APPROXIMATE DYNAMIC PROGRAMMING METHODS FOR COOPERATIVE UAV SEARCH

Matthew Flint [*] **Emmanuel Fernandez** [**]

[*] *BAE Systems, Advanced Information Technologies*
6 New England Executive Park, Burlington, MA 01801
matt.flint@baesystems.com
[**] *Dept. of Electrical and Computer Engineering and Computer Science*
University of Cincinnati, Cincinnati, Ohio 45221-0030
emmanuel@ececs.uc.edu

Abstract: This paper considers the decentralized dynamic programming path planning decision processes of multiple cooperating autonomous aerial vehicles (UAVs) engaged in a search of an uncertain environment. However, what sets this paper apart from previous work is that a functional approximation is used for the dynamic programming (DP) cost-to-go function, resulting in improved quality of the decisions made while retaining computationally feasibility. Some of the effects of the changes are measured by simulation analysis, and the results are presented.

Keywords: Dynamic Programming, Stochastic Modeling, Autonomous Vehicles, Decision Making, Search Methods

1. INTRODUCTION

The usefulness of autonomous vehicles, in both civilian and military contexts, has resulted in a great deal of attention for control of these types of systems. This paper investigates the case where multiple vehicles must cooperatively act together by planning paths to meet some global goal, which is a far more complex case than a single vehicle acting alone. This area, cooperative control of unmanned aerial vehicles (UAVs), has a growing body of research which has been advancing rapidly in recent literature. A great deal of work in cooperative decision and path planning is in the area of "point-to-point" paths, where the vehicles have a given destination or set of destinations to travel to, and must find efficient ways to travel to those destinations, and is a type of vehicle routing and /or task allocation problem. While this area has produced many good results—see e.g. (Bellingham *et al.*, 2002) and many others—the methods utilized are, in general, not directly applicable to cooperative search in an uncertain environment, where the reduction in uncertainty about the environment is of larger importance than the vehicle traveling to specific locations. Some work on cooperative UAV search has been done: see (Rajnarayan and Ghose, 2003); (Hespanha *et al.*, 1999); and previous work by the authors of this paper e.g. (Flint *et al.*, 2003*b*), (Flint *et al.*, 2003*a*), (Flint *et al.*, 2004); and others.

As in previous work by the authors of this paper, a *dynamic programming* algorithm (Bertsekas, 2000) is presented which tries to address the problem of optimality under feasibility constraints. The central problem is the trade-off, made when implementing a dynamic programming algorithm, between optimality and computational complexity. The previous work of the authors utilized a rolling horizon approximation in order to guarantee feasibility. However, this approximation greatly shortened the planning horizon, taking it from the entire mission (in the optimal case) to just a few time steps (which is guaranteed to be feasible). This approach is extended and improved in this paper by using a type of Approximate Dynamic Programming (ADP), which takes the form of several modular approximations that can present information that exists outside of the planning horizon to the vehicles, but at the same time preserving the guaranteed feasibility. Some simulation results show the improvement that can be gained from using the elements outlined in this paper.

2. PROBLEM FORMULATION AND FEASIBLE SOLUTION

2.1 The Dynamic Programming Formulation

For a cooperative UAV search mission, each vehicle must search to locate objects in order to complete the mission. To create a framework in which the best decision of where to place the vehicles' paths can be calculated, first the *state* of the environment is defined. This state is comprised of: the search status, i.e. what objects there are, and their level of uncertainty in location, probability of existence, and probability of classification; the vehicle status, i.e. what vehicles there are whether and whether the vehicles are functioning or not; and the vehicle positions, i.e. their latitudes and longitudes. If the vehicle can communicate with one another, they can form a comprehensive view of the state. (However, it is possible for a vehicle to make decisions based on whatever state information is available in the case of limited or delayed communications.)

With the state defined in this manner, the problem can then be modeled as a Markov decision process, with certain states giving the vehicle rewards (e.g. states in which search objects are located and classified or when a vehicle returns from a mission undestroyed). As the scenario progresses and the vehicles move and act, the system comprised of the team of vehicles transitions from state to state, possibly in a stochastic manner. There are at least three important sources of randomness within this environment. First, unknown locations of the search objects makes the detection or failure of detection of these objects a random event. Secondly, and adding to the first source, the sensors are not assumed to be perfect, and are instead modeled to include errors that occur probabilistically. Lastly, because the problem is decentralized, with each vehicle making its own decisions, the actions of other vehicles are not necessary known to the other vehicles, and the effects of the other vehicles must also be modeled probabilistically.

Now, since the paths that the vehicles take is controllable, it is possible to create a planning algorithm such that some control over the state transitions is gained. Using Dynamic Programming (DP), it is theoretically possible to develop an optimal policy, which dictates the best decisions (Bertsekas, 2000), in terms of the number of targets identified, to make in each possible state so that the maximum expected global reward is achieved over the course of the entire mission. Assume that the mission duration is N decision time steps; i.e. that a vehicle will make N decisions over the course of the mission. Further, define $J_k(x_k)$ as the "cost-to-go" at decision time step k (actually, this is the accepted terminology, even though in this case the problem is formulated to maximize the number of targets identified rather than minimize cost—they are mathematically equivalent problems), and represents the expected number of targets identified by the vehicles as they travel from time step k to the end of the mission. In the ideal case (i.e. unlimited computational power), the optimal decisions can be found by taking the arguments of the maximization of the Dynamic Programming recursion (Bertsekas, 2000),

$$\begin{aligned} J_k(x_k) = \max_{u_k \in \Omega} \{ \mathrm{E}_{w_k} \{ g(x_k, u_k, w_k) \\ + J_{k+1}(f(x_k, u_k, w_k)) \} \}. \end{aligned} \tag{1}$$

In this equation, the state at a time step k is x_k, the vehicle assignments (or control) are u_k which come from a set of possible assignments Ω, such as: turn left, turn right, ascend, descend, or go straight. The stochastic elements are contained in w_k. The term $g(x_k, u_k, w_k)$ is the single step gain. Note that $x_{k+1} = f(x_k, u_k, w_k)$, which produces the next state from the current state, control, and stochastic elements.

For ease of presentation for the remainder of this paper, let Φ_k represent the set $[x_k, u_k, w_k]$, noting that the index of each member is the same as that of Φ. In equation (1), $g(\Phi_k)$ represents the gain that can be had at the present time, and $J_{k+1}(f(\Phi_k))$ represents the predicted gain from future times.

2.2 The Single Step Gain

In order to calculate the cost-to-go of Equation (1) in an algorithmic manner, the expected values of the terms therein must be calculated. The first term, the single step gain, is examined first, following work done in (Flint *et al.*, 2003*a*). Let σ_k be the search gain (i.e. the value of $g(\Phi_k)$ in the best case) for a vehicle at time step k. Let Ψ_k be the probability of another vehicle interfering with the planning vehicle at time k (refer to (Flint *et al.*, 2003*b*) for details on this value). Let δ_k be the probability that the planning vehicle is alive at time k.

Assume that vehicles cause the uncertainty in the environment for all objects to decrease by a set amount, such that there exists some $0 \leq \rho \leq 1$ such that if another vehicle interferes, the vehicle receives gain $(1-\rho)\sigma_k$. If no vehicle interferes, then the vehicle receives gain σ_k. This assumption is good as long as there are no pop-up objects and if the vehicles' sensors can be modeled probabilistically to generate the value of ρ (See (Flint *et al.*, 2004) for a formulation that uses such a model.) Also assume that multiple vehicles interfering will produce a negligible difference from a single vehicle interfering. Next, if a vehicle is destroyed, then it receives 0 gain for the time step. However, if it is alive, then it receives full (unity) gain. Thus, the expected value of the single step gain can be written as:

$$E\{g(\Phi_k)\} = \delta_k \sigma_k (1 - \rho \Psi_k) \tag{2}$$

2.3 The Future Gain

Now that the single step gain is defined, the next step to find the cost-to-go for the system at time k is to iterate the recursion in (1).

The terminal cost of the search mission is assumed to be J_N for all x_N, and is achieved at the end of the search mission. The complete cost-to-go for any time step k is then found by iterating enough times until the terminal cost is reached. This gives

$$\begin{aligned} J_k(x_k) = \max_{u_k \in \Omega} \{ & \mathrm{E}_{w_k}\{g(\Phi_k) + \\ & \max_{u_{k+1} \in \Omega} \mathrm{E}_{w_k}\{g(\Phi_{k+1}) + \ldots + \\ & \max_{u_{N-1} \in \Omega} \mathrm{E}_{w_k}\{g(\Phi_{N-1}) + \\ & J_N(f(\Phi_{N-1}))\}\}\}\}. \end{aligned} \tag{3}$$

However, attempting to solve this equation for the optimal cost to go produces a very complex problem, since the "curse of dimensionality" causes an explosion of possible states to examine over large values of $(N-k)$, i.e. the planning horizon of the entire mission. Past work by the authors (e.g. (Flint *et al.*, 2003*b*) and (Flint *et al.*, 2004)) has utilized a *rolling horizon* limited lookahead policy (see Section 6.3 of (Bertsekas, 2000))) to make the problem feasible, at a cost of optimality, with a good degree of success. This rolling horizon approximation defines a horizon of time steps r, then replaces the value of J_{k+r+1} with the final cost J_N. This gives, instead of equation (3),

$$\begin{aligned} J_k(x_k) = \max_{u_k \in \Omega} \{ & \mathrm{E}_{w_k}\{g(\Phi_k) + \\ & \max_{u_{k+1} \in \Omega} \mathrm{E}_{w_k}\{g(\Phi_{k+1}) + \ldots + \\ & \max_{u_{N-1} \in \Omega} \mathrm{E}_{w_k}\{g(\Phi_{k+r}) + \\ & J_N(f(\Phi_{k+r}))\}\}\}\}. \end{aligned} \tag{4}$$

Since r can be chosen such that $k + r << N$, this produces a much smaller problem space, which can be solved exactly to yield an optimal answer to the sub-problem. However, this solution is not necessarily an optimal solution to the main problem. This creates a fundamental tradeoff between feasibility and optimality.

2.4 Cost-to-go Functional Approximation

A solution to this dilemma is to modify the rolling horizon approximation using a heuristic cost-to-go approximation similar to those shown in Section 6.3.4 of (Bertsekas, 2000). As defined in this reference, this method utilizes an appropriate *scoring function* to evaluate the position of the vehicle at the end of the planning horizon. The limited horizon approximation is still used, but instead of using the terminal cost for time N, the scoring function is used as an approximation of the actual cost-to-go. Thus, an approximate cost-to-go $\tilde{J}$ is created such that

$$\tilde{J}(\Phi_{k+r}) \approx J_N(f(\Phi_{k+r})). \tag{5}$$

The optimization is then performed on the Approximate Dynamic Programming equation

$$\begin{aligned} J_k(x_k) = \max_{u_k \in \Omega} \{ & \mathrm{E}_{w_k}\{g(\Phi_k) + \\ & \max_{u_{k+1} \in \Omega} \mathrm{E}_{w_k}\{g(\Phi_{k+1}) + \ldots + \\ & \max_{u_{N-1} \in \Omega} \mathrm{E}_{w_k}\{g(\Phi_{k+r}) + \\ & \tilde{J}(f(\Phi_{k+r}))\}\}\}\}. \end{aligned} \tag{6}$$

This has the benefit of always producing a feasible result, since the rolling horizon is still in effect. However, if the Approximate Dynamic Programming scoring function accurately approximates the cost-to-go, then the solution will be closer to the true optimum than just the rolling horizon solution alone, which is more myopic. The problem then becomes one of creating an approximation that is accurate and feasible. Since this is expected to be a problem that can be solved incrementally, it is desired that the approximation be developed in a modular fashion, so that the elements and parameters can be added, removed, or refined individually, in order to be able to test the approximations, and adjust what works, and discard what does not.

With this in mind, let this scoring function (that approximates the cost-to-go) be defined as the combination of three modular functions, each of which contains information important to the future gain of the vehicles. This is given by

$$\tilde{J}(\Phi_k) := U\big(L(\Phi_k), I(\Phi_k), M(\Phi_k)\big). \tag{7}$$

The value L is a length factor that determines how far away a potential target is. The function $I(\Phi_k)$ is an interference prediction factor or cooperation factor, which attempts to predict the effect of other vehicle's actions on the environment. The function $M(\Phi_k)$ is a mission position factor. This is high if accomplishing the task puts the vehicle in a good position to complete the other elements of the mission. Lastly, the function $U(\cdot)$ takes the values provided by the other functions and combines them in an intelligent fashion.

To put some basic heuristics into place in order to test the concept of the proposed idea, let $L(\Phi_k)$ be a vector of the same size as the number of objects in the environment such that for object t, there is an element of the vector $L(\Phi_k)$ that can be given by

$$L_t(\Phi_k) = \lambda_t \left(1 - (\frac{d_t^c}{d^*})\right), \tag{8}$$

where $0 \le \lambda \le 1$ is the value, or reward for finding, object t. The value d_t^c is the distance from object t to the planning vehicle c and d^* is a distance that is guaranteed to be larger than d_t^c for any value of c and t. Note that $0 < L_t \le 1$ for all objects t. All of these values can be calculated from (Φ_k). This approximation is based on the fact that, on the average, during the course of an arbitrary search mission, more objects can be detected if vehicles search for objects that are suspected to be close to them, rather than wasting time traveling to those that are far away. Thus, this value is high if an object is nearby, and low if an object is far away. Of course, the presence of the value (or priority) of the object (λ_t) can override this such the vehicles will bypass lower valued, but closer objects in favor of farther but more valuable ones.

Next, let $I(\Phi_k)$ be a vector of the same size as the number of objects in the environment such that for object t, there is an element of the vector $I(\Phi_k)$ that can be given by

$$I_t = \frac{V_N - n_t}{V_N}, \tag{9}$$

where n_t is the number of vehicles that could potentially be closer to the object t than the planning vehicle at the point in time where the planning vehicle would arrive at the state for which $\tilde{J}(\Phi_{k+r})$ is being evaluated. The value V_N is the total number of vehicles. All of these values can be calculated from (Φ_k). Because of the improvement in search performance that is expected when vehicles travel to close objects (which is embodied in $L(\cdot)$), it is expected that any vehicles closer to any given object than the planning vehicle will be more likely to have already searched for it. Thus this approximates the probability that the object will not already have been detected by some other vehicle by the time the planning vehicle gets to it. Note that $0 < I_t \le 1$ for all objects t.

Then, let $M(\Phi_k)$ be a vector of the same size as the number of objects in the environment such that for object t, there is an element of the vector $M(\Phi_k)$ that can be given by

$$\hat{M}_t = \begin{cases} \dfrac{\lambda_t}{T-1} \displaystyle\sum_{z \in \Theta, z \neq t} m_z^t & T > 1 \\ 1 & T \le 1, \end{cases} \tag{10}$$

where in this equation z is an arbitrary object out of the set of all known objects Θ, the total number of objects in the environment is $T \ge 0$, and

$$m_z^t = K_m + (d_{max}^t - d_z^t). \tag{11}$$

This value turns the distance into an inverse, or "closeness", factor, such that the closer something is the larger the factor. In this equation, $K_m > 0$ is a constant that keeps the value positive, and

$$d_{max}^t = \max_{y,z \in \Theta} \{d_z^y\}, \tag{12}$$

where y and z are arbitrary objects in Θ. Lastly,

$$M_t = \frac{\hat{M}_t}{\hat{M}_{max}}. \tag{13}$$

The value $\hat{M}_{max}$ is the largest value of $\hat{M}_t$ for any object, and ensures that $0 < M_t \le 1$ for all objects t. All of these values can be calculated from (Φ_k). What this complicated value produces is a low value for objects that are relatively isolated, or distant from all other objects. Inversely, a high value of M_t means that after reaching object t, relatively many others will be nearby, which should provide more opportunity for detections, which should then produce a better overall search mission.

Lastly, the function $U(\cdot)$ combines these by

$$U(L(\Phi_k), I(\Phi_k), M(\Phi_k)) = \left(\sum_{\forall t \in \Theta} L_t(\Phi_k) I_t(\Phi_k) M_t(\Phi_k)\right) K_u, \tag{14}$$

where K_u is a scaling factor to ensure that the value is appropriate given the single step gains.

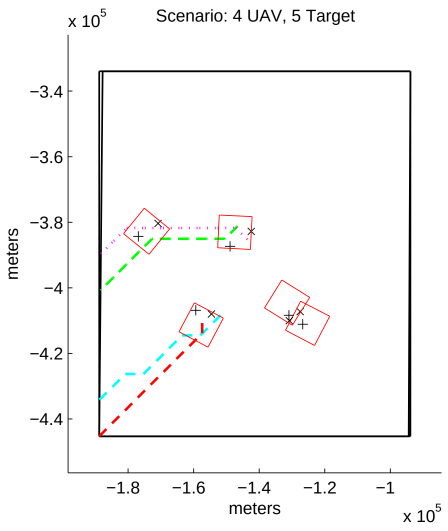

Fig. 1. The simulation environment, showing 4 vehicles' paths, and the targets (with uncertainty regions).

3. SIMULATION RESULTS

A small simulation study using a custom C++ UAV simulator is included in this paper in order to illustrate the feasibility of implementing this approach and to illustrate the improvement to the performance of cooperative search including these factors can have. A typical environment used for these simulations is shown in Figure 1.

In this environment, there exist 4 vehicles, the paths of which are shown by the dotted and dashed lines. The vehicles are searching for 5 real objects, of which 4 are real targets, and 1 is a false target. The targets are initially given to the vehicles as a suspected location, i.e. a coordinate pair, shown as an (x), as well as an uncertainty region which, if the target exists, it is guaranteed to be in. The targets are initially assumed to be distributed equally within the uncertainty region, so the suspected coordinates given do not provide any actual information about where within the uncertainty region the target lies. The actual location of the target, which the vehicles do not know, is shown as a (+). The initial probability of each object being a real object is given as 0.75. The initial estimate of an object being a real target is given as 0.75.

The vehicle will consider an object "found" once its probability of being a real object is greater than 0.95 and its probability of being a real target is either 0.75 or above (i.e. relatively sure it's a real target) or 0.25 or below (i.e. relative sure it's not a real target.) Also, there exists in the vehicles' databases a "ghost" object that is not a real object, but can be used to represent the probability of there existing any further objects in the environment. This ghost object's uncertainty region consists of the entire search environment and has initial probability of existence of 0.50.

The vehicles are allowed 3000 seconds (50 minutes) to conduct their search, and travel at a constant 185.2 m/s (value chosen arbitrarily but assumed to be close to what an actual UAV would travel at). Each vehicle is equipped with a sensor that can detect the ground beneath it as it travels, using sensor parameters (see (Flint *et al.*, 2004)): probability of correct positive detection $\rho = 0.80$, probability of correct positive classification $\psi = 0.80$, probability of correct negative classification $\omega = 0.95$, and probability of correct negative detection $\gamma = 0.99989583$, which for 4 vehicles and 5 targets gives approximately one false positive (i.e. detecting one of the 5 targets where it isn't) every 4 simulation minutes. Each vehicle is allowed to make decisions every 30 seconds (each period of which is a *decision time step* i.e. one value of k.) For the parameters used in the future gain approximations, $\lambda_t = 1$ if the object is not yet located, and is 0 if it has been located. The parameters, $K_m = 1$, and $K_u = 10$, were chosen arbitrarily.

At each decision time step, the vehicle must choose to: go straight ahead; ascend or descend (while going straight); or turn left or right 45 degrees. This limit on maximum turn angle represents the inability of a vehicle to make very wide turns due to the g-force limit on flying vehicles. Once the vehicle has made this choice, the continuous, real-world path of the vehicle is approximated by allowing it to turn (as appropriate) immediately and then travel straight until the next decision time step. In order to execute the simulation in a reasonable amount of time, the rolling horizon of the vehicles is limited to $r = 4$ time steps. The vehicles are assumed to share the results of their sensor actions and their locations each time step with the other vehicles over their communication channels. However, the communication between vehicles is assumed to not be large enough to allow negotiation to occur, thus producing a passive kind of cooperation. Aside from flat ground at an altitude of 0, which will cause the vehicles to crash if they contact it, there are no threats present.

The average number of objects "found" during the course of a mission is shown in Figure 2. The difference in the performance of the case that utilizes the Approximate Dynamic Programming (ADP) and the case that does not use the approximations is statistically significantly different at a level of confidence of 95% or more after the lines separate (about 540 simulation seconds for this figure). The results shown on this figure demonstrate that using these approximations can result in a better

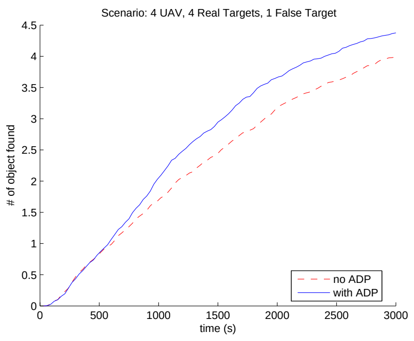

Fig. 2. A comparison of the average number of objects (out of 5 possible) found and classified during the course of a search mission.

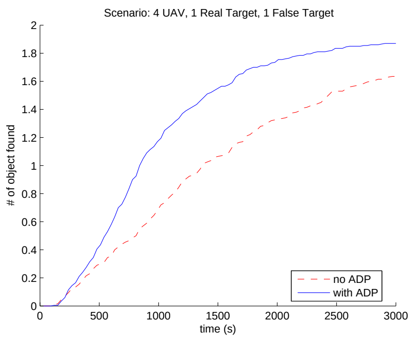

Fig. 3. With only 2 targets in the environment, the ADP methods produce significantly better results than previous cases.

search being conducted, even with the very basic equations developed and presented in this paper.

Two additional results are shown in Figures 3 and 4. In these figures, the same simulation study was run, to the same level of statistical significance, as that which produced Figure 2, except that in the simulation study that produced Figure 3 there were only 2 targets in the environment (1 false and 1 real). In the simulation study that produced Figure 4 there were 10 targets (8 real and 2 false). As can be seen in these figures, the level of improvement of the ADP approximations is dependent on how much information is actually available to the vehicles in the limited planning horizon. In the 2 target case, it is a lot less likely that no target information is available within the vehicle's planning horizon, and so the case without the ADP is much worse than an in the opposite case, where with 10 targets it is much more likely that there is target information to act on in the vehicle's planning horizon.

4. CONCLUSIONS

As can be seen from the simulation results, even these preliminary Approximate Dynamic Programming extensions to the previous formulation can potentially have a large impact on the quality

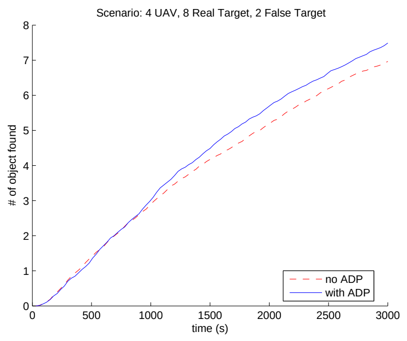

Fig. 4. With 10 targets in the environment, the ADP methods produce a less significant increase in effectiveness.

of a search mission. As was noted earlier, the scoring functions were heuristic in nature. A series of designed experiments or theoretical analysis can be conducted that would allow the identification of important factors in the development of better scoring functions that more accurately reflect the true cost-to-go. Using such scoring functions will cause the search mission to be completed with even better results.

REFERENCES

Bellingham, J. S., M. Tillerson, M. Alighanbari and J. P. How (2002). Cooperative path planning for multiple UAVs in dynamic and uncertain environments. In: *Proc. of the 41st IEEE Conference on Decision and Control*. pp. 2817–22.

Bertsekas, D. P. (2000). *Dynamic Programming and Optimal Control*. Vol. 1. 2nd ed.. Athena Scientific. Belmont, Massachusetts.

Flint, M., E. Fernández-Gaucherand and M. Polycarpou (2003*a*). Cooperative control for UAV's searching risky environments for targets. In: *Proc. of the 42nd IEEE Conference on Decision and Control*. pp. 3567–3572.

Flint, M., E. Fernández-Gaucherand and M. Polycarpou (2003*b*). Stochastic modeling of a cooperative autonomous UAV search problem. *Military Operations Research* **8**(4), 13–32.

Flint, M., E. Fernández-Gaucherand and M. Polycarpou (2004). Efficient Bayesian methods for updating and storing uncertain search information for UAV's. In: *Proc. of the 43nd IEEE Conference on Decision and Control*.

Hespanha, J. P., H. J. Kim and S. Sastry (1999). Multiple-agent probabilistic pursuit-evasion games. In: *Proc. of the 38th Conf. on Decision and Control*.

Rajnarayan, D.J. and D. Ghose (2003). Multiple agent team theoretic decision-making for searching unknown environments. In: *Proc. of the 42st IEEE Conference on Decision and Control*. pp. 2543–48.