PLANNING ALGORITHMS FOR AUTONOMOUS AERIAL VEHICLE

Elodie Chanthery [*] **Magali Barbier** [*]
Jean-Loup Farges [*]

[*] *ONERA - Toulouse Center - FRANCE*
Systems Control and Flight Dynamics Department
{firstname.name}@cert.fr

Abstract: Planning function is essential for increasing the autonomy of aerial systems. This paper presents some improvements dedicated to the management of degraded events in an existing control architecture. These events may start an online replanning. In a military observation mission context, a complex modeling and several efficient algorithms for planning are proposed. Experiments highlight the efficiency of these solutions.

Keywords: autonomous vehicles, algorithms, planning, events, graphs, search methods

1. INTRODUCTION

Robots and unmanned vehicles can be used to perform missions in hazardous environments. The development of unmanned aerial vehicles (UAVs) for military observation missions is a way to reduce human casualties. The autonomy of an unmanned vehicle is characterized by its level of interaction with the operator (Goodrich *et al.*, 2001) : the more abstract the operator decisions are, the more autonomous the vehicle is. In order to assist the operators and to deal with the loss of communication, aerial systems tend to be more and more autonomous. The autonomy consists on the one hand on following the current plan and on the other hand on being able to replan on line after the occurrence of events which degrade or invalidate it.

Many planning problems for vehicles are described and solved in the literature. The scheduling of observations for an airborne telescope (Frank and Kurklu, 2003) requires making choices which lead to other choices later, and contains many interacting complex constraints over both discrete and continuous variables. The planning for mobile robot navigation in an unknown terrain (Koenig and Likhachev, 2001) 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. A real-time route planner named Sparse A^* Search (Szczerba *et al.*, 2000) generates mission-adaptable routes and takes into account various mission constraints cited above.

The planning context treated in this work is a military observation mission for an autonomous aerial system in a three-dimensional, dynamic, uncertain and dangerous environment. This type of mission can be performed by a Medium Altitude Long Endurance (MALE) UAV. The environment includes an unsafe area where the vehicle carries out operations, which are the objectives of the mission. 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. Compared to the cited literature, the aerial system is exposed to danger and there are many objectives whose order is computed by the planning function.
(Chanthery *et al.*, 2004*b*) presents the modeling of the problem and a first planning algorithm implementation. (Chanthery *et al.*, 2004*a*) describes some improvements on the algorithm, the main concepts of the on-board architecture and the details of the planning integration.
This paper presents new research works on the planning algorithms and their capacities to be used in replanning conditions. Section 2 describes the on-board architecture that controls the planning and replanning functions in nominal and degraded situations. Section 3 presents planning algorithms improvements. Tests results are shown in section 4. Section 5 concludes this work and presents future work.

2. CONTROL ARCHITECTURE

In the on-board architectures field, 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 generally adopted in planning, which are a static and deterministic environment. Indeed, new events can occur and invalidate the plan in progress; then the planning function must be executed in order to obtain a new plan. The objective of the execution controller of the mission is to adapt in an asynchronous way to the update of the state of the vehicle and of the environment. This work uses a control architecture developed around the ProCoSA execution controller (ONERA, n.d.).

2.1 Concepts

The on-board architecture is presented on Fig. 1. In accordance with compositional methods (Sinopoli *et al.*, 2001), the problem is broken up into a sequence of several sub-problems. Each sub-problem corresponds to a practical or decisional task carried on by a software program. In accordance with distributed hierarchical architectures (Kim *et al.*, 2001), the system is described by increasingly detailed elements. Those elements are Petri nets modeling the logic of the vehicle behavior. Finally, the supervision is achieved thanks to the ProCoSA Petri player that manages the update of the Petri nets and the communication through events with software programs.

The software programs included are: the planning program; the trajectory computation program calculates the vertical profile between two mission waypoints; the guidance program calculates the

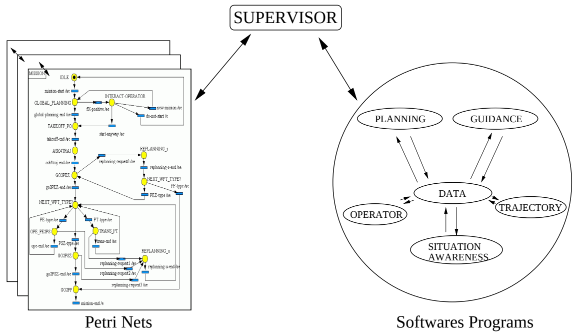

Fig. 1. On-board architecture

controls sent to the vehicle; the data management program centralizes dynamic information; the operator program allows the ground operator to communicate high level decisions; the situation awareness program supervises the mission and the engine state, it sends alert events to the supervisor or/and to the operator.

2.2 Nominal Mode

In a military observation mission, the vehicle takes off from a safe area at an origin take-off waypoint TOW and should come back to one of possible landing waypoints $\{LW\}$ also in the safe area. The unsafe area is defined by sets of entrance $\{ENU\}$ and exit $\{EXU\}$ points. Each objective area is defined by sets of entrance $\{ENO\}$ and exit $\{EXO\}$ points. These waypoints, plus specific waypoints $\{TW\}$ for data transmission, make it possible to define a directed graph that models the steps of the mission. Each phase of the mission (takeoff, navigation to the next waypoint and landing) is broken up in an increasingly detailed way. The low level of this decomposition is the highest level of guidance controls. The Petri nets are developed according to this decomposition.
The main Petri net (dashed block in Fig. 2) describes the general behavior of the vehicle from its takeoff until its landing in the nominal situation that is the following of the initial plan. The marked place in this Petri net indicates the phase in which the vehicle is or the high level action in progress (TO, GO2ENU, OPE_ENO2EXO, TRANS_TW, GO2EXU, GO2LW). These places correspond to the activation of a more detailed Petri net. At the beginning of the mission, the activation of the INIT_PLANNING place indicates the initial plan computation.

2.3 Degraded Mode

At the beginning of the mission, if costs are lower than profits, the path is accepted and the mission starts. Otherwise, the operator is informed.
In a dynamic and dangerous context, the mission achievement follows rarely the initial plan.

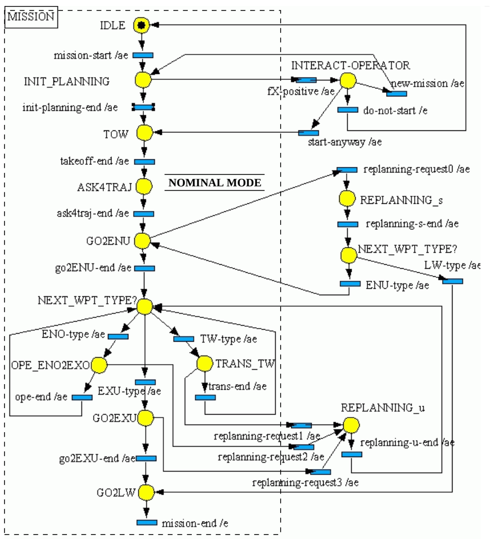

Fig. 2. Mission Petri net

New events can occur and invalidate the plan in progress. The control architecture has to take these events into account. Three types of events can occur. The payload of MALE UAVs can detect close and far threats in the environment. If the threat detection is far, a computation of a new optimal plan is possible; in case of close threat ($< 4km$), a fast admissible plan is needed. The situation awareness program analyzes data coming from aircraft sensors (speed, engine, fuel, ...) and sends alarms in case of failures. When the operator sends new objective, new time constraints, information on new threats or information on new weather data, a new plan is also required.
All these events start the replanning thanks to the transitions called "replanning-request" of the MISSION Petri net (Fig. 2).

3. PLANNING ALGORITHMS

3.1 Basic modeling and algorithm

In the chosen modeling, a directed graph $G(N, A, W)$ is defined. N is a set of nodes that represents the set of waypoints of the mission. The node type is related to the point which it represents among (TOW, ENU, EXU, ENO, EXO, TW, LW). A is the set of arcs. Each arc represents a feasible trajectory between two waypoints. $W = W_1, \ldots, W_p$ is a partition of N. The main particularity of the graph is that there exist two sorts of subsets W_i. The subsets denoted W_{i_d}, as disjunctive, can be visited only once (one input/ one output). The others, denoted W_i, may be visited as many times as wanted. Let $S(W_i)$ be a set of successors of a subset W_i. If n_1 is in W_i and n_2 is in W_j, it exists an arc from n_1 to n_2 if and only if $W_j \in S(W_i)$. The goal of the planning function is to select and order the best sub-sets W_i, to

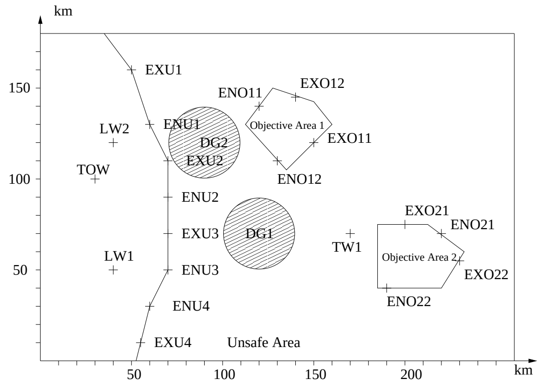

Fig. 3. Mission map

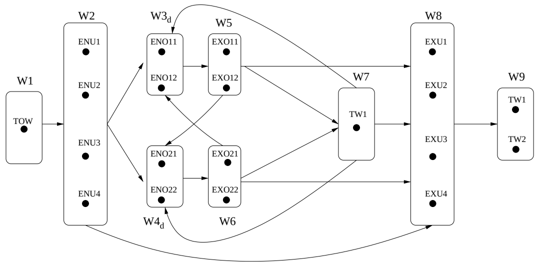

Fig. 4. Partition and successors

determine the waypoint to be used in each subset and the arrival date to this waypoint, maximizing observation profits and minimizing criteria of danger, fuel consumption and durations, while meeting the mission constraints. This planning problem can be seen as a more complex case of the Orienteering Problem with Time Windows (OPTW) (Kantor and Rosenwein, 1992), which is classified as NP-hard. It can also be seen as a more complex International Traveling Salesman Problem (ITSP) (Laporte and Nobert, 1983). The main difference is that profits and danger costs depend on the past route and on the optimized duration between each node. They can not be calculated by summing the cost of each arc as for a traditional OPTW or ITSP. An example of a mission map including two objective areas is given Fig. 3. Fig. 4 illustrates the partition W for this mission.

The criterion and the constraints of the problem are described in (Chanthery *et al.*, 2004*b*). The itinerary search is performed on the tree of possible ways. Proposed 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 for 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 planning algorithm is adapted for on-line replanning and so is able to begin at any point taking into account the new situation. For each developed node, an optimization sub-problem is solved. The problem is transformed into an optimization of a nonlinear criterion under linear constraints. It is solved by the Frank-Wolfe algorithm (Frank and Wolfe, 1956). The basic algorithm is the following:

begin
  Put IW in P
  while P is not empty
    for each v in S($\hat{u}$)
    Build path from IW to v
    Optimize speeds for each edge of the path while meeting the constraints
    Compute the path cost g from IW to v
      if v end waypoint and $g < BOUND$
        $BOUND = g$
      end
      Prune the exploration tree
    end
    Put the elements of $S(\hat{u}, \mathcal{C})$ in P
    Remove $\hat{u}$ from P
    Put $\hat{u}$ in Q
  end
end

Notations: IW point of the planning beginning, $IW = TOW$ at the mission beginning, $IW \neq TOW$ when replanning; g optimal value of the criterion from the origin node to the current node; P list of nodes not yet expanded (frontier of search); Q list of expanded nodes; $\hat{u}$ first element of P; $S(\hat{u})$ children of the node $\hat{u}$ in the data graph complying with the disjunction requirement; $BOUND$ is the current optimal value of the criterion for a path from IW to an end point, initially equal to zero. P and Q are empty at the algorithm initialization.

3.2 Modeling Improvements

Danger zones are modeled by half-balls placed on the ground and centered on the most probable location of the threat. In the first tests, the crossing of danger zones was very penalized and danger zones were avoided. This modeling was quite strict and another one (Method B) is proposed: to follow the outline of the encountered danger zone.
Let C be the center of a sphere and A and B the two points of the path which intercept the sphere (Fig. 5). The distance between A and B is denoted d. It can proved that the arc of minimal length between A and B belongs to the plane containing A, B and C. The length L to bypass the zone is thus given by $L = R.\alpha$. As $d^2 = 2R^2(1 - \cos(\alpha))$, so $L = R.\arccos(\frac{2R^2-d^2}{2R^2})$. To treat bypasses in the algorithm, node v is duplicated in v', except that there is no danger cost and a new length between u and v'. Bypasses are not treated when the current node is in the danger zone.

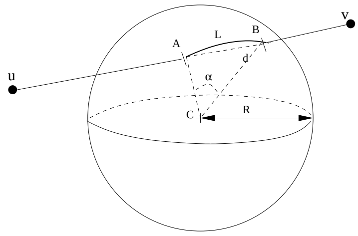

Fig. 5. Intersection of a sphere and a line

3.3 Algorithm improvements

The improvements are performed on the exploration strategy and on the pruning. Indeed, better exploration guidance and pruning will be useful to increase on-line replanning algorithm performances.
A fast first admissible path is very useful for reactive behavior of the system, and for an efficient pruning of the tree. Method F calculates a first path without optimizing speeds and by developing only a limited number of nodes. Speeds are then optimized for this path, given a bounded value for the criterion used by other algorithms.

Several cost evaluation methods are studied. The basic algorithm is modified by implementing a cost evaluation h of the itinerary from an unspecified node to an end node.
Method H_1 uses the evaluation of not yet obtained profits. Let $E_{Profits}$ be an evaluation of maximum future profits.

$$H_1 : h = -E_{Profits}$$

Method H_2 uses the evaluation of not yet obtained profits multiplied by the probability to be alive at the moment of the evaluation P_{alive} (see (Chanthery *et al.*, 2004*b*) for its computation).

$$H_2 : h = -E_{Profits} \times P_{alive}$$

Method H_3 is based on the solving of a relaxed problem: 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. Search will be better guided than for H_1 or H_2; however, the tree could be not pruned enough and search in all the possible paths would be time-consuming. H_3 is neither a maximum bound nor a minimum bound of the criterion cost.
Method H_4 uses the particular structure at two levels of the graph. The high level treats the sub-sets of objectives (W_i level) and the second level treats the choice of the nodes in the considered sub-sets. To sum up, a backward search is done on the partition W, and an heuristic value of the criterion is affected to each node embodying the

shortest distance in the danger zones, the best profit and the shortest covered distance.

Two pruning methods are used. The first one, denoted E_1, is used if h is a minimum bound of the criterion from the current node to an end node.
if $g + h > BOUND$ **then** prune node v.
The second one, denoted E_2, is used in other cases.
if $(g+h) - \gamma|g+h| > BOUND$ **then** prune node v.

In the basic algorithm, the choice of how to put the elements of $S(\hat{u}, \mathcal{C})$ in P is essential. If the arrangement is not efficient, the duration of the search may considerably increase. Four arrangements are considered. R_1 and R_2 are ordered best-first searches (Reif, 1985) guided by g and $g + h$ respectively. They may be sum up in " sort $S(\hat{u})$ in an increasing g (or $g+h$) order and put $S(\hat{u})$ on the top of P". R_3 and R_4 are g (respectively $g+h$) best-first search strategies. They may be sum up in " put $S(\hat{u})$ in P and sort P in an increasing g (or $g + h$) order".

4. EXPERIMENTS

A military mission is defined for a MALE UAV. The waypoints of the mission and the danger zones $DG1$ and $DG2$ are shown on map Fig. 3. 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 is thus defined for the transmission of information concerning this area. The operator defines the mission by giving a set of waypoints (type, coordinates and time windows), the frontier between safe and unsafe areas and information about threats. The goal of the tests is to assess the efficiency of the modeling and algorithm improvements and to compare the performances of the different algorithms. In (Bonet and Geffner, 2004), the criteria to evaluate an algorithm are the number of solved problems, the duration to obtain the solutions, the length of the solution, measured by the number of actions in the plan. These criteria are defined for classical planning problem as blocks-world, logistic, or gripper. Here, evaluation criteria are adapted to the observation mission planning problem. The performances of an algorithm are evaluated by the size of the mission (number of objective areas, number of entrance/exit waypoints per area, types of constraints), the time to obtain the first admissible path and the best path, the value of the criterion for the first admissible and the best solutions.

Four scenarios are defined.
The first one is named "nominal". The initial map of Fig. 3 does not change during the mission execution. Planning is computed from TOW to one end waypoint of the mission. The optimal path is shown on Fig. 6. Algorithms are compared with the one that has been already tested in (Chanthery *et al.*, 2004*a*) ($FH_3E_2R_2$) taken as a reference. Only one parameter is changed for each test. Table 1 sums up the results computed by a Sun Microsystems Sparc Ultra 5 processor. Bypass possibility allows to obtain better criteria but induces a multiplication per 13 and 4 respectively of the times for the obtaining of the best solution and of the algorithm ending. Nevertheless with bypass, the first admissible solution is obtained as quickly as without bypass and is of better quality. Method E_1 associated with H_1 and H_2 is as efficient as the basic method in terms of criterion value and of obtaining times of the first admissible solution or of the best one. Moreover, these combinations reduce the algorithm execution times by 2. The association of E_1 with H_4 degrades the computation times and the first solution quality. For all algorithms using method F, first admissible path has a good quality and is obtained in less than 11 s. When it is not used, computation times are degraded. Best-first search methods, particularly R_4, end the computation more quickly (reduction of the computation time by 4 for R_4).

The three other missions assess replannings on the $FH_3E_2R_2$ and $FBH_3E_2R_2$ algorithms. Replanning 1 is performed after $TW1$: an event of new threat is sent (Fig. 6). Without bypassing,

Table 1. Experiments results: CPU times in seconds, g criterion value

Algorithm	1^{st} admissible		best path		end
	time	g	time	g	time
$FH_3E_2R_2$	2	-57862	2	-57862	60
$FBH_3E_2R_2$	2	-57872	27	-57873	269
$H_3E_2R_2$	11	-57862	11	-57862	57
$FH_1E_1R_2$	2	-57862	2	-57862	32
$FH_2E_1R_2$	2	-57862	2	-57862	32
$FH_4E_1R_2$	7	-57806	50	-57862	69
$FH_3E_2R_1$	2	-57862	2	-57862	60
$FH_3E_2R_3$	2	-57862	2	-57862	21
$FH_3E_2R_4$	2	-57862	2	-57862	15

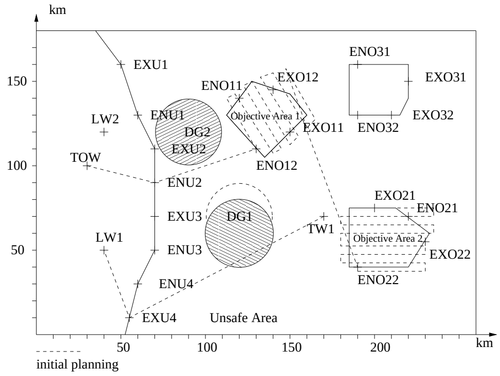

Fig. 6. Replanning maps

replanning result is [$TW1$, $EXU3$, $LW1$]. Replanning takes 0.3 s and the solution is given in 0.05 s. With bypassing, replanning result is [$TW1$, $EXU2'$, $LW2$]. Replanning takes 0.2 s and the solution is given in 0.05 s. The difference between criteria with and without bypassing is 179976, that represents about 18% of the system price.

Replanning 2 and 3 are performed after $EXO11$. For replanning 2, a new objective is added in the mission (Fig. 6). Solutions without and with bypassing are [$EXO11$, $ENO31$, $EXO32$, $ENO21$, $EXO21$, $TW1$, $EXU4$, $LW1$] and [$\ldots$, $TW1$, $EXU2'$, $LW2$] respectively. The gain on cost is 10. These solutions are given in 1.5 s and 1.6 s. End times of algorithms are 29.3 s and 51.7 s. Replanning 3 is an event of failure in the system. The first admissible solution [$EXO11$, $EXU3$, $LW1$] is given in 4.2 s without bypassing. Algorithm with bypassing finds the same solution in 7 s. Tests of replanning stress on the importance of bypassing during online computations: indeed the mission is more constrained concerning possible waypoints and bypasses represent a significant gain on the cost criterion.

5. CONCLUSION

This paper presents some significant advances on planning algorithms to be integrated in a control architecture of an autonomous aerial vehicle. Events, sent by the situation awareness program, the operator program or by the components of the payload, are taken into account by the control architecture based on the combination of the ProCoSA supervisor and the planning function. The planning function has a complex modeling that takes into account the bypassing of threats, time windows and hard constraints of fuel. Planning algorithms have been developed: they solve a shortest path search problem in a graph where costs are dynamic, either positive or negative, and take into account uncertainties. During the search in the graph, for each node expansion, the speed is dynamically optimized for each edge of the path. Future work will concern the improvement of the H_4 method. A better pruning method is envisaged. Algorithm $FH_2E_1R_4$ would be tested first because this combination seems to be the best one. A second track will concern the test of all the algorithms on several missions and in a real-time context. Other research could be performed on the modeling assumptions. The duration of transmissions could be modeled and taken into account. Other types of threats may also be modeled. Petri nets may be used not only for specifying the behavior of the system, but also for modeling some constraints of the planning problem.

REFERENCES

Bonet, B. and H. Geffner (2004). Planning as heuristic search. In: *Proceedings of the 14th International Conference on Automated Planning and Scheduling*. Whistler, British Columbia, Canada.

Chanthery, E., M. Barbier and J.L. Farges (2004*a*). Integration of mission planning and scheduling for unmanned aerial vehicles. In: *ECAI'04 - Workshop on "Planning and Scheduling: Bridging Theory to Practice"*. Valencia, Spain.

Chanthery, E., M. Barbier and J.L. Farges (2004*b*). Mission planning for autonomous aerial vehicles. In: *IAV2004 - 5th IFAC Symposium on Intelligent Autonomous Vehicles*.

Frank, J. and E. Kurklu (2003). Sofia's choice: Scheduling observations for an airborne observatory. In: *13Th International Conference on Automated Planning & Scheduling*.

Frank, M. and P. Wolfe (1956). *An Algorithm for quadratic programming*. Vol. 3. Naval Research Logistic Quaterly.

Goodrich, M.A., D.R. Olsen, J.W. Crandall and T.J. Palmer (2001). Experiments in adjustable autonomy. In: *Workshop on Autonomy Delegation and Control*. IJCAI 2001. Seattle WA.

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

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

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

Laporte, G. and Y. Nobert (1983). Generalized traveling salesman problem through n sets of nodes: an integer programming approach. *INFOR* **21**, 61–75.

ONERA (n.d.). http://www.cert.fr/dcsd/cd/procosa.

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

Sinopoli, B., M. Micheli, G. Donato and T.J. Koo (2001). Vision based navigation for an unmanned aerial vehicle. In: *IEEE International Conference on Robotics and Automation*.

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