# A Cooperative Search Framework for Distributed Agents

Marios M. Polycarpou[†], Yanli Yang[†] and Kevin M. Passino[‡]

[†]Department of Electrical and Computer Engineering and Computer Science
University of Cincinnati, Cincinnati, OH 45221-0030, USA
polycarpou@uc.edu, yangyanl@ececs.uc.edu

[‡]Department of Electrical Engineering, The Ohio State University
2015 Neil Avenue, Columbus, OH 43210-1272, USA
passino@ee.eng.ohio-state.edu

*Abstract*— **This paper presents an approach for cooperative search of a team of distributed agents. We consider two or more agents, or vehicles, moving in a geographic environment, searching for targets of interest and avoiding obstacles or threats. The moving agents are equipped with sensors to view a limited region of the environment they are visiting, and are able to communicate with one another to enable cooperation. The agents are assumed to have some "physical" limitations including possibly maneuverability limitations, fuel/time constraints and sensor range and accuracy. The developed cooperative search framework is based on two inter-dependent tasks: (i) on-line learning of the environment and storing of the information in the form of a "search map"; and (ii) utilization of the search map and other information to compute on-line a guidance trajectory for the agent to follow. The distributed learning and planning approach for cooperative search is illustrated by computer simulations.**

## I. Introduction

During the last decade there has been significant progress in the design and analysis of intelligent control schemes. These techniques have enhanced the overall effectiveness of decision and control methods mainly in two frontiers. First, they enhanced the ability of feedback control systems to deal with greater levels of modeling uncertainty. For example, on-line approximation techniques, such as neural networks, allow the design of control systems that are able to "learn" on-line unknown, nonlinear functional uncertainties and thus enhance the overall performance of the closed-loop system in the presence of significant modeling uncertainty. Second, intelligent control techniques have enhanced our ability to deal with greater levels of uncertainty in the environment by providing methods for designing more autonomous systems with high-level decision making capabilities (outer-loop control). In this framework, high-level decision making may deal, for example, with generating on-line a guidance trajectory for the low-level controller (inner-loop control) to follow, or with designing a switching strategy for changing from one control scheme to another in the presence of changes in the environment or after the detection of a failure.

This research was financially supported by DAGSI and AFRL under the project entitled "Distributed Cooperation and Control for Autonomous Air Vehicles." Please address any correspondence to Marios Polycarpou (polycarpou@uc.edu).

In this paper, we address a problem in the second framework of intelligent control, as described above. Specifically, we present an approach for cooperative search among a team of distributed agents. Although the presented framework is quite general, the main motivation for this work is to develop and evaluate the performance of strategies for cooperative control of autonomous air vehicles that seek to gather information about a dynamic target environment, evade threats, and possibly coordinate strikes against targets. Recent advances in computing, wireless communications and vehicular technologies are making it possible to deploy multiple unmanned air vehicles (UAVs) that operate in an autonomous manner and cooperate with one another to achieve a global objective [1], [2], [3]. A large literature of relevant ideas and methods can also be found in the area of "swarm robotics" (e.g., see [4]) and, more generally, in coordination and control of robotic systems (e.g., see [5]). Search problems occur in a number of military and civilian applications, such as search-and-rescue operations in open-sea or sparsely populated areas, search missions for previously spotted enemy targets, seek-destroy missions for land mines, and search for mineral deposits. A number of approaches have been proposed for addressing search problems. These include optimal search theory [6], [7], exhaustive geographic search [8], obstacle avoidance [9] and derivative-free optimization methods [10].

We consider a team of vehicles moving in an environment of known dimension, searching for targets of interest. The vehicles are assumed to be equipped with: 1) target sensing capabilities for obtaining a limited view of the environment; 2) wireless communication capabilities for exchanging information and cooperating with one another; and 3) computing capabilities for processing the incoming information and making on-line guidance decisions. It is also assumed that each vehicle has a tandem of actuation/sensing hardware and an inner-loop control scheme for path following. In this paper, we focus solely on the design of the guidance controller (outer-loop control), and for convenience we largely ignore the vehicle dynamics.

The vehicles are assumed to have some maneuverability limitations, which constrain the maximum turning radius

of the vehicle. The developed cooperative search framework is based on two inter-dependent tasks: (i) on-line learning of the environment and storing of the information in the form of a "search map"; and (ii) utilization of the search map and other information for computing on-line a guidance trajectory for the vehicle. The distributed learning and planning approach for cooperative search is illustrated by computer simulations.

While there are different command-flow configurations that can be deployed (such as a hierarchical configuration or having one leader coordinate all the activities of the group), in this paper we consider the vehicles as a team of autonomous agents which exchange information but ultimately make their own decisions based on the received information. In the rest of the paper, we will be using the general term "agent" to represent a UAV or other type of appropriate vehicle.

## II. Distributed Guidance and Control Architecture

We consider $N$ agents deployed in some search region $\mathcal{X}$ of known dimension. As each agent moves around in the search region, it obtains sensory information about the environment in the form of automatic target recognition (ATR) data. It also receives information from other agents via a wireless communication channel, which may be coming at a different rate (usually at a slower rate) than the sensor information from its own sensors.

Depending on the specific application, the global objective pursued by the team of agents may be different. In this paper, we focus mainly on the problem of cooperative search, where the team of agents seeks to follow a trajectory that would result in minimization of the uncertainty about the environment; however, the presented framework can be easily expanded to include more advanced missions such as evading threats, attacking targets, etc.

Each agent has two basic control loops that are used in guidance and control, as shown in Figure 1. The "outer-loop" controller for agent $\mathcal{A}_i$ utilizes sensor information $v_i$ from $\mathcal{A}_i$, as well as sensor information from $\mathcal{A}_j$, $j \neq i$, to compute on-line a desired trajectory (path) to follow, which is denoted by $P_i(k)$. The sensor information coming from other agents is represented by the vector

$$V_i = [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_N]^\top ,$$

where $v_j$ represents the information received from agent $\mathcal{A}_j$. The above formulation doesn't necessarily assume that all agents are in range and can communicate with each other; it can also be used for the case where some of the information from other agents is missing, or the information from different agents is received at different sampling rates. The desired trajectory $P_i(k)$ is generated as a digitized look-ahead path of the form

$$P_i(k) = \{p_i(k),\ p_i(k+1),\ \ldots,\ p_i(k+q)\},$$

where $p_i(k+j)$ is the desired location of agent $\mathcal{A}_i$ at time $k+j$, and $q$ is the number of look-ahead steps in the path planning procedure.

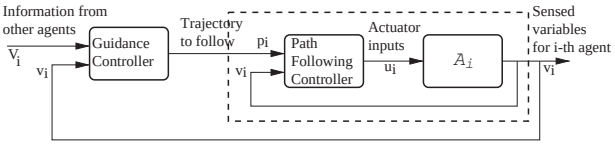


Fig. 1. Inner- and outer-loop controllers for guidance and control of air vehicles.

The inner-loop controller uses sensed information $v_i$ from $\mathcal{A}_i$ to generate inputs $u_i$ to the actuators of $\mathcal{A}_i$ so that the agent will track the desired trajectory $P_i(k)$. We largely ignore the agent dynamics, and hence concentrate on the outer-loop control problem. In this way, our focus is solidly on the development of the controller for guidance, where the key is to show how resident information of agent $\mathcal{A}_i$ can be combined with information from other agents so that the team of agents can work together to minimize the uncertainty in the search region $\mathcal{X}$.

The design of the outer-loop control scheme is broken down into two basic functions, as shown in Figure 2. First, it uses the sensor information received to update its "search map", which is a representation of the environment—this will be referred to as the agent's *learning* function, and for convenience it will be denoted by $\mathcal{L}_i$. Based on its search map, as well as other information (such as its location and direction, the location and direction of the other agents, remaining fuel, etc.), the second function is to compute a desired path for the agent to follow—this is referred to as the agent's *guidance decision* function, and is denoted by $\mathcal{D}_i$. In this setting the guidance control decisions made by each agent are autonomous, in the sense that no agent tells another what to do in a hierarchical type of structure, nor is there any negotiation between agents. Each agent simply receives information about the environment from the remaining agents (or a subset of the remaining agents) and makes its decisions, which are typically based on enhancing a global goal, not only its own goal. Therefore, the presented framework can be thought of as a *passive cooperation* framework, as opposed to *active cooperation* where the agents may be actively coordinating their decisions and actions.

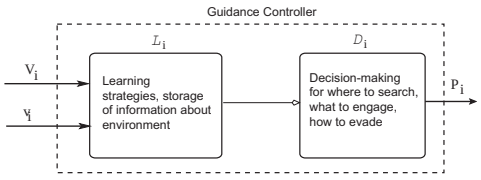


Fig. 2. Learning and decision-making components of the outer-loop controller for trajectory generation of air vehicles.

## III. Distributed Learning

Each agent has a three dimensional map, which we will refer to as "search map," that serves as the agent's knowledge base of the environment. The $x$ and $y$ coordinates of the map specify the location in the target environment (i.e., $(x, y) \in \mathcal{X}$), while the $z$ coordinate specifies the cer-

tainty that the agent "knows" the environment at that point. The search map will be represented mathematically by an on-line approximation function as $z = \mathcal{S}(x, y; \theta)$, where $z \in [0,\ 1]$. If $\mathcal{S}(x, y; \theta) = 0$ then the agent knows nothing (is totally uncertain) about the nature of the environment at $(x, y)$. On the other hand, if $\mathcal{S}(x, y; \theta) = 1$ then the agent knows everything (or equivalently, the agent is totally certain) about the environment at $(x, y)$. As the agent moves around in the search region it incorporates the new information about the environment gathered by itself and received from other agents. Therefore, the search map of each agent is continuously evolving as new information about the environment is collected and processed.

We define $\mathcal{S} : \mathcal{X} \times \Re^q \mapsto [0,\ 1]$ to be an on-line approximator (for example, a neural network), with a fixed structure whose input/output response is updated on-line by adapting a set of adjustable parameters, or weights, denoted by the vector $\theta \in \Re^q$. The weight vector $\theta(k)$ is updated based on an on-line learning scheme, as is common, for example, in training algorithms of neural networks.

While it is possible to create a simpler memory/storage scheme (without learning) that simply records the information received from the sensors, a learning scheme has some key advantages: 1) it allows generalization between points; 2) information from different types of sensors can be recorded in a common framework (on the search map) and discarded; 3) it allows greater flexibility in dealing with information received from different angles; 4) in the case of dynamic environments (for example, targets moving around), one can conveniently make adjustments to the search map to incorporate the changing environment (for example, by reducing the output value $z$ over time using a decay factor).

The search map is formed dynamically as the agent moves, gathers information about the environment, and processes the information. This is illustrated in Figure 3, where we show the area scanned by a "generic" sensor on a UAV during a sampling period $[kT,\ \ kT+T]$ where $T > 0$ is the sampling time. Although in different applications the shape of the scanned area maybe be different, the main idea remains the same. The received data can then be digitized and each grid point is used to adjust the search map $\mathcal{S}(x, y; \hat{\theta})$ by adapting $\hat{\theta}$.

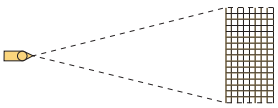

Fig. 3. An example of a scan area for a UAV.

In practice, in addition to the problem of minimizing the uncertainty in the search region, the overall objective may include, for example, finding specific targets, or avoiding certain obstacles and threats. Therefore, the learning scheme described above for minimizing uncertainty may need to be expanded by changing the value region or the dimension of $z$ . For example, one possible way to include a mission of searching for specific targets can be achieved by allowing the output $z$ of the on-line approximator $\mathcal{S}$ to take values in the region $z \in [-1,\ 1]$, where: $z = 1$ represents high certainty that a target is present at $(x, y)$; $z = -1$ represents high certainty that a target is not present at $(x, y)$; and $z = 0$ represents total uncertainty whether a target is present at $(x, y)$.

In this general framework, the tuning of the search map can be viewed as "learning" the environment. Mathematically, $\mathcal{S}$ tries to approximate an unknown function $\mathcal{S}^*(x, y, k)$, where for each $(x, y)$, the function $\mathcal{S}^*$ characterizes the presence (or not) of a target; the time variation indicated by the time step $k$ is due to (possible) changes in the environment (such as having moving targets). Hence, the learning problem is defined in using information from agent $\mathcal{A}_i$ and from other agents $\mathcal{A}_j$, $j \neq i$ at each sampled time $k$, to adjust the weights $\hat{\theta}(k)$ such that

$$\left\| \mathcal{S}(x, y; \hat{\theta}(k)) - \mathcal{S}^*(x, y, k) \right\|_{(x,y)\in\mathcal{X}}$$

is minimized.

Due to the nature of the learning problem, it is convenient to use spatially localized approximation models so that learning in one region of the search space does not cause any "unlearning" at a different region [11]. In general, the learning problem in this application is straightforward, and the use of simple approximation functions and learning schemes is sufficient; e.g., the use of piecewise constant maps or radial basis function networks, with distributed gradient methods to adjust the parameters, provides sufficient learning capability. However, complexity issues do arise and are crucial since the distributed nature of the architecture imposes limits not only on the amount of memory and computations needed to store and update the maps but also in the transmission of information from one agent to another.

At the time of deployment, it is assumed that each agent has a copy of an initial search map estimate, which reflects the current knowledge about the environment $\mathcal{X}$. In general, each agent is initialized with the same search map. However, in some applications it may be useful to have agents be "specialized" to search in certain regions, in which case the search environment for each agent, as well as the initial search map, may be different.

## IV. Cooperative Path Planning

One of the key objectives of each agent is to on-line select a suitable path in the search environment $\mathcal{X}$. To be consistent with the motion dynamics of physical vehicles (and, in particular, air vehicles), it is assumed that each agent has limited maneuverability, which is represented by a maximum angle $\theta_m$ that the agent can turn from its current direction. For simplicity we assume that all agents move at a constant velocity $\mu$ (this assumption can be easily relaxed).

### A. Plan Generation

To describe the movement path of agent $\mathcal{A}_i$ between samples, we define the *movement sampling time* $T_m$ as the time interval in the movement of the agent. In this framework,

we let $p_i(k)$ be the position (in terms of $(x, y)$ coordinates) of $i$-th agent at time $t = kT_m$, with the agent following a straight line in moving from $p_i(k)$ to its new position $p_i(k+1)$. Since the velocity $\mu$ of the agent is constant, the new position $p_i(k+1)$ is at a distance $\mu T_m$ from $p_i(k)$, and based on the maneuverability constraint, it is within an angle $\pm\theta_m$ from the current direction, as shown in Figure 4. To formulate the optimization problem as an integer programming problem, we discretize the arc of possible positions for $p_i(k+1)$ into $m$ points, denoted by the set

$$\overline{\mathcal{P}}_i(k+1) = \left\{ \bar{p}_i^1(k+1),\ \ldots\ \bar{p}_i^j(k+1),\ \ldots \bar{p}_i^m(k+1) \right\}.$$

Therefore, the next new position for the $i$-th agent belongs to one of the elements of the above set; i.e., $p_i(k+1) \in \overline{\mathcal{P}}_i(k+1)$.

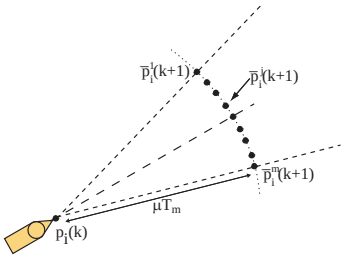


Fig. 4. Selection of the next point in the path of the vehicle.

The agent selects a path by choosing among a possible set of future position points. In our formulation we allow for a recursive $q$-step ahead planning, which can be described as follows:

- When agent $\mathcal{A}_i$ is at position $p_i(k)$ at time $k$, it has already decided the next $q$ positions: $p_i(k+1)$, $p_i(k+2)$, $\ldots$, $p_i(k+q)$.
- While the agent is moving from $p_i(k)$ to $p_i(k+1)$ it selects the position $p_i(k+q+1)$, which it will visit at time $t = k+q+1$.

To get the recursion started, the first $q$ positions, $p_i(1)$, $p_i(2)$, $\ldots$, $p_i(q)$ for each agent need to be selected a priori. Clearly, $q = 1$ corresponds to the special case of no planning ahead. The main advantage of a planning ahead algorithm is that it creates a buffer for path planning. From a practical perspective this can be quite useful if the agent is an air vehicle that requires (at least) some trajectory planning. The recursive $q$-step ahead planning procedure is illustrated in Figure 5 for the case where $q = 6$.

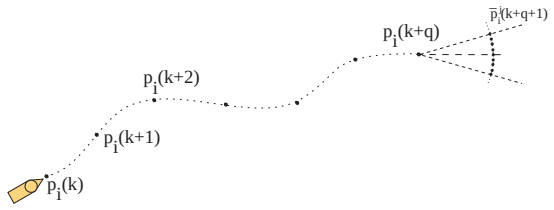


Fig. 5. Illustration of the recursive $q$-step ahead planning algorithm.

### B. Plan Selection

Given the current information available via the search map, and the location/direction of the team of agents (and possibly other useful information, such as fuel remaining, etc.), each agent uses a multi-objective cost function $J$ to select and update its search path. At decision sampling time $T_d$, the agent evaluates the cost function associated with each path and selects the optimal path. The decision sampling time $T_d$ is typically equal to the movement sampling time $T_m$. The approach can be thought of as an "adaptive model predictive control" approach where an adaptive model is developed on-line that is used to predict ahead in time, and on-line optimization methods are employed in the formation of the model and in evaluating the candidate paths to move the agent along.

A key issue in the performance of the cooperative search approach is the selection of the multi-objective cost function associated with each possible path. The approach followed in this paper is quite flexible in that it allows the characterization of various mission-level objectives, and trade-offs between these. In general, the cost function comprises of a number of sub-goals, which are sometimes competing. Therefore the cost criterion $J$ can be written as:

$$J = \omega_1 J_1 + \omega_2 J_2 + \ldots + \omega_s J_s$$

where $J_i$ represents the cost criterion associated with the $i$-th subgoal, and $\omega_i$ is the corresponding weight. The weights are normalized such that $0 \leq \omega_i \leq 1$ and the sum of all the weights is equal to one; i.e., $\sum_{i=1}^{s} \omega_i = 1$. Priorities to specific sub-goals is achieved by adjusting the values of weights $\omega_i$ associated with each subgoal.

The following is a list (not exhaustive) of possible subgoals that a search agent may include in its cost criterion. For a more clear characterization, these sub-goals are categorized according to three mission objectives: Search (S), Cooperation (C), and Engagement (E). In addition to subgoals that belong purely to one of these classes, there are some that are a combination of two or more missions. For example, SE1 (see below) corresponds to a search and engage mission.

**S1** *Follow the path where there is maximum uncertainty in the search map.* This sub-goal simply considers the uncertainty reduction associated with the sweep region between the current position $p_i(k)$ and each of the possible candidate positions $\bar{p}_i^j(k+1)$ for the next sampling time (see the rectangular regions between $p_i(k)$ and $\bar{p}_i^j(k+1)$ in Figure 6). The cost criterion can be derived by computing a measure of uncertainty in the path between $p_i(k)$ and each candidate future position $\bar{p}_i^j(k+1)$.

**S2** *Follow the path that leads to the region with the maximum uncertainty (on the average) in the search map.* The first cost criterion S1 pushes the agent towards the path with the maximum uncertainty. However, this may not be an optimal path, over a longer period of time, if it leads to a region where the average uncertainty is low. Therefore, it's important for the search agent to seek not only the instantaneous minimizing path, but also a path that will

cause the agent to visit (in the future) regions with large uncertainty. The cost criterion can be derived by computing the average uncertainty of a triangular type of region associated with the heading direction of the agent (see the triangular regions ahead of $\bar{p}_i^j(k+1)$ in Figure 6).

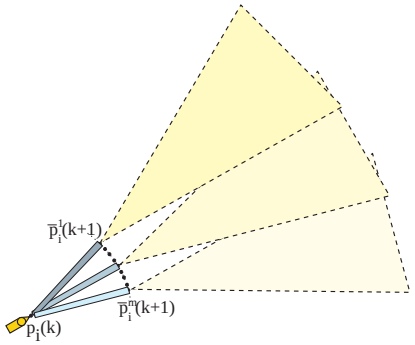


Fig. 6. Illustration of the regions that are used in the cost function for finding the optimal search path.

**C1** *Follow the path where there is the minimum overlap with other agents.* Since the agents are able to share their new information about the search region, it is natural that they may select the same search path as other agents (especially since in general they will be utilizing the same search algorithm). This will be more pronounced if two agents happen to be close to each other. However, in order to minimize the global uncertainty associated with the emergent knowledge of all agents, it is crucial that there is minimum overlap in their search efforts. This can be achieved by including a cost function component that penalizes agents being close to each other and heading in the same direction. This component of the cost function can be derived based on the relative locations and heading direction (angle) between pairs of agents.

**SE1** *Follow the path that maximizes coverage of the highest priority targets.* In mission applications where the agents have a target search map with priorities assigned to detected targets, it is possible to combine the search of new targets with coverage of discovered targets by including a cost component that steers the agent towards covering high priority targets. Therefore, this leads to a coordinated search where both coverage and priorities are objectives.

**E1** *Follow the path toward highest priority targets with most certainty if fuel is low.* In some applications, the energy of the agent is limited. In such cases it is important to monitor the remaining fuel and possibly switch goals if the fuel becomes too low. For example, in search-and-engage operations, the agent may decide to abort search objectives and head towards engaging high priority targets if the remaining fuel is low.

**EC1** *Follow the path toward targets where there will be minimum overlap with other agents.* Cooperation between agents is a key issue not only in search patterns but also—and even more so—in engagement patterns. If an agent decides to engage a target, there needs to be some cooperation such that no other agent tries to go after the same target; i.e., a coordinated dispersed engagement is desirable.

The above list of sub-goals and their corresponding cost criteria provide a flavor of the type of issues associated with the construction of the overall cost function for a general mission. In addition to incorporating the desired sub-goals into the cost criterion (i.e., maximize benefit), it is also possible to include cost components that reduce undesirable sub-goals (minimize cost).

In the next section, we consider some simulation studies that are based on a cost function consisting of the first three sub-goals. Therefore the main goal is to search in a cooperative framework.

## V. Simulation Results

The proposed cooperative search and learning framework has been tested in several simulated studies. Two of these simulation studies are presented in this section. In the first simulation experiment a team of five agents is employed to show the performance of the proposed cooperative search scheme as compared to other search methods, while in the second simulation we use two agents to illustrate the importance of cooperative behavior. In both simulation studies we are using the recursive $q$-step ahead planning algorithm with $q = 3$.

The results for the case of five agents are shown in Figure 7. The upper-left plot shows a standard search pattern(zamboni coverage pattern [12]) for the first 500 time samples, while the upper-right plot shows the corresponding result for a random search, which is subject to the maneuverability constraints. And the lower-left plot shows the result of the cooperative search method. The search region is a 200 by 200 area, and the five search agents start at the location indicated by the triangles. It is assumed that there is some a priori information about the search region: the light polygons indicate complete certainty about the environment (for example, these can represent regions where are no targets for sure due to the terrain); the dark polygons represent partial certainty about the environment. The remaining search region is assumed to be completely uncertain. In this simulation the only mission is for the agents to work cooperatively in reducing the uncertainty in the environment.

The search map used in this simulation study is based on piecewise constant basis functions, and the learning algorithm is a simple update algorithm of the form $\hat{\theta}(k+1) = 0.5\hat{\theta}(k) + 0.5$, where the first encounter of a search block results in the maximum reduction in uncertainty. Further encounters result in reduced benefit. For example, if a block on the search map starts from certainty value of zero (completely uncertain) then after four visits from (possibly different) agents, the certainty value changes to $0 \mapsto 0.5 \mapsto 0.75 \mapsto 0.875 \mapsto 0.9375$. The percentage of uncertainty is defined as the distance of the certainty value from one. The cooperative search algorithm has no pre-set search pattern. As seen from Figure 7, each agent adapts its search path on-line based on current information from its search results, as well as from search results of the other agents. The lower-right plot of Figure 7 shows the per-

centage of uncertainty with time for the standard search pattern, the random search pattern and the cooperative search pattern described above. The ability of the cooperative search algorithm to make path planning decisions on-line results in a faster rate of uncertainty reduction.

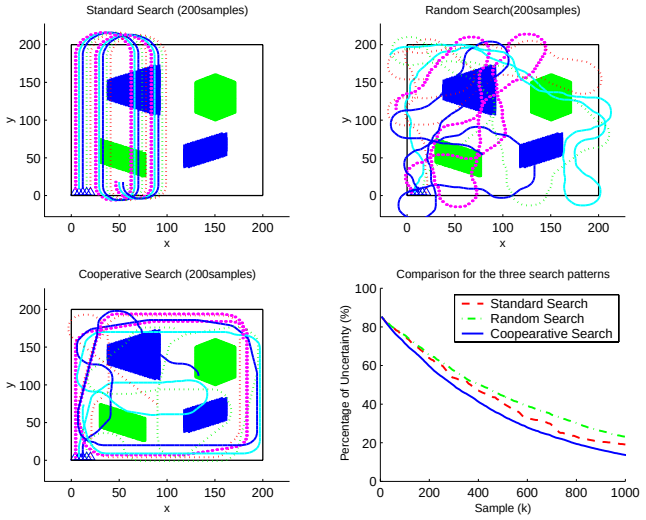


Fig. 7. Comparison of the cooperative search pattern with a "standard" search pattern and a random search pattern for the case of five moving agents.

The simulation results in the case of two agents moving in the same environment is shown in Figure 8. The left plot of Figure 8 shows that the paths of the two agents are overlapping if the cooperation sub-goal C1 is not included in the cost function $J$. The right plot of Figure 8 shows that by including C1 in the cost function, the algorithm can effectively seperate the two agents so as to search the region more efficiently.

In these simulation studies, we assume that the sampling time $T_m = 1$ corresponds to the rate at which each agent receives information from its own sensors, updates its search map and makes path planning decisions. Information from other agents is received at a slower rate at a communication sampling time $T_c = 5T_m$.

It is noted that in these simulations the path planning of the cooperative search algorithm is rather limited since at every sampled time each agent is allowed to either go straight, left, or right (the search direction is discretized into only three possible points; i.e., $m = 3$). As the complexity of the cooperative search algorithm is increased and the design parameters (such as the weights associated with the multi-objective cost function) are fine-tuned or optimized, it is anticipated that the search performance can be further enhanced.

## VI. Concluding Remarks

Advances in distributed computing and wireless communications have enabled the design of distributed agent systems. One of the key issues for a successful and wide deployment of such systems is the design of cooperative decision making and control strategies. Traditionally, feedback control methods have focused mostly on the design

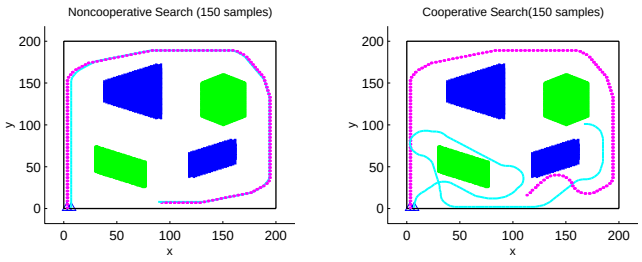


Fig. 8. Comparison of non-cooperative search pattern (left) and cooperative search pattern (right).

and analysis of centralized, inner-loop techniques. Decision and control of distributed agent systems requires a framework that is based more on cooperation between agents, and outer-loop schemes. In addition to cooperation, issues such as coordination, communication delays and robustness in the presence of losing one or more of the agents are crucial. In this paper, we have presented a framework for a special type of problem, the cooperative search. The proposed framework consists of two main components: learning the environment and using that knowledge to make intelligent high-level decisions on where to go (path planning) and what do to. We have presented some ideas regarding the design of a cooperative planning algorithm based on a recursive $q$-step ahead planning procedure and illustrated these ideas with simulation studies.

## References


[1] M. Pachter and P. Chandler, "Challenges of autonomous control," *IEEE Control Systems Magazine*, pp. 92–97, April 1998.

[2] D. Jacques and R. Leblanc, "Effectiveness analysis for wide area search munitions," in *Proceedings of the AIAA Missile Sciences Conference*, (Monterey, CA), Nov. 17–19 1998.

[3] D. Godbole, "Control and coordination in uninhabited combat air vehicles," in *Proceedings of the 1999 American Control Conference*, pp. 1487–1490, June 1999.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. NY: Oxford Univ. Press, 1999.

[5] R. Arkin, *Behavior-Based Robotics*. Cambridge, MA: MIT Press, 1998.

[6] L. Stone, *Theory of Optimal Search*. New York: Acadamic Press, 1975.

[7] B. Koopman, *Search and Screening: General principles with Historical Application*. New York: Pergarnon, 1980.

[8] S. Spires and S. Goldsmith, "Exhaustive geographic search with mobile robots along space-filling curves," in *Collective Robotics* (A. Drogoul, M. Tambe, and T. Fukuda, eds.), pp. 1–12, Springer Verlag: Berlin, 1998.

[9] S. Cameron, "Obstacle avoidance and path planning," *Industrial Robot*, vol. 21, pp. 9–14, 1994.

[10] A. Conn, K. Scheinberg, and P. Toint, "Recent progress in unconstrainted nonlinear optimization without derivatives," *Mathematical Programming*, vol. 79, pp. 397–414, 1997.

[11] S. Weaver, L. Baird, and M. Polycarpou, "An analytical framework for local feedforward networks," *iEEE Transactions on Neural Networks*, vol. 9, no. 3, pp. 473–482, 1998.

[12] V. Ablavsky and M. Snorrason, "Optimal search for a moving target: a geometric approach," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Denver, CO), August 2000.