Multi-Levels Planning for Spacecraft Autonomy

Christophe Guettier, Jean Clair Poncet
Axlog Ingénierie
19-21, rue du 8 mai 1945
94110 Arcueil, F
christophe.guettier@axlog.fr, jean-clair.poncet@axlog.fr

Keywords Spacecraft Formation, Planning & Scheduling, Multi-Agents System, Constraint Programming.

1 Introduction

Unmanned vehicles are a challenging concept for future missions in space, aeronautic and underwater domains. By decreasing human in the loop, the complexity has moved from interoperability requirements to autonomous command and control. Moreover, evolving in an hostile environment, with poor communication facilities, vehicle management is harder to perform remotely. Lastly, possible loss of equipments, communication facilities or a whole vehicle may have dramatic consequences. Relying on distributed computing nodes, a group of spacecraft is characterized by its own activities to be controlled as well as contingent activities that generates failures (communication jamming, components shut down...).

In addition to autonomous behaviors, the increasing amount of system components, combined with the set of modes and services lead to an uncontrolled explosion of combinatorial problems. Raised either at design time or operation time, their complexity becomes untractable by human experts. A trade-off can be formulated by stating incrementally both cost functions and assumptions on the system solution. Using those formulations dynamically to take the right decision in an unexpected situation is the key problem for system's adaptation to its environment. Consequently, it necessitates an intensive use of on board solving capabilities as demonstrated in the Deep Space 1 Remote Agent Experiment [7]. Widely investigated in research and industrial domains, Multi-Agents Systems (MAS) can provide intelligent behaviors to distributed systems[4, 5]. Several approaches exist that consider different levels of reactivity, cognition, sociability, decision capacity and communication expression. The main advantage of those powerful frameworks is to enable the modeling of autonomous functions, such as planning and scheduling, smart sensing and diagnosis together with cooperative and collaborative policies in distributed systems.

Considering spacecraft formations, multiple planning levels are relevant to the efficiency of the autonomous behavior. Within the scope of this paper, we consider a high level that deals with long-term environment and mission updates and a low level for managing short-term command and control. At the high level, global planning has to be performed for the whole constellation during operations. This requires to compute a mission management plan and to broadcast resulting mission goals to the whole formation. However, due to fault-tolerance requirements, time and processing power limits, all the planning details can not be considered centrally. Therefore, the global planning function remains on upper-approximations and/or sufficient statements of feasibility conditions. Spacecraft must locally command and control their actions, fulfilling safely its assigned mission goals. At the low level, current integrated modular avionic approaches make an intensive use of finite-state deterministic reactive automata [1]. However, this necessitates a perfect knowledge of the environment as well as rigid specifications of the system behavior.

Our approach introduces the use of non-deterministic constraint-based automata, so that each system component (such as camera, thruster, gyro wheels, platform) is represented by an automaton model. According to environment changes, a dedicated automata is synthesized automatically from the model by the on-board constraint solver. This approach provides a more adaptive behavior, extending the domain of use of the space system. Furthermore, the limited complexity of those automata enables the construction of plans that satisfy other resources and feasibility

constraints at a fine grain.

These two planning levels involve different abstractions and assumptions over the spacecraft constellation. They raise specific feasibility and synchronization problems that must be solved to guarantee the overall mission execution. Usual planning methods have poor abilities to combine the set of heterogeneous representations such as non-linear feasibility conditions, cumulative resources or disjunctive synchronizations [8]. In our approach, problems are specifically formalized in a multiple models approach, that can be solved jointly or concurrently [4] using a Constraint Programming (CP) approach. This allows the designer to deal with numerous resources, feasibility, coordination, synchronization and other domain-specific constraints, while considering the whole spacecraft formation. Furthermore, this approach enables an easier specialization of models toward dedicated problems instead of using simple deterministic heuristics that are reducing the operational scope of the formation. Deep space probes and earth orbiting formations are relevant examples of potential target domains.

This paper focuses on planning problems that occur at different levels of the multi-agents architecture. We first describe in §2 how we associate a multi-agent system to an autonomous flying formations of spacecraft. Then, we expose in §3 solving approaches for local and global planning functions.

2 A Multi-Agent Approach for Autonomous Formation Flying Design

Introducing autonomous functions in a spacecraft architecture raises several problems for decision making, sensing and communication. The space domain imposes hard constraints on component safety, system provability, architecture security and behavior predictability. The conjunction of these requirements urges to globally specify autonomy-oriented architectures and decision functions. This specification must satisfy heterogeneous constraints extracted from various domains (spacecraft engineering, on-board real-time distributed systems, sensing devices, physics and flight dynamics) as well as mission goals (earth observation, space probes, planetary observations). More particularly, while designing those distributed systems, collaboration, cooperation and synchronization problems must also be taken into account to insure a global efficient and safe behavior. In addition to autonomy problems, these paradigms increase system complexity and must be considered together with the individual behavior of spacecraft in order to design globally consistent architectures [9].

2.1 Multi-Agents System Overview

The intelligent planning and control of a whole flying formation leads to manage a representation of spacecraft interactions within the formation as well as between the environment and the formation. From this viewpoint, MAS is a good approach to model a spacecraft formation as a distributed autonomous architecture [2]. The MAS used in our approach can be characterized as follow:

- Each agent is associated to a unique spacecraft of the formation. At the local agent level, the design relies on a sense-plan-react control loop (time scale is between the second and a minute for spacecraft and platform control). Solving dynamically local command and control problems provides some degrees of autonomy to the agent to achieve short term goals.
- We assume communication network facilities are effective between spacecraft (thanks to Inter Formation Links techniques). We also assume that distributed fault-tolerant algorithms can be used for solving consensus between agents [5]. More particularly, consensus is mandatory for synchronizing planning activities and organizing the agent hierarchy. The long-term global commanding loop (scaled from a minute up to several hours) involves planning problems extended with collaboration and cooperation models.

Associating constraint based planning techniques with multiple models formulation, and advanced search techniques like any-time solving, guarantees decision predictability. This particular combination of architecture and decision method improves the system flexibility, safety, survivability, performance and liveness. As a consequence, the agent architecture integrates both reactive and deliberative behaviors. The resulting system is an hybrid architecture where agents follow long term plan to meet the mission objectives but can also react very quickly to unpredicted events.

2.2 Global MAS Architecture

To perform the distribution of goals to spacecraft, a hierarchy is defined over the agents of the system. This simplifies the design of agents interactions by

only defining a set of strict communication and action rules, such that each agent behavior is specified according to its relative position in the system. At the top of the hierarchy, a single agent is elected by consensus to be the *leader* of the whole formation. This leader is in charge of mission management and goals assignment inside the formation, within the long-term command loop. The agents which are not leader are considered as *subordinates*. Every subordinate is at the same hierarchical level, under the leader agent. They are not in charge of mission management and mission planning. Once the leader has established and communicated a mission plan, each subordinate extracts its partial plan. It remains responsible for a local command and control, by planning and executing a sequence of actions that matches its assigned goals (see §3), within the short term loop.

2.3 Local Agent Architecture

2.3.1 System Components

At the local level, each agent is defined by means of system components and behaviors. In addition to classical spacecraft system (AOCS, FDIR, Engine control), the set of components encompasses functions dedicated to its own behavior as well as activities related to the global formation.

- *A control-command executive* which is in charge of action execution according to the computed plan. This executive can trigger replanning actions when the drift between action prevision and real execution becomes too much important;
- *A knowledge base* to store facts and beliefs relative to current environment and other agents.
- *One or more planning components* to generate mission or local plan;
- *A communication applicative* in order to ensure the message communication with other spacecraft and thus other agents.

Each architectural component is made of several sub-parts. For instance, the knowledge base contains an updatable database that store facts and beliefs, a consistency verification system that checks the information validity, and a user interface for update and consultation.

In the following, we assume that this architecture is common to leader and subordinates spacecraft. The differences are relying only on the behavior within the MAS hierarchy.

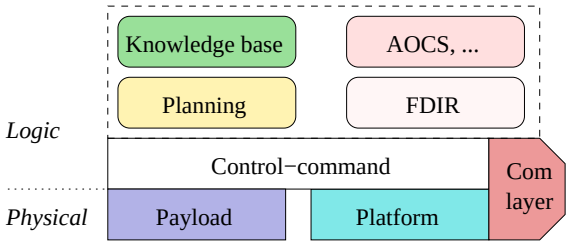

Figure 1: Autonomous agent architecture

2.3.2 Leader Agent Behavior

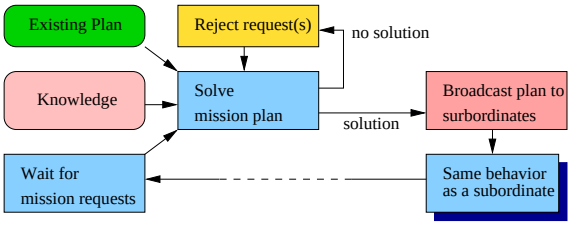

Figure 2: Leader behavior

The leader agent is in charge of the mission plan elaboration that it broadcasts to other agents. It receives goal realization requests from ground operators and shall insert them into a mission plan. For this purpose, the leader collects data from other agents to be permanently aware of the constellation state in terms of resource usage and operational situation (Attitude, Position, Time, Velocity). It decides when a new mission plan shall be computed according to the current formation situation: faulty spacecraft, goal realization requests, validity of the current plan, warning events raised by subordinate, etc.

2.3.3 Subordinate Agent Behavior

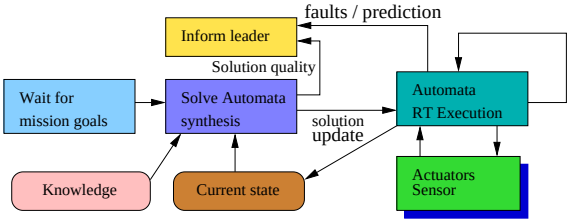

Figure 4: Subordinate behavior

The role of subordinate agents is limited to local command and control for mission plan execu-

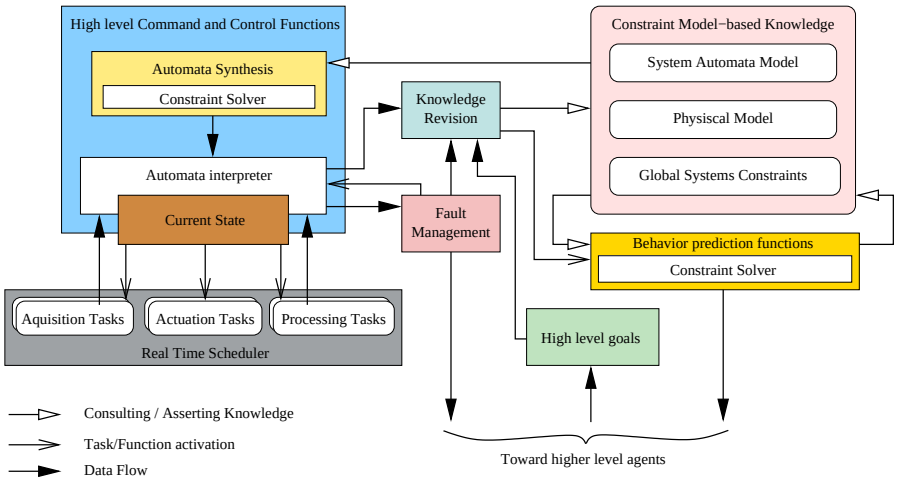

Figure 3: Local architecture

tion. Before a mission plan computing, subordinates send to the leader their own status and availability. This action is also made periodically during the system operation. Once a mission plan has been computed and communicated by the leader, each subordinate extracts the partial plan corresponding to the set of local goals it has to achieve, with associated timeline along the global mission. Each subordinate agent is in charge of achieving those goals by localy commanding and controling the spacecraft, taking into account environment parameters. This involves the dynamic replanning of on-board activities, satisfying local feasibility and safety constraints (see figure 3).

2.4 Agent Coordination

Modeling the actions of agents group needs a specification for the *collaboration*, *cooperation* and *synchronization* processes [6]. Those processes define the overall formation behavior as a consistent group of autonomous entities.

2.4.1 Collaboration & Cooperation

The collaboration considers the way several agents can jointly realize a same goal. On the contrary, the coordination considers the ways several agents or groups of agents can achieve differents goals simultaneously, sharing the same global resources. These two concepts globally bring into play the same things: goals, resources, distributed entities. They are ensured using the same approach in the plan generation, by introducing specific goal allocation and resource consumption models. These models enable the possible sharing of a global resource and the distributed realization of a goal.

In our approach, the collaboration and cooperation rely on the mission plan computing. This plan includes constraint-based models that define the planning and scheduling of inter-spacecraft operations, shared resource utilization as well as common activities [2]. Additionally, cooperation and collaboration are reinforced by distributed mechanisms relying on message communication and knowledge exchange (for instance, the leader election mechanism or the distributed plan checking).

2.4.2 Synchronisation

Inside the MAS, the synchronization is necessary to schedule correctly the sets of distributed actions, involving several spacecraft simultaneously. If we consider two platforms embedding a same distributed observation instrument (a distributed interferometer), their moves and relative positioning must be perfectly synchronized in order to realize an observation. Thus, some synchronization actions shall be included in the plan of each spacecraft.

First, constraint-based timing models are introduced at the mission plan level. Those are usualy a combination of disjunctive and precedence constraints between operations. Second, a synchronization phase is introduced in the local plan, relying on message exchange and/or other mechanism like an accurate spacecraft relative positioning system.

3 Layered Planning

In order to behave autonomously, the spacecraft formation shall be able to anticipate its evolution for short, medium term and long term with a different accuracy. On a planning point of view, meeting all these requirements is equivalent to generate a long term plan with a very fine detail level. Each spacecraft action shall be detailed (thruster ignition, mode switching...) on a long time interval, while the dynamical aspect of spacecraft formation requires a good planning and replanning reactivity to answer rapidly to new requests or unpredicted events. This requires to take into account a lot of parameters and constraints (power level, memory, attitudes, instrument distribution...) while quality plans shall be generated in short time to ensure the safety and liveness of the formation. Generating a plan with such criteria involves complex models resolution and is time and power consuming. Because of the induced complexity, obtaining a great planning reactivity level for this kind of problems is out of the scope of current planners.

Another way to broach this problem is to decompose the planning activity and to distribute it onto the available processing resources of all the spacecraft of the formation: the layered planning concept consists in separating the mission management aspects from the control command of spacecraft. Two planning levels are so distinguished, the first one is centralized onto the leader and affects the whole formation; the second is distributed onto agents and concerns each spacecraft individually. According to this decomposition, the planning relies on:

- Long term activities, that correspond to the mission management loop. It relies on mission planning and considers the evolution of spacecraft formation at a coarse grain, determining formation trajectory and scheduling mission goal;
- Short term prevision, that relies on spacecraft action scheduling and control command reactivity. This results from local planning activity, defining from time to time the actions to be led by each spacecraft to meet the final mission goals;
- Medium term anticipation, which is dispatched on both mission and local planning as a tuning of each one according to planning parameterization (planning horizon, time grain...).

Nevertheless, distributing a part of the planning activity onto spacecraft of the formation raises several problems for the centralized mission plan generation as for local plans generation. As the local spacecraft activity is no more directly taken into account in mission planning, it is necessary to upper approximate or to make use of sufficient conditions to model local behaviors.

Thus, when a spacecraft computes its local action plan, the solving is performed according to the mission plan that has been communicated by its leader. This local plan shall follow the guidelines specified by the mission plan, including goals deadlines and synchronizations with other agents.

3.1 Mission Planning

The global planning is interested in building a mission plan that satisfies as well as possible the current mission objectives. This leads to consider several problems simultaneously, for spacecraft trajectory determination, mission goals scheduling, necessary actions for standard spacecraft operation, and on-board resource management.

3.1.1 Trajectory Determination

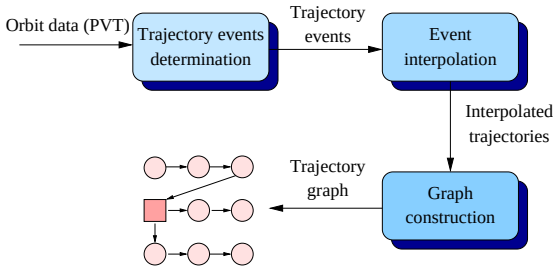

Figure 5: Trajectory graph interpolation process

The trajectory determination is based on the search of a path in a graph [3]. A graph $G(X, U)$ is built using several possible trajectories determined by an orbitography computing that can be realized on-board as on ground (see figure 5). This graph interpollation results in a set of vertices linked together by trajectory edges. Each vertex is associated to a navigation point or to a possible orbit (periodical trajectory). Such *orbit* vertices are useful to model spacecraft cyclic trajectories for earth orbiting formations as for space probes, when they execute in-orbit manoeuvres around a celestial body. Edges represent transition trajectories between navigation points and orbit vertices.

A path model ensures the correctness of spacecraft moves along trajectories by stating flow and transition constraints over the vertices of the associated graph. Then, a vertex can be reached at most once in a path, and when it is reached, the formation must leave it (Kirshhoff law):

$$\forall x \in X, \sum_{e \in \delta_x^+} v(e) = \sum_{e \in \delta_x^-} v(e) \leq 1$$

Where x is a vertex of G, δ_x^+ the set of edges incoming to x, δ_x^- is the set of edges outgoing from x, and $v(e) \in \{0, 1\}$ is the valuation of edge e.

A timing model is used for the expression of entry date d_e and exit date d_s over each vertex of the graph. If navigation points are covered in a null time ($d_e = d_s$), orbit vertices correspond to periodic moves of formation and the expression of time spent over a vertex is slightly different:

$$\forall x \in X_o, d_s(x) = d_e(x) + \overrightarrow{s} M(x) \overrightarrow{e} + kT(x)$$

Where $\overrightarrow{s}$ and $\overrightarrow{e}$ are entry and exit vectors of x, $M(x)$ is the swithching coefficient matrix of x according to incoming and outgoing edges, and T_x is the period of orbit onto x.

Thus, the trajectory model precisely determines the position and time of spacecraft in the mission plan.

3.1.2 Goal Satisfaction

The spacecraft formation mission includes the satisfaction of a list of goals for observation or navigation. Moreover, the standard spacecraft operation at a high level implies several other goals that shall also be included in the mission plan. A goal requires the use of a set of resources, available among the formation (instruments, power...), and can be realized according to specific time intervals and spacecraft positions and attitudes. Thus, a goal can be realized according to specific oportunities determined before the plan generation in accordance with the trajectory graph. Each goal oportunity is attached to a vertex of the graph.

Let O_g be the set of oportunities for goal g, then an oportunity $o \in O$ is used to realize g if the predicate $fixed(o)$ is true. Of course, this means that the time and resource constraints are verified for this oportunity. Thus, the goal g is realized when:

$$\forall g,\ realized(g) = \bigvee_{o \in O_g} fixed(o)$$

Some specific spacecraft activities require a periodic activation. For instance, earth-spacecraft communication that are made in ground station visibility intervals. Thus, a goal can be realized periodically according to its oportunities. In the same way, goals that model a spacecraft activity like thrust or attitude correction shall be realized according to spacecraft position and attitude. A conditional goal notion C is introduced that enforce a goal realization only if the trajectory and attitude of spacecraft require it:

$$\forall g, \exists v, C(g)\ \wedge\ V_{o_g} \in \mathcal{P} \Leftrightarrow realized(g)$$

Where $\mathcal{P}$ is the path and V_{o_g} is a vertex of $\mathcal{P}$ on which an oportunity for g is satisfied.

Due to goal complexity and formation distribution, the realization of several goals in the same time interval introduces a large complexity that is not tractable with actual models. A goal exclusion is so stated to ensure that several goals can't be realized in the same time interval:

$$\forall g_1, g_2,$$
$$[d(g_1), d(g_1) + D(g_1)] \cap [d(g_2), d(g_2) + D(g_2)] = \emptyset$$

Where d is the realization date function and D is the duration function. The realization date is fixed precisely by global plan, duration is determined by upper bounds according to local models.

3.1.3 Resources Consumption

At the mission planning level, both exclusive and cumulative resources are taken into account. Each goal requires a set of exclusive resources (instruments, spacecraft...) and a set of cumulative resources (power, memory, ergol...) for its realization. These amounts are upper approximated according to local models and domain knowledge. Both cumulative and exclusive resources are allocated to goals according to their needs:

$$\forall g, \forall \rho, realized(g) \Leftrightarrow \sum_{s \in F} \Gamma_\rho(s, g) = \mathcal{N}(\rho, g)$$

Where ρ is an exclusive resource, Γ is the affectation function of spacecraft resource to goals and $\mathcal{N}$ is the need function of goals for resource. For cumulative resources, a consumption shall additionaly be stated in order to ensure goal feasibility. As goals can only be realized onto vertices of the graph, this consumption is stated only for time that corresponds to a vertex location in the graph:

$$\rho_v(t) = \rho_v(d_e(v)) - \sum_g \big(\mathcal{N}(\rho, g) \times R(g, v, t)\big)$$

Where $\rho_v(t)$ is the value of resource ρ at time t on vertex v and $R(g, v, t)$ is true if goal g is realized on vertex v before time t. Introducing the bound values of resource ρ, it is possible to state the resource consumption at any time and then before and after each goal execution.

3.2 Local Command and Control

Once a global mission plan has been computed and transmitted by the leader, each spacecraft builds its own local plan to manage its actions in order to realize its part of the mission. In addition to the mission goals given by high level planning, local goals corresponding to spacecraft command and safety procedures have to be realized. Like every traditional scheduling techniques, local planning is bounded by a time horizon $0 \leq t \leq t_{max}$. Plan is enlarged in time as necessary, leading to consider the local planning activity as a task to be included in action schedule. Spacecraft resources are taken into account with high accuracy. The bound given by global plan can be optimized such that available resource are saved. Thus, each part of the mission plan can be locally optimized by each agent, saving time and resources.

3.2.1 Discrete control using constraint-based timed automata

Spacecraft command and control is characterized by different components to manage, including payloads, communications and platform devices. Each component is constrained by continuous physical laws (electric power supply, temperature, engine thrust level, ...). Already in use in the avionic domain, discrete state/transition automata can represent component modes as well as decision making with respect to continuous laws. In fact, a state σ^k is assimilated to a continuous behavior of the k^{th} component, while a transition $\delta(\sigma_i^k, \sigma_j^k)$ models an abrupt change of behavior between states σ_i^k and σ_j^k.

In our approach, a transition δ^{t+1} is triggered at the time $t+1$ when an and-composition of a set of signals S becomes true in the interval $[t, t+1[$:

$$S(t) = \bot \ \wedge \ S(t+1) = \top \Rightarrow \delta^{t+1}(\sigma_i^k, \sigma_j^k)$$

A signal can be a *command selected by the agent* or a *contingent event* raised by the agent environment. The first class of signal represents the agent commanding over the components, while the second one defines the environment stimuli on the agent. Traditional techniques adopted by engineers are based on deterministic reactive automata. In any given state, the automata can reach exactly one state. Those automata can not be adapted dynamically to environment changes and we propose to widen this approach by raising the deterministic assumption (such that multiple states can be reached from a given state):

$$\forall \sigma_i^k, \ \sigma_j,$$
$$\delta^{t+1}(\sigma_i^k, \sigma_j^k) \wedge C^k(t) = \sigma_i^k \Rightarrow C^k(t+1) = \sigma_j^k$$

where $C^k(t)$ is the component state. During the execution, the selection of a unique transition is decided by instanciating commands that optimize the path of future states according to the status of contingent events.

3.2.2 Feasibility conditions

In our model, a component state corresponds to a set of real-time tasks to perform $R(\sigma)$. As several components may share a unique processor, the total workload assigned to a unique processor must remain schedulable. Each real-time task r that belongs to a state execution, (eg. $r \in R(\sigma)$) is represented by an activation period T_r and a worst case duration time c_r. In addition to the previous constraints, we apply the Liu and Layland feasibility conditions to guarantee the schedulability of all the tasks at any time:

$$\forall t, feasible(t) \Leftrightarrow \sum_k \sum_{r \in R(C_k(t))} \frac{c_r}{T_r} \leq 1$$

Other feasibility constraints, involved by onboard cumulative resources (ergol, power supply...) may also be specified. We assume that in a given state, a component consumes a resource in a regular way, such that the overall resource consumption statement is given among the time horizon. The amount $A(t)$ of available resources can be defined as follows:

$$A(t) = A(t-1) + \sum_k a(C_k(t-1))$$

where each $a(\sigma)$ is a worst case consumption of the resource in the state σ.

3.2.3 Generating determistic automata

Those constraints differ from traditional resource scheduling constraints because the cumulated amount depends on the current state of the whole system. Therefore, search methods like task interval or edge finder can not efficiently be used to improve the solving. However, the problem remains

tracktable because the number of states and transitions is limited. The problem consists in solving the command signals and their associated time line in order to reach the state specified by higher level goals. Two kinds of search approaches can be conducted:

- Complete search with certain external events: consists in solving completely the commands to be raised and their associated timing by assuming a perfect knowledge (boolean valuation in time) of contingent events raised by the environment.
- Non-complete search with uncertain external events: consists in solving partially the commands to be raised, keeping open-disjunctions when uncertainty remain on the boolean valuation of contingent events among the time line.

4 Preliminary Results

The layered planning for spacecraft formation has been experimented in a simulation approach. The mission planning has been tested for formations of one to six spacecraft, for earth orbiting and deep space like missions, on a Sun Ultra 5 workstation. No advanced search strategy nor heuristics have been developed, the search has been led with the default global solving mechanism. For orbiting missions, a plan scheduling 150 mission goals (including periodic ones) is generated in 105 seconds. For deep space missions, planning is complexified by trajectory model but up to 40 goals (including conditional and periodic ones) are scheduled in 80 seconds onto a 20 vertices graph.

The local planning component has been less tested. For a thruster example made of two automata with a tenth of states for each one, an optimal solution including synchronization point is found in less than 4 seconds.

5 Conclusion, Further Works

We have demonstrated how relevant are layered constraint model-based planning to the success of missions that involve autonomous spacecraft. Combined to MAS architecture, this enables the solving of complex distributed planning problems and provides a more adaptive and flexible behavior to the group of spacecraft. Nevertheless, the way of modeling does depend on the level of planning and involves many assumptions, either on the planning horizon or on the approximations. The links between planning levels should be investigated deeply in order to enable multiple goal execution at the same time, maximizing the formation performance and efficiency. On the theoretical side, lattices of assumptions should be formalized in order to distribute independent solving goals. On the practical side, further experiments involving huge local planning examples shall be performed to extend preliminary results. Advanced anytime search strategies and solving heuristics shall be developed to improve the planning reactivity.

6 Acknowledgement

This work has been conducted under ESA/ESTEC contract. As a technical officer, we thank Eric Bornschlegl for his continuous support as well as his efforts for providing models of realistic target platforms.

References

[1] G. Berry, A. Bouali, X. Fornari, E. Ledinot, E. Nassor, R. de Simone, "Esterel: a formal method applied to avionic software development", *Science of Computer Programming, 36(2000) 5-25*

[2] C. Guettier, J.C. Poncet, "Automatic Planning for Autonomous Spacecrafts Constellation", *in Proceedings of the 2nd NASA Intl. Workshop on Planning and Scheduling for Space, San-Francisco, 2000*

[3] M. Gondran, M. Minoux, "Graphes et algorithmes", *Editions Eyrolles, 1995*

[4] J. Jourdan, "Concurrence et coopération de modèles multiples dans les langages de contraintes (CLP) et (CC) : Vers une méthodologie de programmation par modélisation", *PhD Thesis, Université Denis Diderot, Paris VII, 1995*

[5] N. Lynch, "Distributed Algorithms", *Morgan Kaufmann Publishers, San Mateo, CA, 1996*

[6] J.M.P. O'Hare, N.R. Jennings, "Foundations of Distributed Artificial Intelligence", *Sixth-Generation Computer Technology series, 1996*

[7] N. Muscettola, P. Pandurang Nayak, B. Pell, B.C. Williams, "Remote Agent: To boldly go where no AI system has gone before", *NASA Ames Research Center, 1998*

[8] D.S. Weld, "Recent advances in AI Planning", *Tech Report UW-CSE-98-10-01, Dept. of CSE, Univ. of Washington, 1998; also in AI Magazine 1999*

[9] M. Woodridge, N.R. Jennings, "Intelligent Agents: Theory and Practice", *Knowledge Engineering Review, October 1994*