Submitted to: *IEEE Control Systems Magazine* (June 2001)

Cooperative Control of Distributed Multi-Agent Systems*

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

‡Department of Electrical Engineering, The Ohio State University
2015 Neil Avenue, Columbus, OH 43210-1272, USA

Abstract

This paper presents an approach for cooperative search by a team of distributed agents. We consider two or more agents moving in a geographic environment, cooperatively 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 interdependent 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. We develop a real-time approach for on-line cooperation between agents, which is based on treating the paths of other vehicles as "soft obstacles" to be avoided. Based on artificial potential field methods we develop the concept of "rivaling force" between agents as a way of enhancing cooperation. The proposed distributed learning and planning approach is illustrated by computer simulations.

1 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 improve the overall performance of the closed-loop

*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).

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.

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 uninhabited air vehicles (UAVs) that operate in an autonomous manner and cooperate with each other to achieve a global objective [1, 2, 3, 4, 5]. A large literature of relevant ideas and methods can also be found in the area of "swarm robotics" (e.g., see [6, 7, 8]) and, more generally, coordination and control of robotic systems (e.g., see [9, 7, 10, 11, 12, 13, 14]). Related work also includes the techniques developed using the "social potential field" method [15, 16, 17] and multi-resolution analysis [18].

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 maneuverability constraint is an issue that is typically not encountered in some of the literature on "collective robotics," which describes swarms of robots moving in a terrain [19]. The main contributions of the work presented in this paper are the formulation of an on-line decision making framework for solving a class of cooperative search problems and the design of a real-time approach for on-line cooperation between agents. 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. We develop a real-time approach for on-line cooperation between agents based on treating the paths of other vehicles as "soft obstacles" to be avoided. Using artificial potential field methods we develop the concept of "rivaling force" between agents as a way of enhancing cooperation. 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 will consider the situation where each vehicle receives information from other vehicles but makes its own decisions on where to go and what to do. In other words, the group of vehicles can be considered as being a group of autonomous agents which exchange information but ultimately make their own decisions based on the received information. The problem of avoiding collisions between

vehicles is not directly addressed in this article; however, the proposed rivaling force approach for reducing path planning overlap can be extended to address the issue of avoiding collisions. In the rest of the paper, we will be using the general term "agent" to represent a UAV or other type of appropriate vehicle.

2 Related Research Work on Search Methods

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 such search problems. These include, among other, optimal search theory [20, 21], exhaustive geographic search [22], obstacle avoidance [23, 24] and derivative-free optimization methods [25].

Search theory deals with the problem of distribution of search effort in a way that maximizes the probability of finding the object of interest. Typically, it is assumed that some prior knowledge about the target distribution is available, as well as the "payoff" function that relates the time spent searching to the probability of actually finding the target, given that the target is indeed in a specific cell [20, 21]. Search theory was initially developed during World War II with the work of Koopmam and his colleagues at the Anti-Submarine Warfare Operations Research Group (ASWORG). Later on, the principles of search theory were applied successfully in a number of applications, including the search for and rescue of a lost party in a mountain or a missing boat on the ocean, the surveillance of frontiers or territorial seas, the search for mineral deposits, medical diagnosis, and the search for a malfunction in an industrial process. Detailed reviews of the current status of search theory have been given by Stone [26], Richardson [27], and Monticino [28].

The optimal search problem can be naturally divided according to two criteria that depend on the target's behavior. The first division depends on whether the target is evading or not; that is, whether there is a two-sided optimization by both the searcher and the target, or whether the target's behavior is independent of the searcher's action. The second division deals with whether the target is stationary or moving. The two divisions and their combinations form four different categories. A great deal of progress in solving stationary target problems in the optimal search framework has been made, and solutions have been derived for most of the standard cases [20]. For the moving target problem, the emphasis in search theory has shifted from mathematical and analytical solutions to algorithmic solutions [28]. A typical type of search problem, called the path constraint search problem (PCSP), that takes into account the movement of the searcher, was investigated by several researchers [29, 30, 31, 32]. Because of the NP-complete nature of this problem, most authors proposed a number of heuristic approaches that result in "approximately optimal" solutions. The two-sided search problem can be treated as a game problem for both the searcher and target strategies. This has been the topic of a number of research works [33, 34, 35]. So far, search theory has paid little attention to the problem of having a team of cooperating searchers. A number of heuristic methods for solving this problem have been proposed by Dell and Eagle [36].

The Exhaustive Geographic Search problem deals with developing a complete map of all phenomena of interest within a defined geographic area, subject to the usual engineering constraints of efficiency, robustness and accuracy [22]. This problem received much attention recently, and algorithms have been developed that are cost-effective and practical. Application examples of

Exhaustive Geographic Search include mapping mine fields, extraterrestrial and under-sea exploration, exploring volcanoes, locating chemical and biological weapons and locating explosive devices [22, 37, 38, 39].

The obstacle avoidance literature deals with computing optimal paths given some kind of obstacle map. The intent is to construct a physically realizable path that connects the initial point to the destination in a way that minimizes some energy function while avoiding all the obstacles along the route [23, 24]. Obstacle avoidance is normally closely geared to the methods used to sense the obstacles, as time-to-react is of the essence. The efficiency of obstacle avoidance systems is largely limited by the reliability of the sensors used. A popular way to solve the obstacle avoidance problem is the potential field technique [40]. According to the potential field method, the potential gradient that the robot follows is made up of two components: the repulsive effect of the obstacles and the attractive effect of the goal position. Although it is straightforward to use potential field techniques for obstacle avoidance, there are still several difficulties in using this method in practical vehicle planning.

Derivative-Free Optimization methods deal with the problem of minimizing a nonlinear objective function of several variables when the derivatives of the objective function are not available [25]. The interest and motivation for examining possible algorithmic solutions to this problem is the high demand from practitioners for such tools. The derivatives of objective function are usually not available either because the objective function results from some physical, chemical or economical measurements, or, more commonly, because it is the result of a possibly very large and complex computer simulation. The occurrence of problems of this nature appear to be surprisingly frequent in the industrial setting. There are several conventional deterministic and stochastic approaches to perform optimization without the use of analytical gradient information or measures of the gradient. These include, for example, the pattern and coordinate search [41, 42], the Nelder and Mead Simplex Method [43], the Parallel Direct Search Algorithm [44], and the Multi-directional Search Method [45]. In one way or another, most derivative free optimization methods use measurements of the cost function and form approximations to the gradient to decide which direction to move. Passino [46] provides some ideas on how to extend non-gradient methods to team foraging.

3 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, which helps to reduce the uncertainty about the environment. This sensory information can be in the form of an image, which can be processed on-line to determine the presence of a certain entity or target. Alternatively, it can be in the form of a sensor coupled with automatic target recognition (ATR) software. In addition to the information received from its own sensors, each agent also receives information from other agents via a wireless communication channel. The information received from other agents can be in raw form or it may be pre-processed, and it may be coming at a different rate (usually at a slower rate) or with a delay, as compared to the sensor information received by the agent 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 maximum gain in information about the environment; i.e., the objective is to minimize the uncertainty about the environment. Intuitively,

each agent wants to follow a trajectory that leads to regions in $\mathcal{X}$ that have not been visited frequently before by the team of agents. Alternatively, if some information about the location of targets is known (e.g., in terms of a probability density distribution), the team of agents seeks to coordinate its activities so as to reach as large a number of targets as quickly as possible. The presented framework can be easily expanded to include more advanced missions such as evading threats, attacking targets, etc. In general, the team may have an overall mission that combines several of these objectives according to some desired priority. However, for simplicity in this paper we will be focusing mostly on the cooperative search problem.

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 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 utilized in the feedback loop is denoted by v_i and may include information from standard vehicle sensors (e.g. pitch, yaw, etc.) and information from on-board sensors that has been pre-processed by resident ATR software. 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 $\mathcal{A}_j$. Although in the above formulation it appears that all agents are in range and can communicate with each other, this is not a required assumption—the same framework can 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, or with a communication delay. 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.

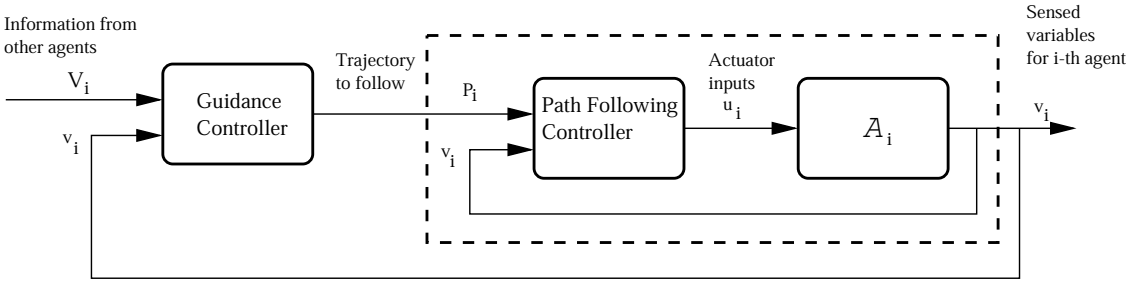

Figure 1: Inner- and outer-loop controllers for guidance and control of air vehicles.

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 we assume that 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.

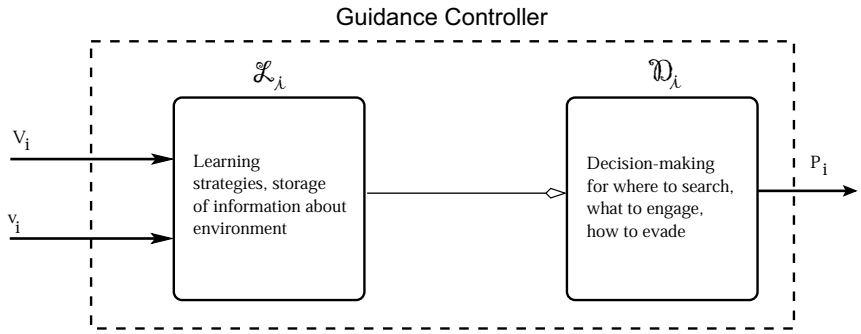

Figure 2: Learning and decision-making components of the outer-loop controller for trajectory generation of air vehicles.

4 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 certainty 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 (x, y) is a point in the search region $\mathcal{X}$, and the output $z \in [0,\ 1]$ corresponds to the certainty about knowing the environment at the point (x, y) in the search region. 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 gathers new information about the environment which is incorporated into its search map. Also incorporated into its search map is the information received by communication with 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 ad-

justable parameters, or weights, denoted by the vector $\theta \in \Re^q$. According to the standard neural network notation, (x, y) is the input to the network and z is the output of the network. 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.

In general, the search map serves as a storage place of the knowledge that the agent has about the environment. 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 $\hat{\theta}$.

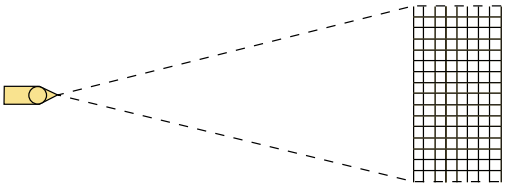

Figure 3: An example of a scan area for a UAV.

In practice, the problem of minimizing the uncertainty in the search region is typically an intermediate goal. The overall objective may include, for example, finding specific targets, or avoiding certain obstacles and threats. Therefore, depending on the application being considered, the learning scheme described above for minimizing uncertainty may need to be expanded. One possible way to include a mission of searching for specific targets is to incorporate the *search map* into a more general *target search map*, which in addition to providing information about the agent's knowledge of the environment, also contains information about the presence (or not) of targets. This 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:

- $z = \mathcal{S}(x, y; \theta) = 1$ represents high certainty that a target is present at (x, y);
- $z = \mathcal{S}(x, y; \theta) = -1$ represents high certainty that a target is not present at (x, y);
- $z = \mathcal{S}(x, y; \theta) = 0$ represents total uncertainty whether a target is present at (x, y).

This representation contains additional information that the agent can utilize in making guidance and path planning decisions. Furthermore, the learning framework can be extended to a multi-dimensional framework, where the output z of the on-line approximator is a vector of dimension

greater than one. For example, one could use the first output to represent the presence/absence of a target (as above), and the second output to represent the priority of the target.

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 as using sensor information from agent $\mathcal{A}_i$ and information coming 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 [47]. The dimension of the input space (x, y) is two, and therefore there are no problems related to the "curse of dimensionality" that are usually associated with spatially localized networks. 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 the special case that no a priori information is available, then each point on the search map is initialized as "completely uncertain." 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.

5 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 θ_m that the agent can turn from its current direction. For simplicity we assume that all agents move at a constant velocity μ (this assumption can be easily relaxed).

5.1 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 μ of the agent is constant, the new position $p_i(k + 1)$ is at a distance μT_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

$$\overline{\mathcal{P}}_i(k+1) = \left\{ \bar{p}_i^1(k+1),\ \bar{p}_i^2(k+1),\ \ldots\ \bar{p}_i^j(k+1),\ \ldots \bar{p}_i^m(k+1) \right\}.$$

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

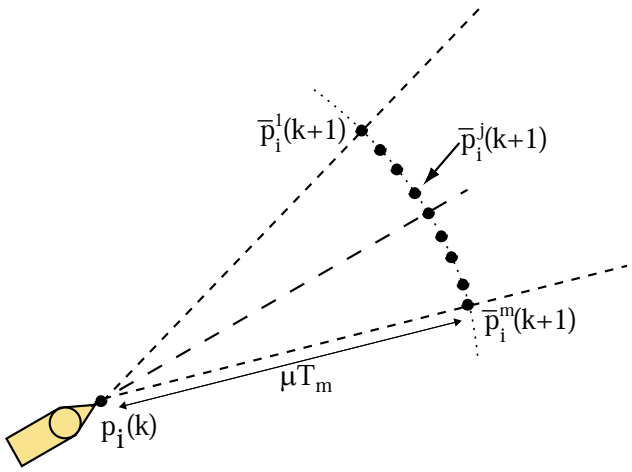

Figure 4: Selection of the next point in the path of the vehicle.

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: $p_i(k+1)$, $p_i(k+2)$, ..., $p_i(k+q)$.
- 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, $p_i(1)$, $p_i(2)$, ..., $p_i(q)$ 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. Planning ahead is also useful for cooperation between agents since it may be communicated to other vehicles as a guide of intended plan selection. This can be especially important if there are communication delays or gaps, or if the sampling rate for communication is slow. On the other hand, if the integer q is too large then, based on the recursive procedure, the position $p_i(k)$ was selected q samples earlier at time $k-q$; hence the decision may be outdated, in the sense that it may have been an optimal decision at time $k-q$, but based on the new information received since then, it may not be the best decision anymore. The recursive q-step ahead planning procedure is illustrated in Figure 5 for the case where $q = 6$.

If practical considerations (such as motion dynamics of the agent and computational demands for path selection) require a relatively large value for q then the problem of "outdated" decision

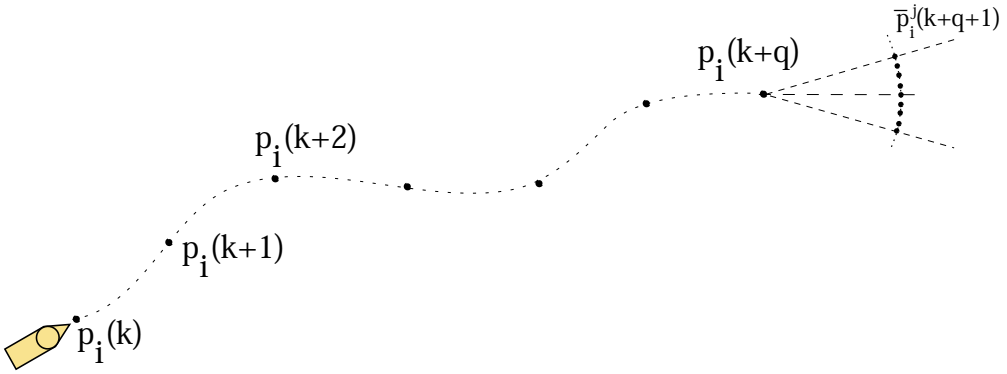

Figure 5: Illustration of the recursive q-step ahead planning algorithm.

making can be ameliorated by an *interleaved* type of scheme. We define a (q, r)-interleaved decision making scheme 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 re-calculates the last r points of the path based on the current data and also selects another new position; i.e., it selects the points $p_i(k+q-r+1)$, $p_i(k+q-r+2)$, $\ldots$, $p_i(k+q)$, $p_i(k+q+1)$.

The term "interleaved" is used to express the fact that decisions are re-calculated over time, as the agent moves, to incorporate new information that may have been received about the environment. According to this formulation, a (q, r)-interleaved decision scheme requires the selection of $r+1$ points for path planning at each sample T_m. The special case of $(q, 0)$-interleaved scheme (actually, strictly speaking it is a non-interleaved scheme) corresponds to the recursive q-step ahead planning scheme described earlier. Similar to the recursive q-step ahead planning scheme, at the beginning, the first q positions for each agent need to be selected a priori. The interleaved path planning procedure is illustrated in Figure 6 for the case where $q = 6$ and $r = 2$.

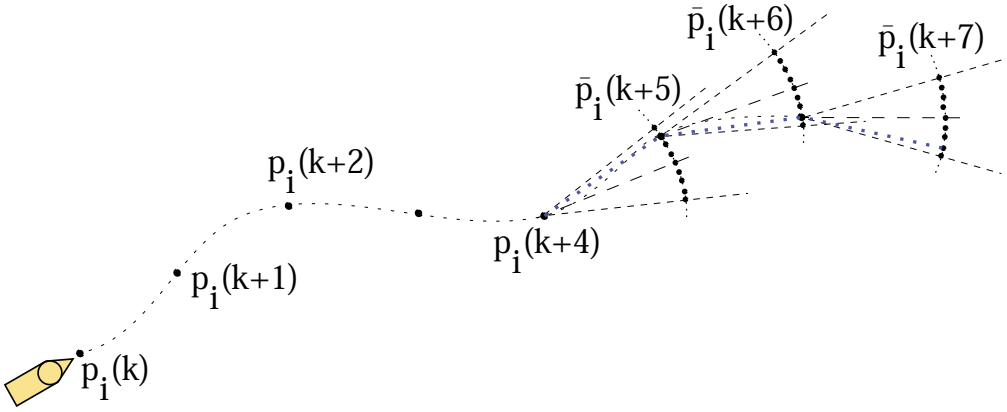

Figure 6: Illustration of the (q, r)-interleaved decision making procedure.

The computational complexity of an interleaved decision making scheme can be significantly

higher than the q-step ahead planning algorithm. Specifically, with the q-step ahead planning algorithm, each agent has to select one position among m possible candidates. With the (q, r)-interleaved algorithm, each agent has to select $r + 1$ positions among a combination of m^{r+1} candidates. Therefore, the computational complexity increases exponentially with the value of the interleaved variable r. This is shown in Figure 6 where $m = 9$, $r = 2$; therefore at each sample time the agent needs to select among $9^3 = 243$ possible paths in order to compute the three positions $p_i(5)$, $p_i(6)$ and $p_i(7)$. The figure shows a path of points generated by the guidance (outer-loop) controller, and then shows a tree of possible directions that the vehicle can take.

5.2 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 we learn the model that we use to predict ahead in time, and we use on-line optimization in the formation of that 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. Our approach 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 ω_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 are achieved by adjusting the values of weights ω_i associated with each subgoal.

The following is a list (not exhaustive) of possible sub-goals that a search agent may include in its cost criterion. Corresponding to each sub-goal is a cost-criterion component that need to be designed. 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 sub-goals 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 cost criterion 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 7). The cost criterion can be derived by computing a measure of uncertainty (or potential "gain" in knowledge) 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 pushes the agent towards the path with the maximum

uncertainty. However, this may not be the best 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 7).

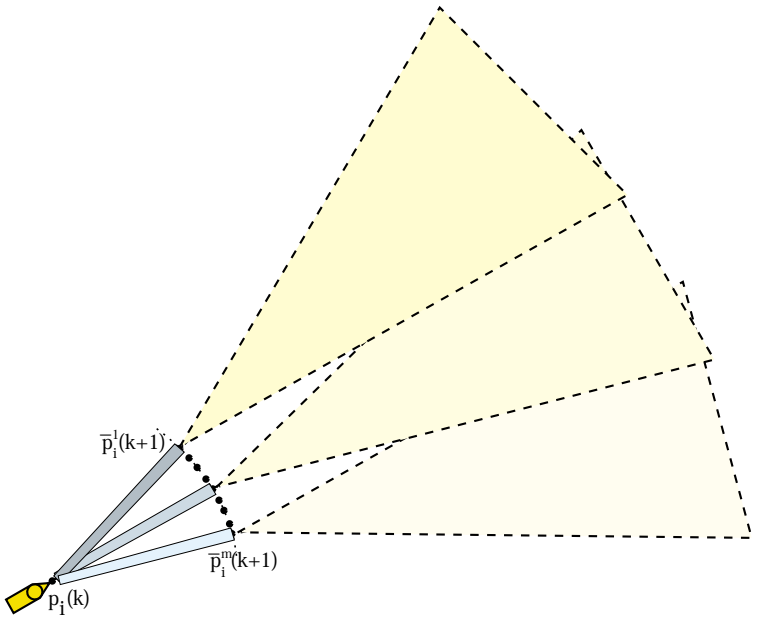

Figure 7: Illustration of the regions that are used in the cost function for finding the optimal search path.

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. This component of the cost function is investigated more thoroughly in Section 6.

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). For example, in order to generate a smooth trajectory for a UAV such that it avoids—as much as possible—the loss of sensing capabilities during turns, it may be desirable to assign an extra cost for possible future positions on the periphery (large angles) of the set $\overline{\mathcal{P}}_i$.

6 On-Line Cooperation Approach for Distributed Agents

The framework developed in this paper is based on distributed agents working together to enhance the global performance of a multi-agent system — in contrast to a framework where distributed agents may be competing with each other for resources. Therefore, one of the key issues in cooperative control is the ability of distributed agents to coordinate their actions and avoid overlap. In a decentralized environment, cooperation between agents may not come natural since every agent tries to optimize its own behavior. In typical complex scenarios it may not be clear to an individual agent how its own behavior is related to the global performance of the multi-agent system.

To illustrate this, consider the following "Easter egg hunt" scenario: each agent is asked to pick up Easter eggs from a field. For simplicity, we assume that the location of the eggs is known (no search is necessary). Each agent is initialized at some location in the field and its goal is to decide which direction to go. The velocity of each agent is fixed and once an agent is at the location of an egg then that egg is considered as having been picked. The global performance criterion is *for all the eggs to be pick up in the minimum possible time.* This simple scenario provides a nice framework for illustrating some of the key concepts of cooperative behavior. For example, an agent $\mathcal{A}_i$ may be tempted to head towards the direction of the closest Easter egg even though this may not enhance the global performance criterion if another agent $\mathcal{A}_j$ is closer to that egg and will get there before agent $\mathcal{A}_i$. On the other hand, just because agent $\mathcal{A}_j$ is closest to that particular egg it does not necessarily imply that it will pick it up before agent $\mathcal{A}_i$ (it may go after some other eggs). If the Easter egg hunt problem was to be solved in a centralized framework then it would be rather easier to assign different eggs to different agents. However, in a distributed decision making setting, each agent is required to make decisions for enhancing the global performance criterion without having a clear association between its own action and the global cost function. In uncertain environments (for example, if the location of a certain Easter egg is not known unless the agent is within a certain distance and possibly within a certain heading angle from the egg) decisions need to be made on-line and therefore the cooperation issue becomes more challenging.

Cooperation between agents can be considered at different levels. For example, if each agent can perform several tasks (such as search for targets, classification, engagement and evaluation of

attack) then cooperation between agents may involve coordinating their behavior while making decisions on which task to perform at what time. In this paper, we are primarily focusing on the *cooperative search* problem. Therefore, the global objective of the team of agents is to update the overall search map (which represents the knowledge of the environment) in the minimum amount of time. To achieve this, each agent has a responsibility to select its path to benefit the team by selecting a path with minimum overlap with other agents' paths, as described in Section 5.2 (sub-goal C1). Next we develop a real-time approach to realize the cooperative search activities among a team of distributed agents.

Before going into the details we present the main idea of the cooperative search scheme. Each agent possesses information about past paths of other agents via inter-agent communication. As discussed before, this information is used for updating the search map of each agent. Therefore, an agent is able to avoid going over paths previously searched by other agents simply by evaluating its search map and following a path that would result in maximum gain. However, this does not prevent an agent from following a path that another agent is about to follow, or has followed since the last communication contact. Therefore, the main idea of the proposed on-line cooperation scheme is for each agent to try to avoid selecting a path that may be followed by another agent in the near future. In this framework, paths of other agents are treated as "soft obstacles" to be avoided in path selection. However, special consideration is given to scenarios where path overlap may occur at approximately right angles, since in this case the overlap time is quite minimum, thereby not worth causing an interception in an agent's path planning. In other words, the scenario that should be avoided is two agents close to each other and heading in approximately the same direction. By treating paths of other vehicles as "soft obstacles" we employ a type of *artificial potential field* method [40] to derive an algorithm for generating the "rivaling force" that neighboring agents' paths may exert on a certain vehicle. The overall rivaling force exerted on an agent is taken into consideration in deciding which direction the vehicle will follow. Next we discuss the details of this approach.

6.1 Generating the Rivaling Force Between Agents

According to the proposed cooperative search framework, at time k, agent $\mathcal{A}_i$ uses the q-step ahead planning to select the position $p_i(k+q+1) \in \overline{\mathcal{P}}_i(k+q+1)$, which it will visit at time $t = k+q+1$. By communicating with other vehicles at time $t = k-d$ (where d is the communication delay), agent $\mathcal{A}_i$ knows their q-step ahead positions $p_j(k+q-d)$ and heading directions $h_j(k+q-d)$ (measured in degrees from a reference direction). The rivaling force $F_{ij}(k)$ exerted by agent $\mathcal{A}_j$ onto agent $\mathcal{A}_i$ at time k is non-zero if both of the following conditions hold:

1. The location $p_j(k+q-d)$ of agent $\mathcal{A}_j$ is within a maximum distance $\bar{\mu}$ and maximum angle $\pm\bar{\varphi}$ from the location of agent $\mathcal{A}_i$ (see the shaded region in Figure 8).
2. The difference in heading angle $\chi_{ij}(k)$ between agent $\mathcal{A}_j$ and agent $\mathcal{A}_i$ lies within either $[-\bar{\chi},\ \bar{\chi}]$ or $[180^0-\bar{\chi},\ 180^0+\bar{\chi}]$, where $\bar{\chi}$ is the maximum allowed difference in heading angle.

The first condition imposes a requirement that agent $\mathcal{A}_j$ needs to be sufficiently close to agent $\mathcal{A}_i$ before it exerts any rivaling force on $\mathcal{A}_i$. In addition to the distance, the angle between the two locations needs to be within $\pm\bar{\varphi}$. This requirement prevents a vehicle $\mathcal{A}_j$ which is behind $\mathcal{A}_i$ from exerting any rivaling force on $\mathcal{A}_i$. In such a situation, there will be a rivaling force in the opposite direction from $\mathcal{A}_i$ to $\mathcal{A}_j$. In the scenario shown in Figure 8, agents $\mathcal{A}_2$ and $\mathcal{A}_3$ satisfy Condition 1 with respect to their position to agent $\mathcal{A}_1$, while agent $\mathcal{A}_4$ does not satisfy Condition 1.

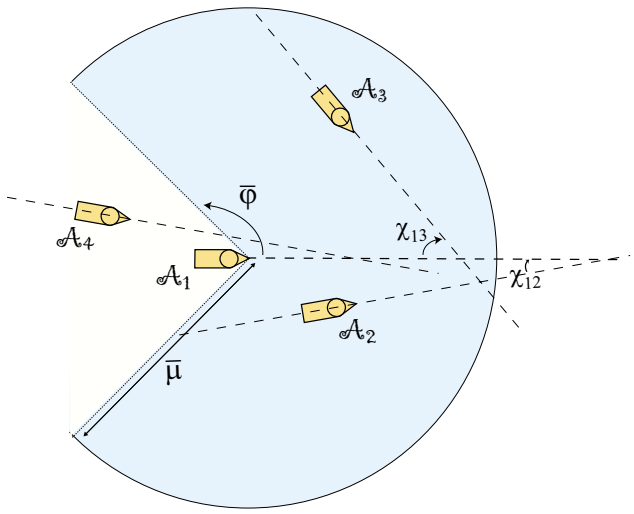

Figure 8: Illustration of conditions that generate non-zero rivaling forces between agents.

The second condition imposes the requirement that in order for agent $\mathcal{A}_j$ to exert a rivaling force on agent $\mathcal{A}_i$ it must either be heading in approximately the same direction, or be coming from approximately the opposite direction. This condition prevents the generation of any rivaling force if the two vehicles are heading in approximately perpendicular directions. Due to maneuverability constraints on the vehicles, the possible overlap in the paths of two agents is significant only if the heading angles are close to each other. At the same time, it is not desirable to impede the path of a vehicle if there is another vehicle coming at approximately right angles. In the scenario shown in Figure 8, agents $\mathcal{A}_2$ and $\mathcal{A}_4$ satisfy Condition 2 with respect to their heading direction in relation to agent $\mathcal{A}_1$ (because both angles χ_{12}, χ_{14} are small), while agent $\mathcal{A}_3$ does not satisfy Condition 2. Therefore, only agent $\mathcal{A}_2$ satisfies both Conditions 1 and 2, and therefore it is the only one that exerts any rivaling force on agent $\mathcal{A}_1$.

For vehicles satisfying both Conditions 1 and 2, the next step is to compute the magnitude and direction of the rivaling force exerted on agent $\mathcal{A}_i$. The main objective here is that the magnitude of the rivaling force $F_{ij}(k)$ exerted by agent $\mathcal{A}_j$ onto agent $\mathcal{A}_i$ at time k should be "large" if agent $\mathcal{A}_i$ is close to the path of agent $\mathcal{A}_j$, and should get smaller as agent $\mathcal{A}_i$ is further away from the path of agent $\mathcal{A}_j$. This approach is similar to artificial potential field methods, which are used in many applications, including the problem of obstacle avoidance of robotic systems. In our case, the obstacle to be avoided is actually the path of another vehicle.

Based on this formulation, we select the rivaling force to be of the form

$$F_{ij}(k) = \begin{cases} k_1 e^{-\alpha \rho_{ij}} \vec{\rho}_{ij} & \text{if Conditions 1 and 2 hold} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where k_1, α are positive design constants, ρ_{ij} is the shortest distance between agent $\mathcal{A}_i$ and the path of agent $\mathcal{A}_j$, and $\vec{\rho}_{ij}$ is a unit vector of the corresponding normalized partial derivative (see Figure 9). Typically k_1 will be a large constant, corresponding to the magnitude of the rivaling force if the distance ρ_{ij} is zero. Note that since we treat paths of neighboring agents as "soft obstacles" there is no need to set the magnitude of the rivaling force to ∞ as is sometimes done in the case of obstacle avoidance problems. The design parameter $\alpha > 0$ corresponds to the rate at which the rivaling force is decreasing as the distance ρ_{ij} is increasing. The rivaling force is not

necessarily symmetric (i.e., $F_{ij}(k) \neq F_{ji}(k)$ since it depends on the relative position and heading direction of the two agents. In fact, as we saw earlier, it is possible for $F_{ij}(k)$ to be zero while $F_{ji}(k)$ is quite large (this would occur if agent $\mathcal{A}_j$ is behind $\mathcal{A}_i$ and heading in approximately the same direction). Figure 9 illustrates the potential field lines associated with the path of agent $\mathcal{A}_2$, and the resulting rivaling force exerted by $\mathcal{A}_2$ onto $\mathcal{A}_1$.

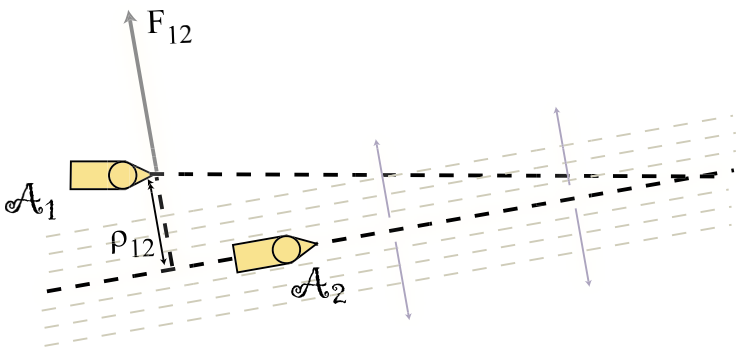

Figure 9: Illustration of the potential field lines associated with the path of agent $\mathcal{A}_2$, and the resulting rivaling force exerted by $\mathcal{A}_2$ onto $\mathcal{A}_1$.

As seen from Figure 9, the path of $\mathcal{A}_2$ that generates a rivaling force onto $\mathcal{A}_1$ includes not only the forward path but also some of the backward (previous) path. The reason for this is that communication delays may cause $\mathcal{A}_1$ to have incomplete (outdated) information about the path followed by $\mathcal{A}_2$. It is also noted that the actual path of an agent may not be a straight line as assumed in Figure 9. However, due to maneuverability constraints, this is a reasonable and simple approximation of the actual path for cooperation purposes.

The overall rivaling force exerted by the entire team of agents upon an agent $\mathcal{A}_i$ at time k is given by

$$F_i(k) = \sum_{j \neq i} F_{ij}(k) \tag{2}$$

Intuitively, according to the overall rivaling force $F_i(k)$ exerted on it, agent $\mathcal{A}_i$ is impelled to select a path $p_i(k+q+1)$, among the possible set of paths $\overline{P}_i(k+q+1)$, that is more in line with avoiding the paths of other vehicles. Therefore, in addition to the magnitude of the rivaling force, a key parameter is the angle difference between the direction of the overall rivaling force $F_i(k)$ and the direction of each possible path from the set $\overline{P}_i(k+q+1)$, which we denote by $\theta_i(j,k)$. From a cooperative viewpoint, the objective is to select the path with the minimum $\theta_i(j,k)$ among $j \in [1, 2, \ldots m]$.

6.2 Formulation of the Cooperation Cost Function

Using the algorithm described in Section 6.1, each agent can compute the rivaling force exerted on it by other agents that are located in close proximity and, based on the overall rivaling force, select an optimal path that would minimize the overlap with paths of other vehicles. However, avoidance of path overlap is only one of an agent's objectives. Indeed, its main objective is to search for (and possibly engage) targets. Therefore, the goal of cooperation needs to be quantified as a cost function component and integrated with the remaining components of the cost criterion.

To integrate the cooperative sub-goal with other objectives, the cooperation cost function is required to generate a performance measure of cooperation associated with each possible path. After normalization, the cost function component for cooperation (denoted by $J(i,j,k)$) should be a function mapping each possible path $j \in [1, 2, \ldots, m]$ into an interval $[0, 1]$. According to the formulation considered in this paper, the value of the cooperation cost function depends on the magnitude of the overall rivaling force $F_i(k)$ and the angle difference $\theta_i(j,k)$ between the direction of the overall rivaling force and the direction of each possible path from the set $\overline{P}_i(k+q+1)$. Figure 10 illustrates the case where there are three possible paths for agent $\mathcal{A}_1$ to follow. The corresponding angles $\theta_1(1,k)$, $\theta_1(2,k)$, $\theta_1(3,k)$ are denoted by θ_1, θ_2, θ_3 respectively for diagrammatic simplicity. Hence, we consider a general function

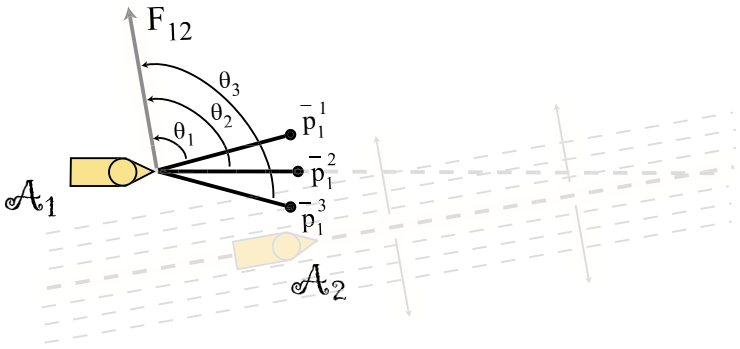

Figure 10: Illustration of computing the cooperation cost function.

$$J(i,j,k) = f(|F_i(k)|,\ \theta_i(j,k))$$

where $f : \Re^+ \times [-\pi,\ \pi] \mapsto [0,\ 1]$ is required to have the following attributes:

- As the magnitude of the rivaling force $F_i(k)$ becomes larger, the differences in the normalized cost function values between alternative paths should become larger. In other words, if $|F_i(k)|$ is large then cooperation is a crucial issue and therefore there should be a significant difference in the cooperation cost function to steer the agent into selecting the path of maximal cooperation. On the other hand, if $|F_i(k)|$ is small then cooperation is not a crucial issue, therefore the cooperation cost function component should be approximately equal for each alternative path plan, thereby allowing the agent to make its path decision based on the cost function associated with the other sub-goals.

- As the magnitude of the angle difference $\theta_i(j,k)$ becomes larger, the differences in the normalized cost function values between alternative paths should become larger. Again, if $|\theta_i(j,k))|$ is small then cooperation is not a crucial issue, therefore, the cooperation cost function component is approximately equal for each alternative path plan. If $|\theta_i(j,k))|$ is large then cooperation is a crucial issue and therefore there should be a significant difference in the cooperation cost function to steer the agent into selecting the path of maximal cooperation.

Deriving an appropriate function f with these attributes is rather straightforward. In the simulations presented in the next section, we use the following cooperative cost function

$$J(i,j,k) = \exp^{\gamma_0 |F_i(k)| \cos(\frac{\theta_i(j,k)}{2})} \tag{3}$$

where γ_0 is a positive design constant.

It is important to note that the specific functions selected in Equation (1) for the rivaling force and in Equation (3) for the cooperative cost function, are not as important as the attributes of these functions. Specifically, other functions with the same attributes can be utilized to obtain similar results.

7 Simulation Results

The approach described in this paper has been implemented and evaluated by several simulation studies. A representative sample of these studies is presented in this section. First, we describe the details of the cost function criterion and then present two simulations studies. In the first simulation study, a team of UAVs is searching in a mostly unknown environment. In the second simulation, we consider a scenario where the environment consists of three targets whose location belongs to a certain probability distribution.

7.1 Design of Simulation Experiment

According to the proposed cooperative path planning approach, each agent uses a multi-objective cost function J to select and update its search path. This approach is quite flexible in that it allows the characterization of various mission-level objectives and facilitates possible trade-offs. The simulation examples presented in this section consider only the first three of the sub-goals (S1, S2, C1) described in Section 5. These sub-goals correspond to the main issues associated with the cooperative search problem.

The cost functions associated with each sub-goal are computed as follows:

- The first cost function $J_{S1}(i,j,k)$ is the gain of agent $\mathcal{A}_i$ on sub-goal S1 if it selects path $j \in [1,2,\ldots,m]$ at time k. It is a positive value denoting the gain on the certainty of the search map by following path j at time k. The following function is used to evaluate the actual gain obtained by selecting the jth path:

$$J_{S1}(i,j,k) = \sum_{(x,y)\in R_{i,j}} [\mathcal{S}(x,y;\theta(k)) - \mathcal{S}(x,y;\theta(k-1))] \tag{4}$$

  where (x,y) denotes any point in the search area $R_{i,j}$ that will be encountered if agent $\mathcal{A}_i$ follows path j, and $\mathcal{S}(x,y;\theta(k))$ is the certainty value of point (x,y) at time k.

- The second cost function $J_{S2}(i,j,k)$ is used to evaluate the potential gain based on the average uncertainty of a triangular region $R'_{i,j}$ associated with the heading direction j. The cost function $J_{S2}(i,j,k)$ is generated by

$$J_{S2}(i,j,k) = \sum_{(x,y)\in R'_{i,j}} (1 - \mathcal{S}(x,y;\theta(k))) \tag{5}$$

  where (x,y) denotes all the points in the region $R'_{i,j}$.

- The third cost-function is used to evaluate the sub-goal C1, which was formulated in Section 6.2.

$$J_{C1}(i,j,k) = \exp^{\gamma_0 |F_i(k)| \cos(\frac{\theta_i(j,k)}{2})} \tag{6}$$

After normalizing the three cost-functions and selecting appropriate weight coefficients, the overall multi-objective cost function is described by

$$J(i,j,k) = w_1 \cdot \overline{J}_{S1}(i,j,k) + w_2 \cdot \overline{J}_{S2}(i,j,k) + w_3 \cdot \overline{J}_{C1}(i,j,k) \tag{7}$$

where $\overline{J}_q$ for $q \in \{S1, S2, C1\}$ denote the normalized cost functions and w_i are the weights, which satisfy $w_1 + w_2 + w_3 = 1$. In the simulation examples different weight values were used to illustrate various aspects of cooperation. The normalized cost functions $\overline{J}_q$ are computed by

$$\overline{J}_q = \frac{J_q(i,j,k)}{\max_j\{J_q(i,j,k)\}}.$$

Therefore, each cost function $\overline{J}_q \in [0,\ 1]$. An agent $\mathcal{A}_i$ selects a path based on which $j \in [1, \cdots, m]$ gives the largest value, as computed by (7).

7.2 High Uncertainty Environment

The first simulation study considers a scenario of high uncertainty in the environment. The search region is a 200 by 200 area. It is assumed that there is some a-priori information about the search region: the green (light) polygons indicate complete certainty about the environment (for example, these can represent regions where it is known for sure—due to the terrain—that there are no targets); the blue (dark) polygons represent partial certainty about the environment. The remaining search region is assumed initially to be completely uncertain. First we consider the case of two agents, and then we use a team of five agents.

In both simulations we are using the recursive q-step ahead planning algorithm with $q = 3$. The weights of the cost function are set to: $w_1 = 0.3125$, $w_2 = 0.375$, $w_3 = 0.3125$, which gives approximately equal importance to each of the three sub-goals. The parameters of the potential field function used for sub-goal C1 are set to: $k_1 = 50$, $\alpha = 1$, $\gamma_0 = 1$. The results for the case of two agents are shown in Figure 11. The upper-left plot shows a standard search pattern 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. The standard search pattern utilized here is based on the so-called zamboni coverage pattern [48]. The lower-left plot shows the result of the cooperative search method based on the recursive q-step ahead planning algorithm.

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. In the above example, after four encounters the block will have 6.25% percentage of uncertainty. The cooperative search algorithm has no pre-set search pattern. As seen from Figure 11, 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.

To compare the performance of the three search patterns, the lower-right plot of Figure 11 shows the percentage 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.

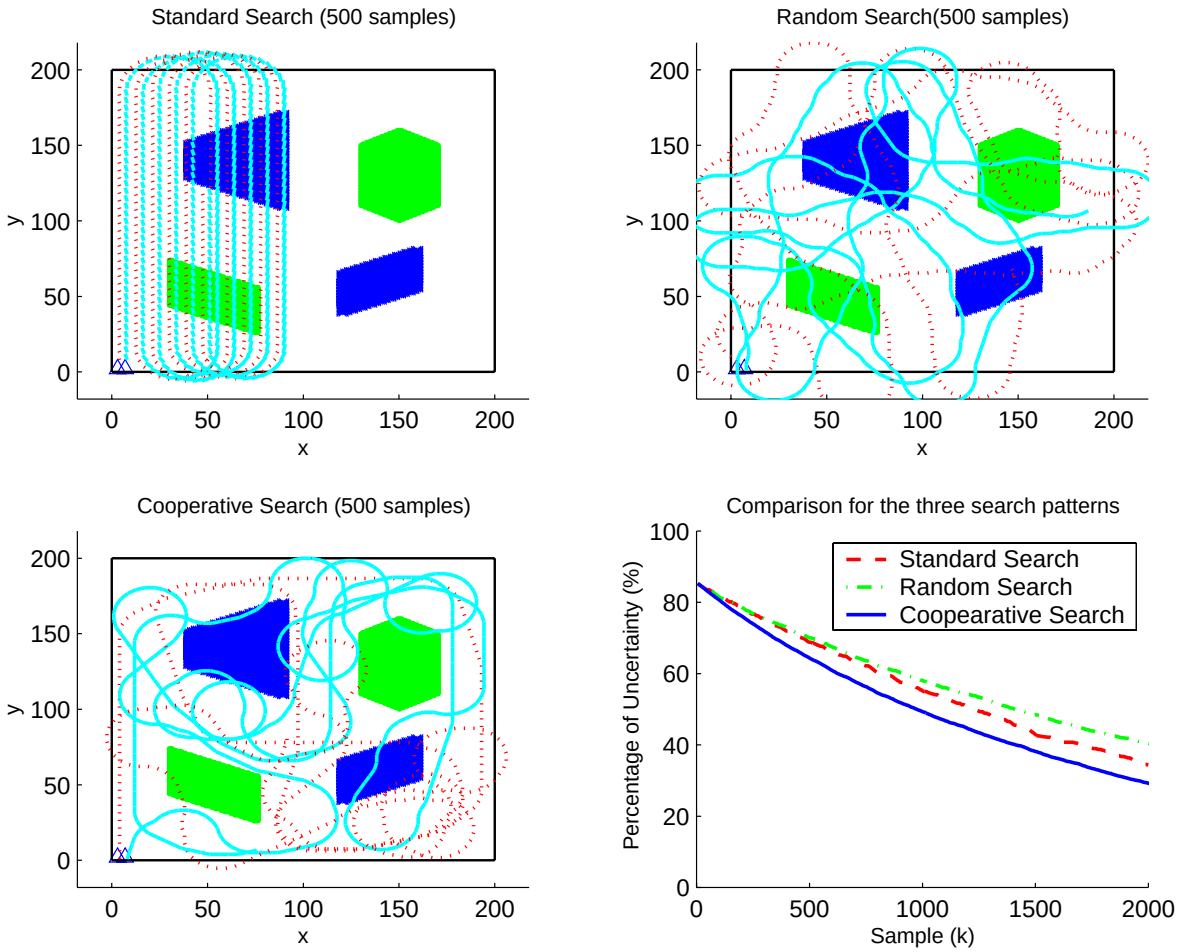

Figure 11: Comparison of the cooperative search pattern with a "standard" search pattern and a random search pattern for the case of two moving agents. The upper-left plot shows a standard search pattern for the first 500 time samples; the upper-right plot shows the corresponding search pattern in the case of a random search, subject to some bounds to restrict the agent from deserting the search region; The lower-left plot shows the cooperative search pattern based on the recursive q-step ahead planning algorithm; the lower-right plot shows a comparison of the performance of the three search patterns in terms of reducing uncertainty in the environment.

Specifically, after 2000 time steps the percentage of uncertainty in the environment reduces from approximately 85% initially to 40.4%, 34.4%, 29.2% for the random search, standard search, and cooperative search, respectively. Therefore, there is approximately a 15% improvement with the cooperative search over the standard search. This is mainly due to the presence of some known regions, which the standard search and random search algorithms are not trying to avoid.

The corresponding results in the case of five agents moving in the same environment is shown in Figure 12. The results are analogous to the case of two agents. After 2000 time steps the percentage of uncertainty in the environment reduces to 13.9%, 12.0%, 7.1% for the random search, standard search, and cooperative search, respectively.

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

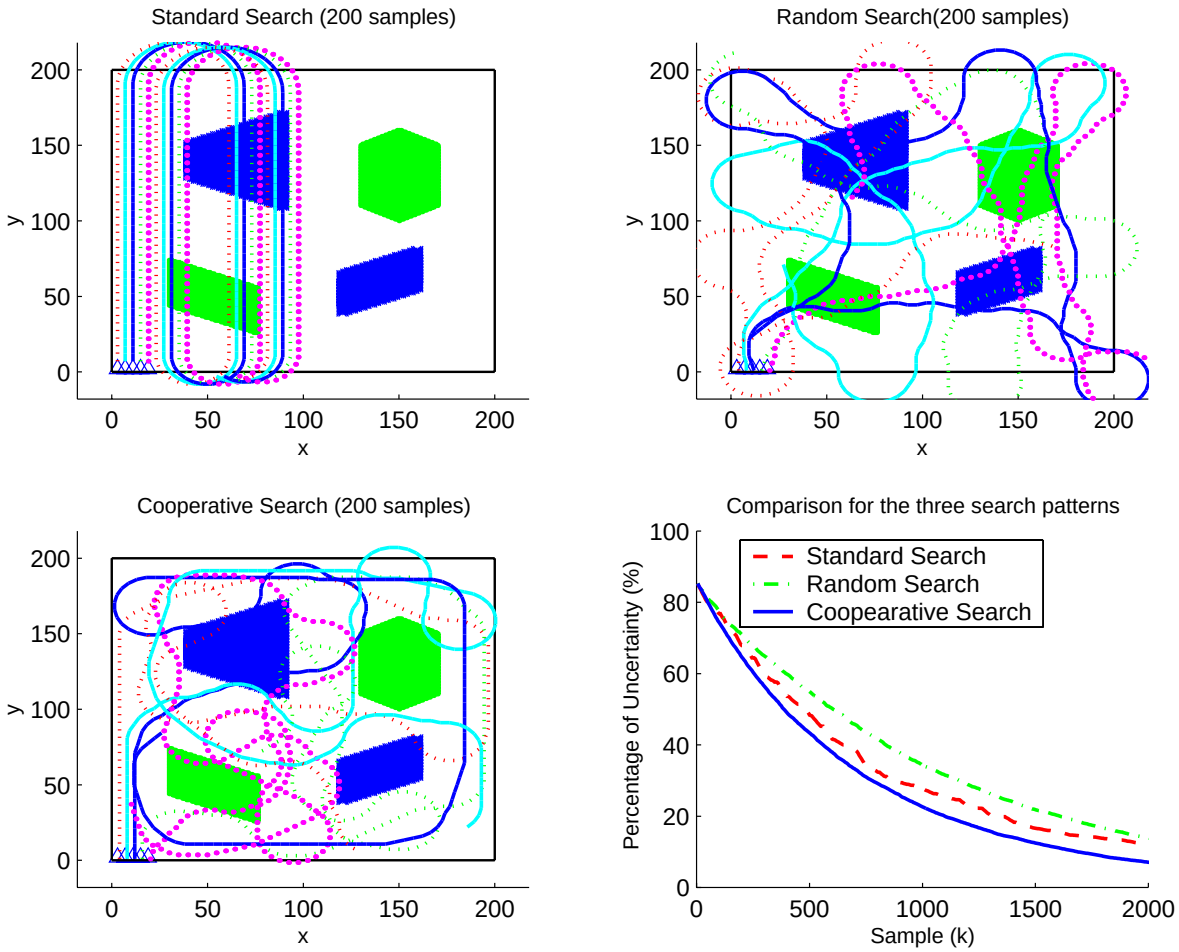

Figure 12: Comparison of the cooperative search pattern with a "standard" search pattern and a random search pattern for the case of five moving agents. The upper-left plot shows a standard search pattern for the first 200 time samples; the upper-right plot shows the corresponding search pattern in the case of a random search, subject to some bounds to restrict the agent from deserting the search region; The lower-left plot shows the cooperative search pattern based on the recursive q-step ahead planning algorithm; the lower-right plot shows a comparison of the performance of the three search patterns in terms of reducing uncertainty in the environment.

path planning decisions. Information from other agents is received at a slower rate. Specifically, we assume that the communication sampling time T_c between agents is five times the movement sampling time; i.e., $T_c = 5T_m$. For fairness in comparison, it assumed that for the standard and random search patterns the agents exchange information and update their search maps in the same way as in the cooperative search pattern, but they do not use the received information to make on-line decisions on where to go.

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., $m = 3$). The left and right directions are at angles of -15^0 and $+15^0$ respectively from the heading direction, which reflects the maneuverability constraints of the vehicles. 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.

7.3 Low Uncertainty Environment

In this second simulation study we consider a more structured environment, where we assume that according to the a-priori information there are three targets whose location is uncertain but satisfies a certain Gaussian distribution. The environment is again a 200 by 200 area and the assumed center of Gaussian probability distributions of each target is located at the coordinates (50, 50), (100, 150), (150, 100), as shown in Figure 13. The probability of the target distribution satisfies a Gaussian distribution of the form

$$p(x, y) = e^{\frac{1}{\sigma} d_c^2(x,y)}, \tag{8}$$

where $d_c(x, y)$ is the minimum distance of the point (x, y) from one of the three target distribution centers, and σ is a constant given by $\sigma = 2\pi\sqrt{1200}$. If an agent passes through a point (x, y) that none of the agents have visited before then the team derives a *target search gain* described by the probability distribution $p(x, y)$ given in (8). Once a point is visited by at least one agent then no further target search gain is assumed available. This is slightly different from the simulation study of the high uncertainty environment where the gain was decreased with every visit to a particular position.

In the simulation shown in Figure 13 we compare the performance of three different runs, all based on the search procedure developed in this paper using the recursive q-step ahead planning algorithm. The team of agents consists of five vehicles with the same maneuverability constraints as in the first simulation study. The only difference between the three runs is the amount of cooperation included, as defined by the third cost function component J_{C1}. The upper left plot shows the trajectories of the team of agents using the cooperative search algorithm with the weights selected as $w_1 = 1/8$, $w_2 = 2/8$, $w_3 = 5/8$. In the second simulation run, shown in the upper right plot, we show the trajectories selected by the five vehicles for a weakly cooperative system with the weights selected as $w_1 = 1/4$, $w_2 = 2/4$, $w_3 = 1/4$. Finally, in the third simulation run there is no cooperation between the five agents, in the sense that the weights are set to: $w_1 = 1/3$, $w_2 = 2/3$, $w_3 = 0$.

As seen from Figure 13, in the case of the cooperative search algorithm (upper left) the five vehicles split up between the two nearest targets and soon they also cover the distant target. In the case of the weakly cooperative search algorithm (upper right) the five agents first go to the nearest target on the lower left, and from there, some agents go to the other two targets. In the case of non-cooperation (lower left plot) all five vehicles head for the nearest target on the lower left and spend considerable time there before they move on to the other targets (in fact, the simulation shows 200 time steps—as compared to 100 samples for the other two simulation runs—because during the first 100 steps all five vehicles remained at the first target). With no cooperation there is significant overlap of the paths of vehicles.

The performance of the three search patterns for the first 200 time steps is shown in the lower right plot of Figure 13 in terms of the percentage of target search gain over time. The percentage of target search gain is computed as the total gain of all five vehicles at time k divided by the initial total target search gain in the environment. After 200 time steps the target search gain for the cooperative search is 59.3%, for the weakly cooperative search it is 54.1% and for the

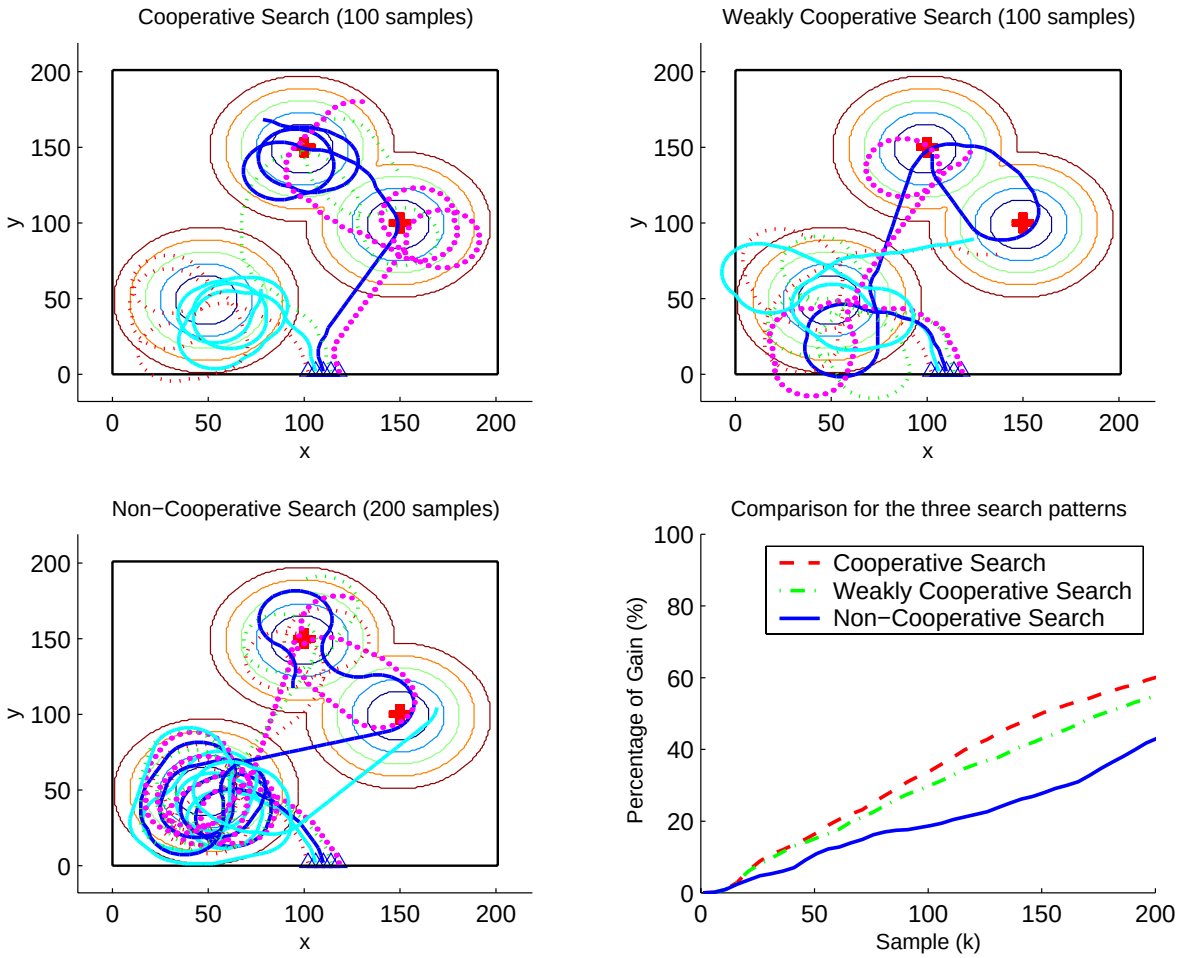

Figure 13: Comparison of the cooperative search pattern with a "weakly cooperative" search pattern and a non-cooperative search pattern for the case of five moving agents searching for three targets located according to a Gaussian distribution function around three center points. The upper-left plot shows a cooperative search pattern for the first 100 time samples; the upper-right plot shows the corresponding search pattern in the case of a weakly cooperative search algorithm; The lower-left plot shows the non-cooperative search pattern for the first 200 time samples; the lower-right plot shows a comparison of the performance of the three search patterns in terms of the percentage of target search gain over time for each of the three search patterns.

non-cooperative search it is 42.8%. It is noted that in this simulation study we do not show the performance of a "standard search pattern" and the random search algorithm because comparably both do not perform well due to the highly structured environment.

8 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 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 an interleaved planning technique, and developed a real-time approach for on-line cooperation between agents. These ideas were illustrated with simulation studies by comparing them to a restricted random search, a standard search pattern, as well as a non-cooperative search algorithm.

Acknowledgment

The authors would like to acknowledge the following co-researchers for extensive discussions, which helped significantly in shaping the concepts and techniques described in this paper: Mark Mears, David Jacques, Philip Chandler, Matt Flint, Ali Minai, Meir Pachter, Rob Murphey and Siva Banda.

References

[1] D. Gillen and D. Jacques, "Cooperative behavior schemes for improving the effectiveness of autonomous wide area search munitions," in *Workshop on Cooperative Control and Optimization*, (University of Florida, Gainesville), Dec. 5–7 2000.

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

[3] 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.

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

[5] D. Hristu and K. Morgansen, "Limited communication control," *Systems & Control Letters*, vol. 37, no. 4, pp. 193–205, 1999.

[6] G. Dudek and et al., "A taxonomy for swarm robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, (Yokohama, Japan), July 1993.

[7] S. Hackwood and S. Beni, "Self-organization of sensors for swarm intelligence," in *IEEE Int. Conf. on Robotics and Automation*, (Nice, France), pp. 819–829, May 1992.

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

[9] B. S. Koontz, "A multiple vehicle mission planner to clear unexploded ordinance from a network of roadways," Master's thesis, Massachusetts Inst. of Tech. 1997.

[10] M. Mataric, "Minimizing complexity in controlling a mobile robot population," in *IEEE Int. Conf. on Robotics and Automation*, (Nice, France), May 1992.

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

[12] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Trans. on Robotics and Automation*, vol. 2, March 1986.

[13] W. Jacak, *Intelligent Robotic Systems: Design, Planning, and Control*. NY: Kluwer Academic / Plenum Pub., 1999.

[14] R. Brooks, ed., *Cambrian Intelligence: The Early History of the New AI*. Cambridge, MA: MIT Press, 1999.

[15] J. Reif and H. Wang, "Social potential fields: a distributed behavioral control for autonomous robots," *Robotics and Autonomous Systems*, vol. 27, pp. 171–194, 1999.

[16] C. Breder, "Equations descriptive of fish schools and other animal aggregations," *Ecology*, vol. 35, pp. 361–370, 1954.

[17] R. Miller and W. Stephen, "Spatial relationships in flocks of sandhill cranes (Grus canadensis)," *Ecology*, vol. 47, pp. 323–327, 1996.

[18] J. Albus and A. Meystel, *Engineering of Mind: An Intelligent Systems Perspective*. New York, NY: John Wiley and Sons, 2000.

[19] A. Drogoul, M. Tambe, and T. Fukuda, eds., *Collective Robotics*. Berlin: Springer Verlag, 1998.

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

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

[22] 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.

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

[24] M. Snorrason and J. Norris, "Vision based obstacle detection and path planetary rovers," in *Unmanned Ground Vehicle Technology II*, (Orlanso, FL), April 1999.

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

[26] L. Stone, "The process of search planning: Current approachs and the continuing problems," *Operational Research*, vol. 31, pp. 207–233, 1983.

[27] H. Richardson, "Search theory," in *Search Theory: Some Recent Developments* (D. Chudnovsky and G. Chudnovsky, eds.), pp. 1–12, New York, NY: Marcel Dekker, 1987.

[28] S. Benkoski, M. Monticino, and J. Weisinger, "A survey of the search theory literature," *Naval Research Logistics*, vol. 38, pp. 469–494, 1991.

[29] J. Eagle and J. Yee, "An optimal branch-and-bound procedure for the constrained path moving target search problem," *Operations Research*, vol. 38, pp. 11–114, 1990.

[30] T. Stewart, "Experience with a branch-and-bound algorithm for constrained searcher motion," in *Search Theory and Applications* (K. Haley and L. Stone, eds.), pp. 247–253, Plenum Press, New York, 1980.

[31] R. Hohzaki and K. Iida, "Path constrained search problem with reward criterion," *Journal of the Operations Research Society of Japan*, vol. 38, pp. 254–264, 1995.

[32] R. Hohzaki and K. Iida, "An optimal search plan for a moving target when a search path is given," *Mathematica Japonica*, vol. 41, pp. 175–184, 1995.

[33] J. Danskin, "A helicopter versus submarines search game," *Operations Research*, vol. 16, pp. 509–517, 1968.

[34] R. Hohzaki and K. Iida, "A search game when a search path is given," *European Journal of Operational Reasearch*, vol. 124, pp. 114–124, 2000.

[35] A. Washburn, "Search-evasion game in a fixed region," *Operations Research*, vol. 28, pp. 1290–1298, 1980.

[36] R. Dell and J. Eagle, "Using multiple searchers in constrainted-path moving-targer search problems," *Naval Research Logistics*, vol. 43, pp. 463–480, 1996.

[37] S. Goldsmith and R. Robinett, "Collective search by mobile robots using alpha-beta coordination," in *Collective Robotics* (A. Drogoul, M. Tambe, and T. Fukuda, eds.), pp. 136–146, Springer Verlag: Berlin, 1998.

[38] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an AUV," *Autonomous Robots*, vol. 3, pp. 91–119, 1996.

[39] H. Choset and P. Pignon, "Coverage path planning: the boustrophedon cellular decomposition," in *International Conference on Field and Service Robotics*, (Canberra, Australia), 1997.

[40] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *International Conference on Robotics and Automation*, (St. Louis), pp. 500–505, March 1985.

[41] V. Torczon, "On the convergence of pattern search algorithms," *SIAM Journal Optimization*, vol. 7, pp. 1–25, 1997.

[42] S. Lucidi and M. Sciandrone, "On the global convergence of derivative free methods for unconstrained optimization," in *Technical Report*, Univ.di Roma, 1997.

[43] J. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.

[44] J. Dennis and V. Torczon, "Direct search methods on parallel machines," *SIAM Journal Optimization*, vol. 1, pp. 448–474, 1991.

[45] V. Torczon, "On the convergence of the multidirectional search algorithm," *SIAM Journal Optimization*, vol. 1, pp. 123–145, 1991.

[46] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *To appear, IEEE Control Systems Magazine*, 2001.

[47] 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.

[48] 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.