

Constraint Model-based Planning and Scheduling with Multiple Resources and Complex Collaboration Schema

C. Guettier

Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304, USA
guettier@parc.xerox.com

B. Allo and V. Legendre

J.-C. Poncet and N. Strady-Lécubin
AXLOG Ingénierie, 19-21 rue du 8 Mai 1945
94110 Arcueil, FRANCE
bertrand.allo;vincent.legendre}@axlog.fr
jean-clair.poncet;nelly.lecubin}@axlog.fr

Abstract

In many domains, planning and scheduling problems have been considered separately. This historical decomposition leads to sub-optimal solving methods as task scheduling and planning share important sub-problems. In fact, on practical examples, there is no reason to separate coordination of activities and synchronization tasks. Furthermore, resource usage is not only relevant for task scheduling problems, but is also strongly affected by planning decisions in real-world problems. This paper proposes a Constraint Model-Based approach to concurrently tackle planning and scheduling problems. We show how constraint-based formulations can take advantage of flow models widely investigated in the operation research community. Our approach extends this model classification in order to derive both timing constraint and resource consumption for each transition of the plan. Therefore, scheduling constraints involving precedences, exclusive disjunctions, and resource capacity limit can be stated over the set of timing and resource variables. On aeronautic and spatial examples, we demonstrate how this approach enables problem-dependent specialization and increases planning and scheduling efficiency. Lastly, by using real-world problem experimentations, we show how the approach supports local / global trade-offs while designing solving methods.

Introduction

Today planners and schedulers are commonly used as off-line tools that search for optimised sequences of actions and tasks to be realized in a given future. On the one hand, planning aims at defining a task set that would perform a given goal. On the other hand, scheduling generally addresses the timeline of a fixed task set while satisfying available resources and precedence constraints. Many applications can be traditionally found in industry resource planners, skill planning for human resources, factories, airports and power plant management. These systems require interactions with human experts who are generally guaranteeing the consistency between planning and scheduling.

The rise of on-board autonomy in space and aeronautics domains(Muscettola 1998)(Allo & al. 2001) and smart matter systems fitted with hundreds of sensors and actuators(Fromherz et al. 1999) are driving planning and scheduling functions. In fact, the human interactions decrease as

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

the complexity of embedded systems increases. Therefore, most of the planning and scheduling approaches that have been considered separately in the past must be nowadays tackled simultaneously. Those systems will also have to deal with restricted on-board resources (energy for spacecraft, kerosene for aircraft, number of sensors and actuators, processing and communication capabilities, etc). Management of these resources is a critical point, as they are also potential sources of system failures(Poncet et al. 2001).

Complex task scheduling has been a major area of investigation within the constraint programming community, since it is representative of an NP-difficult problem class for challenging Constraint Logic Programming (CLP) capabilities(Aggoun and Beldiceanu 1993)(Van Hentenryck 1995). CLP languages(Jaffar and Lassez 1987) can be viewed as an extension of Logic Programming ones where unification is replaced by constraint satisfaction. Logical predicates can be constraints interpreted in a mathematical algebra. The most widely spread one addresses Finite Domains (FD) of \mathbb{Z} with all its classical operators $\{+, -, *, =, >\}$ and is denoted CLP(FD). Considered throughout all the paper, this language has been shown to be a powerful framework for developing complex operators as well as global constraints(Aggoun and Beldiceanu 1993) well-suited for disjunctive and cumulative scheduling problems. In fact, the composition of constraints through logical connectors and quantifiers enables one to address wide, large-scale classes of combinatorial problems(Dincbas et al. 1990)(Jourdan 1995)(Fromherz et al. 1999).

By taking advantage of CLP compositionality, modularity and genericity, Constraint Model-based Programming is a practical approach to keep both problem representation and solving method tractable(Saraswat et al. 1993)(Jourdan 1995). The modelling method captures problem invariants by recursively decomposing the global problem into sub-problems, until predefined constraints (global complex and elementary constraints) can be expressed. As a consequence, global search strategies, combined with branch and bound optimisation algorithms, are easier to elaborate, using structured knowledge of the problem invariants, model structure, and composition.

Following this approach, we propose a flow-based formulation of planning problems, extracted from the state of the art in Operation Research (OR). More generally, graph

path algebra is a powerful way to model practical planning theories and problem instances (Gondran and Minoux 1995). This approach offers a straightforward and natural formulation of constraint-based path planning, as the problem structure matches the physical structure. Also, models with multiple flows and weights, which are known to be difficult using OR techniques, can be efficiently addressed. Moreover, an important feature is the association of a closed form of transition times in order to derive classical scheduling constraints. On two representative examples of planning and scheduling functions, designed for spacecraft formation flying and military aircraft missions, the approach is resistant to domain dependent specialization in spite of the numerous additional constraints to consider. Examples are aircraft dynamics and interoperability for aeronautic missions, orbital manoeuvres for space, as well as specific resources utilization for both domains.

This approach is global in many ways, on both modelling and search strategy sides. However, on practical examples, this global optimisation yields several ways to break up the global search strategy into several smaller structures that perform a more local solution exploration. As a side effect of the approach, experiments show that several incremental search techniques can be designed, relying on heuristics and the decomposition of the problem.

The paper is organized as follows. In a first part, generic models to handle planning and resource management are introduced. Then, two specializations of these constraints on real-world problems will be assessed: the first one relies on aircraft mission planning and the second one spacecraft autonomy. Several experimental results will be given to show that the global approach allows to achieve local reasoning by scaling-up the problem. Before concluding, related works will be addressed.

Constraint-based models for planning multiple complex activities

This section explains the possible composition of path planning and scheduling models in a constraint programming framework. Disjunctive and precedence constraints are formalized using a graph flow model. The models introduced are highly generic and can be instantiated in many ways on real-life domains.

Planning as flow constraints

The plan space is represented as a graph $G(X, U)$ where the set of edges U is representing states and the set of vertices X transitions¹. The graph is considered as acyclic. Further model specializations show how to relax this assumption according to problem structure. A given path of states, also called an activity, is defined by the set of positive flows (1). Flows are represented by variables $\phi_u \in \{0, 1\}$, and the edge u belongs to the path if and only if variable ϕ_u is instantiated to 1.

¹In the remaining of the paper, a vertex is denoted x , while an edge can be denoted either u or (x, x')

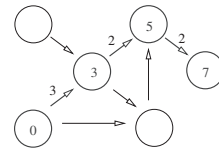


Figure 1: Example of positive flow (bold arrows) and associated weights $\{0, 3, 5, 7\}$ through a graph.

$$\Phi = \{\phi_u \mid u \in U, \phi_u = 1\} \quad (1)$$

From an initial state to a final state, path consistency is asserted by the following constraints, where $\omega^+(x) \subset U$ and $\omega^-(x) \subset U$ are outgoing and incoming edges from vertex x , respectively.

$$\sum_{u \in \omega^+(start)} \phi_u = 1, \quad \sum_{u \in \omega^-(end)} \phi_u = 1 \quad (2)$$

$$\forall x \in X \setminus \{start, end\}, \quad \sum_{u \in \omega^+(x)} \phi_u = \sum_{u \in \omega^-(x)} \phi_u \leq 1, \quad (3)$$

Equation (3) ensures path connectivity and unicity while equation (2) imposes limit conditions for the extremities of the path. This constraint gives a linear chain alternating transitions and states along the graph.

When a given weight w_u is associated with an edge u , the well known path length formulation (4) often addressed in OR (Gondran and Minoux 1995) expresses weights $w(x)$ along path transitions (see example in figure 1):

$$\forall x \in X, \quad w(x) = \sum_{(x',x) \in \omega^-(x)} \phi_{(x',x)} (w_{(x',x)} + w(x')) \quad (4)$$

$$\forall u \in U, w_u \in \mathbb{Z} \quad (5)$$

The set of transition weights that belongs to an activity (data measurement, goal realization ...) is expressed as follows (6):

$$\forall x, t_x = \min(1, w(x)), W = \{w(x) \mid x \in X, t_x = 1\} \quad (6)$$

where t_x states whether transition x is part of the planned activity. For instance, weights w_u can represent time, physical distance, or resource consumption so that associated paths may correspond respectively to plan duration, path distance, or overall consumption.

These constraints define a path algebra denoted $\{\mathbb{N}, \oplus, \times, \varepsilon, e\}$, where e, ε are neutral and null elements. Because we are interested in generating paths as plan solutions, \oplus is the classical $+$, $\varepsilon = 0$ and $e = 1$. It is important to note that $+$ is not idempotent, so that many OR results cannot be applied. Thus, without loss of generality, it is possible to express formulae (3, 2 & 4) using operators of a Constraint Programming language. However, real-life planning and scheduling problems necessitate to deal simultaneously with different resources, distances, and time representations, leading to more complex constraints involving $w(x)$ and/or several weights.

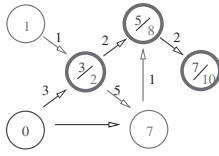


Figure 2: Example of multiple paths (bold arrows) and associated weights $\{0, 3, 5, 7\}$ and $\{1, 2, 7, 8, 10\}$ through a weighted graph.

Predicates for constraint-based planning

For many problems, such as planning for multiple collaborative agents, generic constraint schemes are necessary for defining transition synchronizations, coordination, and interoperability requirements. To formally define those constraint predicates, we need to consider simultaneously multiple paths, represented as a set of activities $\{\Phi_k\}_0^n$ and their associated weights $\{W_k\}_0^n$. Where $k \in \{0, \dots, n\}$ is a given activity (see example figure 2). By allowing multiple paths, when the weights represent time, activities may be conducted in parallel.

Most of the generic constraints defined in this section are composition and/or specialization of several constraints (7) describing a causality between two vertices x and x' of different activities k and k' .

$$\forall k, k' / \text{causal}(k, x, k', x', c) \Rightarrow w_k(x) \geq w_{k'}(x) + c \quad (7)$$

where $c \in \mathbb{N}$ is either a constant or a variable.

By introducing these constraints, it becomes very difficult to take advantage of OR techniques based on path algebra, multicommodity flows or linear programming. However, specific constraints can be designed to solve those disjunctions. By combining (\oplus, \times) formula and disjunctive constraints, one can solve complex planning and scheduling problems involving several disjunctions.

Another set of planning constraints addresses conditions on vertices weights. For a given vertex, conditions on different activities may constrain weights of outgoing or incoming vertices. For example, constraint (8) imposes equal weights on outgoing vertices x', x'' from transition x , opposed to (9).

$$\forall k \in \{0, \dots, n\}, \forall x', x'' / (x, x'), (x, x'') \in U, w_k(x') = w_k(x'') \quad (8)$$

$$\forall k \in \{0, \dots, n\}, \text{alldiff}_{x' / (x, x') \in U} (w_k(x')) \quad (9)$$

Constraints (8) and (9) illustrate how elementary CLP unification and disequation (Dincbas et al. 1990)² can be used for planning. Obviously, as shown in further examples, more complex constraints may be applied on ingoing/outgoing edges of the graph.

²The constraint $\text{alldiff}(V)$ specifies that all variables in the set V have to be different.

Predicates for constraint-based scheduling

When the weight defines a temporal metric, many constraints can be defined as multiple composition and specializations of constraint (7).

- Disjunctive transitions

In many cases such as multiple vehicle trajectory planning or exclusive resource handling, exclusion constraints can be associated with transitions. The predicate $\text{exclusive}(k, k', x)$ specifies that transition time (resp. distance) on x are exclusive for activities k and k' (10).

$$\forall k, k' / \text{exclusive}(k, k', x)$$

$$\Leftrightarrow \text{causal}(k, x, k', x, 1) \vee \text{causal}(k', x, k, x, 1)$$

$$p_x = \min(t_x^k, t_x^{k'}) \quad (10)$$

This constraint is useful to specify temporal (resp. spatial) exclusion when two activities hold the same transition (such that in figure 2, if weights are corresponding to timing information). The 0-1 variable p_x states whether both k and k' trigger transition x . This additional variable complements the usual formulation of disjunctive constraints in task scheduling problems (Aggoun and Beldiceanu 1993).

- Ordered transitions

The predicate $\text{pred}(k, k', x)$ specifies that the weight for k is lower than the weight for k' on transition x (11).

$$\forall k, k' / \text{pred}(k, k', x) \Leftrightarrow \text{causal}(k, x, k', x, 1) \quad (11)$$

Here again, when the weight defines a timing concept, this constraint is useful to specify invariant temporal causalities between activities.

- Same transitions weight

The predicate $\text{eq}(k, k', x)$ specifies that transition weights on x are the same for activities k and k' (12).

$$\forall k, k' / \text{eq}(k, k', x) \Rightarrow w_k(x) = w_{k'}(x) \quad (12)$$

When the weight represents time, this constraint becomes a synchronization point between activity k and k' .

- Toward cumulative constraints

Different kinds of cumulative and capacitive resources can be addressed using the flow-based representation. When considering a given activity, the total (or cumulated) weight of a path must not exceed a given limit L (13). For a given vertex $x \in X$, the total amount of weight must not exceed a given capacity C . If the edge weights are all positive, equation (14) is enough, because the total weight increases monotonically with the number of nodes in the path (Gondran and Minoux 1995). When the edge weights are not all positive, then, the capacity must be checked for each transition (15).

$$\forall w(x) \in W, w(x) < L \quad (13)$$

$$\sum_{k \in \{0..n\}} w_k(x) < C \quad (14)$$

$$\forall d \in \bigcup_{k \in \{0..n\}} d^k(x), \quad \sum_{k \in \{0..n\}/d_k(x) > d} w_k(x) < C \quad (15)$$

In equation (15), each d corresponds to a particular event on vertex x , the set of dates associated to a given vertex. This last constraint holds the same complexity than cumulative constraint (Aggoun and Beldiceanu 1993) in job shop scheduling. A similar constraint set (13,14 & 15) can also be formulated for edges.

Problem instances

A planning and scheduling solution subsequently correspond to a partial or complete assignment of variables $w(x)$, t_x , w_u , ϕ_u , according to the goals to satisfy. By using CLP capabilities, any subpart of the variables can be statically instantiated and any subpart of the models can be solved according to the goal to satisfy. Additional constraints are added by assigning a subset of variables or optimising a cost function such as path weight like distance, timing or a global resource consumption.

Concurrent constraint solving

Planning and scheduling problems are modelled using a constraint-based multiple models approach (Jourdan 1995; Saraswat et al. 1993). Therefore, each model represents a different aspect of the whole problem and owns internally a set of variables and constraints, so that it can be solved either independently or commonly with other models. Models are composed by unifying part of their variables and adding inter-model constraints. Hence, concurrent search strategies over several scheduling, planning and resource models can be implemented using *ask* and *tell* operators (Van Hentenryck et al. 1995) as coordination mechanism. By using these operators, it is possible to freeze the solving over a model, waiting for external constraints to be entailed. For example, concurrent search strategies over multiple activities and resources can be designed by taking advantage of the graph structure. Using a graph exploration, each activity builds a forward search along feasible transitions, while satisfying resources and scheduling constraints. In practice, few transitions are actually explored, due to constraint propagation and early inconsistent partial solutions. Many other concurrent strategies can be defined as shown in the experiments.

Problem examples

Two problem instances will be considered to illustrate the effectiveness of our approach. The first one is a military aeronautic problem where formations have to coordinate themselves to perform a strike on a given target. The second one illustrates the application of the generic model to a space observation problem for a spacecraft formation where resource utilization is critical.

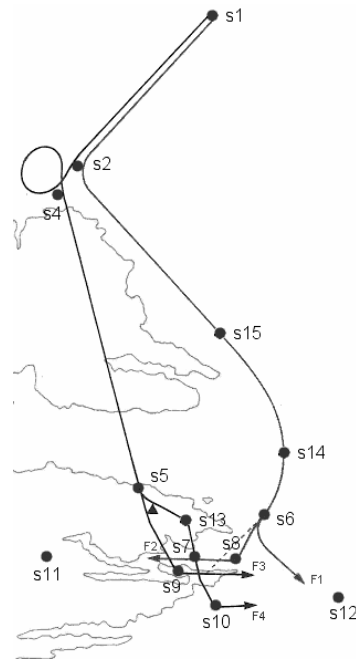


Figure 3: Expert solution on the real map.

Multiple path planning for air operations

The discussed model has been instantiated and specialized for modelling an aircraft mission planning problem. The focused air mission involves multiple formations which have their own objectives. Moreover, overall mission objectives are achieved thanks to coordinated actions between formations. In the addressed domain of air mission planning, a complete plan is a set of edges to fly by for each formation, constrained by feasible mechanical parameters (altitude, aircraft attitude, speed ...), on-board resources (kerosene, jamming, decoys ...), coordination, interoperability between aircraft and by objectives to achieve (imposed time window for corridor entry/exit).

Figure 3 describes the mission environment. All formations are coming from the same nav point s_1 . Formations F_1 and F_2 must cross simultaneously nav point s_6 to perform Suppression of Enemy Air Defence (SEAD). Formations F_3 and F_4 must cross nav point s_5 , and then take different routes (resp. by s_7, s_{10}, s_{12} and s_9, s_{12}). Threats are localized in the s_9, s_8 zone, where the mission objective is (imposed as a constraint).

Figure 4 shows mission objectives. Formation F_2 must cross nav point s_7 before formation F_3 , to perform a Battle Damage Assessment (BDA). In the same way, to satisfy system inter-operability, formation F_3 must cross nav point s_7 before formation F_4 crosses nav point s_9 . Formations F_1, F_3 and F_4 must escape by nav point s_{12} and formation F_2 by nav point s_{11} . Those mission interoperability requirements have been represented using the coordination formalisms (10 & 11).

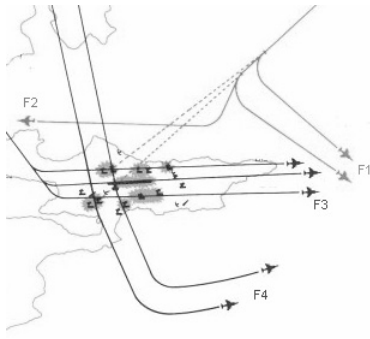


Figure 4: Detailed view of threats and targets.

Extending planning formulations for realistic navigation model

In this context, the planning phase consists in solving all the models, finding a route for each formation within the set of possible navigation points to achieve the whole mission. The graph $G = (X, U)$ corresponds to possible paths, where X corresponds to navigation points, and U to fly-by areas. Linked navigation points are entry and exit points of the defined area. A transition is a manoeuvre on a navigation point and a state a nominal flight through a fly-by area. The model is extended to consider complex tactical needs: holding patterns within an area is allowed by introducing a decision variable for each edge $u \in U$: $c_u \in \mathbb{N}$. The formation can hold on in an area by flying back to the entry point after the exit navigation point has been reached, defining elliptic patterns. However, the formation must leave the area through the exit navigation point. Thus the formation can fly by the area $2 \cdot c_u + 1$ times. The variable c_u corresponds to the number of waiting cycles performed on the edge u . When u does not belong to the path, the number of waiting cycles is zero (16):

$$\forall u \in U, \phi_u = 0 \Rightarrow c_u = 0 \quad (16)$$

Several weights are specifying the navigation model. For instance, let s_u be the speed of the aircraft when flying along the edge u . The domain of s_u corresponds to a discretisation of the continuous speed domain (tenth of mach in the implemented example).

In addition, the model takes into account two physical aspects of the air mission. The spatial representation aims at modelling the static geographical map of the area in which the mission takes place. The threat model matches the enemy's positions.

- Navigation points are defined with their coordinates in the three-dimensional space so that the length l_u and a cost of fly in terms of kerosene resource are weighting the edge u . Constants α_u and β_u correspond to consumption characteristics of the aircraft along the edge u . These parameters take into account altitude, air pressure and covered distance.
- The different formations are threatened by a group of ground air defences distributed along the way to the target. Each aircraft can protect itself from the enemy by a

limited ability to hide. Based on air-ground defence positions, this is modelled using a self-protection weight p_u cumulated along the mission path vertices. On each vertex x , the self-protection used, cumulated from the beginning of the path, must not exceed an available amount P_x . This weight is computed according to two components along vertical and horizontal axis. Threat value is integrated along these two axes according to the distance between the threat source and the edge vertices, and using a threat distribution law.

Problem dependent models as path weights specialization By introducing waiting cycles c_e , the generic cumulative constraint (4) has to be substituted by the following specialization (17):

$$\begin{aligned} \forall x \in X, \\ w(x) = \sum_{(x',x) \in \omega^-(v)} \left[\phi_{(x',x)} \right. \\ \left. (w_{(x',x)} (2 \cdot c_{(x',x)} + 1) + w(x')) \right] \end{aligned} \quad (17)$$

The multiple resource models are formalized by replicating eq. (17), constraining the weight formulation (13), and substituting the following specific formulations to the vertices and edges weights (resp. $w(x)$ and w_u).

• Self-protection

The self-protection $p(x)$ on each vertex x is given by directly substituting $p_{(x,x')}$ to $w_{(x,x')}$ in eq. (17). An additional limit constraint is also applied (18).

$$\forall x \in X, p(x) < P_x \quad (18)$$

• Timing

With the introduction of aircraft speed s_u , the timing $d(x)$ as the flyby dates over each navigation point x , leads to consider the following edge weight d_u (19).

$$\forall u \in \Phi, d_u = \lfloor \frac{l_u}{s_u} \rfloor \quad (19)$$

• kerosene

Lastly, when considering C as the total available kerosene, the consumption $c(x)$ when arriving to a navigation point x , leads to the following edge weight c_u (20). The consumption on each vertex also satisfies a given limit (21).

$$\forall u \in \Phi, c_u = \alpha_u + \beta_u \cdot s_u \quad (20)$$

$$\forall x \in X, c(x) < C \quad (21)$$

The aircraft performances are also taking advantage of the graph representation, by constraining adjacent edges speed pairwise.

• Acceleration

For each pair of adjacent edges u and u' , we statically calculate a new coefficient $\Psi_{(u,u')}$ that depends on the angle between the two edges and the maximum acceleration worth both to the cell structure and the pilot. Speed

on successive followed edges is then linked by the constraints (22):

$$\begin{aligned} \forall u, u' / adjacent(u, u'), \\ (s_{u'} - s_u) \cdot (s_{u'} + s_u) \leq \Psi_{(u, u')} \end{aligned} \quad (22)$$

• Turning

Similarly, we calculate another parameter $\gamma_{(u, u')}$ that also depends on the angle between two adjacent edges u and u' , and the aircraft turning capabilities. The average speed during the turn is approximated by $\frac{s_u + s_{u'}}{2}$ which is then constrained with $\gamma_{(u, u')}$, leading to (23):

$$\begin{aligned} \forall u, u' / adjacent(u, u'), \\ \frac{s_u + s_{u'}}{2} < \gamma_{(u, u')} \end{aligned} \quad (23)$$

Both static parameters $\Psi_{(u, u')}$ and $\gamma_{(u, u')}$ are characterizing aircraft performances on the transition (u, u') . This model could be easily refined with additional constraints sets modelling other performance aspects such as aircraft acceleration and braking capabilities.

Lastly, several constraints composed with planning and scheduling predicates (7), (8) and (10-15) enable to define tactical constraints such as exclusive or joint flyby of an area using disjunctions (eq. (10)) or timing equalities (eq. (12)), formation covering while entering, crossing or exiting an area, and successive operation in an area using (eq. (11)). In all these constraints, the use of a timing interval $[d_{min}, d_{max}]$ remains useful for defining minimum and maximum delays using causal predicates (eq. (7)). The width of the interval makes constraints more or less flexible.

Experimentations In our approach, solving the global problem corresponds to a long-term planning function which is solved before the mission starts. The problem under consideration³ involves solving a conjunction of all the models previously described and minimizing a cost objective⁴. Once a first global plan has been delivered, each formation leader can refine and adapt its own plan according to a more accurate representation of its immediate environment. In this case, coordination constraints have been already solved in the initial plan and are relaxed as a cost objective for each formation. Starting from the initial global plan, each formation can solve incrementally its own plan by minimizing delays with pre-planned meeting dates. Then, scalability issues can be addressed, decentralizing the solving onto the different formations.

The global problem is decomposed into independent sub-problems according to the mission hierarchy. When the solving time is limited, all the formation leaders can be designated to solve *locally* easier instances by recasting coordination constraints as objective function. On the opposite, if an important time is allowed for searching a *global* plan-

³Experiments have been run using a SICStus Prolog implementation on a Pentium 400Mhz with 128 MB RAM.

⁴The cost function can be taken from cumulative models, minimizing a critical data. A combination of data can also be used in case of a multi-criterion objective.

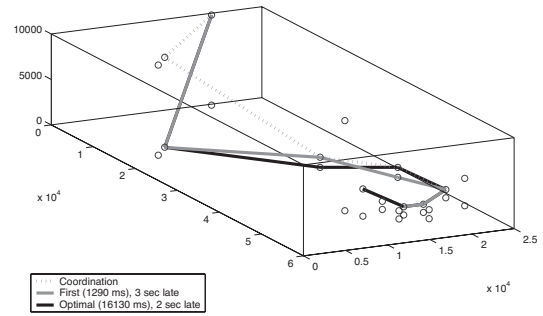


Figure 5: Planning and scheduling solution updates for a group of two formations after limiting resources. Dots are possible vertices. The first solution (in grey) is close to the optimal one (in black).

ning and scheduling solution, a single leader can plan centrally for all the formations, keeping all coordination constraints. Intermediate cases can be considered when some sub-leaders solve the planning and scheduling problem for different subsets of the formations.

Figure 5 gives an example of distribution of the planning process on two different formations. The two formations are no more exactly synchronized, but they fly by their previous synchronization point with a short delay. In this case, the pre-planned meeting dates will be considered as a heuristic on the best plan to guide the search for each formation. This heuristic is not optimal, since it removes solutions from the search space (the strategy is not complete), but quickly gives solutions and allow local reasoning schemes.

A global coordination quality (24) can be defined to evaluate solutions effectiveness emerging from the decentralized solving process. In this case, the quality is a scalar product defined as follows:

$$q(t) = \vec{s} \cdot \vec{c}(t), \quad (24)$$

where \vec{s} is a probability of success vector⁵ and $\vec{c}(t)$ a 0–1 vector representing coordination constraints violated at a given moment t of the solving process. Because the quality characterizes the global state of the decentralized system, it cannot be part of a cost function, but only evaluated statically over several scales of decomposition.

Realized onto variations of the specification of an actual strike scenario, experiments presented in figures 6 and 7 show worst and average cases of quality measure according to different level of decentralization. Because the full decomposition involves more local explorations, solutions can be found incrementally and faster. On the opposite, global search always provide complete solutions (with 100 of quality) but are slower.

Search strategy For this experimentation, decision variables are path variables (ϕ_u), waiting cycles (c_u) and aircraft

⁵This is generally a predefined domain-dependent vector, that is out of the scope of this paper.

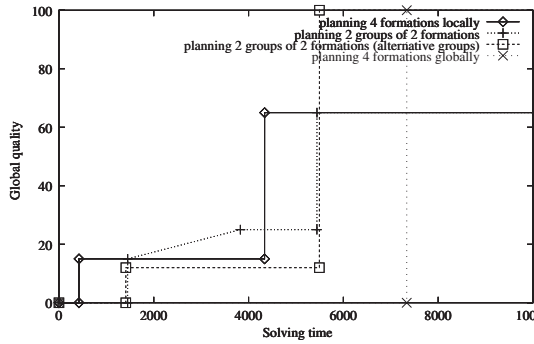


Figure 6: Average quality of planning and scheduling with decentralization trade-offs (times are in ms).

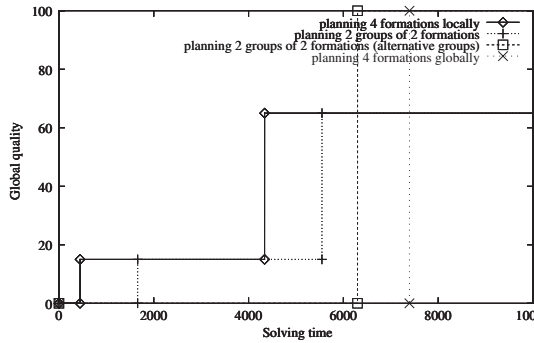


Figure 7: Worst quality of planning and scheduling with decentralization trade-offs (time are in ms).

speeds (s_u). The variable selection is done in this order, so that the path is entirely resolved before a coarse grain synchronization can be done with the waiting cycle enumeration, and lastly refined with a speed adjustment. The variables domain enumeration is done in increasing order for path and cycles variables, and in decreasing order for speeds.

Spacecraft Mission Planning and Scheduling Problems

Planning specialization for space domain considers a formation of spacecraft in a deep space mission. This formation is made of less than a ten of spacecraft, flying close one from each other (few hundreds of meters). The set of possible trajectories that can be used by formation are represented through a graph $G(X, U)$ containing navigation point joined by transition trajectory edges (figure 8).

Orbitography planning According to the specificity of space trajectory, the spacecraft formation can place itself onto an halo orbit around a celestial body or virtual point (like a Lagrange point). This necessitates to introduce in the graph-based representation specific vertices called *orbit vertices* in order to model orbital trajectories (8)(Guettier and Poncet 2000). These vertices are characterized by a transition time (exit date is different from entry date) and a transition matrix M expressing the time $d(x, y)$ between injec-

tion and extraction orbit points (Guettier and Poncet 2001). These enlarged graph representation allows one to express trajectory timed model for the whole formation. However, *traditional* vertices are also considered as *navigation points*. They are characterized by a date of exit that equals the date of entry. This way, formation path planning always relies on generic path constraints (2) and (3).

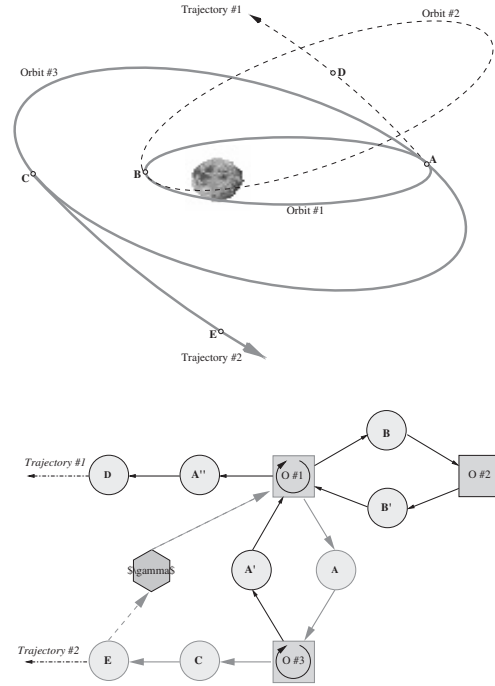


Figure 8: Orbits selection using path constraints

Goal Scheduling Considering the space domain, goals to be realised by a deep space formation are slightly different from aircraft mission domain. Goals (observation, measurements, manoeuvres, earth communications ...) have to be scheduled according to time constraints (possible time realization windows), spacecraft position in space and, resource availability (electric power, fuel, on-board memory ...). They can be operational (i.e. goals necessary to the formation safety and liveness) or mission oriented. Both the number of goals realized and cumulative resources are represented using multiple weights (4) on the graph representation. These weights are respectively a 0 – 1 variable specifying the realization of a goal on a given vertex and the amount of resource consumption/production on an edge. Exclusive resources, are defined as disjunctive transitions (10). Additionally, some of the goals shall be realized regularly according to a given period, like communications between spacecraft formation and earth based control stations. This is defined using a causality (7) predicate and a period of P :

$$\forall x, x', k, P, \text{periodic}(x, x', k, P) \Leftrightarrow \text{causal}(k, x, k, x', c) \wedge \text{causal}(k, x', k, x, c + P) \quad (25)$$

A goal also owns a priority level directly related to its

importance for mission success and liveness (High priority goals are mandatory). This mechanism ensures that spacecraft formation security, safety and liveness will be preserved at each moment during the mission life as safety critical actions are included in the plan.

Space planning goals are scheduled along the path determined onto the trajectory graph, but only onto vertices of the graph (Poncet et al. 2001) (26). Generic planning constraints inherited from predicates (7), (8) and (10-15) are used to constrain goals execution onto a given vertex of the graph in a given time window.

$$\forall k, \forall x; \text{ Let } d(k,x) \text{ the realization date of } k \text{ on } x$$

$$\text{if } x \text{ is an orbit vertex then } E_x < d(k,x) < S_x$$

$$\text{else } d(k,x) = E_x = S_x \quad (26)$$

Where E_x is the date of entry onto vertex x and S_x is the date of leaving vertex x .

Obviously, it remains necessary to determine all vertices and edges properties using advanced trajectory computing models before starting the plan elaboration.

Resource Allocation In order to obtain a feasible mission plan, it is necessary to represent the evolution of main resources values through time, according to vehicle motions and goal realization. Resource levels depends on the goal affectation model that specifies which spacecraft of the formation is involved in which goal realization. Thus, resource consumption model is instantiated for each resource of each spacecraft of the formation.

Considering resource consumption, it is necessary to distinguish exclusive, cumulative and capacitive resources. The figure 9 illustrates the evolution of resources levels for three different resources (namely R1, R2 and R3). R1 is an exclusive resource, R2 is a capacitive resource and R3 is a cumulative resource.

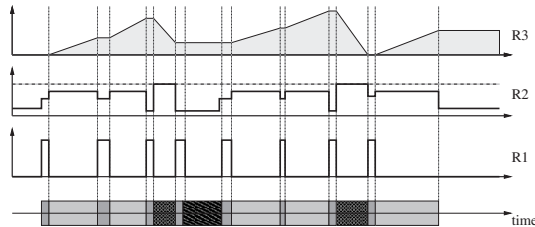


Figure 9: Resource consumption examples according to plan.

Cumulative resource R1 evolves during plan execution because some of the weight terms can be positive or negative according to plan action (eq. 15)⁶. Exclusive and capacitive resource consumption are interesting only when realizing goals. It is then necessary to check the availability of

⁶For instance, on-board memory can be consumed (occupied) when realizing observation and freed when uploading data to earth station or transmitting results to another spacecraft.

exclusive resources needed for goal execution and the sufficient level of capacitive resources involved in this realization (like electrical power, for instance).

Cumulative resources necessitate adopting a different approach. Obviously, a goal will be feasible only if the required resources are available at the starting date of the goal, so level of resources shall be known before each goal execution in order to state its possible realization.

$$\forall k, \forall x, \text{ feasible}(k, d(k,x)) \Leftrightarrow \begin{cases} \forall R \mid \text{required}(k, R, N) \\ \exists r \in R, r(d(k,x)) \geq N \end{cases} \quad (27)$$

Constraint (27) expresses that an activity k will be feasible on a vertex x at a date $d(k,x)$ if and only if for each class of resource of class R needed to realize k it exists a physical resource $r \in R$ such that the amount of r at date $d(k,x)$ is at least equals to the level N required to realize k .

Obviously, levels of resource must also be known after each goal execution in order to compute remaining parts of the plan. Moreover, the cumulative resource level shall also be expressed when coming on and leaving out each vertex of the formation path into the trajectory graph.

$$r_s(E_x) = \left[\sum_{u=x}^{\mathcal{J}x \in \omega_x^+} (r_s(S_{x'}) - \Delta_c(u)) \times \delta u \right] \quad (28)$$

Equation (28) expresses the value of resource r of spacecraft s when entering on vertex x at date E_x as the value $r_s(S_{x'})$ of the resource when exiting from the predecessor x' of x in the path, minus the value $\Delta_c(u)$ of resource spent on the edge leading from x' to x . Constraint expressing value of a cumulative resource between two goals execution looks like (28) except that it must express possible resource value according to all possible goals schedule, introducing numerous disjunctions.

Experimentations In this example, mission plans have been searched for a single spacecraft, for a formation of two spacecraft and then for a formation of four spacecraft. Several planning experiments have been run requiring the scheduling from 10 and up to 100 goals. Experiments have been divided into two classes, the first one including resource constraints (figure 10), the second one with exclusive constraints only (figure 11). Constraint resources have been stated in this examples for cumulative resources (memory and propellant), capacitive resources (electrical power) and exclusive resource (payload instrument). Both experiments include mode synchronization constraints for the 2 spacecraft and 4 spacecraft formations.

Figures reported in 10 and 11 show that solving duration does not explode according to the number of goals. This is due to the multiple constraint propagations between the different weights of the global problem. Likewise scheduling problems, cumulative resources generate more difficult problems than exclusive ones⁷. Those experiments show that decentralizing the search according to the number of

⁷Results have been obtained using an Ilog Solver Implementation onto a Sun Ultra 5 workstation with 128 MB RAM.

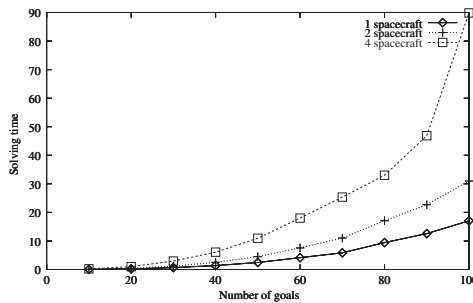


Figure 10: Planning and scheduling with cumulative and exclusive resources.

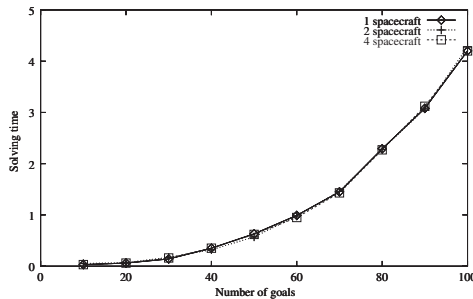


Figure 11: Planning and scheduling with exclusive resource only.

spacecraft is more interesting when dealing with cumulative resources and large set of goals. Hence, without changing the modelling, different search strategies can be dynamically selected according to planning and scheduling parameters (number of goals, spacecraft, resources properties).

Search strategy The strategy used to solve these problem instances is based on a dynamic goal scheduling approach. Goal schedules are explored one after another, so that each schedule propagates feasibility constraints over resource models through consumption constraints along the path of feasible orbits. Associated to dedicated scheduling heuristics, this enables an efficient solving even if this search strategy can be improved in further works.

Related Works in Planning

Much planning work has been led by Honeywell for both military or civilian purposes, using dynamic programming. In their approach, potential fields are defined to describe the environment. Attractive fields represent targets and repelling fields model threats. The optimisation process is then to find an appropriate path in the resulting potential field. Several criteria can be considered such as fuel consumption or synchronization time. Another classical approach relies on Hierarchical Task Network (HTN). This is what has been implemented on MACBeth (see (Goldman et al. 2000)), which provides a large set of possibilities to specify a problem constrained in time and resource. This planner combines HTN with constraint propagation and is more generic

than the previously cited systems, since it has already been applied to ground vehicles (UGV DemoII), UAVs and soldier platoons.

On the other hand, our work can be compared to generic planning under resource management. According to (Long and Fox 2000), our problem can be identified as an instance of transportation problem using a set of pre-planning algorithms. Then, once identified, the class of problem can orientate the choice of heuristics. The pre-planning algorithms can be used in many planners, and so allows to consider generic planners as a possibility to solve our problem.

Compared to domain-independent planning using the PDDL formalism (see (Ghallab et al. 1998) for a definition of PDDL), the expressiveness of constraint satisfaction problems gives a larger freedom in the problems which can be designed. Nevertheless, the latest version of PDDL extends the language to take into account resource management and timing concerns. The language allows the definition of continuous or discrete resources and the management of time-consuming actions. These new improvements of the language traduce recent work led on this subject.

Two alternatives are currently used for planning under resource management. The first approach uses data structures to model the resources, and apply the solving process to the problem composed of actions and resources. This approach is fairly similar to ours, except that our formalism allows a greater expressiveness in the specification of resources management for intermediary states. Usually, only the final state is specified using this approach. Specifying other states urges to consider time resource as a parameter of every action. This approach is used in IxTeT (Laborie and Ghallab 1995), HSTS (Muscettola 1994) or RIPP (Koehler 1998) for example. LPSAT uses floats and fluents to model constants and variables and couples a SAT solver with a simplex method to manage both actions and resources. RIPP is an implementation of Graphplan (Blum and Furst 1997) allowing resource management.

Another orientation consists in selecting the actions in a first planning process and then consider the resource management as a second scheduling process. This is the case in parCPlan (Lever and Richards 1994) and RealPlan (Srivastava 2000). In RealPlan, both the planning and the scheduling problem are modeled and solved using dynamic CSP, and dependency directed backtracking can be provided between the two processes according to defined policies.

Lastly, the use of Concurrent Constraint (CC) language has been the focus of researches led in NASA and MIT. This approach takes advantage of the ask and tell paradigm to respectively request and broadcast agent states. The same paradigm is also used to constrain planning representations also assimilated as path formulations. Therefore, a partial plan solution can be dynamically updated by other agents status. These techniques have been used to implement the planner embedded in the Remote Agent Experiment (Muscettola 1998) (Williams & al 2001) mission launched within the Deep Space One spacecraft in October 1998. Remote agent is able to build mission plan, using constraint and previous set of existing partial plans.

At the moment, few generic planners are able to handle

both resource and timing issues. Recent studies gave first results in this domain, but the results remain still far less good than those obtained with dedicated planning systems.

Conclusion

Generally investigated separately, planning and scheduling algorithms may become increasingly complex and difficult to design when considering additional constraints (flight dynamics and coordination for air mission planning, spacecraft behaviour) and real size problems. This paper proposed a generic constraint-based representation for solving concurrently planning and scheduling problems. Flow models over graph appear to be a generic and powerful means to formalize the composition of scheduling and planning models.

Therefore, on the modelling side, properties offered by Constraint Programming languages allow designers to elaborate global models that can be specialized onto different problems. For this purpose, generic constraint-based predicates have been prototyped and implemented, addressing resource consumptions as well as activities synchronization and coordination. On the solving side, search method completeness can be compromised in different ways. Using the same modelling and according to system requirements (such as trade-off between duration and quality), one can decide to perform a complete search, relax some constraints or decentralize the solving in independent strategies.

The effectiveness of the approach has been demonstrated on real-world planning and scheduling examples with realistic parameters. Despite the numerous domain dependent constraints, the planning and scheduling function gives results compliant with system requirements and realistic problem size. Moreover, for these particular examples, practical rules for automatic selection of search strategy have been emphasized. As a result, those rules do not break the concurrency between planning and scheduling models.

References

- Aggoun,A. and Beldiceanu,N. 1993. Extending CHIP in order to Solve Complex Scheduling and Placement Problems, *Mathematical Comp. Modelling*, vol. 17, no. 7, pp. 57-73, Pergamon Press Ltd.
- Allo,B.; Guettier,C.; Lecubin,N. 2001. A Demonstration of Dedicated Constraint-based Planning within Agent-based Architectures for Autonomous Aircraft. In *Proceedings of the IEEE Symposium on Intelligent Control*.
- Blum,A.L. and Furst,M.L. 1997. Fast Planning through Planning Graph Analysis, *Artificial Intelligence*, 90(1-2):281-300.
- Dincbas,M.; Van Hentenrick,P. and Simonis,H. 1990. Solving Large Scale Combinatorial Problems in Logic Programming. *Journal of Logic Programming*, Vol. 8, p.75-93
- Fromherz,M.; Hoeberechts,M. and Jackson,W. 1999. Towards Constraint-based Actuation Allocation for Hyper-redundant Manipulators. In *Proceedings of CP'99 Workshop on Constraints in Control*.
- Ghallab,M.; Howe,A.; Knowblock,C.; Mac Dermott,D.; Ram,A.; Veloso,M.; Weld,D.; and Wilkins,D. 1998. PDDL- The Planning Domain Definition Language. Technical Report, CVC TR-98-003/DCS TR-1165. Yale Center for Computational Vision and Control.
- Gondran,M. and Minoux,M.1995. *Graphes et Algorithmes*, ed. Eyrolles, Paris.
- Goldman,R.P.; Haigh,K.Z.; Musliner,D.J.; and Pelican,M. 2000. MACBeth: A Multi-Agent Constraint-Based Planner. In *Proceedings of the AAAI Workshop on Constraints and AI Planning*. Menlo Park, Calif.: AAAI Press.
- Guettier,C. and Poncet,J.C. 2000. Automatic Planning for Autonomous Spacecraft Constellations. In *Proceedings of the International NASA Workshop on Planning and Scheduling for Space*, San-Francisco, USA, 2000.
- Guettier,C. and Poncet,J.C. 2001. Multi-Levels Planning for Spacecraft Autonomy. In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotic for Autonomy in Space*. St Hubert, Canada: Canadian Space Agency.
- Constraint Logic Programming. 1987. In *Proceedings of the 14th ACM POPL*, Munich.
- Jourdan,J. 1995. Concurrence et cooperation de modeles multiples dans les langages de contraintes CLP et CC. Univ. Paris VII.
- Koehler,J. 1998. Planning under resource constraints. In *Proceedings of ECAI'98*.
- Laborie,P. and Ghallab,M. 1995. Planning with sharable resource constraints. In *Proceedings of IJCAI'95*. Menlo Park, Calif.: International Joint Conference on Artificial Intelligence, Inc.
- Long, D. and Fox, M. 2000. Automatic Synthesis and use of Generic Types in Planning. In *Proceedings of AAAI 2000*. Menlo Park, Calif.: AAAI Press.
- Lever,J.M. and Richards,B. 1994. ParcPLAN: a planning architecture with parallel actions, resources and constraints. In *Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems*.
- Muscettola,N. 1994. Integrating Planning and Scheduling. In *Intelligent Scheduling*. Zweben and Fox, eds. Morgan-Kaufmann.
- Muscettola,N.; Pandurang Nayak, P.; Bell,P. and Williams, B.C. Remote Agent: To boldly go where no AI system has gone before, NASA Ames Research Center. 1998.
- Poncet,J.C.; Guettier,C.; Le Lann,G.; and Bornschlegl,E. 2001. Constraint-Based Layered Planning and Distributed Control For an Autonomous Spacecraft Formation. In *Proceedings of the 1st ESA Workshop on Space Autonomy*, 209-219. Noordwijk, NL: European Space Agency.
- Srivastava,B. 2000. RealPlan: Decoupling Causal and Resource Reasoning in Planning. In *Proceedings of AAAI 2000*. Menlo Park, Calif.: AAAI Press.
- Saraswat,V.;Bobrow,D. and de Kleer,J. Infrastructure for Model-based computing TR, Xerox PARC, Palo Alto Ca.
- Van Hentenryck,P.; Saraswat,V and Deville,Y. 1995. Design, Implementation and Evaluation of the Constraint Language CC(FD), In *Constraint Programming: Basics and Trends*, A. Podelski Ed., Springer-Verlag.
- Van Hentenryck, Constraint Solving for Combinatorial Search Problems: A Tutorial. In *Proceedings of CP'95*.
- B. Williams & al, Model Based Reactive Programming of Cooperating Vehicles for Mars Exploration. Multi-Levels Planning for Spacecraft Autonomy. In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotic for Autonomy in Space*. St Hubert, Canada: Canadian Space Agency.