3rd US-European Competition and Workshop on Micro Air Vehicle Systems (MAV07) & European Micro Air Vehicle Conference and Flight Competition (EMAV2007), 17-21 September 2007, Toulouse, France

Automated Mission Planning for a Fleet of Micro Air Vehicules

Pierre-Selim Huard* and Nicolas Barnier†

École Nationale de l'Aviation Civile, Toulouse, 31055, France

Cédric Pralet‡

Office Nationale d'Études et de Recherches Aérospatiales, Toulouse, 31055, France

Abstract. The ENAC University is using and developping the Paparazzi UAV system, which aims to provide a free and fully autonomous autopilot for a fleet of MAV, including fixed wing and rotary wing MAVs. The Paparazzi project has now succeeded in providing a fully autonomous navigation system for multiple fixed-wing aircraft system. One of the main concern of the project is the development of autonomous decision making algorithms and cooperative behaviour between the different MAV of a fleet to increase the level of autonomy of the system. We can distinguish 4 different levels of autonomy: the manual level, the augmented stability level with or without pilot through video, the flight plan level where the mission is described in terms of trajectories and waypoints, and the mission level where the mission is described in terms of goals. In this paper, we focus on the latter, which implies to solve the automated mission planning problem for a fleet of MAV. We first present a formal model for this problem inspired by the MAV06, EMAV2006 and MAV07 contests, then we propose two different approaches to solve it.The first one is based on dynamic programming and computes a policy which gives the action to take for each state, regardless of prior history and taking the uncertainties into account, whereas the second one is a real-time planning and replanning algorithm based on the A* algorithm.

I. A formal model of a mission

Automated planning[1] is a branch of Artificial Intelligence which aims to provide algorithms which can compute policies or plans to realize a set of fixed goals performed by agents such as ground robots or UAV. To modelize classic planning problems we use restricted state-transition systems (see figure 1). These systems are determinist (there is no uncertainty), finite (there is a finite number of state and action), and completely observable (there is no uncertainty on the state observation). We denote this system $\Sigma = (S, A, \gamma)$ where S is a finite set of states, A is a finite set of actions, $\gamma : S \times A \rightarrow S$ is a transition function.

A planning problem is then defined as a triple $\mathcal{P} = (\Sigma, s_0, g)$ with Σ a restricted state-transition system, s_0 the initial state, and g a set of final states. Solving the mission planning problem consists in finding a valid sequence of action which permits to go from an initial state to a final state. In our case, the final states are defined as a state where all MAVs are either at home or on ground (or out-of-order).

Definition 1 *A plan is sequence of action $\pi = (a_i)_{i\in[1..n]}$ where n is the size of the sequence. A solution to $\mathcal{P}$ is a plan $\pi(a_i)_{i\in[1..n]}$ that satisfies:*

$$s_n \in g \text{ where } s_n = \gamma(\gamma(\ldots\gamma(s_0, a_1)\ldots, a_{n-1}), a_n)$$

*huard@recherche.enac.fr

†barnier@recherche.enac.fr

‡cpralet@onera.fr

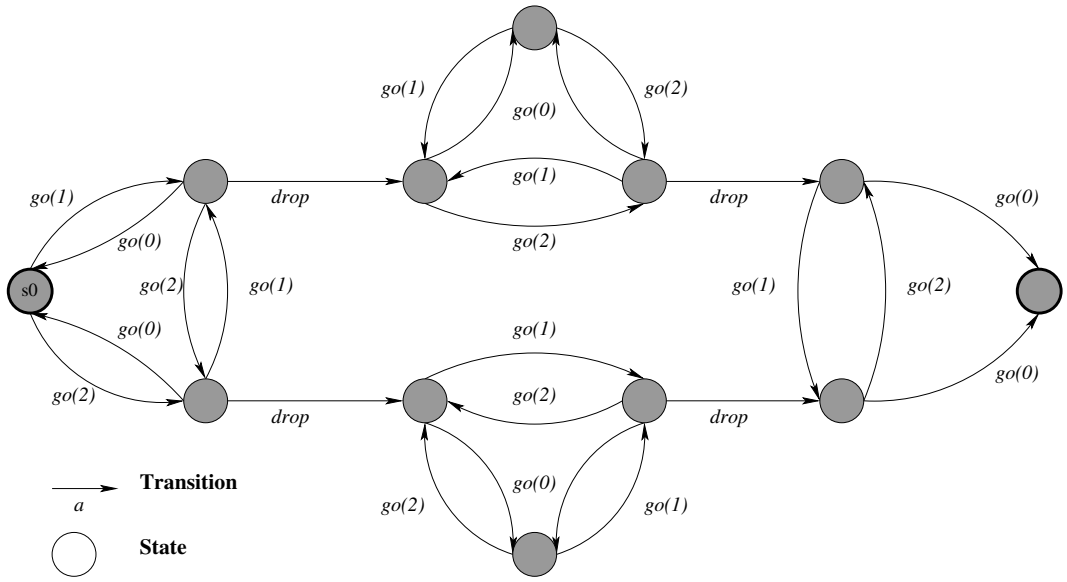

Figure 1. A state transition system that describes a mission with 1 vehicle and 2 drop zones. 0 is home, 1 is the first drop zone, 2 is the second drop zone.

I.A. States description

Each state can be described with first-order logic or with variables representation. In this paper we only use the variables representation. We define each state as a set of variables which represents the state of each MAV of the fleet, and the achievement of each goal. Moreover, for a fleet of MAVs we need to synchronize the actions of the various aircraft. To do this synchronization, we add a variable for each MAV describing the time left to the end of the current action.

For each MAV i we have the following variables at decision step t:

- $position[i,t] \in [0,...,n-1]$ is the position of the MAV and n is the number of waypoints in the mission.
- $ball[i,t] \in [0,\ldots,balls_{max}]$ is the number of paintballs left in the airframe.
- $video[i,t] \in [0,1]$ is set to 1 if the MAV has a video camera sensor. This variable is constant if we consider that the video camera won't fail during the mission.
- $flying[i,t] \in [0,1]$ represents if the MAV is flying or not.
- $time[i,t] \in [0,...,time_{max}]$ is the time to the end of the current action and $time_{max}$ is the maximum time of an action.
- $action[i,t] \in \{nop, takeoff, go(x), eight, drop, land, nop\}$, $x \in [0,\ldots,n-1]$ is the current action.

For each goal k we have : $goals[k,t] \in [0,1]$, deepending whether the goal k is achieved or not.

At the initial state all MAVs are on ground.

I.B. Actions and transitions description

The transition function γ describes the constraints,[2] i.e. the preconditions and postconditions of an action that bind two successive states. Therefore, we use the language of constraints to express γ. An action for a

fleet of MAVs is the vector $a = (a_1, a_2, \ldots a_i \ldots, a_{nb\_aircraft})$ of the actions of each MAV. In this paper we consider for each MAV the following actions: Takeoff, Land, Move, Eight[a] and Drop[b].

I.B.1. The preconditions

A MAV has to finished its current action before it begins a new one, therefore for each action we have the following precondition: $time[i,t] = 0$.

In this paper we consider the following constraints:

Takeoff The MAV is on the ground and Takes off, which gives the following constraint: $(flying[i,t] = 0)$

Land The MAV is flying and lands on the ground: $(flying[i,t] = 1)$

Move The MAV is flying and move to a waypoint x : $(flying[i,t] = 1)$

Eight The MAV does a eight figure centered over the target x to identify it: $(flying[i,t] = 1) \wedge (postion[i,t] = x) \wedge (video[i,t] = 1)$

Drop The MAV drops a ball on the target x: $(flying[i,t] = 1) \wedge (position[i,t] = x) \wedge (ball[i,t] > 0)$

Nop The MAV is on the ground and does no action: $((time[i,t] = 0) \vee (action[i,t] = nop)) \wedge (flying[i,t] = 0)$

I.B.2. Action in progress

Additional other constraints need to be met as well by the following variables.

If $time[i,t] > 1$ the current action is not finished, so the state variables are constants except for the time variable:

$$\begin{aligned} position[i,t+1] &= position[i,t] \\ ball[i,t+1] &= ball[i,t] \\ flying[i,t+1] &= flying[i,t] \\ action[i,t+1] &= action[i,t] \end{aligned}$$

The evolution of the time to the end:

$$\begin{aligned} time[i,t] > 0 &\Rightarrow time[i,t+1] = time[i,t] - 1 \\ time[i,t] = 0 &\Rightarrow time[i,t+1] = duration(action[i,t+1]) - 1 \end{aligned}$$

I.B.3. The effects

If $time[i,t] = 1$, the constraints translating the effects of actions are applied to the state variables:

- After taking off the aircraft is flying, the $flying$ variable is set to 1 (see equation (1)).
- After landing the aircraft is on the ground, the $flying$ variable is set to 0 (see equation (2)).
- After moving to x the aircraft position is x (see equation (3))

[a] *Eight*: MAV does a eight figure centered on a target
[b] *Drop*: MAV drops a paintball on a target

$$action[i,t] = takeoff \Rightarrow flying[i,t+1] = 1 \quad (1)$$
$$action[i,t] = land \Rightarrow flying[i,t+1] = 0 \quad (2)$$
$$action[i,t] = move(x) \Rightarrow position[i,t+1] = x \quad (3)$$

Let k be the goal corresponding to the realisation of $action[i,t]$ at $position[i,t]$. If $time[i,t] = 1$ then:

$$\begin{aligned} action[i,t] = eight &\Rightarrow goal[k,t+1] = 1 \\ action[i,t] = drop &\Rightarrow (goal[k,t+1] = 1) \wedge (ball[i,t+1] = ball[i,t] - 1) \\ goal[k,t] &\leq goal[k,t+1] \end{aligned}$$

I.C. Optimisation

To compare the quality of different solutions we use a criterion, also called utility function ($U : S \rightarrow \mathbb{R}$). This criterion permits to choose the best solution. In some cases, finding the optimal solution can be too expensive computationaly speaking. In those cases we use the criterion to find a "good" solution (but not the best).

In this paper, we chose to first maximize the amount of goal achieved (or the sum of the utilities if some goal are more important than others) within a given time and then to minimize the duration of the mission.

II. Mission planning under uncertainty

In the first part we have not modelised the uncertainties in the problem. We write it as if it traduces the reality of operations. However, the result of an action is not always deterministic. Markov Decision Processes[3] are a frequently used model to take into account the uncertainty of the effects of an action. It introduces the probability of transition between two states as a function $T : S \times A \times S \rightarrow [0,1]$. In this paper we choose to use a formalism based on Uncertainties, Feasibilities[c], and Utilities[4] which is better suited to our problem because we have already describe feasibilities and utilities in the previous section. Therefore we only need to describe uncertainties.

We suppose the result of an action does not depend on the prior history, it depends of the current state and of the enviroment only.

II.A. The uncertainties

We assume that only the achievement of a goal is uncertain. We define the probability of identification of a target at the end of the eight figure by the MAV i:

$$P_{id}(i) = P\left(goal[k,t+1] = 1 \mid goal[k,t] = 0, action[i,t] = eight, time[i,t] = 1, position[i,t] = pos(k)\right)$$

Let $\{i_1, \ldots, i_m\}$ be a set of MAV verifying $action[i,t] = eight, time[i,t] = 1, position[i,t] = pos(k)$. The event "MAV i identify the target k" does not depend of the event "MAV j identify the target k", therefore using the Poincaré formula we have:

$$\begin{aligned} P_{id} &= P(goal[k,t+1] = 1 \mid goal[k,t] = 0) \\ &= \sum_{j=1}^{m} (-1)^{j-1} \sum_{1 \leq i_1 < \cdots < i_j \leq m} [P_{id}(i_1) \times P_{id}(i_2) \times \cdots \times P_{id}(i_j)] \end{aligned}$$

[c] Feasibilities corresponds to the transition constraints.

And if $P_{id}(i) = p$ does not depend of the MAV i we have:

$$P_{id} = \sum_{j=1}^{m} (-1)^{j-1} \sum_{l=1}^{j} \binom{j}{l} p^l \tag{4}$$

We can define the probability of dropping a ball close to the target using the same properties as in (4).

II.B. Solving

To solve a planning mission with uncertainty, we propose to generate an optimal policy $\pi : S \rightarrow A$ which will maximize the expected utility after each step of decision. We use Bellman's optimality equations (see equations 5 and 6) and dynamic programming[3,5] to compute the optimal value $V_n^*(s)$ at each step of decision n for each state s and the corresponding action $\pi_n(s)$.

$$V_n^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s, s', a) \left((U(s') - U(s)) + V_{n+1}^*(s') \right) \tag{5}$$

$$\pi_n(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s, s', a) \left((U(s') - U(s)) + V_{n+1}^*(s') \right) \tag{6}$$

However some problems cannot be solved within acceptable computation times (e.g. a mission with 3 targets, 1 MAV and 120 decision steps can take as much as 3 hours to solve on a PIV 3GHz). Moreover, it might be impossible to know the probabilities of transition between state, or to a long time before the mission the field conditions such as wind.

III. Real time mission planning and replanning

Our second method is a real-time planning and replanning algorithm. It permits to solve problems that cannot be modelled with probabilities. The algorithm we use to compute the plan is based on the classic A* algorithm.[6] We build a plan *offline* assuming all actions are deterministic and will succeed. During the execution of the plan we check whether an action fail or does not end in the forecasted state, and replan when necessary.

We define three different stages of the real-time planning[7] and replanning algorithm. The *A* planner* which builds a new plan from current state, the *execution* which runs the plan, and the *controller* which checks if the plan needs to be corrected (see figure 2).

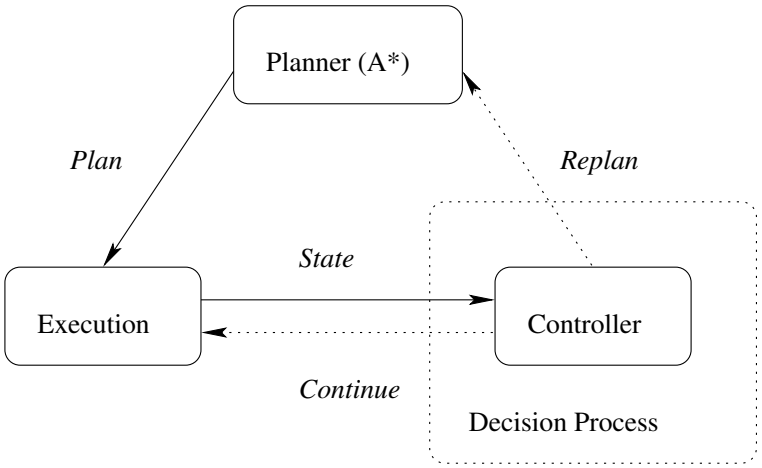

Figure 2. Execution / planning scheme.

III.A. Planning

The A* algorithm is a pathfinding algorithm[d] often used in IA planning because of it simplicity and efficiency due to the fact that it expands estimated best nodes first. To estimate the cost of the best solution that goes through state n we use:

$$f(n) = g(n) + h(n)$$

Where $g(n)$ is the time to go from initial state to n. and $h(n)$ is an estimated time to the solution from n.

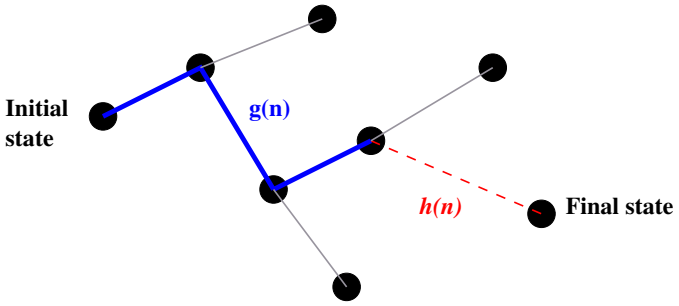

Figure 3. Estimated cost of a partial solution.

The A* algorithm usually stores partial solutions into a priority queue using a function f to estimate the cost of a partial solution. It expands the node with the best priority, and store the new partial solutions into the same priority queue. To minimize the cost of the mission we include a penality in g for solution state where a goal is not done. Let G be the set of goal that are not achieved, the penality for a solution state is:

$$penality = \left(\sum_{k \in G} U[k] \right) \times time_{mission}$$

$$cost = g + penality$$

Where $time_{mission}$ is the maximum time allocated to the execution of the mission.

III.B. Execution Control and replanning

The planner sends to the execution module of the MAV a plan $\pi = (a_k)_{1 \leq k \leq n}$ and the predicted resulting states $p = (s_k)_{1 \leq k \leq n}$. After each step of the plan, we enter a decision process where the controller decides whether to continue with the execution of the plan or to build a new plan to cope with an unpredicted state, i.e. if $\gamma(s_{t-1}, a_t) \neq s_t$. In those cases, the planner needs to give a new plan as soon as possible. The online computing time ct of a new plan needs to be very small compared to the decision time step $time\_step$. We choose the following condition:

$$ct < \frac{time\_step}{10} \tag{7}$$

In case the previous condition fails, we repair the previous plan thus If the last action of MAV i failed, the MAV i redo this action before resuming his plan. The condition 7 is likely to fail when few goals have been achieved.

[d] Pathfinding = find the shortest path.

IV. Results and Further works

We have successfully written a formal model of mission for a fleet of MAVs. We have also proposed and implements two different algorithms to solve the mission planning problem using this model. Our Dynamic Programming based algorithm, which takes into account uncertainties, is an offline algorithm (computed on ground before the mission), and the resulting policy could be airborne in a paparazzi equiped MAV as a conditional flight plan. We successfully solved a 2 MAVs mission with 4 targets and 120 decision steps. Our real-time planning and replanning algorithm successfully solved a 2 aircrafts mission with 8 targets. Our first results show that for it is harder to solve problems if the fleet of MAVs is homogenous (i.e. all the MAVs have the same sensors), it makes the problem highly symmetrical. The next step is to work on symmetry breaking and goal allocation to tackle the complexity of the problem.

References

[1]Ghallab, M., Nau, D., and Traverso, P., *Automated Planning: Theory and Practice*, Morgan Kaufmann Publishers, 2004.

[2]van Beek, P. and Chen, X., "CPlan: A Constraint Programming Approach to Planning," *AAAI/IAAI*, 1999, pp. 585–590.

[3]Putterman, M., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, 1994.

[4]Pralet, C., Verfaillie, G., and Schiex, T., "Decision with uncertainties, feasibilities, and utilities: towards a unified algebraic framework," *The 17th European Conference on Artificial Intelligence*, 2006.

[5]Bellman, R., *Dynamic Programming*, Princeton University Press, 1957.

[6]Nielson, "Problem Solving Methods in Artificial Intelligence," 1971.

[7]Paolucci, M., Shehory, O., and Sycara, K., "Interleaving planning and execution in a multiagent teamplanning environment," 2000.