and the *value* of the timeline x at step i.

See Figure 1 for a graphical representation of a timeline x whose horizon variable is equal to 4. For example, variable t_3 represents the temporal position of step 3 and variable x_3 the value of timeline x at step 3. See Figure 2 for an equivalent tabular representation.

Classical variables can be added to the variables associated with timelines. Between all these variables, constraints can be defined in order to limit the possible combinations of values. The result is a kind of *dynamic CSP* (*Constraint Satisfaction Problem*; Mittal & Falkenhainer, 1990; Rossi *et al.*, 2006), in which the set of variables and constraints is not fixed, but depends on the assignment of the horizon variables of the timelines.

In (Verfaillie *et al.*, 2008), it has been shown that the CNT framework subsumes existing frameworks used to model discrete event dynamic systems, such as *automata*, *timed automata*, or *Petri nets*. It has also been shown how planning and scheduling problems, expressed in generic frameworks such as *STRIPS* (Ghallab *et al.*, 2004) or *RCPSP* (Baptiste *et al.*, 2001), can be modeled using the CNT framework.

In this article, we adopt another point of view. Using two examples of mission management problems in the aerospace domain we recently had to deal with (management of a team of unmanned air vehicles and of an Earth observing satellite), we want to demonstrate the modeling

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

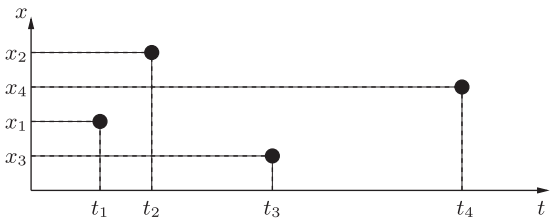


Figure 1 Graphical representation of a timeline x

steps	1	2	3	4
temporal positions	t_1	t_2	t_3	t_4
values	x_1	x_2	x_3	x_4

Figure 2 Equivalent tabular representation

power and the versatility of the CNT framework. More precisely, we want to show its ability to support various modeling styles: *action-based* modeling styles used in the Planning community, *constraint-based* modeling styles used in the Scheduling and Constraint Programming (CP) communities, as well as various combinations of them.

The article is organized as follows. In Section 2, we introduce a small toy planning problem we will use to illustrate the basic definitions of the CNT framework. These definitions are given in Section 3. In Section 4, various models of the problem of management of the mission of a team of unmanned air vehicles are proposed and discussed. In Section 5, a mixed model of the problem of management of the mission of an Earth observing satellite is also proposed. Section 6 provides the reader with some guidelines about the use of the CNT framework. Section 7 discusses related works aiming at modeling planning and scheduling problems. Finally, Section 8 concludes with several current and future research directions in terms of modeling frameworks and solving algorithms.

In this article, we focus on modeling issues and say almost nothing about algorithmic ones. This is not because we would consider the algorithmic issues as being secondary. This is simply because we think that modeling all the features of real planning and scheduling problems is a key question which has not been answered satisfactorily as yet and remains one of the main obstacles to the development and the use of planning and scheduling technologies. However, for each of the two mission management problems we consider, we mention how we solved them, using either generic or specific tools. Moreover, the interested reader may look at (Pralet and Verfaillie, 2008b) for CNT models of problems from the International Planning Competition (IPC), first generic CNT solving algorithms, and experimental results.

2 An illustrative example

Let us consider the following toy planning problem. We assume a team of robots, made of two robots that are different in terms of efficiency, speed, and energy consumption, but are both initially present at the same location. The team is given the following mission: to deliver a packet to another location by a deadline and with a final level of energy greater than or equal to a given threshold, by choosing for that one of the two robots. No energy production is possible when in motion. More precisely, we assume the following data:

- a set $\mathcal{L}s$ of *locations* the robots can use to move from the initial location $Li \in \mathcal{L}s$ to the final one $Lg \in \mathcal{L}s \mid Lg \neq Li$;
- an initial *time* Ti and a deadline Tg, both positive integers;

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

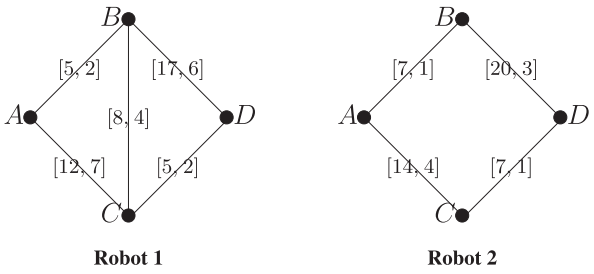


Figure 3 Graphical representation of an instance of the toy planning problem. For each robot and each possible direct move is associated an edge labeled by a pair *[d, c]*, where *d* is the duration and *c* is the energy consumption

- for each robot $r \in \{1, 2\}$, an initial level of *energy* $Ei[r]$ and, for both robots, a minimum final level of energy Eg, all positive integers;
- for each robot $r \in \{1, 2\}$ and each pair of locations $l_1, l_2 \in \mathcal{L}s \mid l_1 \neq l_2$:
  - a boolean $Mo[r, l_1, l_2]$ equal to 1 if robot r can *move* directly from l_1 to l_2 and conversely, and equal to 0 otherwise;
  - a positive integer *duration* $Du[r, l_1, l_2]$ of the direct move of robot r from l_1 to l_2 and conversely, when possible;
  - a positive integer energy *consumption* $Co[r, l_1, l_2]$ of the direct move of robot r from l_1 to l_2 and conversely, when possible too.

Let us consider the specific instance of Figure 3, in which are represented, for each robot, the possible direct moves and, for each of them, the associated duration and energy consumption. In addition, we assume that $Li = A$, $Lg = D$, $Ti = 0$, $Tg = 20$, $Ei[1] = 10$, $Ei[2] = 8$, and $Eg = 2$. We observe that the first robot is more efficient, faster, but more energy consuming than the second one.

The only solution of this instance is to use the first robot to move from A to B, C, and finally D.

3 The Constraint Network on Timelines framework

Let us use now the toy planning problem described in the previous section to illustrate the basic definitions of the CNT framework. Although these definitions differ significantly from the ones given in Verfaillie *et al.* (2008), they are semantically equivalent.

3.1 Horizon variables

Horizon variables are used to represent the number of steps to be considered.

DEFINITION 1. *A horizon variable h is a variable whose domain of values is any subset of* $\mathbb{N}$. *We will use* $\mathbf{D}(h)$ *to denote the domain of a horizon variable h.*

In the toy planning problem, the minimum number of steps is equal to 2 (solution with a direct move from Li to Lg and thus with two steps: the initial and the final ones) and the maximum number of steps is equal to $|\mathcal{L}s|$ (solution using all the locations to move from Li to Lg; going several times through the same location is physically possible, but counterproductive). Hence, we will use a horizon variable h whose domain $\mathbf{D}(h)$ is $[2..|\mathcal{L}s|]$[1].

Note that although the domain of h is bounded in this problem, the domain of any horizon variable is in general bounded or unbounded.

[1] $[a..b]$ denotes the set of integers greater than or equal to integer a and less than or equal to integer b.

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

3.2 Time references

Time references are used to represent the temporal positions of the successive steps.

DEFINITION 2. *A time reference t is a pair $\langle D, h \rangle$ where:*

- D is any subset of $\mathbb{R}$; D is the domain of values of t;
- h is a horizon variable; h is the horizon of t; it represents the number of steps in t.

We will use $\mathbf{D}(t)$ *and* $\mathbf{h}(t)$ *to denote, respectively, the domain and the horizon of a time reference* t.

In the toy planning problem, the minimum time to be considered is Ti and the maximum is Tg. Hence, we will use a time reference t whose domain $\mathbf{D}(t)$ is $[Ti..Tg]$ and whose horizon variable $\mathbf{h}(t)$ is the horizon variable h introduced in the previous section.

Note that although the domain of t is a finite set of integers in this problem, the domain of any time reference is in general bounded or unbounded, and discrete or continuous. Note also that although only one time reference is useful to model this problem, several time references, using several horizon variables, may be useful to model other problems. See, for example, Section 4.

3.3 Timelines

Timelines are used to represent the values of the relevant attributes of the system at successive steps.

DEFINITION 3. *A timeline x is a pair $\langle D, t \rangle$ where:*

- D is the domain of values of x;
- t is the time reference of x.

We will use $\mathbf{D}(x)$ *and* $\mathbf{t}(x)$ *to denote, respectively, the domain and the time reference of a timeline* x. *In short, we will often use* $\mathbf{h}(x)$ *to denote the horizon of the time reference of a timeline* x: $\mathbf{h}(x) = \mathbf{h}(\mathbf{t}(x))$.

Two timelines are fully synchronized if they share the same time reference. Two timelines that do not share the same time reference can, however, be partially synchronized using dynamic constraints between their time references (see Section 3.8).

In the toy planning problem, the relevant attributes at each step are the current location and the current level of energy of the chosen robot. Hence, we will use two timelines l and e, which represent, respectively, the location and the level of energy. The domain $\mathbf{D}(l)$ of timeline l is $\mathcal{L}s$ and the domain $\mathbf{D}(e)$ of timeline e is $[Eg..Ei]$, if $Ei = max(Ei[1], Ei[2])$. Both timelines are fully synchronized and share the same time reference t introduced in the previous section.

Note that although the domains of l and e are finite in this problem, there is in general no restriction on the domain of any timeline. It is symbolic or numeric, discrete or continuous, and finite or infinite. Note also that although only two fully synchronized timelines are useful to model this problem, several timelines, using several time references and consequently only partially synchronized, may be useful to model other problems. See, for example, Section 4.

3.4 Static variables

DEFINITION 4. *A static variable is either a horizon variable, or any other variable independent from time references and timelines.*

In the toy planning problem, it may be convenient to introduce a static variable r whose domain $\mathbf{D}(r)$ is $\{1, 2\}$ to represent the robot that is chosen for the mission.

Note that although the domain of r is finite in this problem, there is in general no restriction on the domain of any static variable, if it is not a horizon variable. If it is a horizon variable, its domain must be indeed a subset of $\mathbb{N}$ according to Definition 1.

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

3.5 Dynamic variables

Dynamic variables are associated with steps of time references and timelines.

DEFINITION 5. *A time reference t and an assignment $a \in \mathbf{D}(\mathbf{h}(t))$ of its horizon variable $\mathbf{h}(t)$ together induce a finite set of variables $\mathbf{VI}(t, a) = \{t_i \mid i \in [1..a]\}$. This set is empty when $a = 0$. All these variables share the same domain of values $\mathbf{D}(t)$. These variables can be referred to as dynamic temporal variables.*

DEFINITION 6. *Similarly, a timeline x and an assignment $a \in \mathbf{D}(\mathbf{h}(x))$ of its horizon variable $\mathbf{h}(x)$ together induce a finite set of variables $\mathbf{VI}(x, a) = \{x_i \mid i \in [1..a]\}$. This set is empty when $a = 0$. All these variables share the same domain of values $\mathbf{D}(x)$. These variables can be referred to as dynamic a temporal variables.*

For example, in the toy planning problem, $h = 2$ induces the dynamic temporal variables t_1 and t_2, and the dynamic atemporal variables l_1, l_2, e_1, and e_2.

3.6 Constraint Satisfaction Problem constraints

The classical definition of a constraint in the CSP framework (Rossi *et al.*, 2006) is the following one:

DEFINITION 7. *A CSP constraint c is a pair $\langle S, R \rangle$ where:*

- S is a finite set of variables; S is the scope of c;
- R is any explicit or implicit representation of the set of allowed combinations of values of the variables in S: $R \subseteq \Pi_{v \in S}\mathbf{D}(v)$; R is the relation associated with c.

We will use $\mathbf{S}(c)$ and $\mathbf{R}(c)$ to denote, respectively, the scope and the relation of a CSP constraint c.

3.7 Static constraints

Static constraints are used to limit the possible combinations of assignments of static variables.

DEFINITION 8. *A static constraint c is simply a CSP constraint whose scope $\mathbf{S}(c)$ is limited to static variables.*

In the toy planning problem, there is *a priori* no such constraint. However, let us assume that we use graph algorithms to precompute for each robot the minimum and the maximum length, in terms of number of edges, to move from the initial to the final location, without going twice through the same location, ignoring the time and energy constraints. Let $Nmin[r]$ and $Nmax[r]$ be these minimum and maximum lengths. For example, in the specific instance we consider, we have: $Nmin[1] = Nmin[2] = 2$, $Nmax[1] = 3$, and $Nmax[2] = 2$. This would allow us to define the following static constraint between static variable r and static horizon variable h:

$$(Nmin[r] + 1) \leqslant h \leqslant (Nmax[r] + 1) \tag{1}$$

3.8 Dynamic constraints

Dynamic constraints are used to limit the possible combinations of assignments of static and dynamic variables. The difficulty is that the set of dynamic variables, associated with time references and timelines, is not fixed. It depends on the assignments of the horizon variables. This leads us to a definition of what is a dynamic constraint which is not as obvious as the previous definitions are. We will use several examples to illustrate it.

DEFINITION 9. *A dynamic constraint c is a tuple $\langle SV, ST, SX, f \rangle$ where:*

- SV is a finite set of static variables; SV is the scope of c in terms of static variables;
- ST is a finite set of time references; ST is the scope of c in terms of time references;
- SX is a finite set of timelines; SX is the scope of c in terms of timelines;

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

- let SH be the set of horizon variables associated with the time references in ST and the timelines in SX; we have $SH = (\cup_{t \in ST} \mathbf{h}(t)) \cup (\cup_{x \in SX} \mathbf{h}(x))$[2]; f is a function which associates a finite set of CSP constraints with each assignment A of the horizon variables in SH.

For each assignment A of the horizon variables in SH, let $SD(A)$ be the set of dynamic variables induced by A in the time references in ST and the timelines in SX. We have $SD(A) = (\cup_{t \in ST} \mathbf{VI}(t, A[\mathbf{h}(t)])) \cup (\cup_{x \in SX} \mathbf{VI}(x, A[\mathbf{h}(x)]))$[3]. It is assumed that, for each assignment A of the horizon variables in SH, the scope $\mathbf{S}(c)$ of each CSP constraint c in $f(A)$ is included in $SV \cup SD(A)$.

We will use $\mathbf{SV}(c)$, $\mathbf{ST}(c)$, $\mathbf{SX}(c)$, and $\mathbf{SH}(c)$ to denote the scope of a dynamic constraint c in terms of, respectively, static variables, time references, timelines, and horizon variables, and $\mathbf{f}(c)$ to denote its associated function.

To illustrate this definition, let us come back to our toy planning problem. To specify the initial state, we need the following three constraints c_2, c_3, and c_4:

$$t_1 = Ti \tag{2}$$

$$l_1 = Li \tag{3}$$

$$e_1 = Ei[r] \tag{4}$$

The first one is a dynamic constraint defined by the tuple $c_2 = \langle \emptyset, \{t\}, \emptyset, f_2 \rangle$ with f_2 the function which associates with each assignment of h the unique unary CSP constraint $t_1 = Ti$ on dynamic temporal variable t_1 (in this particular case, function f_2 is a constant).

The second one is defined by the triple $c_3 = \langle \emptyset, \emptyset, \{l\}, f_3 \rangle$ with f_3 the function which associates with each assignment of h the unique unary CSP constraint $l_1 = Li$ on dynamic atemporal variable l_1.

The third one is defined by the triple $c_4 = \langle \{r\}, \emptyset, \{e\}, f_4 \rangle$ with f_4 the function which associates with each assignment of h the unique binary CSP constraint $e_1 = Ei[r]$ between static variable r and dynamic atemporal variable e_1.

Then, to specify state transitions, we need the following three constraints c_5, c_6, and c_7:

$$\forall i \in [2..h] : Mo[r, l_{i-1}, l_i] = 1 \tag{5}$$

$$\forall i \in [2..h] : t_i = t_{i-1} + Du[r, l_{i-1}, l_i] \tag{6}$$

$$\forall i \in [2..h] : e_i = e_{i-1} - Co[r, l_{i-1}, l_i] \tag{7}$$

For example, the first one is a dynamic constraint defined by the tuple $c_5 = \langle \{r\}, \emptyset, \{l\}, f_5 \rangle$ with f_5 the function which associates with each assignment a of h the set $\{(Mo[r, l_{i-1}, l_i] = 1) \mid i \in [2..a]\}$ of $(a-1)$ ternary CSP constraints, each one connecting static variable r and dynamic atemporal variables l_{i-1} and l_i.

The second one is defined in a similar way by a triple $c_6 = \langle \{r\}, \{t\}, \{l\}, f_6 \rangle$ and the third one by a triple $c_7 = \langle \{r\}, \emptyset, \{e, l\}, f_7 \rangle$.

To specify the goal state, we need the following constraint c_8:

$$l_h = Lg \tag{8}$$

This constraint is defined by the tuple $c_8 = \langle \emptyset, \emptyset, \{l\}, f_8 \rangle$ with f_8 the function which associates with each assignment a of h the unique unary CSP constraint $l_a = L_g$.

The deadline Tg and the minimum level of energy Eg are enforced via the domains of time reference t and timeline e.

Finally, if we want to enforce that the robot does not go several times through the same location, because this would be counterproductive, we can use the following constraint c_9:

$$AllDifferent(\{l_i \mid i \in [1..h]\}) \tag{9}$$

[2] In general, ST is not included in the set of time references associated with the timelines in SX.

[3] If A is an assignment and v a variable assigned by A, $A[v]$ denotes the projection of A on v. By extension, if A is an assignment and V a set of variables assigned by A, $A[V]$ denotes the projection of A on V.

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

This constraint is defined by the tuple $c_9 = \langle \emptyset, \emptyset, \{l\}, f_9 \rangle$ with f_9 the function which associates with each assignment a of h the unique global CSP constraint $AllDifferent(\{l_i \mid i \in [1..a]\})$.

3.9 Constraint networks on timelines

All these definitions can be put together in order to define constraint networks on timelines.

DEFINITION 10. *A constraint network N on timelines (CNT) is a tuple* $\langle V, T, X, CV, CX \rangle$ *where:*

- V is a finite set of static variables;
- T is a finite set of time references; it is assumed that $\forall t \in T : \mathbf{h}(t) \in V$: the horizons of all the time references belong to V; they are considered as static variables;
- X is a finite set of timelines; it is assumed that $\forall x \in X : \mathbf{t}(x) \in T$;
- CV is a finite set of static constraints; it is assumed that $\forall c \in CV : \mathbf{S}(c) \subseteq V$;
- CX is a finite set of dynamic constraints; it is assumed that $\forall c \in CV : (\mathbf{SV}(c) \subseteq V) \wedge (\mathbf{ST}(c) \subseteq T) \wedge (\mathbf{SX}(c) \subseteq X)$.

We will use $\mathbf{V}(N)$, $\mathbf{T}(N)$, $\mathbf{X}(N)$, $\mathbf{CV}(N)$, *and* $\mathbf{CX}(N)$ *to denote, respectively, the static variables, the time references, the timelines, the static constraints, and the dynamic constraints associated with a constraint network N on timelines.*

It is assumed that a default dynamic constraint c_t *is associated with each time reference* $t \in T$. *This constraint enforces that the temporal variables associated with a time reference are totally ordered. We have* $c_t = \langle \emptyset, \{t\}, \emptyset, f_t \rangle$ *with* f_t *the function which associates with each assignment a of* $\mathbf{h}(t)$ the set $\{(t_{i-1} \leqslant t_i) | i \in [2..a]\}$ *of* $(a-1)$ *binary CSP constraints.*

It is moreover assumed that a default dynamic constraint c_x *is associated with each timeline* $x \in X$. *This constraint enforces that if two temporal positions in a timeline are equal, the associated timeline values are equal too (a timeline can have only one value at a given time). We have* $c_x = \langle \emptyset, \{\mathbf{t}(x)\}, \{x\}, f_x \rangle$ *with* f_x *the function which associates with each assignment a of* $\boldsymbol{h}(x)$ *the set* $\{((\mathbf{t}(x)_{i-1} = \mathbf{t}(x)_i) \rightarrow (x_{i-1} = x_i)) \mid i \in [2..a]\}$ *of* $(a-1)$ *CSP constraints.*

The CNT which results from the modeling of our toy planning problem is defined by the tuple $\langle \{r, h\}, \{t\}, \{l, e\}, \{c_1\}\ \{c_i \mid i \in [2..9]\} \rangle$.

3.10 Assignments

An assignment of a CNT is an assignment of all its static variables (including all the horizon variables) and of all the induced dynamic variables.

DEFINITION 11. *An assignment A of a constraint network N on timelines is an assignment of all the static variables in* $\mathbf{V}(N)$ *and of all the dynamic ones in* $(\cup_{t \in \mathbf{T}(N)} \mathbf{VI}(t, A[\mathbf{h}(t)])) \cup (\cup_{x \in \mathbf{X}(N)} \mathbf{VI}(x, A[\mathbf{h}(x)]))$.

For example, $\langle r = 1, h = 2, t_1 = t_2 = 0, l_1 = l_2 = A, e_1 = e_2 = 10 \rangle$ is a possible assignment of the instance of the toy planning problem we consider.

3.11 Constraint satisfaction

The classical definition of the satisfaction of a constraint in the CSP framework (Rossi *et al.*, 2006) is the following one:

DEFINITION 12. *A CSP constraint c is satisfied by an assignment A if and only if:*

- all the variables in $\mathbf{S}(c)$ are assigned by A;
- the projection $A[\mathbf{S}(c)]$ of A on the variables in $\mathbf{S}(c)$ is allowed by $\mathbf{R}(c)$: $A[\mathbf{S}(c)] \in \mathbf{R}(c)$.

This allows the satisfaction of any static constraint to be defined. For example, constraint c_1 is satisfied by the assignment $\langle r = 2, h = 3 \rangle$, but not by the assignment $\langle r = 2, h = 4 \rangle$.

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

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

Figure 4 Tabular representation of the only solution, with $r = 1$ (first robot chosen)

For dynamic constraints, we adopt the following definition:

DEFINITION 13. *A dynamic constraint c is satisfied by an assignment A if and only if:*

- all the horizon variables in **SH**(c) are assigned by A;
- all the CSP constraints in **f**(c) (A[**SH**(c)]) are satisfied.

For example, constraint c_5 is satisfied by the assignment $\langle r = 2, h = 3, l_1 = A, l_2 = B, l_3 = D \rangle$, but not by the assignment $\langle r = 2, h = 3, l_1 = A, l_2 = B, l_3 = C \rangle$ (no direct move possible from B to C for robot 2).

3.12 Solution of a constraint network on timelines

DEFINITION 14. *A solution S of a constraint network N on timelines is an assignment of N such that all the constraints in* **CV**(N) *and* **CX**(N) *are satisfied.*

For example, $\langle r = 1, h = 4, t_1 = 0, t_2 = 5, t_3 = 13, t_4 = 18, l_1 = A, l_2 = B, l_3 = C, l_4 = D, e_1 = 10, e_2 = 8, e_3 = 4, e_4 = 2 \rangle$ is a solution of the instance of the toy planning problem we consider and, in fact, the only solution of this instance. See Figure 4 for a tabular representation of this solution.

3.13 Consistency of a constraint network on timelines

DEFINITION 15. *A constraint network N on timelines is consistent if and only if a solution exists.*

Since it admits a solution, the instance of the toy planning problem we consider is consistent. However, if *Ei*[1] would, for example, be equal to 9 instead of 10, it would be inconsistent.

4 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.nal.res.in/mav08/`). Its data are as follows:

- a number NA of *areas* to be visited, each containing a *target*; areas are assumed to be all different; they are numbered from 1 to NA;
- for each area $a \in [1..NA]$, its *type* TY_a, with three possible values EI, SC, and DR:
  - if $TY_a = EI$, a contains a target to be *identified* by using a camera and performing an *eight* over a;
  - if $TY_a = SC$, a contains a target to be *localized* by using a camera too and performing a *scan* over a;
  - if $TY_a = DR$, a contains a target to be *touched* by using one marble and performing a *drop*;
- a unique *takeoff* and *landing area HO* for all the vehicles; number 0 is associated with this area;
- a number NV of *vehicles*; vehicles are numbered from 1 to NV;

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

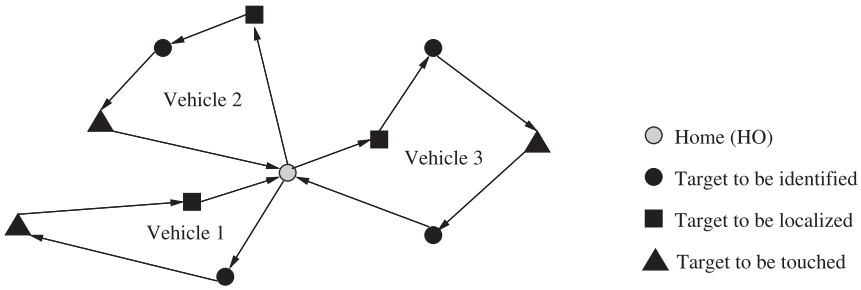


Figure 5 Graphical representation of a solution of the mission management problem for a team of unmanned air vehicles

- for each vehicle v$\in$[1..*NV*], a boolean CA_v equal to 1 if and only if *v* embeds a *camera*, the number NM_v of *marbles* initially available in *v*, and the *speed* SP_v of *v*;
- for each vehicle $v\in[1..NV]$ and each action *ac* among actions *TO*, *LA*, *EI*, *SC*, and *DR* (takeoff, landing, eight, scan, and drop), the expected *duration* $DU_{v,ac}$ of *a* when performed by *v*;
- for each pair $a, a'\in[0..NA]$ of areas, including *HO*, the *distance* $DI_{a,a'}$ from *a* to a';
- a *minimum duration DUmin* between two takeoffs or landings of different vehicles at *HO*.
- a maximum *mission duration MD*;

The problem is to visit and handle all the areas using the team of vehicles. See Figure 5 for a graphical representation of a solution of an instance involving three vehicles, four targets to be identified, three to be localized, and three to be touched.

To show the modeling power and the versatility of the CNT framework, we propose three models of this problem:

1. in Section 4.1, an *action-based* model, inspired from SAT or CSP encodings of planning problems (Kautz & Selman, 1992), which describes the preconditions and the effects of the possible actions and uses only dynamic variables and constraints, except the static horizon variables;
2. in Section 4.2, a *mixed* model, which modifies the previous one by introducing some static variables and constraints;
3. in Section 4.3, a constraint-*based* model, inspired from CSP encodings of resource constrained scheduling problems (Baptiste *et al.*, 2001), which uses only static variables and constraints.

4.1 Action-based model

Let $AC = \{TO, LA, EI, SC, DR, GO, NO\}$ (takeoff, landing, eight, scan, drop, goto, and nothing) be the set of possible actions. The *NO* action is useful because all the vehicles but one must wait before taking off because of the minimum duration between takeoffs and landings.

We have *NV* vehicles that will evolve in parallel, with no synchronization between the starting times of the actions performed by each vehicle. As a consequence, *NV* sets of timelines can be used to model this problem, each set associated with one vehicle and having its own time reference. With each vehicle $v\in[1..NV]$, we associate a time reference ti_v and four timelines, which share the same time reference ti_v: a timeline fl_v to represent the fact that *v* is *flying* or not, a timeline at_v to represent its current *position* (area), a timeline nm_v to represent the current number of available *marbles*, and a timeline ac_v to represent the current *action*.

Since, for each vehicle *v*, there is at least one step associated with the initial state and at most $NSmax = 2 \cdot NA + 3$ steps in the case in which all the areas are handled by the same vehicle, the domain of the horizon of each time reference ti_v is [1..*NSmax*]. Since *MD* is the maximum duration, the domain of each time reference ti_v is [0..*MD*]. For each vehicle *v*, the domains of the

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

timelines fl_v, at_v, nm_v, and ac_v are, respectively, $\{0, 1\}$, $[0..NA]$, $[0..NM_v]$, and AC. These choices result in the following constraints:

Initial state of each vehicle:

$$\forall v \in [1..NV]: \quad fl_{v,1} = 0 \tag{10}$$
$$at_{v,1} = 0 \tag{11}$$
$$nm_{v,1} = NM_v \tag{12}$$
$$ac_{v,1} \in \{TO, NO\} \tag{13}$$
$$ti_{v,1} = 0 \tag{14}$$

Final state of each vehicle:

$$\forall v \in [1..NV]: \quad fl_{v,\mathbf{h}(ti_v)} = 0 \tag{15}$$
$$at_{v,\mathbf{h}(ti_v)} = 0 \tag{16}$$
$$ac_{v,\mathbf{h}(ti_v)} = NO \tag{17}$$

Conditions and effects of a takeoff:

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1]:$$
$$(ac_{v,i} = TO) \rightarrow ((at_{v,i} = 0) \wedge (fl_{v,i} = 0) \wedge (fl_{v,i+1} = 1) \wedge (ti_{v,i+1} - ti_{v,i} = DU_{v,TO})) \tag{18}$$

Conditions and effects of a landing:

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1]:$$
$$(ac_{v,i} = LA) \rightarrow ((at_{v,i} = 0) \wedge (fl_{v,i} = 1) \wedge (fl_{v,i+1} = 0) \wedge (ti_{v,i+1} - ti_{v,i} = DU_{v,LA})) \tag{19}$$

Conditions and effects of an eight:

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1]:$$
$$(ac_{v,i} = EI) \rightarrow ((TY_{at_{v,i}} = EI) \wedge (fl_{v,i} = 1) \wedge (CA_v = 1) \wedge (ti_{v,i+1} - ti_{v,i} = DU_{v,EI})) \tag{20}$$

Conditions and effects of a scan:

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1]:$$
$$(ac_{v,i} = SC) \rightarrow ((TY_{at_{v,i}} = SC) \wedge (fl_{v,i} = 1) \wedge (CA_v = 1) \wedge (ti_{v,i+1} - ti_{v,i} = DU_{v,SC})) \tag{21}$$

Conditions and effects of a drop:

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1]:$$
$$(ac_{v,i} = DR) \rightarrow$$
$$((TY_{at_{v,i}} = DR) \wedge (fl_{v,i} = 1) \wedge (nm_{v,i} - nm_{v,i+1} = 1) \wedge (ti_{v,i+1} - ti_{v,i} = DU_{v,DR})) \tag{22}$$

Conditions and effects of a move:

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1]:$$
$$(ac_{v,i} = GO) \rightarrow ((at_{v,i+1} \neq at_{v,i}) \wedge (fl_{v,i} = 1) \wedge (ti_{v,i+1} - ti_{v,i} = DI_{at_{v,i}, at_{v,i+1}}/SP_v)) \tag{23}$$

Conditions of a null action:

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)]: (ac_{v,i} = NO) \rightarrow ((at_{v,i} = 0) \wedge (fl_{v,i} = 0)) \tag{24}$$

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

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1]: \quad (fl_{v,i+1} \neq fl_{v,i}) \rightarrow (ac_{v,i} \in \{TO, LA\}) \tag{25}$$
$$(at_{v,i+1} \neq at_{v,i}) \rightarrow (ac_{v,i} = GO) \tag{26}$$
$$(nm_{v,i+1} \neq nm_{v,i}) \rightarrow (ac_{v,i} = DR) \tag{27}$$

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

Each area must be handled once and only once by the team of vehicles[4]:

$$\forall a \in [1..NA] : \sum_{v=1}^{NV} \sum_{i=1}^{\mathbf{h}(ti_v)} ((at_{v,i} = a) \wedge (ac_{v,i} = TY_a)) = 1 \tag{28}$$

Minimum duration between takeoffs and landings:

$$\begin{aligned}&\forall v, v' \in [1..NV] | v < v', \forall i \in [1..\mathbf{h}(ti_v)-1], \forall i' \in [1..\mathbf{h}(ti_{v'})-1] : \\ &\quad ((ac_{v,i} \in \{TO, LA\}) \wedge (ac_{v',i'} \in \{TO, LA\})) \rightarrow \\ &\quad ((ti_{v',i'} \geqslant ti_{v,i+1} + DUmin) \vee (ti_{v,i} \geqslant ti_{v',i'+1} + DUmin))\end{aligned} \tag{29}$$

For each vehicle, an action cannot be followed by the same action:

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1] : ac_{v,i+1} \neq ac_{v,i} \tag{30}$$

No landing can be followed by a takeoff:

$$\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1] : (ac_{v,i} = LA) \rightarrow (ac_{v,i+1} \neq TO) \tag{31}$$

For each vehicle, null actions occur only at the beginning or at the end of the sequence of actions:

$$\forall v \in [1..NV], \forall i \in [2..\mathbf{h}(ti_v)-1] : ac_{v,i} \neq NO \tag{32}$$

Such a modeling is driven by two principles used by SAT or CSP encodings of planning problems (Kautz & Selman, 1992):

- the presence of an action implies its preconditions and its effects, and effects are effective changes in timeline values (see constraints c_{18} to c_{24});
- any effective change in a timeline value is justified by an action (see constraints c_{25} to c_{27}).

Constraints c_{28} are global constraints on the actions performed by the team of vehicles. Constraints c_{30} to c_{32} constrain the sequence of actions performed by each vehicle and rule out inconsistent or sub-optimal solutions, such as performing a landing followed by a takeoff.

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

4.2 Mixed model

We can modify the previous model by introducing static variables such as, for each area, the vehicle in charge of handling it. For each area $a \in [1..NA]$, let v_a be a static variable of domain $[1..NV]$, which represents the *vehicle* in charge of a.

The introduction of such variables modifies constraints c_{20}, c_{21}, and c_{22}. For example, constraint c_{20}, which specifies the conditions and effects of an eight, becomes:

$$\begin{aligned}&\forall v \in [1..NV], \forall i \in [1..\mathbf{h}(ti_v)-1] : \\ &\quad (ac_{v,i} = EI) \rightarrow \\ &\quad ((TY_{at_{v,i}} = EI) \wedge (v_{at_{v,i}} = v) \wedge (fl_{v,i} = 1) \wedge (CA_v = 1) \wedge (ti_{v,i+1} - ti_{v,i} = DU_{v,EI}))\end{aligned} \tag{33}$$

More importantly, it simplifies constraint c_{28}, because it suffices now to specify that each area must be handled once and only once by the vehicle in charge of it:

$$\forall a \in [1..NA] : \sum_{i=1}^{\mathbf{h}(ti_{v_a})} ((at_{v_a,i} = a) \wedge (ac_{v_a,i} = TY_a)) = 1 \tag{34}$$

[4] As is usual in CP tools, a constraint c is considered equivalent to a boolean variable b equal to 1 if c is satisfied and 0 otherwise (constraint reification).

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

It would be possible to go further by introducing other static variables, such as, for each vehicle v and each position p in the sequence of the areas visited by v, the area visited by v at position p in the sequence. We do not detail the associated model and go directly to a pure constraint-based model.

4.3 Constraint-based model

It is indeed possible to build a pure constraint-based model, using only static variables and constraints. Let us consider the following static variables:

- for each area $a \in [1..NA]$, a variable V_a of domain $[1..NV]$, which represents the vehicle in charge of a;
- for each vehicle $v \in [1..NV]$, a variable n_v of domain $[0..NA]$, which represents the number of areas handled by v;
- for each vehicle $v \in [1..NV]$ and each position $p \in [1..NA]$, a variable $a_{v,p}$ of domain $[0..NA]$, which represents the area visited by v at position p in the sequence of the areas visited by v; by default, $a_{v,p} = 0$ when no area is visited by v at position p;
- for each vehicle $v \in [1..NV]$, a variable s_v of domain $[0..MD]$, which represents the starting time of the takeoff of v;
- finally, for each vehicle $v \in [1..NV]$, a variable e_v of domain $[0..MD]$, which represents the ending time of the landing of v.

Note that some variables, such as n_v or e_v, are redundant, but useful to express some constraints in a more concise way. This choice of variables induces the following constraints:

Constraints between variables v_a and variables n_v:

$$\forall v \in [1..NV] : n_v = \sum_{a=1}^{NA} (v_a = v) \tag{35}$$

Constraints on variables $a_{v,p}$:

$$\forall v \in [1..NV], \forall a \in [1..NA] : \sum_{p=1}^{NA} (a_{v,p} = a) \leqslant 1 \tag{36}$$

Constraints between variables v_a and variables $a_{v,p}$:

$$\forall v \in [1..NV], \forall a \in [1..NA] : (v_a = v) \leftrightarrow \left(\sum_{p=1}^{NA} (a_{v,p} = a) = 1 \right) \tag{37}$$

Constraints between variables n_v and variables $a_{v,p}$:

$$\forall v \in [1..NV], \forall p \in [1..NA] : (a_{v,p} = 0) \leftrightarrow (p > n_v) \tag{38}$$

A camera is required for identification and localization:

$$\forall a \in [1..NA] : (TY_a \in \{EI, SC\}) \rightarrow (CA_{v_a} = 1) \tag{39}$$

One marble is required for each touching:

$$\forall v \in [1..NV] : \sum_{a=1}^{NA} ((v_a = v) \wedge (TY_a = DR)) \leqslant NM_v \tag{40}$$

Duration of the mission of each vehicle:

$$\forall v \in [1..NV] : (n_v = 0) \rightarrow (s_v = e_v = 0)$$

$$(n_v > 0) \rightarrow (e_v - s_v = DU_{v,TO} + DU_{v,LA} + \sum_{a=1}^{NA} (v_a = v) \cdot DU_{v,TY_a} +$$

$$(DI_{0,a_{v,1}} + DI_{a_{v,NA},0} + \sum_{p=1}^{NA-1} DI_{a_{v,p},a_{v,p+1}})/SP_v) \tag{41}$$

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

Minimum duration between takeoffs and landings. Let $tlints = \cup_{v \in [1..NV] \mid n_v > 0}\{[s_v..(s_v + DU_{v,TO} + DUmin)], [(e_v - DU_{v,LA})..(e_v + DUmin)]\}$ be the set of the takeoff and landing temporal intervals enlarged by $DUmin$. For any set $ints$ of temporal intervals, let $NoOverlap(ints)$ be the global scheduling constraint which enforces that there is no overlap between temporal intervals in $ints$:

$$NoOverlap(tlints) \tag{42}$$

4.4 Discussion

These three models of the same problem show the versatility of the CNT framework which can support various modeling styles. People from the Planning community would certainly prefer the first action-based model, whereas other people from the Constraint and Integer Linear Programming (CP and ILP) communities would naturally prefer the third constraint-based one.

The three models can be implemented using classical CP tools, taking advantage, for the first two models, of the bounded number of steps and introducing null actions when steps are not used. For the moment, we only implemented the first action-based model using *OPL* (`http://www.ilog.com/products/oplstudio/`) and did not perform intensive experiments using the three models on instances of various sizes.

5 Modeling the mission management problem for an Earth observing satellite

The problem we consider now is a very simplified version of the problem of management of the observations performed by an autonomous Earth observing satellite. For example, data memorization and downloading constraints are omitted. A description of the whole problem, a timeline-based model, and approximate solving algorithms are presented in Pralet and Verfaillie (2008a). The data of the simplified version are as follows:

- a *planning horizon* defined by its *starting* and *ending times* STA and END;
- *satellite features* such as the minimum and maximum levels of *energy* $ENmin$ and $ENmax$, the *power* $Psun$ delivered by the solar panels during day periods, the *powers* $Psat$ and Pob consumed, respectively, by the platform and by any observation action, and the *rotation speed* MS of the sight mirror;
- an *initial state* defined by the initial *energy* level EN_0 and the initial *orientation* OR_0 of the sight mirror;
- a set OB of *observations* to be performed with, for each observation $o \in OB$, the observation duration DO_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;
- 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.

Two observations of different ground areas cannot be performed in parallel. Moreover, sufficient time is necessary between the end of an observation and the beginning of the following one to allow change in the sight mirror orientation to be performed.

The problem is to perform an optimal subset of OB over the planning horizon. The definition of the optimization criterion is omitted in this article. See Figure 6 for a temporal representation of a solution involving nine observation windows and two eclipse windows.

Although we proposed three different models of the mission management problem for a team of unmanned air vehicles in Section 4, we propose only one mixed model of this problem which we consider to be the most appropriate. This CNT model combines static variables to represent observation decisions with dynamic variables to represent the way state attributes such as energy evolve with time. It therefore combines static and dynamic constraints.

To represent observation decisions, we can first use the following static variables: for each observation $o \in OB$, the chosen *observation window* $no_o \in [0..NO_o]$ (value 0 represents the fact

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

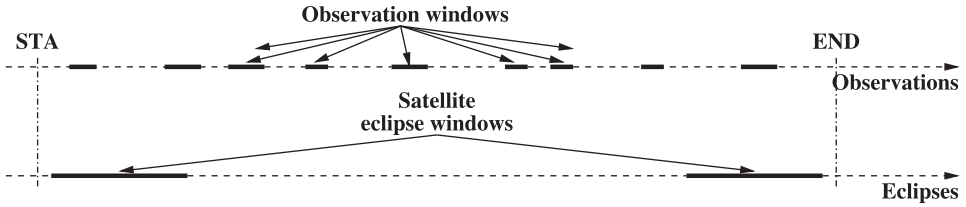


Figure 6 Temporal representation of a solution of the mission management problem for an Earth observing satellite

that o is not performed) and the associated observation *starting* and *ending times* so_o and $eo_o \in [STA..END]$.

Then, to represent the evolution of the state of the satellite, we can use a set of fully synchronized timelines. In fact, we consider one time reference *ti* and three timelines, which share the same time reference *ti*: two timelines *ob* and *ec*, which respectively, represent the fact that the satellite is *observing* or not and in an *eclipse* period or not, and one timeline *en* which represents the current level of available *energy*.

Since there are at least two steps associated with times STA and END and at most $NSmax = 2 \cdot (NO + NE) + 2$ steps in the case where all the observations are performed and all the window starting and ending times are different, the domain of time reference *ti* is $[STA..END]$ and the domain of its horizon is $[2..NSmax]$. The domains of timelines *ob*, *ec*, and *en* are, respectively, $\{0, 1\}$, $\{0, 1\}$, and $[ENmin..ENmax]$. These choices result in 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 the starting and ending times are arbitrarily set to STA:

$$\forall o \in OB: \quad (no_o \neq 0) \rightarrow (so_o = SO_{o,no_o}) \tag{43}$$

$$(no_o \neq 0) \rightarrow (eo_o = so_o + DO_o) \tag{44}$$

$$(no_o = 0) \rightarrow (so_o = eo_o = STA) \tag{45}$$

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:

$$\forall o, o' \in OB, \forall k \in [1..NO_o], \forall k' \in [1..NO_{o'}] \mid$$
$$((o \neq o') \wedge (SO_{o,k} \leqslant SO_{o',k'}) \wedge (SO_{o',k'} < SO_{o,k} + DO_o + \frac{|OR_{o,k} - OR_{o',k'}|}{MS})):$$
$$(no_o = k) \rightarrow (no_{o'} \neq k') \tag{46}$$

$$\forall o \in OB, \forall k \in [1..NO_o] \mid SO_{o,k} < STA + \frac{|OR_{o,k} - OR_0|}{MS} : no_o \neq k \tag{47}$$

Initial state:

$$en_1 = EN_0 \tag{48}$$

$$ti_1 = STA \tag{49}$$

Observing and eclipse states:

$$\forall i \in [1..\mathbf{h}(ti)]: \quad (ob_i = 1) \leftrightarrow \vee_{o \in OB} (so_o \leqslant ti_i < eo_o) \tag{50}$$

$$(ec_i = 1) \leftrightarrow \vee_{k=1}^{NE} (SE_k \leqslant ti_i < SE_k + DE_k) \tag{51}$$

Energy evolution: it is assumed to be piecewise linear; the available energy is therefore 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 and by observation if

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

the satellite was observing, with a maximum of *ENmax*; note that the minimum level of energy *ENmin* is enforced via the domain of timeline *en*:

$$\forall i \in [1..\mathbf{h}(ti)-1]:$$
$$en_{i+1} = min(ENmax, en_i + (ti_{i+1}-ti_i)\cdot((1-ec_i)\cdot Psun - Psat - ob_i \cdot P_{ob})) \quad (52)$$

Temporal positions of the successive steps: let $ots = \cup_{o \in OB | so_o \neq eo_o}\{so_o, eo_o\}$ and $ets = \cup_{k \in [1..NE]}\{SE_k, SE_k + DE_k\}$ be, respectively, the sets of effective observation and eclipse starting and ending times; let $ts = ots \cup ets \cup \{END\}$; the temporal position of the next step is defined as follows:

$$\forall i \in [1..\mathbf{h}(ti) - 1] \; : \; ti_{i+1} = max\{t \in ts \mid t > ti_i\} \quad (53)$$

Final state:

$$ti_{\mathbf{h}(ti)} = END \quad (54)$$

Constraints c_{43} to c_{47} are static, whereas the following ones are dynamic.

Taking advantage of the bounded number of steps, this model has been first implemented using the CP tool *Comet* (`http://www.comet-online.org/`). Then, we designed a more efficient dedicated algorithm, implemented in *Java* and intended to be embedded in the on-board control software of a satellite.

6 How to use the Constraint Network on Timelines framework

The objective of the CNT framework is to offer a modeler all the basic elements that enable him/her to model any discrete event dynamic system and the properties he/she knows or wants to verify or to enforce on it. However, since these elements are very basic, it may be difficult to select the right ones and to organize them in order to model properly a complex system. In this section, we provide a modeler with some guidelines that may help him/her in this task.

First, since the CNT framework is an extension of the CSP framework, it is possible to reuse all the modeling experience accumulated in the CSP framework and in related frameworks such as SAT or ILP, by looking at models of scheduling and planning problems in these frameworks.

Second, experience in modeling using variable and constraint-based frameworks such as CSP, SAT, or ILP shows that the crucial choice is the choice of the variables. Since the CNT framework distinguishes static variables from dynamic ones, it is crucial to decide on what the static and dynamic variables will be. Although the three models of the same problem in Section 4 show that several options are often candidate, some sensible rules can be followed. Static variables are useful to represent choices that must be made once and only once such as, for example, in the illustrative problem of Section 2, the robot that is chosen for the mission or, in the problem of Section 4, for each area *a*, the vehicle in charge of *a*. Time references are naturally useful to represent the temporal positions of the successive steps. As for the timelines, they are useful to represent the choices that must be made at each step such as, for example, in the illustrative problem of Section 2, the location that is visited at each step. They are also useful to represent the consequences at each step of the choices made for static or dynamic variables such as, for example, in the illustrative problem of Section 2, the level of energy at each step. In fact, it is sensible to associate a timeline with each dynamic attribute x of the system, that is, with each attribute x whose value evolves step after step, whatever x represents: a component of the state of the system or its environment, or an event from the system, its environment, or its controller. Then, the definition of constraints naturally follows the choice of variables. If constraints cannot be easily expressed with the chosen variables, the choice of variables must be reconsidered.

Third, timelines may be either synchronized by sharing the same time reference, or non-synchronized by using several time references. Synchronized timelines are useful when evolutions are fully synchronized, as in the illustrative problem of Section 2, or partially but strongly

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

synchronized as in the problem of Section 5, where observations and eclipses together impact the level of energy. In contrast, non-synchronized timelines are useful when evolutions are fully non-synchronized or partially but weakly synchronized as in the problem of Section 4, where vehicles must synchronize their actions only when taking off and landing. Using several time references implies defining temporal constraints between time references in order to partially synchronize them, as Constraint 29 does in the problem of Section 4.

7 Related work

Contrary to classical frameworks used in the planning and scheduling community, such as STRIPS, RCPSP, or PDDL (Fox & Long, 2003), which are all built around the notions of *action*, *precondition*, *effect*, *duration*, and resource *consumption*, the CNT framework is a more basic modeling framework, built around the notions of *time reference*, *timeline*, and *constraint*. As shown in this article, this is one of its strengths because this allows complex dynamic phenomena such as concurrent interdependent evolutions or complex requirements such as constraints on sequences of actions or states to be precisely modeled. From the knowledge engineering point of view, this may be one of its weaknesses because building and checking models may be more difficult when using lower-level constructs. However, nothing prevents us from building on top of the basic CNT framework generic higher-level constructs that would be closer to the modeler point of view. This would enable the modeler to choose freely, according to his own habits and needs, between low and high level modeling constructs. However, the identification of the right constructs and their implementation remains a work to be done, for which experience in planning, scheduling, and CP will be useful.

With regard to the use of *constraints* in planning and scheduling, the CSP framework and associated CP tools have been used for many years to model and solve scheduling problems (Baptiste *et al.*, 2001). Things are more recent and a bit more problematic with planning problems. The main obstacle to a CSP modeling of planning problems is clearly the unknown, possibly unbounded, number of steps, and thus of variables and constraints. Such an obstacle is absent from scheduling problems, where the number of tasks and resources to manage is known. Nareyek *et al.* (2005) propose a survey of various attempts to overcome this obstacle and to combine Artificial Intelligence planning and CP. The CNT framework follows the last approach among the three they distinguish:

- in the first approach used by planning tools such as IxTeT (Ghallab & Laruelle, 1994) or RAX-PS (Jónsson *et al.*, 2000), CP is used as a kind of subroutine to efficiently solve subproblems, such as time or resource problems, generated by the main planning procedure;
- in the second approach initialized by Kautz and Selman (1992)in the SAT framework and extended by van Beek and Chen (1999)and Do and Kambhampati (2001) to the CSP framework, a CSP is built to solve planning problems over a fixed horizon of k steps, and k is incremented when no plan is found;
- in the last approach, a planning problem is tackled as a kind of dynamic CSP with a horizon that is not fixed; such an approach is used by the Visopt ShopFloor system (Barták, 2002), by the EUROPA planner, built on top of the CAIP framework (Constraint-based Attribute and Interval Planning (Frank & Jónsson, 2003)), and by the CPT planner (Vidal & Geffner, 2006).

With regard to the use of *timelines* and constraints on timelines to model planning and scheduling problems, this idea is present in many works at the frontier between planning and scheduling (Ghallab & Laruelle, 1994; Muscettola, 1994; Cesta & Oddi, 1996; Barták, 2002; Frank & Jónsson, 2003; Fratini *et al.*, 2008).

However, in Ghallab and Laruelle (1994), Muscettola (1994), and Cesta and Oddi (1996), constraints are mainly used to manage temporal and resource constraints, whereas a classical planning procedure is used to manage causal links between actions. In contrast, in the CNT framework, all the decisions related to actions, time, and resources are managed the same way via constraints and via the same procedure, which can combine constraint propagation and search.

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

The CAIP framework (Frank & Jónsson, 2003) is probably the closest one to the CNT framework. Attributes in the CAIP framework are equivalent to timelines in the CNT framework. However, in the CAIP framework, relations between attributes take the form of *intervals* on which a given predicate holds, linking a given set of attributes. Constraints between intervals take the form of *configuration rules*, which specify that the presence of a given interval implies the presence of at least one set of intervals among a given set of sets of intervals. In contrast, the CNT framework uses more basic notions of step and of temporal position and value associated with any step. This allows any kind of constraint to be defined between any kind of variable: dynamic temporal and atemporal variables and static variables, including horizon variables (see, for example, Constraint 8 (resp. 54) for a constraint which links a horizon variable with atemporal (resp. temporal) variables). However, if we consider that the notion of interval is important, nothing prevents us from defining it by using the basic elements of the CNT framework, following the same philosophy that led to the introduction in CP Optimizer (`http://www.ilog.com/products/cpoptimizer/`) of *interval* variables that can be used in any model together with classical variables.

The modeling principles used by the Visopt ShopFloor system (Barták, 2002) are close to the ones of the CAIP framework: use of intervals, called *slots*. Each slot is filled by an activity and a sequence of slots is associated with each resource. Constraints link temporal positions and activities of slots. Constraint Logic programming (CLP) allows slots and constraints on them to be dynamically introduced. As previously said, the CNT framework is more basic and thus more flexible. However, the notions of resource, activity, and slot could be built on top of it, if useful.

Last, (Fratini *et al.*, 2008) propose a generic software architecture, called OMPS, which allows timeline-based models of *component* and inter-component behaviors to be defined and managed. Such an architecture could be used on top of the CNT framework.

Although many works inspired by First-order Logics, such as PDDL or CAIP, adopt a predicate-based modeling style, the CNT framework, inspired by practice in Automatic Control (see, for example, automata and synchronous languages (Benveniste *et al.*, 2003)), adopts a more natural attribute-based modeling style, as the recent ANML language (Smith *et al.*, 2008) does.

To sum up, we think that the key advantage of the CNT framework is to be a pure CSP framework from which it inherits a clear semantics and all the associated solving algorithms (constraint propagation, backtrack search, local search …), which only need to be extended to the new framework.

In this article, we focused on planning and scheduling problems, but it must be stressed that the modeling capabilities of the CNT framework can be used in other contexts such as failure diagnosis, situation recognition, or validation. For example, the problem of building a consistent activity plan and the problem of identifying a consistent failure scenario can be modeled the same way as a CNT consistency problem. Moreover, the problem of proving that a given controller will never lead a system to an undesirable state or to a constraint violation can be modeled as a CNT inconsistency problem.

8 Research directions

Extending the CNT framework to optimization is straightforward, either by introducing optimization variables that are constrained by other problem variables, as is usually done in CP, or by allowing local soft constraints that are local cost functions as done in the WCSP framework (Weighted CSP (Schiex *et al.*, 1995)).

Our current work focuses on algorithmic issues. We have already designed and implemented generic CNT solving algorithms, using a combination of tree search and lazy introduction of variables and constraints, on top of the CP *Choco* solver (`http://choco-solver.net/`). These algorithms work assuming finite domains for horizon variables, time references, and timelines. They show that it is possible and may be beneficial not to systematically assign horizon variables first. They have been successfully experimented on planning problems coming from the IPC (Pralet & Verfaillie, 2008b). Further work will focus on generic CNT solving algorithms, using greedy or local search.

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press

Another important direction of research is the extension of the CNT framework to uncertainty (Puterman, 1994) via a necessary distinction between controllable and uncontrollable timelines and via the introduction of local feasibility, plausibility, and utility functions, as was done, for example, in the PFU framework (Pralet *et al*., 2007) in a static context.

References

Baptiste, P., Pape, C.L. & Nuijten, W. 2001. *Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers.

Barták, R. 2002. Visopt ShopFloor: on the edge of planning and scheduling. *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, Lecture Notes in Computer Science **2470**, 587–602.

Benveniste, A., Caspi, P., Edwards, P., Halbwachs, N., Guernic, P.L. & de Simone, R. 2003. The synchronous languages twelve years later. *Proceedings of the IEEE* **91**(1), 64–83.

Cesta, A. & Oddi, A. 1996. DDL.1: a formal description of a constraint representation language for physical domains. In *New Directions in AI Planning*, Ghallab, M. & Milani A. (eds). IOS Press.

Do, M. & Kambhampati, S. 2001. Planning as constraint satisfaction: solving the planning-graph by compiling it into CSP. *Artificial Intelligence* **132**(2), 151–182.

Fox, M. & Long, D. 2003. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* **20**, 61–124.

Frank, J. & Jónsson, A. 2003. Constraint-based attribute and interval planning. *Constraints* **8**(4), 339–364.

Fratini, S., Pecora, F. & Cesta, A. 2008. Unifying planning and scheduling as timelines in a component-based perspective. *Archives of Control Sciences* **18**(2), 5–45.

Ghallab, M. & Laruelle, H. 1994. Representation and control in IxTeT: a temporal planner. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, AAAI Press, 61–67.

Ghallab, M., Nau, D. & Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Jónsson, A., Morris, P., Muscettola, N., Rajan, K. & Smith, B. 2000. Planning in interplanetary space: theory and practice. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, AAAI Press, 177–186.

Kautz, H. & Selman, B. 1992. Planning as satisfiability. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, John Wiley & Sons, 359–363.

Mittal, S. & Falkenhainer, B. 1990. Dynamic constraint satisfaction problems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, AAAI Press, 25–32.

Muscettola, N. 1994. HSTS: integrating planning and scheduling. In *Intelligent Scheduling*, Zweden, M. & Fox, M. (eds). Morgan Kaufmann, 169–212.

Nareyek, A., Freuder, E., Fourer, R., Giunchiglia, E., Goldman, R., Kautz, H., Rintanen, J. & Tate, A. 2005. Constraints and AI planning. *IEEE Intelligent Systems* **20**(2), 62–72.

Pralet, C. & Verfaillie, G. 2008a. Decision upon observations and data downloads by an autonomous earth surveillance satellite. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation for Space (iSAIRAS)*, AAAI Press.

Pralet, C. & Verfaillie, G. 2008b. Using constraint networks on timelines to model and solve planning and scheduling problems. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (ICAPS)*, 272–279.

Pralet, C., Verfaillie, G. & Schiex, T. 2007. An algebraic graphical model for decision with uncertainties, feasibilities, and utilities. *Journal of Artificial Intelligence Research* **29**, 421–489.

Puterman, M. 1994. *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Rossi, R., Beek, P.V. & Walsh, T. (eds) 2006. *Handbook of Constraint Programming*. Elsevier.

Schiex, T., Fargier, H. & Verfaillie, G. 1995. Valued constraint satisfaction problems: hard and easy problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Morgan Kaufman, 631–637.

Smith, D., Frank, J. & Cushing, W. 2008. The ANML Language. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (ICAPS)*, AAAI Press.

van Beek, P. & Chen, X. 1999. CPlan: a constraint programming approach to planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, AAAI Press, 585–590.

Verfaillie, G., Pralet, C. & Lemaître, M. 2008. Constraint-based modeling of discrete event dynamic systems. *Journal of Intelligent Manufacturing, Special Issue on ''Planning, Scheduling, and Constraint Satisfaction''*. Published online.

Vidal, V. & Geffner, H. 2006. Branching and pruning: an optimal temporal POCL planner based on constraint programming. *Artificial Intelligence* **170**, 298–335.

https://doi.org/10.1017/S0269888910000172 Published online by Cambridge University Press