Integration of Mission Planning and Flight Scheduling for Unmanned Aerial Vehicles

Elodie Chanthery, Magali Barbier, Jean-Loup Farges
ONERA, Toulouse Center - 2 av E. Belin - Toulouse, France

Abstract. This article presents the integration of on-line mission planning and flight scheduling for an unmanned aerial vehicle in military observation missions. Planning selects and orders the best subset of observations to be carried out and schedules the observations while accommodating time windows. The vehicle is subjected to speed, fuel supply and flight constraints in a uncertain and dynamic environment. The modeling of the problem and the algorithms implementation based on the exploration of a graph by an ordered depth-first search are presented. Mission planning and flight scheduling function is integrated in an on-board architecture based on Petri nets.

1 Introduction

Robots and unmanned vehicles have been used to perform missions in hazardous environment. Among these applications is the development of unmanned aerial vehicles (UAVs) for military observation missions in order to reduce human casualties. The autonomy of an unmanned vehicle is characterized by its level of interaction with the operator [6] : the more abstract the operator decisions are, the more autonomous the vehicle is. On-board mission planning and flight scheduling, computing a time-stamped itinerary for the vehicle in order to achieve the objectives of the mission, is one of the main challenges for intelligent UAV development in order to increase the autonomy level.
On-board functions for autonomous vehicles have to comply with three kinds of requirements : adaptability to a dynamical and, most of the time, uncertain environment, on-line problem solving and respect of real-time constraints for reactivity needs. Those functions thus should be integrated in an on-board mission management architecture including reactive capabilities. In this paper, the execution control function performs the global control of the execution of a mission. This includes the interaction with the physical system and the supervision of deliberative tasks. The main decisional component carries out the planning and scheduling function. The architecture makes it possible to react to degraded situations : the calculation of a new plan takes into account the new constraints. For this function, a long computation time only induces a degradation of the relevance of the plan with respect to the situation at the instant it becomes available but does not imply the vehicle destruction : mission planning problem is a soft real time problem.
The application context of the planning problem (including scheduling), its modeling, the planning algorithms implementation and the simulation results are provided in Section 2. Section 3 details the planning function integration in a real time architecture. The choice of intermixed planning and execution control is justified, the main concepts of the on-board architecture are presented, then the details of the planning integration are given. Section 4 concludes this work and presents a discussion of future work.

2 Planning Algorithms

2.1 Observation Mission Planning Problem

The context is a military observation mission for an autonomous aerial system in a three-dimensional, dynamic, uncertain and dangerous environment [2]. The environment of the mission includes an unsafe area where the vehicle carries out operations, that are the objectives of the mission. A mission is modeled by an origin waypoint, one or several end waypoints located outside the unsafe area and by the definition of a set of objective areas located inside the unsafe area. The information collected on an objective area may be transmitted either on line or at the exit of the area or at a transmission point, located in the range of the ground station and defined for all objectives that have their exit points outside this range. Any collected information not already transmitted is obtained on the end waypoint. Danger zones are defined by the expected localization of potential threats. The mission constraints are due to the objectives, the environment and the engine. The planning function has to select and order the best sub-set of objectives and to determine the arrival date at each waypoint, maximizing observation profits and minimizing criteria on danger, fuel consumption and durations, while meeting the mission constraints.
Since there are few advantages and for the moment few means of anticipating replanning points (role of a prediction function), planning is performed *a priori* : the solution is based on probabilistic information on the future events. All useful information for planning is supposed to be known before computation.

2.2 Previous Works

Many planning problems for vehicles are described and solved in the literature. The scheduling of observations for an airborne telescope [4] requires making choices which lead to other choices later, and contains many interacting complex constraints over both discrete and continuous variables. It is similar to our problem except that there is no danger. The planning for mobile robot navigation in unknown terrain [11] is solved by a heuristic search method that repeatedly determines a shortest path from the current robot coordinates to the goal coordinates while robot moves along the path. There is only one goal while our objective includes several sub-goals that are the observation areas. These solutions are thus not suited for observation missions. It is necessary to implement algorithms adapted not only to the overall goal of the mission, but also to the danger of the mission and which are able to take into account various mission constraints including the finite fuel supply or the flight altitude. A real-time route planning named SAS route planner [17] generates mission-adaptable routes and takes into account various mission constraints cited above, but there is no scheduling.
Our planning problem can be seen as a more complex case of the Orienteering Problem with Time Windows (OPTW) [9], which is classified as NP-hard. The next section describes how our problem is modeled as an itinerary search in a graph with a costs optimization.

2.3 Modeling

In the chosen modeling [2], the vehicle takes off from the safe area at PO and comes back at an end waypoint PF. The unsafe area is defined by entrance points $\{PEZ\}$ and exit points $\{PSZ\}$. Each objective area is defined by a set of entrance points $\{PE\}$ and a set of

exit points $\{PS\}$. With the transmission waypoints PT, these waypoints make it possible to define a directed graph as a set of nodes and a set of arcs. A node is an encapsulation of a 3D physical waypoint. The type of the node is related to the point which it represents among (PO, PEZ, PSZ, PE, PS, PT, PF). An example of a mission map including two objective areas is given Figure 1. Figure 2 illustrates the data graph for this mission where the graph is represented by subsets of nodes having the same predecessors and the same successors.

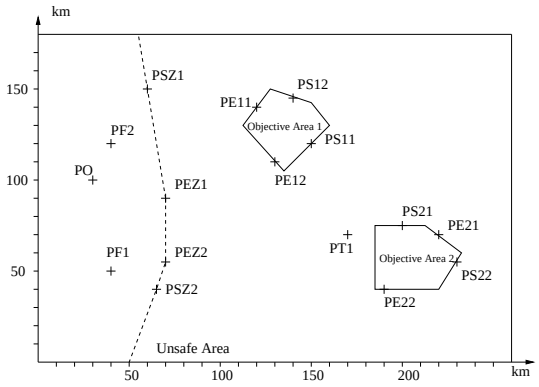

Figure 1: Mission map

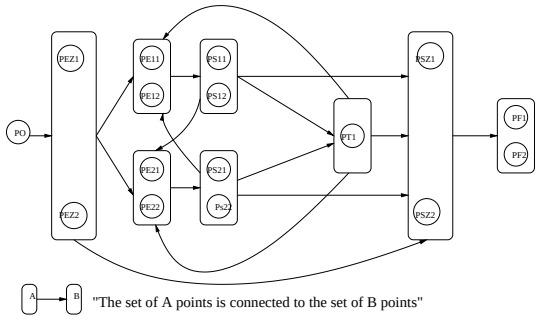

Figure 2: Data graph example

One hypothesis denoted “cycles elimination hypothesis” reduces the number of possible paths in the graph : an entrance point of an objective area which objective was already carried out can not be crossed again.

The criterion to be minimized [2] is the difference between the costs and the profits obtained on the selected path. The modeled costs are relative to the consumption, the danger and the durations. The total profit can be seen as a reward associated with information obtained on objective areas. Costs and profit take into account uncertainties. The decision variables are the flight duration of each arc and a boolean variable expressing the fact that an arc is used in the itinerary or not. The components of the vector of flight durations for an itinerary is speeds related. The consumption and duration costs may be expressed as sums on the arcs of functions of these decision variables. The major difficulty is that profits and danger costs depend on the past route and on the optimized durations between each node. They can not be calculated by arc as for a traditional OPTW.

The problem presents linear and nonlinear constraints. Linear constraints, applied to arcs, depend on the characteristics of the vehicle (minimum and maximum speeds) and on the data graph : the date of arrival on some waypoints is included in a time window out of which the observations are not valid any more. Nonlinear constraints are global and are related to danger (loss of the vehicle) and consumption limits (maximum fuel quantity).

At the mission beginning, the aim of planning is thus to find an optimal path starting by PO, finishing by an end node via the arcs of the graph. During the mission, the planning function computes an optimal path beginning at a predicted vehicle position in the graph updated according to the objectives already carried out.

2.4 Planning Algorithms Description

The itinerary search is performed on the tree of possible ways. Proposed algorithms are based on an alternative of the ordered depth-first search [15] guided by a best-first strategy. These

algorithms are different from the ones of the literature : for each developed node, the precise evaluation of the criterion requires an optimization of the speeds of the whole itinerary.
The output is a path defined by an ordered list of nodes and a vector of optimized durations between each pair of nodes. The pruning function makes it possible to cut the branches of the exploration tree and thus to reduce the number of node expansions. The planning algorithm is adapted for on-line replanning and so is able to begin at any aerial point taking into account a new environment or new objectives. For each developed node, an optimization subproblem is solved. The nonlinear danger and fuel constraints are integrated in the criterion as penalty functions. The problem becomes the optimization of a nonlinear criterion under linear constraints (speed limits and arrival dates). The problem is solved by the Frank-Wolfe algorithm [5]. This basic algorithm [2] is called Algorithm 1.
In order to improve the guidance of the exploration in the tree of the possible paths and to improve the pruning function, Algorithm 1 is modified by implementing a cost evaluation of the itinerary from an unspecified node to an end node : the problem is solved with a constant speed on the itinerary. The selected value is the optimal value for a problem without constraint and danger. The calculation takes into account the transmissions differed at the transmission waypoints. The pruning of Algorithm 1 is improved by the calculation of a first path without optimization speeds and by developing only a limited number of nodes. Speeds are then optimized for this path, given a bounded value for the criterion. The only difference between Algorithms $2a$ and $2b$ is the pruning function. Algorithm $2a$ uses the pruning function of Algorithm 1. It has the advantage of not cutting a branch of the exploration tree containing the optimal path. Search will be better guided than for Algorithm 1 because of the evaluation of the cost path to an end point. However, the tree could be not pruned enough and search in all the possible paths would take time. Algorithm $2b$ uses an evaluation of the future cost for its pruning function. The disadvantage is that if parameters are badly selected, the branch containing the optimal path may be cut. The advantage is that it makes possible to cut a higher number of branches and thus to deal with problems of big size.

2.5 Results

The algorithms are implemented using C++ language. A military mission is defined for a Mean Altitude Long Endurance vehicle. The waypoints of the mission are shown on the map Figure 1. The transmission for the objective area 1 is done at an exit point of the area. The objective area 2 is located out of the range of the ground station, a transmission point PT1 is thus defined for the transmission of information concerning this area. The operator defines the mission by giving the set of waypoints (type, coordinates, time windows), the frontier between safe and unsafe areas and information about threats.

The first test is carried out without danger and with a total observability but with time constraints on each entrance point of the unsafe area. Optimal path is shown on Figure 3. The speed adapts to time constraints. For the second test, the quantity of fuel is limited.

For the third test, a threat $DG1$ is added ofter the entrance in the unsafe area. The profit is degraded compared with the first test because of the risk to lose the vehicle. The path bypasses the danger zone (Figure 4). The last test is an on-line replanning that occurs between the entrance in the unsafe area and the first objective area in the second test because a new threat has been detected. The autonomous system bypasses the new threat (Figure 4). This test highlights the vehicle ability to autonomously adapt to a new environment.

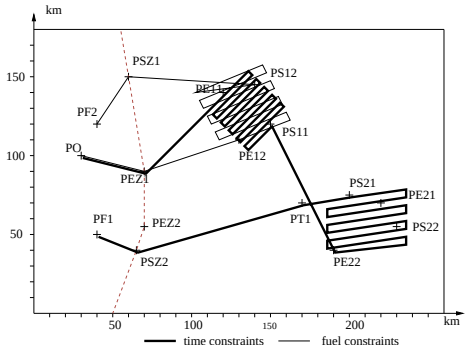

Figure 3: Results for time and fuel constraints

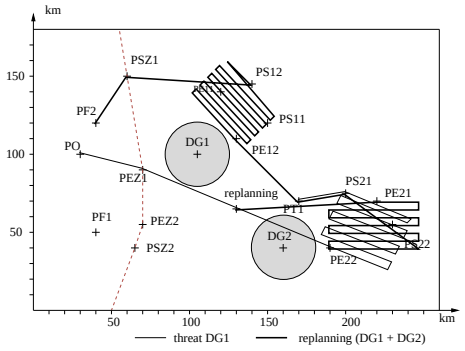

Figure 4: Results for threats and replanning

Table 1: Experiments results

case	time windows			fuel			threat $DG1$			replanning		
Algorithm	1	2a	2b	1	2a	2b	1	2a	2b	1	2a	2b
first admissible path	9	0	0	9	0	0	12	0	0	5	0	0
optimal path	173	0	0	9	0	0	70	0	0	234	8	8
algorithm end	449	170	148	192	162	23	122	60	28	286	114	105

Comparative results for the four tests, treated by the three algorithms, are shown on table 1. The values correspond to the number of developed nodes to obtain an admissible path, the optimal path and to end the algorithm. For Algorithms 1 and $2a$, the total number of developed nodes to end the algorithm corresponds to the number of developed nodes to obtain the optimality proof of the best cost path obtained. For Algorithm 2, 0 indicates that a route is obtained after the search phase at constant speed. The average of CPU time for one node expansion is about 0.2s for a Sun Microsystems Sparc Ultra 5 processor.

A first advantage of the algorithms is that a suboptimal admissible solution is obtained after a few node expansions. For Algorithm 2, the first admissible path is obtained after 0.9s maximum. For all the tests, the number of developed nodes is lower than the maximum number of developed nodes without pruning (2103). Algorithm 2 is efficient for pruning and search guidance : the total number of developed nodes is less than for Algorithm 1 and the optimal solution is obtained more quickly. As expected, Algorithm $2b$ prunes more the exploration tree than Algorithm $2a$. The optimal solution is always found by Algorithm $2b$.

3 Planning and Scheduling Integration in a Real-Time Architecture

3.1 Intermixed Planning and Execution Control Choice

Recent studies on the links between the calculation of plans and their executions show a growing number of practical applications. The context of these studies casts doubt over the assumptions which are generally adopted in planning, that are a static environment and no failure. Indeed, new events can occur and invalidate the plan in progress; the execution of the planned actions can fail and the planning must modify the current plan. The execution controller of the mission must adapt in an asynchronous way to the update of the state of the vehicle.

Various types of links between the planning and the execution control exist in the literature.

A probabilistic or conditional planning such as the Markov Decision Processes (MDP) allows to control uncertainties relating to the state of the environment and the effects of the actions. A problem of exploration planning is solved in [18]. An hybrid approach between state space factorization and decomposition is proposed. The supervisor directly applies the strategy found by planning for the current state of the system. A library of plans calculated off line or in line [8] can describe the various behaviors of a vehicle. The supervisor starts the suitable plan according to its knowledge of the environment, its objectives and its intentions. New objectives or new environment are not taken into account. The IxTeT-eXEC planning system [12] interleaves planning and execution control. Its key component is a temporal executive which interacts with the planning system. It regularly updates the plan under execution, repairs the plan in case of failure and replans upon arrival of new goals. However, the plan is not optimized and the planner cannot deliver early a partial plan.

Our approach proposes to use a structure based on the combination of the planning algorithm described in the previous section with the ProCoSA execution controller [14]. The originality of ProCoSA is that it implements tasks scheduling and event handling strategies that are described with Petri nets [13]. Petri nets are directed graphs with two kinds of nodes, the places and the transitions. The marking of the graph is given by tokens in places. In ProCoSA, an automaton, called the Petri Player, manages the update of the Petri nets marking. The data-processing components of the deliberative part of the vehicle such as a planning algorithm are implemented as on-board software programs. The real-time level of this language will depend on the application; in this work, the architecture behaves like a soft real-time system. Indeed, there is no guarantee on time response mainly because ProCoSA internally uses a socket communication protocol and events treatments depend on the Petri Player state. An original functionality of the ProCoSA Petri nets is the possibility to assign events and requests to transitions. A transition is fired if the marking validates it and if an assigned event occurs. The crossing of this transition produces the assigned requests. These requests are either events towards other transitions or messages toward a software program. Finally, software programs can produce events towards ProCoSA. The Petri nets are developed off line by means of a convivial graphic user interface.
In the Petri nets that describe the vehicle behavior during the mission, places model the possible states of the vehicle and transitions model the changes between states: at any moment, the set of marked places describes the state of the vehicle.

3.2 The On-Board Architecture Concepts

Two things make it possible to reduce the complexity of on-board architectures. In compositional methods [16], the problem is broken up into a sequence of several small problems of manageable complexity; in distributed hierarchical architectures [10], the system is described by increasingly detailed elements. The on-board architecture presented in this paper has such an implementation (Figure 5) : software programs carry out the decisional and practical tasks, a set of Petri nets hierarchically models the logic of the vehicle behavior and a supervisor manages the update of the vehicle behavior and the communication with decisional tasks.

The architecture thus includes several software programs : the planning program; the trajectory computation program calculates the vertical profile between two mission waypoints; the guidance program calculates the controls sent to the vehicle by taking account its flight

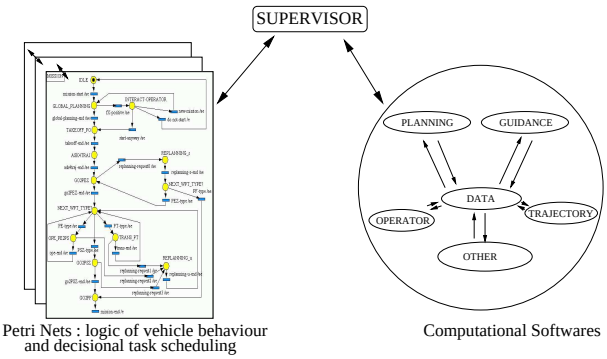

Figure 5: On-board architecture

dynamic characteristics; a data management program centralizes dynamic information necessary to all components; the operator program allows the ground operator to communicate high level decisions on the mission (abandon, new objective . . .); a situation awareness program supervises the mission and the engine state and sends alerts events for replanning or operator information. Planning program is the one described in section 2. Both guidance and data management programs are described in detail in a first implementation [1]. The other programs are to be developed.

Petri nets detail the vehicle behavior during the mission in nominal mode and in degraded situations. The typical phases of an observation mission are : takeoff, navigation to the next waypoint as long as there remains one, return to the ground station and landing. Each phase is broken up in an increasingly detailed way. The low level of this decomposition is the highest level of piloting controls communicated to the engine. The Petri nets are developed in a hierarchical way according to this decomposition. An advantage of this architecture is to allow an easy integration of new behaviors. In addition, comprehension is easier for an operator located in a ground station.

The main Petri net “Mission” (Figure 6) describes the general behavior of the vehicle from its takeoff until its landing. The marked place in this Petri net indicates the phase in which the vehicle is or the high level action in progress (TAKEOFF_PO, GO2PEZ, OPE_PE2PS, TRANS_PT, GO2PSZ, GO2PF). These places correspond to the activation of a more detailed Petri net. At the beginning of the mission, the activation of the GLOBAL_PLANNING place indicates the global planning computation. If the optimal value of the criterion is negative, it means that costs are lower than profits. The path is then accepted and sent for treatment, the mission starts. On the contrary, if the criterion is positive, the mission is either abandoned or interrupted. During the mission, the places REPLANNING_s and REPLANNING_u detail the behavior of the vehicle during replanning either in the safe area (s), or in the unsafe one (u). Places OPE_PE2PS and TRANS_PT detail the operation between PE and PS and the data transmission. Specific transitions and the associated Petri nets control the reaction to operator and degraded events (eventually from situation awareness program).

The hierarchy of the Petri nets is presented Figure 7. The Petri net “Takeoff” describes the takeoff of the vehicle at the beginning of the mission, corresponding to the PO waypoint. The Petri net “Operation” activates the Petri net corresponding to the type of the operation. During this net, PE and PS of the mission area carried out are joined. “Transmission” describes the behavior of the aerial system during the data transmission to the ground station on a PT

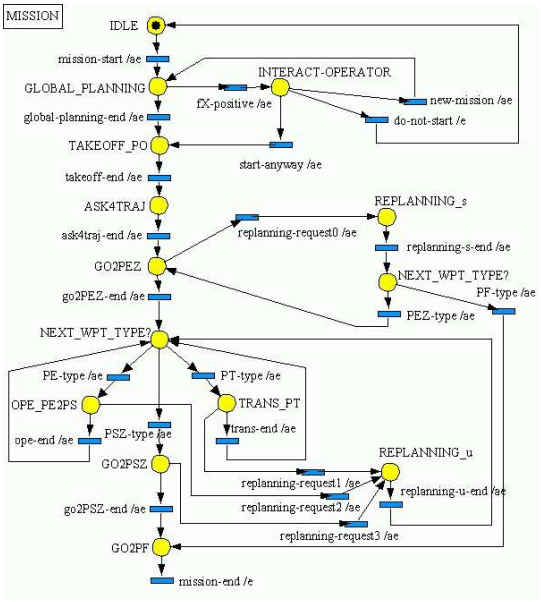

Figure 6: Mission Petri net

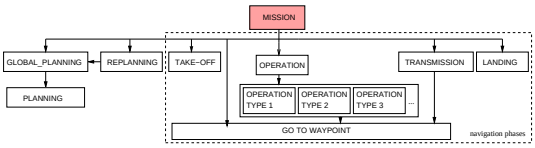

Figure 7: Hierarchy of the Petri nets

waypoint. The Petri net “Landing” describes the end of the mission on PF. The Petri net “Go to waypoint” specifies the behavior of the vehicle during the navigation phase to a specified waypoint. It is used by the net “Mission” for the waypoints of type PEZ and PSZ, by the net “Operation” and by the net “Transmission” to navigate to the transmission waypoint.

3.3 Planning Integration

Planning is integrated in the on-board architecture thanks to three Petri nets “Global_planning”, “Planning” and “Replanning” that call the planning algorithm with different parameters. Figure 8 illustrates the links between these three Petri nets.

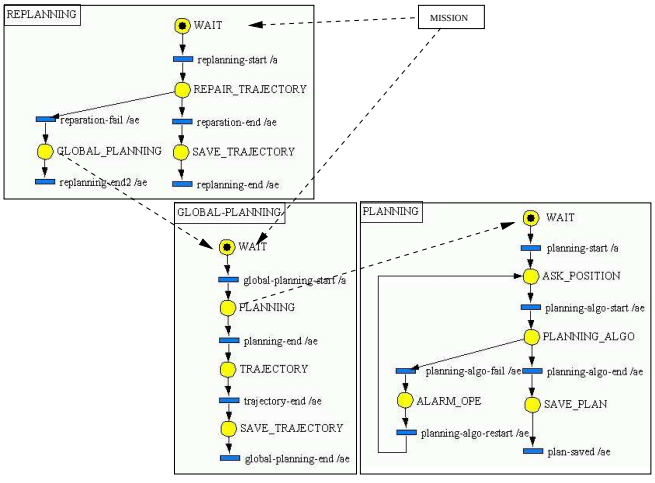

Figure 8: Planning integration in Petri nets

The Petri net “Global-planning” is run when the place GLOBAL-PLANNING of “Mis-

sion" is activated. The Petri net "Replanning" is run when one of the two places REPLANNING of "Mission" is activated.
Global planning consists in the determination of the best plan, resulting of the planning computation described in section 2, and then in the calculation of the trajectories between the waypoints of the plan. The net "Planning" asks for the current position of the vehicle, then runs the planning algorithm. If the computation fails, an alarm is sent to the operator via an Human Machine Interface. "Replanning" consists in trying to repair the trajectories. If it fails, it runs the global planning.

4 Conclusions and future work

This paper proposes the integration of on-line mission planning and flight scheduling for an unmanned aerial vehicle.
The planning problem is defined in the context of an observation mission in a three-dimensional, dynamic, uncertain and dangerous environment. The mission is modeled by a directed graph where each node is an encapsulation of a physical waypoint and edges represent feasible trajectories between waypoints. The goal of the planning is to find a path from the origin waypoint to one of the end waypoints and to schedule observations to accommodate time windows. The criterion to minimize is the sum of the consumption, danger and durations costs minus the profits obtained from the observations. The constraints are induced by the mission objectives, the environment and the autonomous aerial system. Three planning algorithms are proposed : they solve a shortest path search problem in the graph where costs are dynamic, either positive or negative, and take into account uncertainties. The tree exploration is based on an ordered depth-first search algorithm. For each node expansion, the speed is dynamically optimized for each edge of the path. Several tests show the good behavior of the algorithms with time and fuel constraints, in presence of dangers and in a replanning way.
The planning and scheduling function is integrated in a hierarchical architecture based on its combination with the ProCoSA execution controller. A user interface allows an operator to define various observation missions. The vehicle behavior is described by increasingly detailed Petri nets. The Petri net of the highest level illustrates the modeling of the mission. It exists a close link between the modeling of the mission (Section 2) and this Petri net. If an other modeling is chosen [1], the main Petri net "Mission" could be adapted. The advantage of this architecture is the hight abstraction level of Petri nets. They allow to specify every kind of modeling (space state and plan space planning). Three Petri nets are dedicated to the planning and replanning during the mission. They allow the specification of the planning function running using an intermixed approach between execution control and planning.
To conclude, the operator only defines the mission in terms of waypoints and constraints. The planning function gives a high autonomy level to the aerial vehicle. Moreover, the system is adapted to a dynamical and uncertain environment.

Future work will first concern the improvement of the tree exploration of the planning algorithm. A solution considering a factored tree where each objective is globally considered is under investigation. A second track will concern the test of the algorithm in a real time context. In that case and for a given computer technology, a trade-off has to be performed between the quality of the solution and the reactivity of the autonomous aerial system. This trade-off could be studied by limiting the number of node expansions by associating a reac-

tion time to a number of node expansions and by observing the resulting vehicle behavior. Other research could be performed on the modeling assumption, the use of a weighted criterion, the use of penalty functions. Other types of threat may also be modeled. An any-time planning approach [7] could face the problem of having a partial admissible solution at any time in urgency case. The final integration of planning and scheduling in the architecture is still in hand. Future work will finalize this implementation and simulations with the global architecture should be done in future months.

References

[1] E. Chanthery and M. Barbier, 'Functional modules for intermixed planning and execution of an observation mission', in *Proceedings of the 18th Bristol UAV Systems Conference*, (April 2003).

[2] E. Chanthery, M. Barbier, and J.L. Farges, 'Mission planning for autonomous aerial vehicles', in *IAV2004 - 5th IFAC Symposium on Intelligent Autonomous Vehicles*, (2004).

[3] G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, 1963. Princeton, NJ.

[4] J. Frank and E. Kurklu, 'Sofia's choice: Scheduling observations for an airborne observatory', in *Proceedings of the 13Th International Conference on Automated Planning & Scheduling*, (2003).

[5] M. Frank and P. Wolfe, *An Algorithm for quadratic programming*, volume 3, Naval Research Logistic Quaterly, 1956.

[6] M.A. Goodrich, D.R. Olsen, J.W. Crandall, and T.J. Palmer, 'Experiments in adjustable autonomy', in *Workshop on Autonomy Delegation and Control*, Seattle WA, (August 2001). IJCAI 2001.

[7] N. Hawes. 'Anytime planning for agent behaviour', in *Proceedings of the Twelth Workshop of the UK Planning and Scheduling Special Interest Group*, 157–166, (2001).

[8] F. Ingrand, R. Chatila, and R. Alami, 'An architecture for dependable autonomous robots', in *IARP-IEEE Workshop on Dependable Robotics*, Seoul, South Korea, (2001).

[9] M.G. Kantor and M.B. Rosenwein, 'The orienteering problem with time windows', *Journal of Operational Research Society*, **43**(6), 629–635, (1992).

[10] H. Jin Kim, R. Vidal, D. H. Shim, O. Shakernia, and S. Sastry, 'A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles', in *IEEE Conference on Decision and Control*, Orlando, (December 2001).

[11] S. Koenig and M. Likhachev, 'Improved fast replanning for robot navigation in unknown terrain', Technical report, College of Computing, Georgia Institute of Technology, (2001).

[12] S. Lemai and F. Ingrand, 'Interleaving temporal planning and execution: Ixtet-exec', in *ICAPS 03 Workshop on Plan Execution*, (2003).

[13] T. Murata, 'Petri nets : properties, analysis and applications', in *IEEE*, pp. 77(4), 541–580, (1989).

[14] DCSD ONERA, 'http://www.cert.fr/dcsd/cd/procosa'.

[15] J. Reif, 'Depth-first search is inherently sequential', *Information Processing Letters*, **20**, 229–234, (1985).

[16] B. Sinopoli, M. Micheli, G. Donato, and T.J. Koo, 'Vision based navigation for an unmanned aerial vehicle', in *IEEE International Conference on Robotics and Automation*, (2001).

[17] R.J. Szczerba, P. Galkowski, I.S. Glickstein, and N. Ternullo, 'Robust algorithm for real-time route planning', *IEEE Transactions on Aerospace and Electronics Systems*, **36**(3), 869–878, (2000).

[18] F. Teichteil-Konigsbuch and P. Fabiani, 'An hybrid probabilistic model for autonomous exploration', in *Proceedings of RFIA 2004*, (2004).