Chapter I
Multi-Vehicle Missions: Architecture and Algorithms for Distributed Online Planning

Johan Baltié
Sophis Technology, France

Eric Bensana
ONERA, France

Patrick Fabiani
ONERA, France

Jean-Loup Farges
ONERA, France

Stéphane Millet
Dassault Aviation, France

Philippe Morignot
Axlog Ingénierie, France

Bruno Patin
Dassault Aviation, France

Gérald Petitjean
EURODECISION, France

Gauthier Pitois
Axlog Ingénierie, France

Jean-Clair Poncet
Sophis Technology, France

ABSTRACT

This chapter deals with the issues associated with the autonomy of vehicle fleets, as well as some of the dimensions provided by an artificial intelligence (AI) solution. This presentation is developed using the example of a suppression of enemy air defense mission carried out by a group of unmanned combat air vehicles (UCAV). The environment of the mission management system (MMS) includes the theatre of operations, vehicle subsystems, and the MMS of other UCAV. An MMS architecture, organized around a database including reactive and deliberative layers, is described in detail. The deliberative layer includes a distributed mission planner developed using constraint programming and an agent framework. Experimental results demonstrate that the MMS is able, in a bounded time, to carry out missions, to activate the contingent behaviors, to decide whether to plan or not. Some research directions remain open in this application domain of AI.

INTRODUCTION

The autonomy of vehicle fleets is a major artificial intelligence (AI) challenge. Indeed, the behavior of each agent associated to each vehicle of the fleet has to be specified not only in terms of actions on the environment but also in terms of flexible group decision making. Three AI topics may be useful to achieve this objective:

- Agent architectures can help in designing the architecture of the software of each vehicle
- A multi-agent approach can address the problem of interactions between vehicles
- Automated planning can provide the basis of vehicle intelligence

The purpose of this chapter is to present the problems associated with the autonomy of vehicle fleets together with some solutions via the example of a suppression of enemy air defense (SEAD) mission carried out by a group of unmanned combat air vehicles (UCAV). The first section of the chapter is devoted to the presentation of the example problem.

The problems linked to architectural choices are addressed in the second section. Indeed, autonomous vehicles are a specific kind of agent: they have to be very reactive in order to move at high speed and they have to be intelligent in order to aim at the right goal. Research conducted in the AI and robotic domains give some architectural answers to these challenges.

The capability of an agent to plan for itself and for other agents is also a key component of agent intelligence. In the third section, the planning problems arising in multi-vehicle management and their distribution across the agents are presented. The methods used for treating those problems are detailed.

The fourth section gives some experimental results obtained on the example problem. The behavior of the planning module is illustrated and the complete multi-vehicle mission management system (MMS) is tested with a simulation tool. Finally some conclusions are presented along with directions for future research.

EXAMPLE PROBLEM

The example problem proposed in this chapter is the development of a MMS for a group of UCAV. The environment of the group includes a safe area and a dangerous area, no flying zones, a command and control (C2) center, the terrain, some threats, and some targets. This environment is dynamic: threats and targets may be discovered during the course of the mission. The dynamic flight constraints of the UCAV are considered. Moreover, some subsystems interact with the MMS. The functions of those subsystems are location, flight management, communication, self-defense, sensor management, and weapon management. A mission plan is defined before take-off, and the aims of the MMS are to follow the mission plan, to ensure safety, to ensure survivability, and to ensure the success of the mission. Some requirements are deduced from those aims: the plan must be applied, disruptive events must be detected and analyzed, reactive actions must be carried out and, if needed, the mission plan must be recomputed online.

The example problem is more complex than the sum of every single agent mission management problem. The set of actions that can be performed by a group of UCAV is larger than the one for a single UCAV. For instance, the group can split: some UCAV fly to a convenient place to perform detection, identification, and localization of targets and other UCAV fly to another place to perform the strike itself. After the action the group can merge. Military pilots in SEAD missions apply this type of action rules.

Mission Description

The considered mission is an air to ground mission which can combine SEAD, suppression of targets (STRIKE), and battle damage assessment (BDA) operations. Take-off, landing, and refuelling are not considered. The considered UCAV is a subsonic stealth aircraft with an internal weapon bay. Figure 1 illustrates a typical mission.

The mission can be split in a sequence of phases:

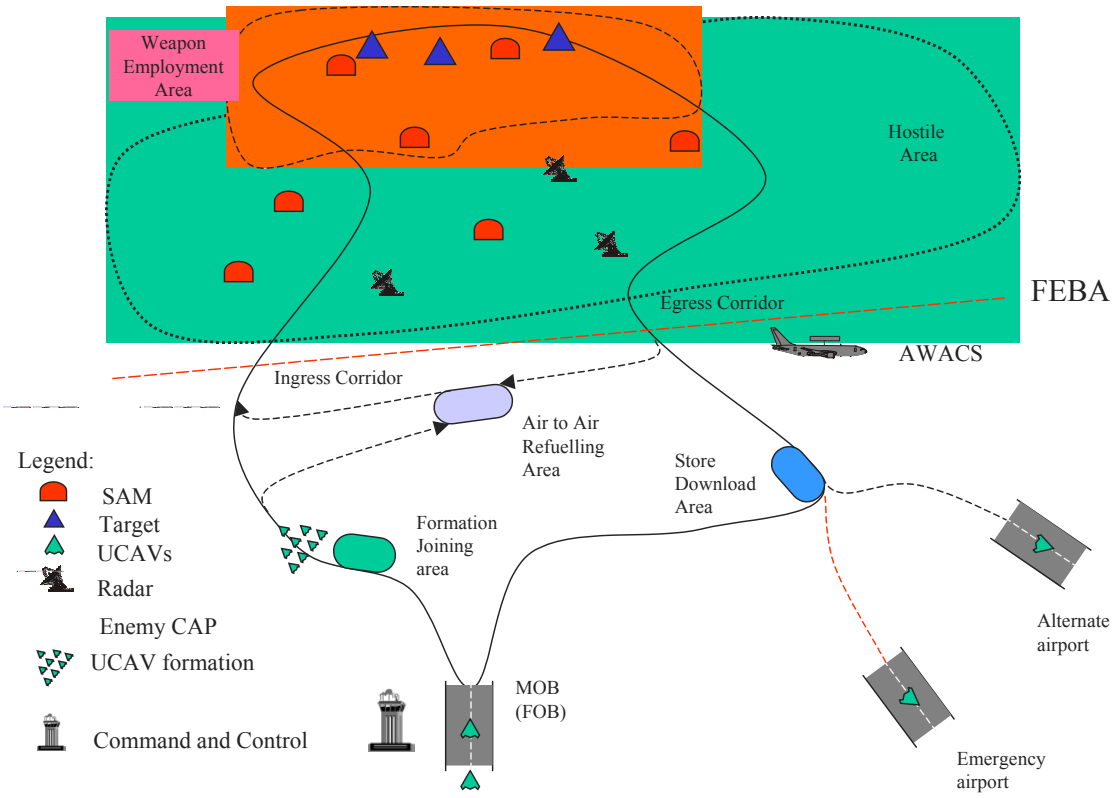
- **Domestics in:** transit to forward edge of battle area (FEBA)
- **FEBA crossing:** cross the FEBA. This area can be very protected.
- **Tactics in:** transit from FEBA to target area
- **Attack:** localize, identify, acquire, destroy the mission targets
- **Tactics out:** transit from target area to FEBA
- **FEBA crossing:** cross the FEBA. This area can be very protected.
- **Domestics out:** transit from FEBA to end of mission point

All of these phases can be described in term of constraints. For example, which equipment is usable, or in which condition a piece of equipment can be used to fire on a target.

UCAV System Description

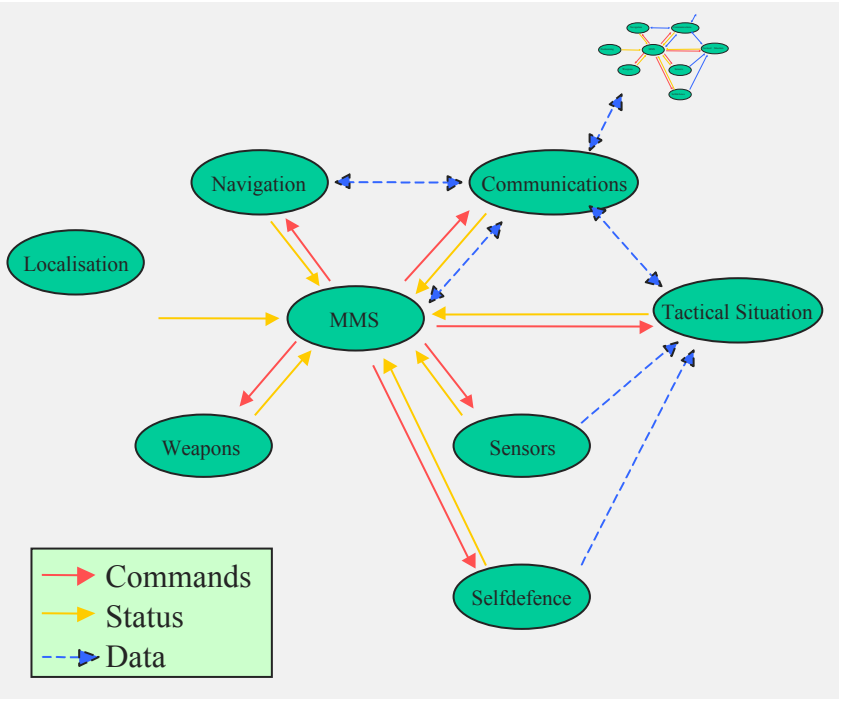
Figure 2 illustrates the UCAV system and the exchanges between the different system modules.

Figure 1. Illustration of a typical mission; the environment includes surface to air missiles (SAM), a main operating base (MOB), and an airborne warning and control system (AWACS)



- The navigation module is responsible for applying the navigation plans. A navigation plan is a sequence of waypoints.
- The localization module is responsible for computing the current position of the UCAV from different sources (global positioning system, or GPS, inertial, numeric terrain).
- The sensor module is responsible for managing the onboard sensors (synthetic aperture radar, or SAR, electro optical).
- The weapon module is responsible for managing the onboard weapons.
- The communication module is responsible for managing the datalinks:
  - **Intra formation datalink** (short distance, bidirectional, stealth)
  - **Low bandwidth datalink** (long distance, bidirectional)
  - **High bandwidth datalink** (long distance, images upload)
- The self-defence function is responsible for managing the following pieces of equipment:
  - **Missile approach warner**, in charge of detecting incoming missiles
  - **Radar warning receiver**, in charge of detection and identifying ground to air threats
  - **Active electronic counter-measure**, in charge of jamming ground to air threats
- The tactical situation module is in charge of elaborating tracks on ground to air threats with data coming from internal Radar Warning Receiver or other aircraft.
- The MMS is in charge of managing the overall mission at two levels: vehicle and group levels. At vehicle level, it is responsible for:
  - Elaborating the current vehicle mission status and tactical situation with the information pushed by all the other modules,
  - Executing the current mission plan,
  - Executing emergency actions when

Figure 2. The UCAV system



needed (reaction to threats, for example).

- At group level (all the UCAV participating in the mission), it is responsible for:
  - Elaborating the group mission status and tactical situation and
  - Creating a new mission plan from the old one, the current status, and tactical situation when the old one is no longer applicable.

ARCHITECTURAL CHOICES

Existing architectures for autonomous vehicles may be purely reactive or may have a deliberative layer. Those architectures may integrate mission management systems.

Pure Reactive Architectures

The idea beyond this kind of architecture is that mobile agents do not need online problem solving algorithms and can rely solely on modules quickly processing signals and logical conditions. The modules may then be organized in an undetermined number of asynchronous layers, the lowest layer being in direct connection with sensors and actuators and each layer being controlled and parameterized by the next upper layer. Brooks (1986) has formalized a reactive architecture: the subsumption architecture where each layer is generalized by the next upper layer. The wiring of the interconnection between modules is predefined and the actuation of a module is either internal or made by the reception of a message from another module. Exchanges between layers are performed using special kind of generic functions: inhibitor and suppressor. An inhibitor inhibits output messages and a suppressor suppresses the usual input message and provides a replacement.

A reactive architecture can be built using the model that flexible planning and scheduling with contingencies is performed on the ground and that vehicle autonomy is ensured by a conditional executive, a resource manager and a model-based mode identification and reconfiguration system. Washington, Golden, Bresina, Smith, Anderson, and Smith (1999) derive this reactive vehicle architecture from the Remote Agent (Muscettola, Nayak, Pell, & Williams, 1998) reactive and deliberative architecture. The plan commands are sent to the vehicle real time control system, with results coming back via state monitors into the mode identification and reconfiguration system. This system infers the state of the system from the monitored information and updates the state for the conditional executive. If commands fail or schedule constraints are violated, the conditional executive tries to recover using contingency plans. Reactive architectures are not able to support problem solving but react quickly to events. They are proposed for planetary exploration missions.

Another approach is based on formalisms close to the decision trees: universal plan (Schoppers, 1995) or teleo-reactive trees (Benson, 1995). The instantaneous behavior of the agent is defined through a tree of tests. The tested variables may be given by sensors or internal values. Each test leads to two other tests (if-yes and if-not) and leaves of the tree are quick atomic actions. The tree is permanently tested from root to leaves by a looping procedure. Modal logic can be integrated in this formalism in order to take into account the temporal extent of actions and differences between the agent's expectation and reality (Schoppers, 1995). Trees can also modify themselves by learning (Benson, 1995). The advantage of this approach is to allow a fluid behavior of the vehicle. However, even if it is theoretically possible, the integration of deliberative features in this approach seems practically difficult.

Reactive and Deliberative Architectures

Deliberative architectures are fully based on problem solving and usually react slowly. They are not used for vehicle mission management.

Practical intelligence of a vehicle consists in a mix of reactive and deliberative behaviors. The architecture includes two layers: the reactive layer that interacts with the environment using sensors and actuators and the deliberative layer that includes reasoning modules. For instance, the Entropy Reduction Engine architecture of Bresina and Drummond (1990) includes a reactor that provides reactive behavior in an environment. In this architecture, the deliberative layer provides plans with the help of two modules: the projector and the reductor:

1. The projector explores possible futures and provides advice about appropriate behaviors to the reactor.
2. The reductor reasons about behavioral constraints and provides search control advice to the projector.

Gat (1992) proposed a third layer that supervises the modules of the reactive and deliberative layers. This third layer activates the modules, receiving their termination messages. It has to react quickly taking into account the reasons of a module execution failure. This supervision layer may be implemented in a flexible way as a Petri net player playing Petri nets described in a hierarchical way (Verfaille and Fabiani, 2000). Application of that type of architecture has been proposed for planetary exploration rovers.

One important requirement for architectures having both reactive and deliberative layers, is the independence of the two layers in terms of execution of modules. Indeed, even with high environmental pressure, the architecture must avoid stopping the execution of a reactive module to start a deliberative module. For this reason, reactive and deliberative layers may each present an independent supervision functionality (Hayes-Roth, Pfleger, Lalanda, & Morignot, 1995). For instance, an independent controller for each layer (deliberative, reactive) dictates which behavior (cognitive behavior at the cognitive layer, physical behavior at the physical layer) to activate now, given information present on a common structure of each layer. One of these cognitive behaviors may be "planning," in our case. One of these reactive behaviors may be, "take an evasive maneuver," in our case. This architecture has been demonstrated on a mobile robot performing secretarial tasks in a laboratory.

Then comes the discussion: what is a plan? Investigators have first considered a plan as a simplified program (in the sense of a program in a programming language), for example, a sequence of programming instructions. Executing the plan means executing the instructions. That is the view of a pure computer scientist. Other authors have considered plans as a communication medium (Agre & Chapman, 1987): a plan is considered as data in a formal language that provides information to other agents about the intentions (short term, long term) of the agent. For other authors, plans are considered as intended behaviors (Hayes-Roth et al., 1995) capturing the idea that an action in a plan is not a simple one-shot action but a more continuous and meaningful activity of the agent named behavior. Hence, the sequence of action descriptions in the plan corresponds, when executed, to a sequence of behaviors, not of actions.

We follow this latter approach, with the notable difference that behaviors, in our architecture, always unfold in the same order. Hence there is no need for a declarative approach for expressing (encoding) behaviors (at the reactive layer and at the deliberative layer). In other words, the logical unfolding of behaviors (at the reactive and at the deliberative layers) is simply hard-wired through a simple sequential graph.

An intermediate layer, between the reactive and deliberative layers, is proposed by Alami, Chatila, Fleury, Ghallab, and Ingrand (1998). In the resulting architecture, the deliberative layer includes a planning module and a supervision module that activates one plan in function of a stack of agent's intentions. The reactive layer is reduced

to modules controlling sensors and actuators. The intermediate layer is called the execution control layer. It chooses and parameterizes the adequate module of the reactive layer with respect to the tasks of the plan given by the deliberative layer. It also elaborates, from the information returned by the reactive layer, execution reports. Finally, it transmits the reports to the supervision module of the decisional layer.

The Remote Agent architecture (Muscettola et al., 1998) integrates not only a deliberative layer and an execution control layer but also a model-based mode identification and reconfiguration. The deliberative layer includes a mission manager that formulates short-term planning problems for a planner and scheduler on the basis of a long-range mission profile. The executive achieves robustness in plan execution by exploiting the plan flexibility, for example, by being able to choose execution time within specified windows or being able to select different task decompositions for a high-level activity. The mode identification tracks the most likely vehicle states by identifying states whose models are consistent with sensed values and commands sent. The mode reconfiguration uses the vehicle model to find a command sequence that establishes or restores desired functionality.

Level of Detail in the Plan

Architectures with an advanced execution control layer raise the issue of plan abstraction and contingencies. Indeed, the more flexibility is given to the execution control layer, the less precise is the planned action and the more unpredictable are its effects. The basic formalism for abstraction of action is the Hierarchical Task Network (HTN). This formalism states that elementary actions are derived from abstract actions that are the purpose of the plan. This decomposition is made through non-elementary actions called methods. A method includes an identifier, arguments and a list of pairs (conditions–list of actions, elementary action or method). Activating a method means to choose a pair whose conditions are compatible with the current state and to activate the actions of its list. It is possible to specify if the actions have to be activated in sequence or if the actions may be activated in parallel. This formalism decomposes the most abstract action in an "and/or" tree. Bresina and Washington (2001) use such decompositions in their Contingent Rover Language to provide a flexible plan to a conditional utility-based executive.

Architecture for the Example Problem

The vehicle architecture for the example problem is already defined. In that architecture a part of the reactive layer is outside the scope of the mission management problem. However the mission management system is subject to time pressure and cannot include only a deliberative layer. The design principles for the example problem are:

1. Online planning shall be activated only when the current plan is invalidated by the current situation.
2. The reactive layer shall not only execute the plan but also handle emergency situations.
3. Sensor inaccuracy is managed through planning of behavioral procedures for inaccuracy reduction.

Figure 3 presents the mission management system architecture for the example problem. It includes reactive and deliberative layers. This architecture is organized using an underlying database that stores information about the vehicle, the other vehicles, the mission, and the environment.

The reactive layer includes the following modules: "observe and compare," "pre-empt," and "act."

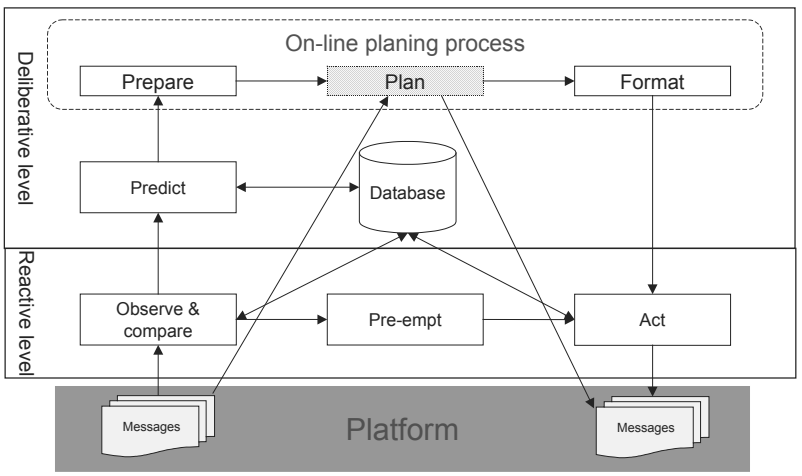
The "observe and compare" module receives the messages from the platform, marshals them

into structured data, updates the database information, and performs tests about the short-term situation. It tests the possible disruptive events such as unexpected exposure to a threat, presence in a No Fly Zone (NFZ), loss of communication, and failure of other vehicle subsystems using some simple threshold-based functions applied on limited time horizon and geographical range. It includes a function of inhibitor, in the sense of Brooks (1986), of the messages in direction to "pre-empt."

The "pre-empt" module is activated once a threshold has been passed over. According to the nature and to the emergency level of this event, it determines candidate contingent behaviors and selects the one that has to be applied in the system to secure aircraft situation as soon as possible. This could necessitate to quickly check the overall UCAV situation and, if necessary, to compute some parameters to determine the contingent behavior and activate it. Finally, "Preempt" determines the sequence of unit actions generated by the behavior. Four behaviors may be activated by the module:

- The behavior for new radar threat detection implements updates of the radar list and radar locations, active electronic counter measure under specified conditions, maneuver for avoidance or information gathering under other specified conditions.
- The behavior for approaching missile detection implements unconditional use of active electronic counter measures, chaff and flare decoy, and evasive maneuvers.
- The behavior for loss of communication between neighboring vehicles consists in commanding the altitude of the vehicles to different predetermined flight levels in order to ensure the absence of collision between them. The secured altitude slots are attributed to each aircraft at plan generation time, ensuring their uniqueness for each aircraft of a formation.
- The behavior for NFZ violation avoidance is implemented in two parts. First, a modification of the current trajectory is computed, attempting to avoid incoming NFZ, going round it by the shortest way. If this fails or if the situation evaluation reports that the aircraft suddenly appears to be inside a NFZ, the solution consists in a fast trajectory computing that will attempt to exit the NFZ by crossing the closest frontier point.

Figure 3. The mission management system architecture



All behaviors have a date parameter, a time-out, and a homing waypoint parameter. They are updated each time the "preempt" component is activated. This update can be null according to the current aircraft situation, meaning that the system can return to nominal plan execution. Otherwise, the contingency ends once its associated timeout has been passed out or when homing point is reached. All behaviors, except loss of communication, have a Boolean parameter indicating whether the global path can be modified. Finally, the behavior for new radar threat detection has an additional parameter indicating the origin waypoint.

Four types of information represent the behaviors:

- The warning processing information indicates what should be done in terms of knowledge management.
- The system management information indicates what should be done in terms of auto-protection actions.
- The flight plan modification information indicates what should be done in terms of navigation actions.
- The end of contingency information indicates conditions for terminating the behavior.

The "act" module carries out the unit actions of the plan or of the "pre-empt" module and determines whether an action is correctly performed or not. Hence, the module analyzes the status of the subsystems stored in the database. When actions of the plan and from "pre-empt" are conflicting, it always gives the priority to preemption actions in order to ensure platform safety. This mechanism corresponds to a suppressor function in the sense of Brooks. Finally the module sends messages to the platform subsystems.

The deliberative layer includes the following modules: "predict," "prepare," "plan," and "format."

The "predict" module assesses the feasibility of the on going plan and decides whether to compute a new plan or not. Probabilities of UCAV survival and of target killing are updated and compared to a threshold. The possibilities of fulfilling time constraints at some waypoints and of having enough fuel to finish the mission are checked. It should be noted that the "predict" component faces a dilemma. On the one hand, deciding to re plan all the time leads the agent to an erratic behavior, always starting the beginning of new unrelated plans, and therefore not leading to any goal at all (too often replanning). On the other hand, deciding to plan too infrequently leads the agent to follow unusable plans, since the behavior of the agent does not adapt to what actually happens in the environment (too infrequent replanning). The solution we propose for this "predict" component is a medium term on the previous spectrum, by using variables representing states of the agent. When the mean of these variables is above some threshold, then the replanning decision is taken (and replanning occurs). This solution is not satisfactory in principle, since it does not solve the problem of the continuity of the behavior of the agent over successive replanning activities. But at least it provides a practical and simple (but not elegant) solution, even if these variables and thresholds need careful tuning for a realistic replanning frequency to be adopted. Moreover, the occurrence of a replanning request while replanning has to be managed. This management is performed using priorities on replanning reasons. If the priority of the reason of the present replanning request is lower than the one of the on going replanning, the request is ignored. Otherwise, the on going replanning activity is stopped and the replanning is started with a context including the present request.

The "prepare" module gathers and generates data for the "plan" module. The data includes:

- Participating vehicles.
- Available resources for each vehicle.

- Environment including threats, targets, and NFZ.
- A graph including possible paths for acquisition, attack, and return to base. This graph is built in two steps. An initial graph is deduced from the initial mission plan by associating mission waypoints to graph nodes and transitions between these waypoints to graph edges. Nodes and edges are tagged according to their strategic properties for acquisition, shooting, and so forth. The second step is done each time the component is activated. It consists in the generation of different alternative paths for each strategic action, including Return To Base. These paths are generated using a potential field based algorithm in which threats and NFZ are associated to repulsing potential while targets and base airport are associated with attractive potentials. Motion planning is fully explained in the next section.
- Time intervals at waypoints.

The "format" module refines the macro actions of the plan into sequences of unit actions. For instance the macro action "launch bomb 1 on target 101 at time t" is refined in the sequence of unit actions: "select resource type bomb 1 at time t-d" then "initialize selected resource with target 101 features at time t-e" then "ask to C2 go/no go at time t-f" then "if C2 answer is go fire bomb 1 at time t-g."

The "plan" module directly receives some messages and is also able to send directly other messages. This feature of the architecture allows multi-UCAV distributed planning.

The MMS is activated every 0.1 seconds. For most of the cycles only three modules are activated: "observe and compare," "predict," and "act." For those cycles, the result of the analysis of messages from the platform by "observe and compare" and "predict" indicates that no preemptive behavior has to be activated and no new plan has to be computed. The "act" module continues carrying on actions of the current plan.

For cycles where "observe and compare" indicates that a pre-emptive behavior has to be activated, the "preempt" module is additionally activated. This module may remain activated for several consecutive cycles until the behavior is finished. Meanwhile the "act" module applies the actions issued from the behavior. For instance, if the disruptive event is the detection of a missile launch, the module remains activated until the end of the escape maneuver.

For cycles during which "predict" indicates that a new plan has to be computed, the "prepare" module is additionally activated. In the same cycle, the "plan" module is activated in the background. Several cycles afterwards, the "plan" module provides a plan and activates the "format" module just for a single cycle.

Special attention is given to the way the computation time constraints are taken into account: different priorities are assigned to the input messages, only a bounded number of messages are processed each cycle and long processing is performed over several cycles or in a separate thread. The conjunction of those techniques ensures that a computation time bound for a time cycle of the mission management system can be predetermined.

PLANNING

Specifying the Mission and the Planning Problem

A specific grammar can be used to specify the mission of a multi-vehicle system (Brumitt & Stentz, 1998). The basic elements of the grammar are the vehicles, the goals, and the motions of combination of vehicles towards combination of goals. Those elements can be combined using "or," "and," and "and then" operators. The specification given by

the user is translated in an expression allowing planning. The basic elements of this expression can be treated as:

- A shortest path problem
- A traveling salesman problem
- A path selection problem
- A multiple traveling salesman problem

However most of the time missions have to be specified through formats that are specific to the application. For the example problem, the following information has to be given to the planner each time a new plan is requested:

- The date for the plan to start, because problems are not stationary. For instance, the FEBA shall be crossed only in specific time windows.
- The UCAV to be considered. Indeed the vehicles are basic elements of mission specification. However additional information must also be provided. For instance, the UCAV predicted state in terms of geometry and resources at the date for the plan to start. Moreover, because a permanent communication network cannot be established and because enemies may destroy the UCAV, three classes of vehicles are to be distinguished: (i) the UCAV involved in the communication cluster in which the planning is carried on, (ii) the UCAV not involved in the communication cluster but presenting a plan assumption, and (iii) the UCAV not involved in the communication cluster and assumed out of order.
- The goals to be considered. Indeed goals are basic elements of the mission specification. Goals are described through action prototypes for target destruction. A prototype includes the resources to be used by the UCAV at specified places in the space and at specified times in order to have a specified probability of destroying the target. Figure 4 gives an example of an action prototype as a xml text. This prototype specifies that the target 1168 is a SAM site at a given latitude, longitude, and altitude. The target can be attacked either by delivering one bomb of type 1 with a UCAV with global positioning system capability or by delivering three bombs of type 1 with one or several UCAV. In the first case , the probability of destruction is 0.95. In the second case it is 0.80. Moreover the attack can be conducted either through node 1072 and edge 1069 with a heading of 270 degrees or through node 1074 and edge 1070 with a heading of 0 degrees.
- The navigation data including relevant characteristics of the environment. This includes a graph and the description of the airspace parts threatened by the different threats.

Motion Planning

Planning for multi-vehicle missions obviously includes motion planning for each vehicle of the fleet. The result of this motion planning is a sequence of "go to" actions to different points in the space. Different requirements for the path can be found in the literature (Allo, Guettier, Legendre, Poncet, & Strady-Lecubin, 2002; Kuwata, 2003; Szczerba, Galkowski, Glickstein, & Ternullo, 2000):

- The angle between two successive legs at each point of the path must be below a given value.
- The angle between two successive legs at each point of the path must be feasible through a sequence of three circle arcs that do not conflict with forbidden area.
- The distance between two consecutive points must be larger than a value that depends on the distance to the origin and on the distance to destination.
- The distance between two consecutive points must be large enough not to have any

Figure 4. Action prototype

```
<target id="1168" type="SAM_SITE" lat="45.241111" lon="8.810983" alt="0.">
        <mode bomb1="1" bomb2="0" pr="95" gps="1"></mode>
        <mode bomb1="3" bomb2="0" pr="80" gps="0"></mode>
        <node id="1072" type="weapon" axis="270.000000" edge="1069"></node>
        <node id="1074" type="weapon" axis="0.000000" edge="1070"></node>
</target>
```

interference from the sequence of circle arcs at the upstream point with the sequence at the downstream point.

- The length of the path must be below a given value.
- The heading of the last leg is given.
- The total combustible consumption must be lower than the combustible available at the beginning of the plan.

Two kinds of approaches for motion planning exist: the covering of the space by cells (Szczerba et al., 2000) and the construction of a graph (Allo et al., 2002; Fabiani et al., 2005; Kuwata, 2003). If the space is covered by cells, the solution of the problem can be provided by an A* algorithm. For each cell a local cost and an optimal cost to go under relaxing assumptions are computed. The A* algorithm uses the optimal cost to go as a heuristic function. If a graph is built, it can be done either by the Voronoi method (Fabiani et al., 2005) or by the visibility method (Kuwata, 2003). Then the path on the graph is found either by a Dijkstra (1959) algorithm, a modified Dijkstra algorithm (Kuwata, 2003), or constraint programming (Allo et al., 2002; Strady-Lécubin & Poncet, 2003).

For the example problem, the graph given to the planning module is complemented by the visibility method in order to provide paths around the threatened areas. Each node j of the graph presents a set of upstream edges (In_j) and a set of downstream edges (Out_j). Then the motion-planning component of the problem is formulated using constraint programming. Some variables are associated to UCAV and nodes; the fact that the UCAV i passes by the node j ($P_{i,j}$), the arrival time of the vehicle at the node, the altitude, speed, fuel, mass, logarithm of survival probability, and the fuel spent for the heading change at the node. Other variables are associated to vehicles and edges; the fact that the UCAV i flies the edge k ($U_{i,k}$), the flight time, altitude variation, fuel consumption, logarithm of conditional survival probability, and exposure time to the different threats of the UCAV on the edge. Constraints describe navigation possibilities. They include:

- A vehicle mass definition constraint,
- Constraints ensuring consistency between passing at a node and flying edges,
- Constraints modeling the feasibility of the heading change at a node and its consequence on the speed and consumption,
- Constraints associated to the initial state of each vehicle,
- Constraints ensuring consistency between the values of variables at the upstream node, the downstream node, and on the edge when a vehicle flies the edge, and
- Constraints associated to the terminal nodes of the graph.

For instance, consistency between passing at node and flying edges is given by:

$$\text{If } In_j \neq \varnothing, \sum_{k \in In_j} U_{i,k} = P_{i,j} \quad (1)$$

$$\text{If } Out_j \neq \varnothing, \sum_{k \in Out_j} U_{i,k} = P_{i,j} \quad (2)$$

If the heading change from edge k to edge l at not j is not possible:

$$U_{i,k} + U_{i,l} \leq 1 \quad (3)$$

Otherwise:

$$U_{i,k} + U_{i,l} = 2 \Rightarrow CS_{SC} \quad (4)$$

Where CS_{SC} is a set of constraints on speed and consumption of UCAV *i* at node *j*.
For the node *jinit(i)* where the UCAV *i* is at the date the plan begins:

$$P_{i,jinit(\mathrm{i})}=1 \quad (5)$$

For each edge *k*:

$$U_{i,k} = 1 \Rightarrow CS_{UD} \quad (6)$$

Where CS_{UD} is a set of constraints between upstream and downstream nodes of edge *k* on arrival time, altitude, speed, fuel, logarithm of survival probability involving flight time, altitude variation, fuel consumption, logarithm of conditional survival probability, and exposure time on the edge *k*.

Finally the path ends in a node without downstream links:

$$\sum_{j / Out_j = \varnothing} P_{i,j} = 1 \quad (7)$$

The constraint satisfaction problem derived includes linear, nonlinear, disjunctive, and conditional constraints. The variable domains are finite.

Consistent Group Motion

Multi-vehicle missions may also imply a consistent group motion: vehicles must remain close one to the other but must not collide. To take into account this kind of requirement, flocking has been adapted and implemented on fleets of actual robots by Hayes and Dormiani-Tabatabaei (2002). Moreover Olfati-Saber and Murray (2003) studied the adaptation of flocking algorithms in case of communication constraints and obstacle avoidance. Usually, flocking is not used at the deliberative layer but at the reactive layer. However, the use of a flocking criterion in a Dynamic Programming successive approximation scheme allows the computation of coordinated motion for two unmanned vehicles (Corre, 2003). For missions where vehicles move in an encumbered environment, the absence of collision and deadlock is ensured through planning. In Alami, Ingrand, and Qutub (1997), the edges of a navigation graph are resources to be shared among the mobile robots. The planning is then performed incrementally in a decentralized way: each time a robot receives a new goal, it builds a new plan compatible with the activities of its current plan and with the resource usage of the activities of the plans of the other robots. In other missions, the absence of collision is not handled by planning but is treated locally in a reactive way, for instance by stopping the robot with lower priority (Brumitt & Stentz, 1998).

A different kind of consistent group motion can be found with exploration missions: the vehicles must share the area to be explored. In the work of Walkers, Kudenko, and Strens (2004), agents are in charge of mapping a two dimensional area. Each agent builds heuristically a value function for each cell of the space. The path is obtained by selecting the neighboring cell with the best value. Several ways of computing the value are proposed. The computation of the value may take into account, when the communication between agents is possible, the path planned and the area already explored by other agents.

For the example problem, consistent group motion is not handled by the planner. The planner may produce a plan where some UCAV fly on the same edge at the same time. In that case the reactive layer gathers those UCAV in a formation and specific flying rules are applied.

Performing Tasks

In a group of vehicles, the vehicles may have different capabilities of action and observation. In that context, one vehicle may perform an action for another one and the exchange of information between vehicles may be explicitly planned. Some actions using resources of the sub-systems of the vehicle are in the plan besides the "go to" actions. Tavares and Campos (2004) give an example of such a situation: two helicopters have to travel a given path as quickly as possible but there is, somewhere on the path, one threat that can be perfectly observed and destroyed only by one of them. The plan is obtained using a team Partially Observed Makov Decision Process where the path is divided in steps and each agent has, in addition to original actions and observations, communication actions and observations. However, solving the simple problem exactly, where the path is already given, is not tractable and the heuristics used are sometimes not optimal.

Multi-vehicle mission planning may also include task selection and assignment. Several approaches for task assignment exist:

- Using rules on the state of the vehicles such as assigning the task to the nearest vehicle with the capability to perform it (Beard, McLain, Goodrich, & Anderson, 2002).
- Making requests to compute the cost of a path to a task destination or of a visit of several task locations for vehicles and using the results of the requests for optimizing the decision (Brumitt & Stentz, 1998).
- Computing for each vehicle the travel time associated to permutations of combinations of feasible, in terms of resources and tasks, selecting a limited number of permutations for each vehicle. Finally optimizing a criterion based on time of achievement of the last task, the permutation travel time and the total waiting time. This optimization selects one permutation per vehicle and set the instants of task achievement (Kuwata, 2003). This approach is based on the use of libraries of linear programming with mixed variables.

Finally, vehicles assigned to the same task may have to be synchronized. Kuwata (2003) proposes to solve synchronization and assignment in a single problem. Another solution consists in generating, for each vehicle, a set of good paths to the point it has to perform the task and then to find among those sets the best feasible instant for task achievement (Beard et al., 2002). The assignment or synchronization is performed by computing, for each vehicle, a set of initial paths and selecting one path in each set. Then the selected path is refined in a last step by optimizing, taking into account the given synchronization instants (Beard et al., 2002; Kuwata, 2003).

The example problem involves different aspects: selection of goals, selection of an action mode for each goal, assignment of vehicles and their resources to each selected action mode, and scheduling attacks. Constraint programming and integer programming are powerful approaches for integrating those different aspects. Indeed, this approach is efficient even for planning problems expressed in a propositional representation (van Beek & Chen, 1999; Vossen, Ball, Lotem, & Nau, 1999). Moreover, the use of constraint programming allows a formulation consistent with the one for motion planning. For additional details about the encoding of a planning problem as a constraint programming formulation, see the chapter entitled Extending Classical Planning for Time: Research Trends in Optimal and Suboptimal Temporal Planning in this book. Variables indicating quantities of different resources (bombs, missiles, and so forth) of the UCAV when passing the node are associated to vehicles and nodes. For nodes j attached to the attack of a target, additional variables are the participation of the UCAV to the attack ($I_{i,j}$) and the quantities of different resource k used by the UCAV at the node ($Q_{k,i,j}$). Finally variables are

associated to goal achievement. For each target, the variables are:

- The fact that the target *o* is attacked (A_o),
- The fact that it is attacked at a given node ($A_{o,j}$),
- The fact that it is attacked at a given node in a given mode ($A_{o,j,m}$),
- The attack time,
- The gross efficiency of the attack (Eff_o),
- The gross efficiency of the attack at a given node ($Eff_{o,j}$),
- The logarithm of the efficiency discounted by survival probability of participant UCAV.

Constraints describe conditions for goal achievement. The constraints associated to goal achievement include constraints associated to the attack of the targets and constraints defining some global criteria such as global efficiency (*Eff*) and global survivability. An important aspect is the link between the motion planning part of the model and the goal achievement part of the model. This link is ensured by constraints of the type:

$$I_{i,j} \le P_{i,j} \tag{8}$$

$$I_{i,j} \le \sum_k Q_{k,i,j} \tag{9}$$

$$Q_{k,i,j} \le K_{k,i} I_{i,j} \tag{10}$$

$$\sum_i Q_{k,i,j} \ge R_{k,m} A_{o,j,m} \tag{11}$$

$$\sum_m A_{o,j,m} = A_{o,j} \tag{12}$$

$$\sum_j A_{o,j} = A_o \tag{13}$$

Equations (8) and (9) indicate that preconditions for an UCAV to participate to an attack at a node are to pass by that node and to have some resource to use at that node. Equation (10) bounds the resources usage by zero if the UCAV does not participate and by the available quantity, $K_{k,i}$, otherwise. Equation (11) indicates that the precondition for the group to attack a target in a given mode is to have at least the resource amount requested for that mode, $R_{k,m}$. Equations (12) and (13) indicate that a single mode and a single node are selected for the attack of the target. Similar equations ensure the link between attack times and passage times at points.

A simplified equation for efficiency of the attack of target *o* at node *j* is:

$$Eff_{o,j} = \sum_m p_{o,m} A_{o,j,m} \tag{14}$$

where $p_{o,m}$ is the probability of destruction of the target o when attacked in mode m. This probability is directly taken from the action prototype. Then the efficiency of the attack of target *o* is given by:

$$Eff_o = \sum_j Eff_{o,j} \tag{15}$$

Finally the global efficiency is defined as the sum of the probability of destruction of the different targets:

$$Eff = \sum_o Eff_o \tag{16}$$

The modeling of the example problem using a constraint programming approach entails the definition of a large number of variables. But the graph of variables and constraints associated to a multi-vehicle mission presents a star structure: the variables associated to goal achievement are connected by constraints to the variables associated to the different vehicles but there is no direct constraint between the variables of two vehicles. This structure allows the decomposition of the initial problem into a problem associated to each vehicle and a goal achievement problem. The decomposition of the initial problem has the advantage of permitting the use of the computing resources of all vehicles and it corresponds to the approaches of Brumitt and Stentz (1998),

Beard et al. (2002), and Kuwata (2003). Several techniques are available to perform the distributed search of a solution. Among those techniques, it is possible to make the distinction between methods that start by solving the goal achievement problem and then making requests to motion planning solvers (Brumitt & Stentz, 1998) and methods that start by creating several proposals with the motion planning solvers (Beard et al., 2002; Kuwata, 2003). For the example problem, it seems that the first method would be blind starting the search mainly for setting attack time variables and performance evaluation variables. Thus the problem is solved with a three step technique where: (1) sets of solutions are searched for the problems associated to each vehicle, (2) a coordination problem, including the goal achievement problem and the selection of one solution per set, is solved, and (3) the solution is refined for each vehicle.

Important technical points are the assumptions made at step 1 to compute feasible paths between the points associated to goal achievement and the assumptions made for the representation of the different schedule on the same path with their impact on fuel consumption and threat exposure. It has been decided to compute feasible paths with a simplified consumption model and to provide the impact of the timing on other variables by linear relations.

Implementation

Implementation of distributed planning algorithms for multi-vehicle missions is often performed using custom software built without off-the-shelf packages. However, Kuwata (2003) uses CPLEX as a mixed variable linear programming engine for solving the assignment problem.

For the example problem the implementation relies on JADE (Bellifemine, Poggi, & Rimassa, 1999), a FIPA agent compliant framework. This framework allows implementing the three steps of the distributed planning algorithm as two behaviors of the planner agent. The first behavior is activated by the "to prepare" module, sends a request for proposals to the different UCAV, waits for the reception of the proposals and solves the coordination problem. The second behavior is activated by a request for proposals. It computes and sends the set of solutions, waits for the selected solution, and refines it. The CHOCO (Laburthe, 2000) tree searching constraint solver is used for enumerating the set of solutions, solving the coordination problem and refining the selected solution. For additional information about constraint satisfaction techniques used by current constraint solvers, see the chapter entitled Principles of Constraint Processing in this book.

Computation Time Constraints

The control of the time to find a plan may also be an important implementation issue for some multi-vehicle online mission planning. One approach to avoid performing long plan computation on line is to provide a policy, computed off-line, that for each possible state gives almost immediately an action. In that case the performance is grounded on the introduction of expert knowledge or on dynamic programming or on reinforcement learning (Harmon & Harmon, 1996). However, the drawbacks of this approach are the limitation of the amount of memory for policy storage, the time taken off-line to compute the policy and the difficulty of assessing the quality of the learned policy.

Another approach for controlling the computation time is to plan with different levels of detail, either in time or in state space. For instance the receding horizon approach, used by Kuwata and How (2004) for UAV, consists in computing at each sample time a very detailed plan from the end of the current sample time to the end of a short-term horizon. The computation takes into account a rough plan evaluation from the end of

this horizon to the end of the planning problem. Damiani, Verfaillie, and Charmeau (2004) propose for observation satellites an approach with a planning horizon not defined a priori. Indeed, breadth first tree search algorithms, like forward dynamic programming, can be interrupted at anytime and a rough evaluation can be added to the criterion of the leaves of the last fully developed level. For a classical planning domain, the propositions describing the state are classified in different classes of abstraction (Knoblock, 1991). The planning is performed starting with the most abstract class of propositions and then treating progressively less abstract classes. This approach may reduce computation time, but leads sometime to degraded performances (Smith & Peot, 1992). Zilberstein (1993) proposes algorithms that progressively improve the solution and studies the problem of splitting a given computation time in a sequence of such algorithms using their performance profiles. Zilberstein and Russel (1993) demonstrate the practical application of computation time splitting on a sequence made of an image analyzer feeding a path planner. At each less abstract level, the terrain description is refined by using a grid with twice number of discrete values in each dimension.

For some missions with ground vehicles and stationary environment it could be possible to stop the vehicles while the plan is computed. For the example problem this is not possible. The plan has to be ready at the time at which it should begin. The time spent by the solver to solve the sub-problems is controlled by the selection of variables to be assigned, by the selection of values for those variables, and by interruption of the tree search. For instance, an obvious solution of the coordination problem is to attack no targets. The selection of variables and values for the solver is performed in order to find this obvious solution first and then to improve it. Moreover before beginning to plan, the remaining time is split and assigned to each step of the solution.

EXPERIMENTAL RESULTS

Experimental results are given in the context of SEAD and STRIKE scenarios.

Behavior of the Planning Module

In order to assess the performance of the planning function, tests are conducted on a single Sun Blade 1500 computer. The planning module is requested to provide a plan for four UCAV with four targets on a graph with 50 nodes and 57 edges. The searches for a set of solutions by each vehicle correspond to a constraint satisfaction problem of about 300 variables and 500 constraints. This step leads to the generation of 46 solutions for the first UCAV after 1.0 seconds, 46 solutions for the second UCAV after 1.6 seconds, 46 solutions for the third UCAV after 1.7 seconds, and 138 solutions for the fourth UCAV in 2.4 seconds. It is interesting to note that the number of solutions found in the first step corresponds to the size of the domains of four of the variables of the second step. The second step of the method induces an optimization problem with about 1500 variables and 4200 constraints. The first constraint propagation made by CHOCO reduces the number of free variables to about 920. The following figure illustrates the anytime behavior of the second step of the planning method. A solution with no target attacked is found in about 1.0 seconds. Then as solution time increases, the efficiency of the solution, as defined by equation (16), is improved and more targets are attacked. Finally, the third step conducts to optimization problems of about 300 variables and 200 constraints for each UCAV. The refinement of the solution is given in 0.05 seconds for the first UCAV, 0.09 seconds for the second UCAV, 0.14 seconds for the third UCAV and 0.24 seconds for the fourth UCAV. Note that the solution times for the first and third step of the method are over-estimated because the tests are conducted using a single computer. Finally, it can be observed that if optimality is not required

the computation time for the coordination step can be reduced to few seconds.

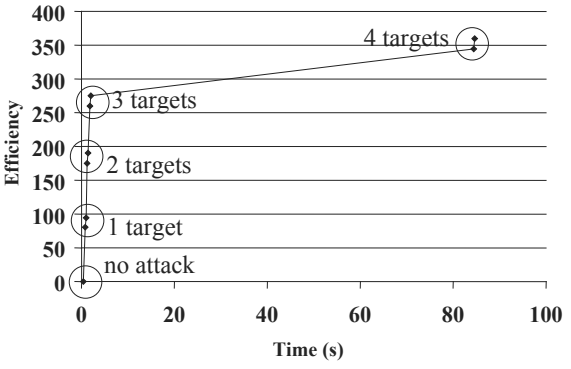
FULL SIMULATION RESULTS

Those results demonstrate the capacity of a fleet of vehicles integrating a distributed planning module within a reactive and deliberative architecture. The mission management system is able to carry on nominal missions as specified, to activate the contingent behaviors on disruptive events, to decide whether or not to plan and, if necessary, to plan and to run in a bounded time. Moreover, datalink requirements for the functions of the mission management system and performance of distributed planning are assessed.

The MMS is evaluated on different scenarios, starting from a very simple situation, one aircraft, one target, and moving on to more complex situations with multiple aircraft, threats, and targets. The following figure illustrates the nominal simplest scenario. The white line is the navigation of the aircraft. The red area is the FEBA, the green ones no-fly-zones, and the yellow one is a threat detection range. The MMS is able to execute this mission correctly and in time.

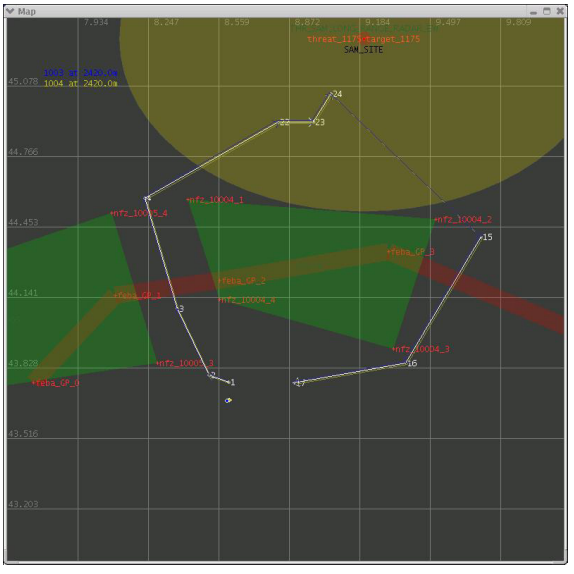
On this scenario, events are injected:

Figure 5. Performance profile for the second step of the planning algorithm. The efficiency is the sum of the destruction probabilities, expressed in %, of the targets attacked.



- **Discovery of a new threat on the path of the UCAV:** The reaction of the UCAV is first a modification of the flight path, induced by the reactive layer, in order to localize the threat. Then the deliberative layer plans online the mission in order to respect its timings. The new plan is applied and executed successfully. It is correct with respect to mission goals.
- **C2 sends new threat data when a jammer is available:** The reaction of the UCAV is no reaction at all as it considers it will be able to cross it using its jammer for self-defense.
- **C2 sends new threat data when a jammer isn't available:** The reaction of the UCAV is replanning in order to maximize survivability. The new plan is applied and executed successfully. It is correct with respect to mission goals. The new flight path goes around the new threat.
- **C2 sends data about two new threats while a jammer is available:** The reaction of the UCAV is replanning in order to maximize survivability. The new plan is applied and executed successfully. It is correct with respect to mission goals.

Figure 6. The nominal simplest scenario



During a test with two aircraft, the nominal scenario is executed correctly and in time. Events are also injected in this scenario.

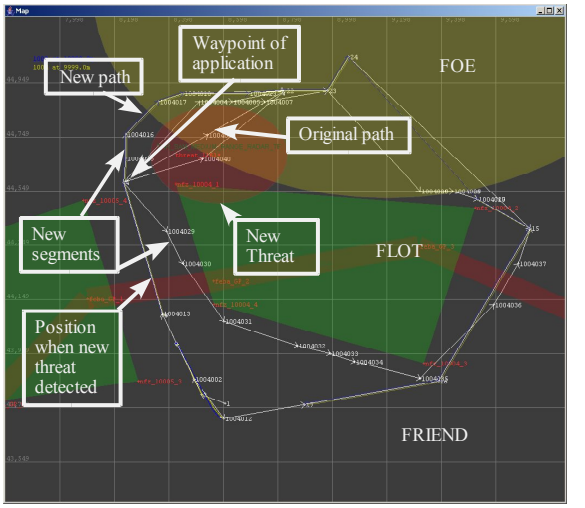
- **Loss of a sensor on the UCAV responsible for target acquisition:** The reaction of the group is mission replanning, the acquisition task is reallocated to the other UCAV. The new plan is applied and executed correctly.
- **C2 sends new threat data when a jammer is available:** The reaction of the group is no reaction at all as it considers it will be able to go through it using its jammer for self-defense.
- **C2 sends new threat data when a jammer isn't available:** The reaction of the group is replanning in order to maximize survivability. The new plan is applied and executed successfully. It is correct with respect to mission goals. The new flight path goes around the new threat.
- **C2 sends new mission target: the reaction of the group is replanning in order to be able to treat all mission targets.** The new plan is applied and executed successfully. It is correct with respect to the new mission goals. Each UCAV is responsible for a target.

The conjunction of those experimental results demonstrates the feasibility of the proposed mission management system.

CONCLUSION

The proposed architecture and distributed planning method for multi-vehicle missions contribute to the increase of vehicle intelligence and autonomy. Indeed, with the integration of online planning, disruptive events in absence of human intervention do not lead necessarily to aborting the mission. However, it is important to note that

Figure 7. Illustration of change of trajectory resulting from a new plan



the architecture proposed for the example problem addresses a specific class of multi-vehicle missions. For this class the plan exists at the beginning of the mission and provides actions up to the end of the mission. In a context where there is a large uncertainty about the ending conditions of the mission or where there are systematically a large difference between the situation expected at planning time and the actual situation, other architectures based on a more systematic activation of the planning module are more suited.

Some research directions remain for this application domain of AI:

- Study of the link between the geometry and the actions.
- Study of the method for taking into account uncertainty about the state of the vehicles and the environment as distance from current date increases. Indeed, in the real world sensing is not perfect and actions may have uncertain outcomes not only in terms of rewards, as considered in the example, but also in terms of future state. The solution provided through the architecture proposed for the example is to compute a plan for the

current context using a deterministic model and to recompute it when the context changes significantly from the initial hypothesis. A more proactive solution could be obtained by using for instance probabilistic planning as proposed by Teichteil-Königsbuch and Fabiani (2006) for the mission of a search and rescue rotorcraft. The extension of this approach to multi-vehicle missions is a very promising research direction.

- Study of the efficiency of other distributed methods.
- Study of mixed initiative planning for fleets with manned and unmanned vehicles.
- Study of the sustainability of mission consistency despite the ability to compute several new plans during the mission.

ACKNOWLEDGMENT

The French "Délégation générale de l'armement," a part of the ministry of defense, has funded this work from 2003 to 2006 in the scope of the ARTEMIS project, thanks to this institution. The authors acknowledge the ARTEMIS partners for their contribution, thanks to Jean-Francois Gabard and Catherine Tessier. Finally, the authors thank Richard Washington for correcting many English mistakes.

REFERENCES

Agre, P.E., & Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial intelligence* (pp. 268-272).

Alami, R., Chatila, R., Fleury, S., Ghallab, M., & Ingrand, F. (1998). An architecture for autonomy. *International Journal of Robotic Research, 17*(4), 315-337.

Alami, R., Ingrand, F., & Qutub, S. (1997). Planning coordination and execution in multirobots environement. In *Proceedings of the 8th International Conference on Advanced Robotics* (pp. 525-530).

Allo, B., Guettier, C., Legendre, V., Poncet, J.C., & Strady-Lecubin, N. (2002). Constraint model-based planning and scheduling with multiple resources and complex collaboration schema. In *Proceedings of the 6th International Conference on Artificial intelligence Planning and Scheduling* (pp. 284-293).

Beard, R.W., McLain, T.W., Goodrich, M.A., & Anderson, E.P. (2002). Coordinated target assignment and intercept for unmanned air vehicles. *Institute of Electrical and Electronics Engineers Transactions on Robotics and Automation, 18*(6), 911-922.

Bellifemine, F., Poggi, A., & Rimassa, G. (1999). JADE–A FIPA-compliant agent framework. In *Proceedings of the Fourth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* (pp. 97-108).

Benson, S.S. (1995). *Learning action models for reactive autonomous agents.* Stanford: Stanford University.

Bresina, J.L., & Drummond, M. (1990). Integrating planning and reaction. A preliminary report. In *Proceedings of the American Association of Artificial intelligence Spring Symposium on Planning in Uncertain, Unpredictable or Changing Environments* (pp. 24-28).

Bresina, J.L., & Washington, R. (2001). Robustness via run-time adaptation of contingent plans. In *Proceedings of the American Association of Artificial intelligence Spring Symposium on Robust Autonomy* (pp. 24-30).

Brooks, R. (1986). A robust layered control system for a mobile robot. *Institute of Electrical and*

Electronics Engineers Journal of Robotics and Automation, 2(1), 14-23.

Brumitt, B.L., & Stentz, A. (1998). GRAMMPS: A generalized mission planner for multiple mobile robots in unstructured environements. In *Proceedings of the Institute of Electrical and Electronics Engineers International Conference on Robotics and Automation* (vol. 3, pp. 2396-2401).

Corre, J. (2003). *Planification distribuée sous contrainte de communication*. Master of Science dissertation, ESIGELEC, UFR des sciences de Rouen, Rouen, France.

Damiani, S., Verfaillie, G., & Charmeau, M.C. (2004). An anytime planning approach for the management of an Earth watching satellite. In *4th International Workshop on Planning and Scheduling for Space* (pp. 54-63). Darmstadt, Germany.

Dijkstra, E.W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik, 1,* 269-271.

Fabiani, P., Smith, P., Schulte, A., Ertl, C., Peeling, E., Lock, Z., et al. (2004). *Overview of candidate methods for the «autonomy for UAVs » design challenge problem*. Group for Aeronautical Research and Technology in EURope, Flight Mechanics Action Group 14 Report.

Gat, E. (1992). Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the National Conference on Artificial intelligence* (pp. 809-815).

Harmon, M.E., & Harmon, S.S. (1996). Reinforcement learning: A tutorial. Retrieved on August 14, 2007, from http://iridia.ulb.ac.be/~fvandenb/qlearning/rltutorial.pdf

Hayes, A.T., & Dormiani-Tabatabaei, P. (2002). Self-organized flocking with agent failure: Offline optimization and demonstration with real robots. In *Proceedings of the Institute of Electrical and Electronics Engineers International Conference on Robotics and Automation* (p. 4).

Hayes-Roth, B. (1993). *An architecture for adaptive intelligent systems*. Stanford University: Knowledge Systems Laboratory.

Hayes-Roth, B., Pfleger, K., Lalanda, P., & Morignot, P. (1995). A domain-specific software architecture for adaptive intelligent systems. *Institute of Electrical and Electronics Engineers Transactions on Software Engineering, 21*(4), 288-301.

Knoblock, C.A. (1991). *Automatically generation abstractions for problem solving.* Unpublished doctoral dissertation, Carnegie Mellon University, School of Computer Science.

Kuwata, Y. (2003). *Real-time trajectory design for unmanned aerial vehicles using receding horizon control.* Master of Science dissertation, Massachusetts Institute of Technology.

Kuwata, Y., & How, J.P. (2004). *Three dimentional receding horizon control for UAVs.* Paper presented at the American Institute of Aeronautics and Astronautics Guidance, Navigation, and Control Conference and Exhibit.

Laburthe, F. (2000). CHOCO: Implementing a CP kernel. In *Proceedings of the Workshop on Techniques for Implementing Constraint Programming Systems,* Singapour (pp. 71-85).

Muscettola, N., Nayak, P.P., Pell, B., & Williams, B.C. (1998). Remote agent: To boldly go where no AI system has gone before. *Artificial intelligence, 103*(1/2), 5-47.

Olfati-Saber, R., & Murray, R.M. (2003). Flocking with obstacle avoidance: Cooperation with limited information in mobile networks. In *Proceedings of the 42nd Institute of Electrical and Electronics Engineers Conference on Decision and Control* (pp. 2022-2028). Retrieved on August 14, 2007, from http://www.cds.caltech.edu/~olfati/papers/cdc03/cdc03b_ros_rmm.pdf

Schoppers, M. (1995). The use of dynamics in an intelligent controller for a space faring rescue robot. *Artificial intelligence, 73*(1/2), 175-230.

Smith, D.E., & Peot, M.A. (1992). A critical look at Knoblock's hierarchy mechanism. In *1st International Conference on Artificial intelligence Planning Systems* (pp. 307-308).

Strady-Lécubin, N., & Poncet, J.C. (2003). Mission management system high level architecture, report 4.3. MISURE/TR/4-4.3/AX/01, EUCLID RTP 15.5.

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

Tavares, A.I., & Campos, M.F.M. (2004). Balancing coordination and synchronization cost in cooperative situated multi-agent systems with imperfect communication. In *Proceedings of the 16th European Conference on Artificial intelligence* (pp. 68-73).

Teichteil-Königsbuch, F., & Fabiani, P. (2006). Autonomous search and rescue rotorcraft mission stochastic planning with generic DBNs. In M. Bramer (Ed.), *International federation for information processing* (p. 217), *Artificial intelligence in theory and practice* (pp. 483-492). Boston, MA: Springer.

van Beek, P., & Chen, X. (1999). CPlan: A constraint programming approach to planning. In *Proceedings of American Association for Artificial intelligence* (pp. 585-590).

Verfaillie, G., & Fabiani P. (2000). Planification dans l'incertain, planification en ligne. Presentation at Rencontres Nationales des Jeunes Chercheurs en Intelligence Artificielle.

Vossen, T., Ball, M., Lotem, A., & Nau, D. (1999). On the use of integer programming models in AI planning. In *Proceedings of the 16th International Joint Conference on Artificial intelligence* (pp. 304-309).

Walkers, T., Kudenko, D., & Strens, M. (2004). Algorithms for distributed exploration. In *Proceedings of the 16th European Conference on Artificial intelligence* (pp. 84-88).

Washington, R., Golden, K., Bresina, J., Smith, D.E., Anderson, C., & Smith, T. (1999). Autonomous rovers for Mars exploration. In *Proceedings of the Institute of Electrical and Electronics Engineers Aerospace Conference* (Vol. 1, pp. 237-251).

Zilberstein, S. (1993). *Operational rationality through compilation of anytime algorithms*. Unpublished doctoral dissertation, Computer Science Division, University of California at Berkeley.

Zilberstein, S., & Russel, S.J. (1993). Anytime sensing, planning and action: A practical model for robot control. In *Proceedings of the 13th International Joint Conference on Artificial intelligence,* Chambery, France (pp. 1402-1407).