15th Triennial World Congress, Barcelona, Spain

COOPERATIVE PATH-PLANNING FOR AUTONOMOUS VEHICLES USING DYNAMIC PROGRAMMING [1]

Matthew Flint, Marios Polycarpou, Emmanuel Fernández-Gaucherand [2]

Dept. of Electrical and Computer Eng. and Computer Science
University of Cincinnati, Cincinnati, Ohio 45221-0030, USA

flintmd@email.uc.edu, polycarpou@uc.edu,
emmanuel@ececs.uc.edu

Abstract: It is shown how to model a cooperative path planning system for multiple autonomous air vehicles within the framework of a stochastic (dynamic programming) decision process. The proposed approach allows the vehicles to cooperate and find near-optimal search paths over a given environment in the presence of uncertainty and constraints on movement and computational power.

Keywords: Agent, Autonomous Vehicle, Co-operative Control, Distributed Control, Dynamic Programming, Stochastic Control

1. INTRODUCTION

1.1 *Overview*

This paper presents a stochastic decision process formulation for cooperative control among a team of distributed, uninhabited air vehicles (UAV's) which leads to a solution based on dynamic programming. The work presented here is part of a larger research effort aimed at developing cooperative control algorithms that will allow a team of UAV's to fly high-level missions without direct human intervention, and in the presence of uncertainty (Pachter and Chandler, 1998). This allows for missions that are particularly suited for autonomous flying agents such as wide area searches, in cases like searching for lost or stranded persons in dangerous environments, or "smart" munitions that can be launched without knowing precisely where the targets are (Jacques and Leblanc, 1998).

What differentiates this problem from the classical or standard search problems is the fact that optimal paths are desired, rather than an optimal allocation of effort, and those paths are constrained. Additionally, the requirement of cooperation among a team of vehicles makes this a more challenging problem from those which have already been well researched.

1.2 *The Vehicle and the Environment*

For simplicity, a typical vehicle is assumed to fly at a constant velocity and altitude, and avoid colliding with other vehicles; these are reasonable assumptions in the case of "smart" munitions (Jacques and Leblanc, 1998). The ability of the vehicle to make turns is constrained to a minimum radius (or maximum turn angle.) The vehicle also has the ability to communicate with other vehicles over a non-perfect wireless channel.

[1] This work was supported by DAGSI (Dayton Area Graduate Studies Institue) and AFRL (Air Force Research Laboratory).
[2] Partially supported by a Faculty Research Support Program Grant, University Research Council, University of Cincinnati.

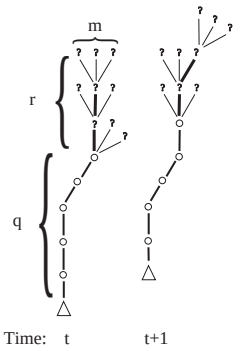

Fig. 1. Illustration of the vehicle's decision process in time.

A vehicle carries a map of the environment in its memory. This map contains all the information that the vehicle knows about its environment, and the information on this map is what the vehicle bases its decisions on. Information from this map can be shared with other vehicles, but because communication is assumed not to be perfect, all vehicles do not necessarily have the same information on their maps. As a vehicle passes over the terrain, a corresponding area of the map is updated to reflect the information gained from the vehicle's sensor. The uncertainty about that area decreases, and any threats or targets found are recorded. The environment is given by a bounded search region, about which some a priori information may be available.

1.3 *Search and Path Planning Requirements*

The goal of each air vehicle in the search problem is to move over the environment such that, at the end of the search, the maximum amount of information about the environment has been gained by the team of vehicles. A vehicle must therefore plan a *local* path over the environment to meet this *global* goal. The path planning problem is discretized in time by allowing the vehicle to only make decisions at discrete time intervals. It is further discretized in space by only allowing the vehicle to make a limited number of choices at each time step (such as: turn 15° left, go straight, or turn 15° right.)

The amount of gain received for a single sensor sweep (ρ) is one of the parameters that can be adjusted as part of the scenario. However, searching a region that has already been searched does not produce as much of a gain as searching an area that has never been searched before. Nevertheless, unless a vehicle can get complete certainty about an area with one pass, there is still some gain from searching an area more than once.

The lifetime of the vehicle, determined by the amount of fuel it carries, will be N time steps. Additionally, at each time step the vehicle must also choose a path that is at least q steps ahead. Figure 1 shows an abbreviated search tree for a typical vehicle. This planning ahead is particularly useful for flying vehicles–ones that have to move forward at all times and thus require something of a buffer in their decisions on where to move next–and is also useful in terms of allowing autonomous vehicles to cooperate with one another, if the moves that they plan can be communicated to each other. At each time step, the vehicle augments its path plan by choosing a point from m possible choices. Drawing the possible choices that the vehicle can take produces a tree q steps ahead of the vehicle. The tree is of depth r, which is how far the vehicle searches in one time step.

2. THE DYNAMIC PROGRAMMING (DP) APPROACH

2.1 *The need for DP*

Over the years, there has been a significant body of research work on path planning and cooperative control. Most of this research is based on robotic systems. However, there are some important distinctions between cooperative control of robots and air vehicles. One of the key differences is the maneuverability constraint of air vehicles imposed by the limited turning angle. Patek et al. (Patek *et al.*, 1999) use directed graphs to constrain the direction of the vehicle, exploring the problem with a (widely) different set of initial conditions. Other methods, such as that employed in (Polycarpou *et al.*, 2001) develop a general framework for cooperative control and use heuristic policies to guide the vehicles.

A basic dynamic programming, or stochastic control, model has three key elements which can be found in this problem. First, the underlying system is a discrete time stochastic dynamic system, due to the discretization in time assumption. Secondly, the cost function is additive over time, since a vehicle cannot "forget" things by moving to the wrong areas, i.e. the least amount of gain a vehicle can obtain is zero, even if it follows an unsuitable path. And, the nature of the decision process is sequential in time, since the vehicle takes an action, collects information, makes a decision based on this information, and then acts on that decision, sequentially repeating this process, looking for a best path over all time. These place this problem and modeling paradigm within the realm of discrete-time stochastic control, amenable to DP solutions (Bertsekas, 2000).

DP provides a convenient modelling and analytical tool that many of the other, more intuitive or heuristic, approaches sometimes lack. For ex-

ample, DP can achieve a provably optimal global solution to the problem, at least in theory. In practice this is rarely the case, since this may be computationally infeasible in the absence of further structural properties, but DP can nevertheless serve as a blueprint for nearly optimal solutions.

2.2 *Modeling the Problem in a DP Framework*

The first step in modeling the problem in a stochastic control / DP formulation is to identify a state, a control (decision options), and a cost (or gain) function. In the problem described in this paper, the state of the system comprises the locations of all the vehicles and the current state of the map that shows where the vehicles have searched. However, the whole state will not necessarily be known to each vehicle. Each vehicle keeps track of its own state, i.e., its location and where it has searched. This information can be derived easily from the path list that the vehicle followed, thus, x_k, the path list, or the set of the choices the vehicle has made so far, is defined to be the state (at time step k). The state $x_k \in S_k$, where S_k is the set of all possible choices the vehicle can make up to time k. The vehicle's decision, or control, at time k is u_k, which is the choice of which direction to follow. The control $u_k \in U$, where U is a set of size m and contains the choices that the vehicle can take. U is not a function of k since the vehicle is assumed to have the same choices at every time step (for example: turn left, go straight, turn right). Because the vehicle chooses its path, the state it is in is completely under its own control, so the system evolves as a *deterministic* function of x_k and u_k, in other words, $x_{k+1} = f(x_k, u_k)$, where $f(\cdot,\cdot)$ is a logical operator that concatenates the current choice (u_k) with the list of previous choices (x_k). J_k is defined as the "cost-to-go" function from time step k to N, or equivalently, the gain to be had from taking the path from time step 1 to $k-1$.

In the case where there is only one vehicle in the environment and the vehicle has unlimited computational power, the gain function, $g(x_k)$ would be purely a deterministic function. In this case, it would be possible to treat the problem in an open-loop framework, which would be able to be solved before the vehicle is put into flight by letting the final cost (at step N) be $J_N = 0$ (no bonus for being in any particular state at time N), and solving the DP recursion from time steps 1 to N:

$$J_k(x_k) = \min_{u_k \in U} (E_{w_k}\{g(x_k, u_k) + J_{k+1}(f(x_k, u_k))\}) \quad (1)$$

However, the vehicle is not assumed to have unlimited computational power, nor to be alone.

2.3 *Complexity – Limited Lookahead Policies*

A strict DP approach will have to compute to the very end of the vehicle's search the state, which would include every possible decision it could make. Since this state is a location and path on a map, it is natural to view this as a line on the search map extending behind the vehicle, connecting all the points the vehicle has travelled, and to view the planning of the vehicle as a tree starting from the vehicle's current location, as in Figure 1. This tree would then grow exponentially as the depth of the search (the number of steps ahead planned) increases. This quickly becomes intractable when the idealistic assumption of unlimited computation is removed.

To implement this scheme in a realistic setting, then, there has to be some simplifying assumptions. To address the problem of expanding a tree out to the end of the vehicle's own lifetime, N, a limited look-ahead policy is used, in which rather than computing the state to the end of the vehicle's lifetime (a tree of depth N), it replaces the cost-to-go at state $k + r$ (J_{k+r}) with the final cost (J_N), as shown in (Bertsekas, 2000). This allows the system to calculate the tree generated by only going to a depth of r, and to choose the best path visible in this sub-tree. The size of the abbreviated tree is in general much smaller than the size of the tree of the original problem (i.e. $m^r \ll m^N$). Thus r can be chosen such that the problem is tractable.

2.4 *Adding Utility*

The introduction of the limited look-ahead policy can produce a sub-optimal solution. This arises because of the now limited scope of the vehicle's planning, in which the best action to take is only able to be discovered past the visible "horizon" of the vehicle's sight (planning).

Adding utility functions to the cost function allows for taking steps to reduce the effects of these so-called horizon problems. This can be illustrated by a situation where a vehicle is forced to go out-of-bounds because the out-of-bounds region does not enter into its horizon in time for the vehicle to turn away. Since the vehicle receives zero gain for the time it searches out-of-bounds, this is a situation that should be avoided. The solution to this problem is to include a utility function inside the gain function that can utilize information normally outside the vehicle's horizon in order to allow the vehicle to react to the information.

Consider as an example the case of a vehicle approaching a corner of a rectangular search region from inside the region, where the vehicle detects that there is an out-of-bounds region on either side of it, but does not see that the point of the corner also leads out of bounds, because this point lies outside its depth of vision. In this example, the utility function would have to take into account the reduction in gain from leaving the search at the corner point by reducing the gain from an area within the vehicle's horizon so that the total reduction in gain visible to the vehicle will be approximately the same as if it could see all the way to the end of this horizon problem.

To do this, we define a function Γ such that $0 < \Gamma \leq 1$. The value of Γ is low when the vehicle is on a path that would lead out-of-bounds, and is unity when it is on a path that does not. Then, the new cost function becomes

$$g(x_k, u_k) = \Gamma * g_o(x_k, u_k) \tag{2}$$

where g_o is the old cost function.

(Note: our choice of terminology has only tangential relation to Utility Theoretic formulations, e.g., as in risk-sensitive control.)

2.5 *Interference – Random variable w_k*

Now the formulation is expanded to include the presence of other vehicles. Because of the limited look-ahead and the limited communication assumptions, each vehicle can no longer correctly calculate where the other vehicles are going to be, because each vehicle is now viewing only its own section of the environment. Because there are other vehicles in the search region over which the searching vehicle has no control and cannot accurately predict their positions, and because actual gain comes q time steps after the vehicle makes its choices, some consideration must be given to the fact that a vehicle's gain may not be what it had expected. Consider, for example, the situation depicted in Figure 2. Vehicle A's decision may have been optimal when it made it, but now another vehicle might come through the area, cause a certain region to be searched twice, and the result would be a gain that is less than if the vehicles were searching completely different areas. This is taken into account with a random variable w_k, which represents the loss in gain at time $k+q$ (compared to what was expected when the decision is made at time k) because of interference by another vehicle.

The function σ is defined to be a function that takes the state and a direction choice at time k and returns the gain from making that choice. The gain at each step is then what one would get if no interference were present minus the amount of gain one would lose from another vehicle interfering,

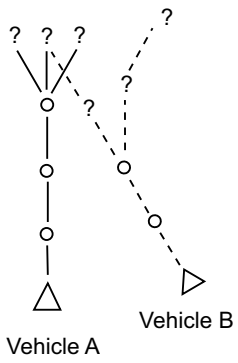

Fig. 2. Illustration of potential interference by other vehicles.

$$g(x_k, u_k, w_k) = \sigma(x_k, u_k) - w_k \tag{3}$$

This modified gain is then used in the DP recursion (Equation (1)) to determine the optimal path on the sub-tree.

The reduction in gain from searching an area where another vehicle is searching is dependent on the uncertainty level in that area as provided by the search map.

The amount of interference expected at a node (n_j) of the search tree is thus a function of $\sigma(n_j)$ and the probability of the vehicle interfering. So

$$E(w_k) = \rho \times \sigma(n_j) \times \mathrm{P}(n_j) \tag{4}$$

can be computed, where $\mathrm{P}(n_j)$ is the probability of vehicle j being at n_j, and ρ is a scalar that represents how much one sensor sweep reduces the uncertainty (since this calculation is being done by the second vehicle to sweep the area). It is important to note that the interference factors are not symmetric between interfering vehicles, since the first vehicle to search a region will receive full gain, even if the second vehicle will not.

2.6 *Complexity – Simplifying w_k*

Calculating $\mathrm{P}(n_j)$ exactly would require each vehicle to expand every other vehicle's planning tree, and assign a likelihood to each path that would cause that vehicle to interfere. Even assuming that this information is correct, this approach is impractical in the face of the computational complexity requirements.

A simplification is therefore introduced in order to produce an implementable solution. It is assumed that the distribution of probability over the tree has some structure in space. This structure is simplified into several spatial regions where each

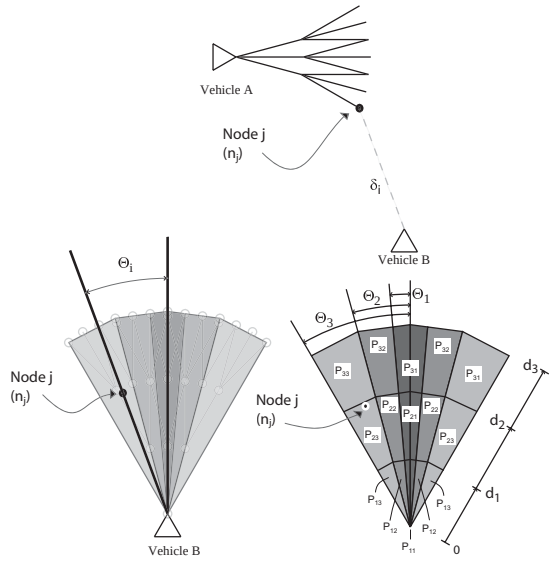

Fig. 3. A node (n_j) is checked for possible interference.

region represents a certain fixed probability of the vehicle causing an interference. Figure 3 shows a typical situation when this process would be used. Vehicle A has created a tree of possible paths to travel, and is checking to see to what degree Vehicle B may interfere, if any. At the end of each of these paths is a node (n_j). At each node, the distance to another vehicle (B in this case), called δ_i, is checked. If this distance is less than d_3, the maximum distance at which vehicles might interfere with one another, then the distance is compared to other distances, (d_1, d_2, and d_3 from the figure). Then the angle from the line extending along the interfering vehicle's current trajectory (θ_i) is compared to certain angles (θ_1, θ_2, and θ_3). These distances and angles form the regions from which the P(n_j) is drawn. If the node turns out to be outside of these regions, then the process produces a zero probability of interference term ($P(n_j) = 0$). Once this approximated P(n_j) is known, then $E(w_k)$ can be calculated using Equation (4).

Thus cooperation is achieved without the need for a vehicle to expand any planning trees but its own. The cooperation is of a passive sort, also, since the vehicles do not negotiate. Instead, they use the communication channel only to send knowledge of their locations and headings. The structure of the regions, also, is flexible, and the values can be tuned for better results or to incorporate new information, opening up the possibility of online adjustment.

3. SIMULATION

The simulation presented here show the proposed DP formulation in use. The environment used for this example, in Figure 4, shows a typical scenario a vehicle may face when searching an area for ground targets e.g. as part of a military strike.

The vehicle's paths are overlaid on top of the map of this environment. The lightly shaded round region represents a lake that is known a priori to have no value to be gained by being searched (and starts at 0% uncertainty), since, in this example, the ground targets are known not to be able to float. The dark shaded square regions are forests that are known a priori to have a low (25% uncertainty) likelihood of having targets of interest, justified in this example by saying that the targets have difficulty moving through rough terrain. The thin bar running vertically across the terrain (and through a forest) represents a road, which a priori is known to provide a large gain when searched (100% uncertainty), since the targets are most likely to have moved here. The remaining terrain has no a priori information associated with it, and is assumed to be of some importance to search (95% uncertainty). This represents flat ground that can easily have something of interest.

For this scenario, $\rho = 50\%$, so one pass of the vehicle over a region with 100% uncertainty would make it become 50%; a region with 50% would become 25%, and so on. A best-first label correcting method was used to search the path trees (Bertsekas, 2000). A vehicle's visible planning horizon (r) was six time steps ahead, or 30 units (they fly at 5 units per time step.) The utility function discussed in this paper was also included in the cost function, in which Γ was proportional to the distance from a vehicle to the corner if the vehicle was heading towards the corner and was within a minimum range (30 units).

The vehicle's paths in Figure 4 show a sample of some of the typical behaviors of the vehicles, after running the simulation for a short time. Desired behavior for the vehicles are for them to avoid the areas with low uncertainty, and also to make sure to search the road, which a priori information indicated should be searched well. Logically, they should also avoid searching the same area searched by another vehicle. Due to the interest of space constraints, just this one example is provided to show how the algorithm can make the vehicles behave in an efficient manner, in which the vehicles avoid the low gain areas. Also, one vehicle finds the road and travels along it for most of its length. Initially the vehicles spread out, and after that, whenever their paths cross, it tends to avoid low angles of intersection in order to keep the amount of overlap as small as possible.

Figure 5 shows the proposed DP search algorithm, the results of a standard, or lawnmower, type search, and a random type search, all after running for a short time. A lawnmower search, in which the vehicles systematically traverse the

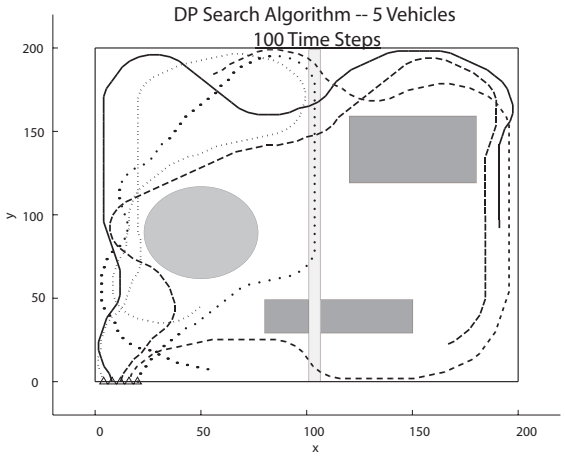

Fig. 4. Five vehicles searching an environment where the a priori information is shown in the darker shapes.

environment in lanes (just like someone who is mowing a lawn,) is very effective in the case where no a priori information is known, but Figure 6 shows that, in the presence of a priori information such as that shown in the problem of Figure 4, the search algorithm that is proposed in this paper gives better results. The bottom scale in Figure 6 is the time spent searching. The vertical scale shows the average uncertainty of a unit on the map of the environment, where lower uncertainty is the desired outcome. Note that the time spent searching in Figure 6 is longer than that displayed for the other figures. Note also that the map does not start at 1 (100%) uncertainty at time 0 because of the a priori information. As expected, both types of structured search perform better than a purely random search, in which the vehicles simply travel without regard for the environment or each other unless they reach an out-of-bounds, when they turn around.

4. CONCLUSION

Modeling the cooperative path planning problem in a Stochastic Dynamic Programming framework produces a unique and powerful approach that provides many tools for the problem of autonomous agents planning in an uncertain environment. It provides a platform for not only feasible solutions to existing problems but also a flexible framework to expand upon.

5. REFERENCES

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

Jacques, D.R. and R. Leblanc (1998). Effectiveness analysis for wide area search munitions.

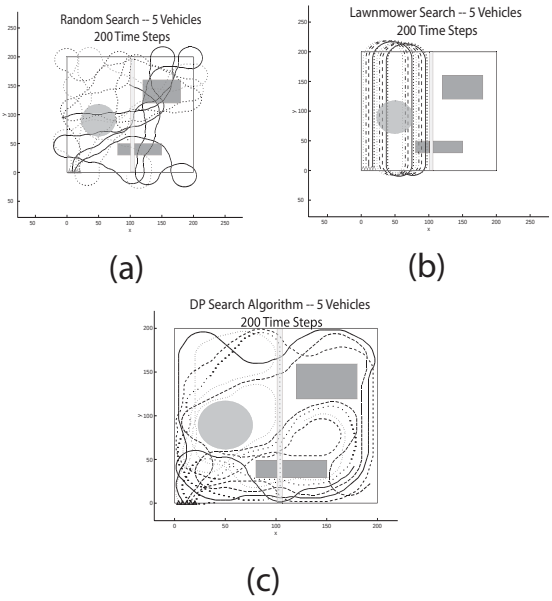

Fig. 5. Examples of the (a) random search (b) standard (lawnmower) algorithm (c) DP algorithm.

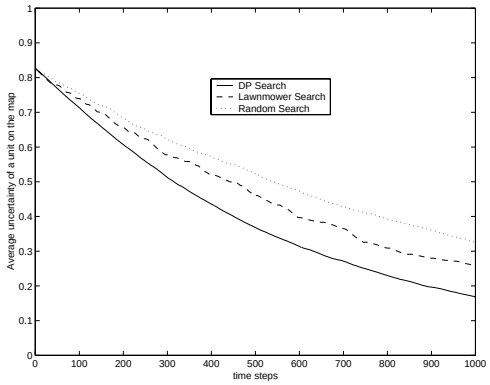

Fig. 6. Performance comparison of the proposed DP algorithm to other types of search algorithms.

In: *Proceedings of the AIAA Missile Sciences Conference*. Monterey, CA.

Pachter, M. and P. Chandler (1998). Challenges of autonomous control. *IEEE Control Systems Magazine* pp. 92–97.

Patek, S.D., D.A. Logan and D.A. Castanon (1999). Approximate dynamic programming for the solution of multiplatform path planning problems. In: *In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. 1999*. Vol. 1. pp. 1061–1066.

Polycarpou, M., Y. Yang and K. Passino (2001). A cooperative search framework for distributed agents. In: *Proceedings of the 2001 IEEE International Symposium on Intelligent Control*. pp. 1–6.