Chapter 1

COOPERATIVE CONTROL FOR AUTONOMOUS AIR VEHICLES

Kevin Passino[†], Marios Polycarpou[‡], David Jacques[⋆], Meir Pachter[⋆], Yang Liu[†], Yanli Yang[‡], Matt Flint[‡] and Michael Baum[†]

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

[‡]Dept. of Electrical and Computer Engineering and Computer Sciences
University of Cincinnati, Cincinnati, OH 45221-0030, USA

[⋆]Air Force Institute of Technology, AFIT/ENY
Wright Patterson Air Force Base, OH 45433-7765, USA

Abstract The main objective of this research 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 coordinate strikes against targets. The air vehicles 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. They 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 "target search map"; and (ii) utilization of the target search map and other information to compute on-line a guidance trajectory for the vehicle to follow. We study the stability of vehicular swarms to try to understand what types of communications are needed to achieve cooperative search and engagement, and characteristics that affect swarm aggregation and disintegration. Finally, we explore the utility of using biomimicry of social foraging strategies to develop coordination strategies.

1. Introduction

1.1. UAAV Operational Scenarios

Recent technological advances are making it possible to deploy multiple Uninhabited Autonomous Air Vehicles (UAAVs) that can gather information about a dynamic threat/target environment, evade threats, and coordinate strikes against targets (see Figure 1.1). Here, we show a section of land (the black regions indicate mountains/rugged terrain where targets/threats generally do not reside) where there are targets with certain priorities (e.g., set by the command center) and threats with different severities relative to their ability to eliminate the UAAV or other assets. A target may also be a threat, but there may be threats that are not high priority targets, or targets that are not severe threats to the UAAV (e.g., the autonomous munition is assumed small and expendable, so targets are not considered to be threats to the munition itself). Mobile target/threats are designated by showing their path of travel. There are also stationary targets/threats. There are some targets/threats that are in groups, while others are distant from a group. There are groups with certain types of compositions (e.g., some in a high threat/low target priority group that are protecting a low threat severity/high target priority site). The proposed formulation can also allow for negative target priorities, which may include undesirable targets such as a truckload of civilians (collateral damage). The UAAVs are designated with shaded triangles (the black one has extra UAAV supervisory responsibility), and the ellipses show the region where the sensor resources of the UAAVs are currently focused (and they can only "see" targets/threats within this ellipse). In our general formulation we assume that in addition to sensing and maneuvering capabilities, the UAAVs have an ability to strike a target with on-board munitions, or the UAAV is a munition itself. There are some communications that are allowed between UAAVs, and these are represented via the bidirectional dotted arrows; however, there are communication limitations (e.g. bandwidth, delays, or range constraints). The general problem is to coordinate the movement of the UAAVs so that they can evade threats and locate and destroy the highest priority targets, before they expend their own resources (e.g. fuel).

1.2. UAAV Cooperation to Enhance Mission Effectiveness

While each UAAV could certainly operate independently of the others, the overall mission effectiveness can be improved via communica-

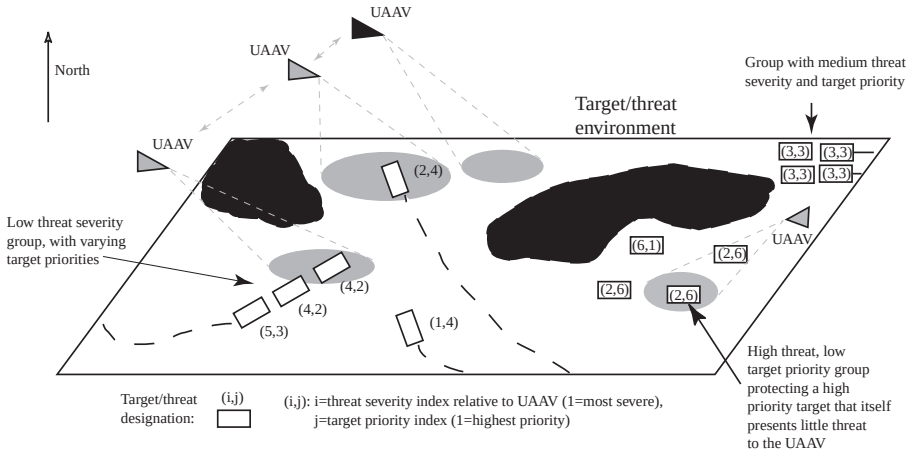

Figure 1.1. UAAVs operating in a dynamic target/threat environment.

tions and coordination of their activities. For instance, for the situation depicted in Figure 1.1 suppose that one UAAV has identified the $(2,4)$-target/threat (since it is in the focus of its sensors) and another UAAV has just identified a $(4,2)$-target/threat. The notation used here is that (i,j) implies threat severity index i and target priority index j, where 1 denotes the highest priority (or the most severe threat). Further suppose that in the recent past the second UAAV also identified the $(4,2)$-target/threat to its southwest. Note that this "identification" entails estimating the position of the target/threat, and the type of the target/threat (i.e., an estimate of the target/threat indices given in the figure), and there is uncertainty associated with each of these. In Figure 1.1 the "supervisory" UAAV is capable of receiving information from all other UAAV's subject to range limitations and data loss assumptions. This communication could be direct or via retransmission by all UAAV's. In this situation suppose the supervisor decides that since the one UAAV has encountered a target/threat with high threat severity, but low target priority, while the second has encountered two target/threats with higher target priority, but lower threat severity, the one UAAV ought to evade the threat, and go assist the second UAAV in identifying and possibly eliminating the high priority targets. Notice that these decisions could have been made without the supervisor UAAV, but if it is available, and has established communications with other nearby UAAVs it can make more effective decisions.

In the case, however, where there are no nearby UAAVs to communicate with, each UAAV must be able to act entirely on its own in the most effective manner possible. This situation is depicted in the southeast corner of the target/threat environment. There, we have a group of low target priority/high threat severity vehicles that are "protecting" a relatively defenseless, but high priority target (i.e., the $(6, 1)$ site). Suppose that the nearby UAAV has identified two of the three in the group that is protecting the valuable target. The UAAV should then have a strategy where it tries to evade the threats and continue to seek a higher priority target, and decide in an optimal fashion when to strike targets. Hence, we see that UAAVs must be able to flexibly operate independently, but if communication is possible with nearby UAAVs, then the information should be exploited to optimize the balance between the need to identify target/threats, and to evade and destroy these.

This leads to the following principle: Wide-area search and identification of targets/threats is desirable, but it must be balanced with fuel/time constraints, threats that may dictate the need to evade and hence obtain poor identification, and the need to ultimately engage the highest priority targets with the limited resources available (e.g., for the autonomous munition problem we assume a single shot, expendable UAAV).

1.3. Distributed Guidance and Control Architecture

Consider N vehicles deployed in some search region $\mathcal{X}$ of known dimension. As each vehicle 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 is typically in the form of an image, which can be processed on-line to determine the presence of a certain item 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 vehicle also receives information from other vehicles via a wireless communication channel. The information received from other vehicles can be in raw data form or it may be pre-processed, and it may be coming at a different rate (usually at a slower rate) than the sensor information received by the vehicle from its own sensors.

Depending on the specific mission, the global objective pursued by the team of vehicles may be different. In this paper, we focus mainly on the problem of cooperative search, where the team of vehicles 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 vehicle wants to follow a trajectory that leads to regions in $\mathcal{X}$ that have not been visited frequently before by the team of vehicles. 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.

Each vehicle has two basic control loops that are used in guidance and control, as shown in Figure 1.2. The "outer-loop" controller for vehicle $\mathcal{A}_i$ 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 vehicles 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 vehicle $\mathcal{A}_j$. Although in the above formulation it appears that all vehicles 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 vehicles is missing, or the information from different vehicles 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 vehicle $\mathcal{A}_i$ at time $k+j$, and q is the number of look-ahead steps in the path planning procedure.

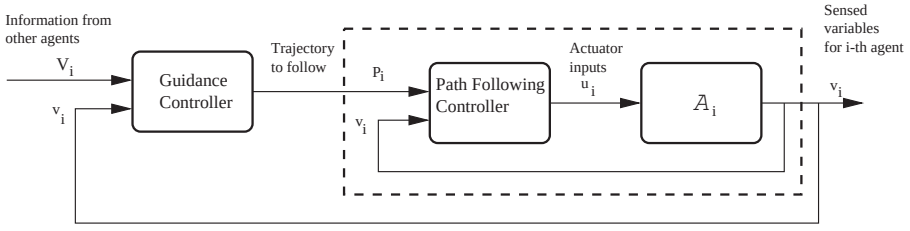

Figure 1.2. Inner- and outer-loop controllers for guidance and control of air vehicles.

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 vehicle will track the

desired trajectory $P_i(k)$. We largely ignore the vehicle 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 vehicle $\mathcal{A}_i$ can be combined with information from other vehicles so that the team of vehicles 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 1.3. 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 vehicle'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 vehicles, remaining fuel, etc.), the second function is to compute a desired path for the vehicle to follow—this is referred to as the vehicle's *guidance decision* function, and is denoted by $\mathcal{D}_i$. In this setting we assume that the guidance control decisions made by each vehicle are autonomous, in the sense that no vehicle tells another what to do in a hierarchical type of structure, nor is there any negotiation between vehicles. Each vehicle simply receives information about the environment from the remaining vehicles (or a subset of the remaining vehicles) 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 vehicles may be actively coordinating their decisions and actions.

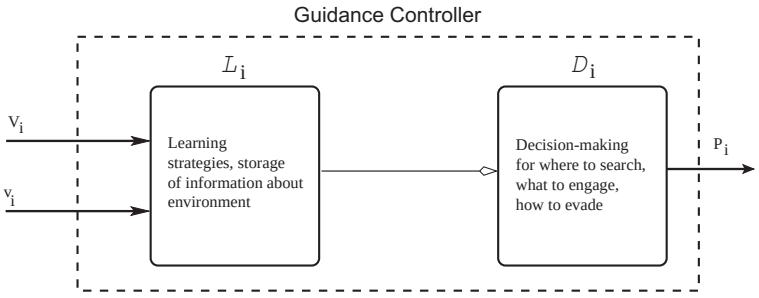

Figure 1.3. Learning and decision-making components of the outer-loop controller for trajectory generation of air vehicles.

Generally, the characteristics of the N trajectories of the group of UAAVs combine to result in the "emergence" of a set of vehicle trajectories that represent, for example:

- Cooperative search so that search resources of the group are minimized in obtaining maximum information about the environment, or
- Cooperative engagement where an attack is coordinated between UAAVs (e.g., for a particularly high priority target).

While the development of a guidance algorithm for a single UAAV is challenging, in this project it is especially important to develop learning and decision-making strategies that exploit information from other vehicles in order to realize the benefits of cooperative control for mission effectiveness enhancement.

2. Autonomous Munition Problem

There are several types of UAAVs at various stages of development in the US Air Force. For instance, it is envisioned that two smart Uninhabited Combat Air Vehicles (UCAVs) may be used for coordinated strike missions, and work is progressing on this topic at the Air Force Research Laboratory (AFRL). Alternatively, there is another class of UAAVs where there is a larger number of *inexpensive and expendable* autonomous munitions (AMs) with wide area search capability that could be deployed in a seek/destroy mission against critical ground mobile targets, the so-called LOCAAS problem (Jacques and Leblanc, 1998). In our research work we use this type of UAAV problem as a benchmark for the study of design and evaluation of distributed coordination and control strategies for UAAVs. Here, we will summarize the AM problem and our current progress on developing and evaluating search and engagement strategies.

2.1. Autonomous Munitions: Characteristics and Goals

For the AM problem we will assume that there are no threats, only mobile targets, so that we do not need to consider evasive maneuvers (i.e., so we only have a seek/destroy mission). We consider the coordination of multiple munitions ($N \geq 4$) and consider the case where there are multiple target types, with different priorities that are specified a priori. Although we assume that the targets lie in the (x, y) plane, we intend to consider terrain effects that could restrict target movement. Terrain considerations can be used to discourage munitions from searching areas where targets are not expected to be located, thus improving the overall search efficiency. We assume that each AM has a "generic sensor" (our methods will not be sensor-specific) that "illuminates" (allows it to see)

only a fixed (e.g., elliptical or trapezoidal) region on the (x, y) plane (i.e., like the shaded regions in Figure 1.1). If an object is in its illumination area, the sensor will have the opportunity to recognize the object (target priority i or non-target), and if the munition declares it to be a target, provide an estimate of the (x, y) location. The generic target recognition for this project will be done using probability table look-ups, a confusion matrix, as is known to those familiar with ATR terminology. In addition to mis-identifying real targets, the munitions have the potential for declaring non-target objects to be a real target. For a declared target (real or otherwise), the minimum information we expect the munition to communicate is the target priority, (x, y) location, and whether or not the munition is committed to engaging the target.

The current state-of-the-art in munition wide area search is such that a ground mobile target free to move in any direction can increase its target location uncertainty faster (quadratic with time) than a single munition can search (linear with time). Further, each individual munition has limited sensing/computing capability with which to positively identify a target. Assuming the target is found and properly identified, each munition has a probability of kill given a hit which is less than unity. An obvious way to compensate for these factors is to use multiple munitions to help find, identify and/or strike critical targets, but this must be done in such a way that the probability of missing other targets in the area is minimized. An additional constraint on the solution is the level of communication that is possible between munitions. For example, it may be that only a low-bandwidth channel is available between munitions that are physically near each other. Currently we are also assuming range degraded communications.

The overall goal should be to increase the macro-level effectiveness (as measured by standard mission effectiveness performance metrics), while at the same time maintaining or reducing the required level of technology for each individual munition. The effectiveness of our cooperative control strategies is being compared to a baseline autonomous (non-cooperating) wide area search munition. Scenarios used as test cases are parameterized by factors such as target density, munition/target ratio, ATR performance (probability of correct target report and False Target Attack Rate, FTAR), warhead lethality, and others. The often competing objectives being addressed for each scenario are illustrated by the following cases:

1 *Maximize probability of correct target identification:* In this scenario we specify that the highest priority is to identify targets, and that they are only destroyed if there are not further opportunities to identify more aspects of the target environment. In this

case the group of AMs will likely miss the opportunity to destroy some targets, but it may be able to communicate a more accurate picture of the target environment back to a command center for a follow-on strike.

2 *Maximize the probability of high priority kills:* In this scenario the emphasis is on the destruction of the highest priority targets. For this reason, an AM finding a high priority target will immediately initiate an engagement, and a nearby munition may also decide to engage the same target. However, an AM finding a lower priority target may communicate information on the target but continue searching for a high priority target. In this case the lower priority targets will only be engaged by munitions that are running low on fuel and therefore have a low probability of finding a high priority target through continued search.

3 *Maximize overall mission effectiveness:* In this case, the set of AM's will seek to maximize a formula consisting of a weighted sum of the expected number of kills on real targets (higher priority counts more) with a negative contribution for an attack on a non-target vehicle. This case, more than the previous two, provides a very challenging problem for cooperative behavior strategies because the munition false target attack rate can result in degraded performance for the cooperative munitions as compared to the baseline autonomous munitions.

2.2. Simulation Testbeds for the Autonomous Munitions Problem

There is a simulation testbed written in FORTRAN that is currently being used at AFIT for evaluating mission effectiveness for cooperative control strategies (see Gillen and Jacques, 2000). The simulation (PSUB) was originally developed by Lockheed Martin Vought Systems for the LOCAAS program, and has been modified for evaluation of cooperative attack strategies. This simulation, however, is proprietary and not well suited to the cooperative search and classification problems. For these reasons, we developed an open simulation within the MATLAB environment. The capabilities of the MATLAB simulation are similar to those of PSUB, namely multiple munitions with generic target recognition capability (probability table look-ups) searching for multiple moving targets of differing priority, with user specified environmental characteristics (e.g., target densities, false target densities, etc.). Two example plots from our testbed are shown in Figure 1.4. Here we show the use of

"serpentine" and "random" search for a rectangular region (the squares designate targets and triangles are vehicles).

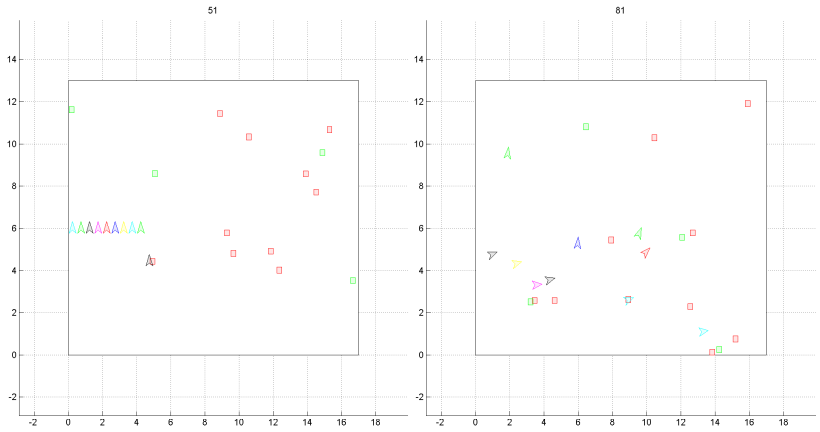

Figure 1.4. Output from autonomous munitions simulation testbed.

The Matlab cooperative munitions simulation test bed has a friendly graphical user-interface that facilitates visualization of UAAV swarm dynamics and the effects of coordination strategy design choices. It uses object-oriented features of Matlab, has Monte-Carlo simulation abilities, and provides numerical measures of mission effectiveness. We use it in the evaluation of the effectiveness of coordination strategies, and to study trade-offs in design. A detailed description of the progress made on the development and evaluation of a "behavioral rule scheme" for the autonomous munition problem is given in the companion paper (Gillen and Jacques, 2000).

3. Cooperative Control via Distributed Learning and Planning

In this section, we present a general cooperative search framework for distributed agents. This framework is used to explain how the N vehicles work together to achieve distributed learning about their environment and how distributed learning can facilitate distributed planning of UAAV activities for search and engagement missions.

3.1. Distributed Learning

Each UAAV has a three dimensional map, which we will refer to as "search map," that serves as the vehicle'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 vehicle "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 the certainty about knowing the environment at the point (x, y) in the search region. If $\mathcal{S}(x, y; \theta) = 0$ then the vehicle 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 UAAV knows everything (or equivalently, the vehicle is totally certain) about the environment at (x, y). As the vehicle 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 vehicles. Therefore, the search map of each vehicle 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 $\theta \in \Re^q$. 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 vehicle 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 vehicle moves, gathers information about the environment, and processes the information based on automatic target recognition software, or other image processing methods. This is illustrated in Figure 1.5, where we show the area scanned by a "generic" sensor on a UAAV 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 $\hat{\theta}$.

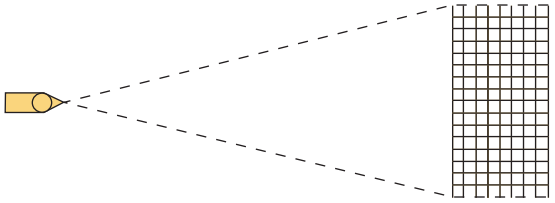

Figure 1.5. An example of a scan area for a UAAV.

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 vehicle's knowledge of the environment, it 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 vehicle 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 in using sensor information from vehicle $\mathcal{A}_i$ and information coming from other vehicles $\mathcal{A}_j$, $j \neq i$ 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 (Weaver et al., 1998). 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 vehicle to another.

At the time of deployment, it is assumed that each vehicle 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 vehicle is initialized with the same search map. However, in some applications it may be useful to have UAAVs be "specialized" to search in certain regions, in which case the search environment for each UAAV, as well as the initial search map, may be different.

3.2. Cooperative Path Planning

One of the key objectives of each vehicle is to on-line select a suitable path in the search environment $\mathcal{X}$. To be consistent with the motion dynamics of air vehicles, it is assumed that each vehicle has limited maneuverability, which is represented by a maximum angle θ_m that the vehicle can turn from its current direction. For simplicity we assume

that all vehicles move at a constant velocity μ (this assumption can be easily relaxed).

Plan Generation. To describe the movement path of vehicle $\mathcal{A}_i$ between samples, we define the *movement sampling time* T_m as the time interval in the movement of the vehicle. In this framework, we let $p_i(k)$ be the position (in terms of (x, y) coordinates) of i-th vehicle at time $t = kT_m$, with the vehicle following a straight line in moving from $p_i(k)$ to its new position $p_i(k+1)$. Since the velocity μ of the UAAV 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 1.6. 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 vehicle belongs to one of the elements of the above set; i.e., $p_i(k+1) \in \overline{\mathcal{P}}_i(k+1)$.

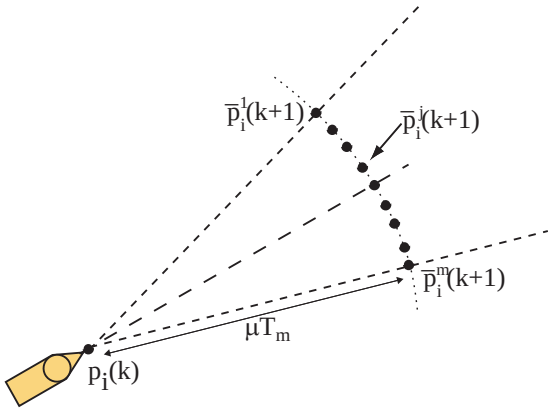

Figure 1.6. Selection of the next point in the path of the vehicle.

The UAAV 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 vehicle $\mathcal{A}_i$ 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 vehicle 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 vehicle 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 since air vehicle require (at least) some trajectory planning. 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 1.7 for the case where $q = 6$.

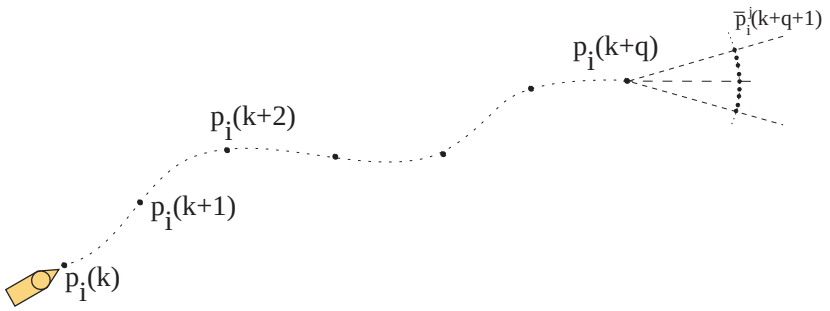

Figure 1.7. Illustration of the recursive q-step ahead planning algorithm.

If practical considerations (such as motion dynamics of the vehicle and computational demands for path selection) require a relatively large value for q then the problem of "outdated" decision making can be ameliorated by an *interleaved* type of scheme. We define a (q, r)-interleaved decision making scheme as follows:

- When vehicle $\mathcal{A}_i$ 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 vehicle 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 $p_i(k+q-r+1)$, $p_i(k+q-r+2)$, ..., $p_i(k+q)$, $p_i(k+q+1)$.

The term "interleaved" is used to express the fact that decisions are re-calculated over time, as the vehicle 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 vehicle need to be selected a priori. The interleaved path planning procedure is illustrated in Figure 1.8 for the case where $q = 6$ and $r = 2$.

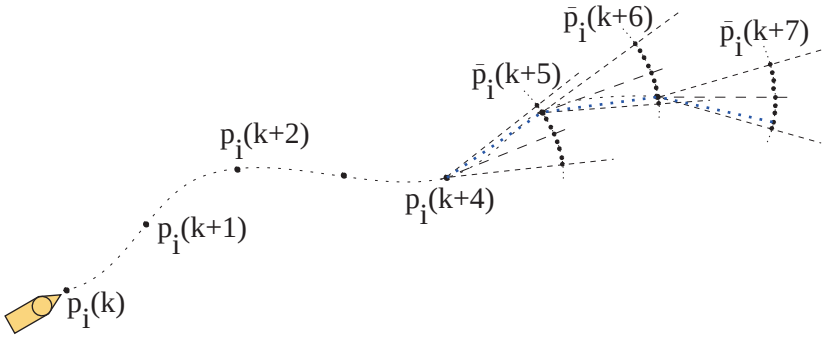

Figure 1.8. Illustration of the (q, r)-interleaved decision making procedure.

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 vehicle has to select one position among m possible candidates. With the (q, r)-interleaved algorithm, each vehicle 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 1.8 where $m = 9$, $r = 2$; therefore at each sample time the vehicle 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.

Plan Selection. Given the current information available via the search map, and the location/direction of the team of vehicles (and possibly other useful information, such as fuel remaining, etc.), each vehicle uses a multi-objective cost function J to select and update its search path. At decision sampling time T_d, the vehicle 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 vehicle 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 sub-goal.

The following is a list (not exhaustive) of possible sub-goals that a search vehicle may include in its cost criterion. Corresponding to each sub-goal is a cost-criterion component that can be readily 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 1.9). 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 cost criterion (S1) pushes the vehicle 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 vehicle to seek not only the instantaneous minimizing path, but also a path that will cause the vehicle 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 vehicle (see the triangular regions ahead of $\bar{p}_i^j(k+1)$ in Figure 1.9).

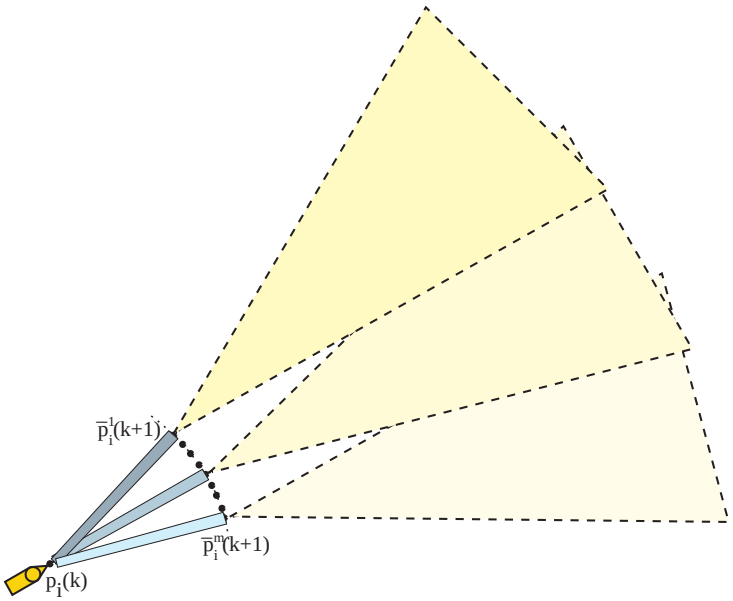

Figure 1.9. Illustration of the regions that are used in the cost function for finding the optimal search path.

C1 *Follow the path where there is the minimum overlap with other vehicles.* Since the vehicles are in frequent communication with each other, they are able to share their new information about the search region, thereby the search map of each individual vehicle is similar to the search maps of the other vehicles. Consequently, it is natural that they may select the same search path as other vehicles (especially since in general they will be utilizing the same search algorithm). This will be more pronounced if two vehicles happen to be close to each other. However, in order to minimize the global uncertainty associated with the emergent knowledge of all vehicles, it is crucial that there is minimum overlap in their search efforts. This can be achieved by including a cost function component that penalizes vehicles 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 vehicles.

SE1 *Follow the path that maximizes coverage of the highest priority targets.* In mission applications where the vehicles 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 vehicle 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 vehicle is a key limiting factor. 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 vehicle 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 vehicles.* Cooperation between vehicles is a key issue not only in search patterns but also—and even more so—in engagement patterns. If an vehicle decides to engage a target, there needs to be some cooperation such that no other vehicle 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 $\overline{\mathcal{P}}_i$.

In the next subsection, 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.

3.3. Cooperative Search and Learning Results

The proposed cooperative search and learning framework has been tested in several simple simulated studies. Three of these simulation studies are presented in this subsection. In the first simulation experi-

ment there are two vehicles, while in the second simulation we will be using a team of five vehicles. In both of these simulation studies we are using the recursive q-step ahead planning algorithm with $q = 3$. In the final simulation experiment we compare the q-step ahead planning algorithm with the interleaved scheme, and also examine the effect of a dynamic environment, where the location of targets maybe changing over time.

The results for the case of two vehicles are shown in Figure 1.10. 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 (Ablavsky and Snorrason, 2000). The lower-left plot shows the result of the cooperative search method based on the recursive q-step ahead planning algorithm. The search region is a 200 by 200 area, and we consider two search vehicles which start at the location indicated by the triangles. 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 to be completely uncertain. For simplicity, in this simulation the only mission is for the vehicles to work cooperatively in reducing the uncertainty in the environment (by searching in highly uncertain areas).

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) vehicles, 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 preset search pattern. As seen from Figure 1.10, each vehicle adapts its search path on-line based on current information from its search results, as well as from search results of the other vehicles.

To compare the performance of the three search patterns, the lower-right plot of Figure 1.10 shows the percentage of uncertainty with time for the standard search pattern, the random search pattern and the co-

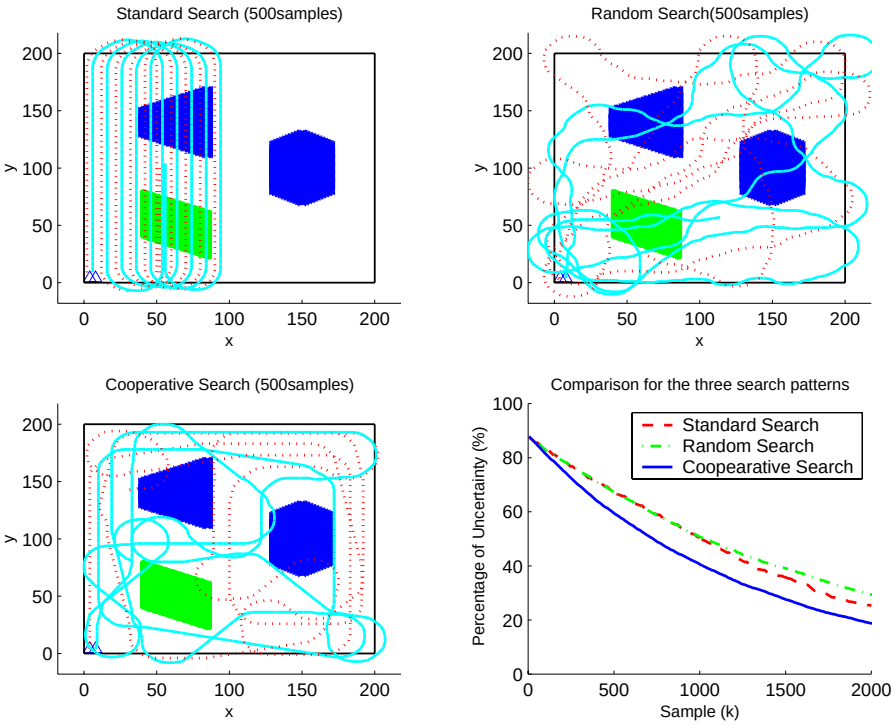

Figure 1.10. Comparison of the cooperative search pattern with a "standard" search pattern and a random search pattern for the case of two moving vehicles. The upper-left plot shows a standard search pattern for the first 500 time samples; the upper-right plot shows the corresponding search pattern in the case of a random search, subject to some bounds to restrict the vehicle 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.

operative 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.

The corresponding results in the case of five vehicles moving in the same environment is shown in Figure 1.11. In this simulation study we assume that the initial information about the environment is slightly different from before. The results are analogous to the case of two vehicles.

In these simulation studies, we assume that the sampling time $T_m = 1$ corresponds to the rate at which each vehicle receives information from its own sensors, updates its search map and makes path planning de-

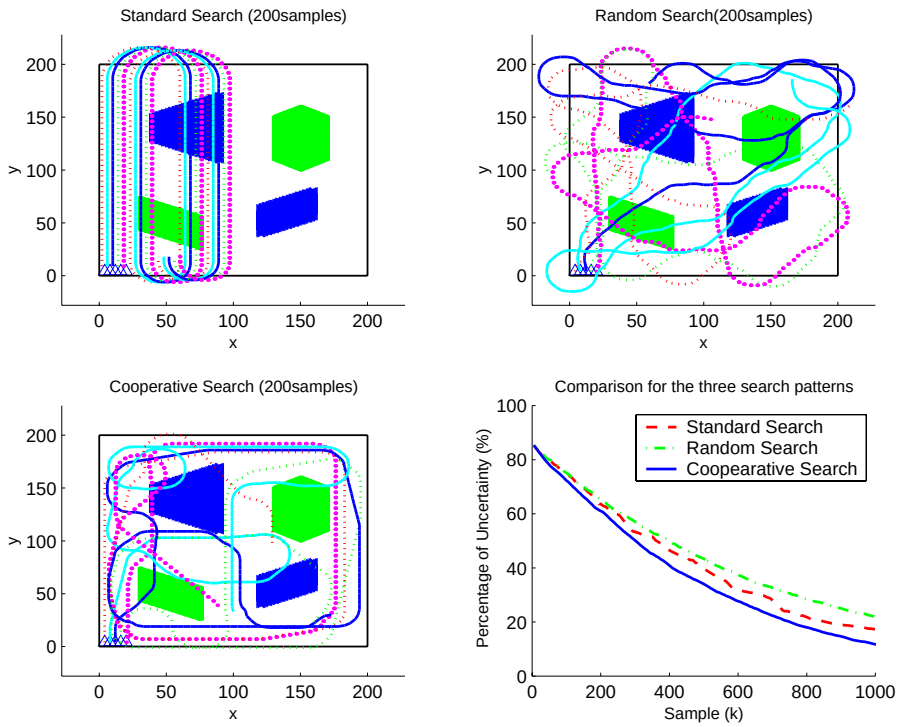

Figure 1.11. Comparison of the cooperative search pattern with a "standard" search pattern and a random search pattern for the case of five moving vehicles. The upper-left plot shows a standard search pattern for the first 500 time samples; the upper-right plot shows the corresponding search pattern in the case of a random search, subject to some bounds to restrict the vehicle 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.

cisions. Information from other vehicles is received at a slower rate. Specifically, we assume that the communication sampling time T_c between vehicles is five times the movement sampling time; i.e., $T_c = 5T_m$. For fairness in comparison, it is assumed that for the standard and random search patterns the vehicles 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.

In the first two simulations it was assumed that learning of the environment was monotonically increasing with time; in other words, as the team of UAAVs moved around the environment, they were able to en-

hance their knowledge, leading to decreasing levels of uncertainty. In the case of a dynamic environment the targets may be moving around, and therefore the represented level of uncertainty increases with time. In the framework followed in this paper, reduction in the certainty levels due to a dynamic environment can be implemented by using a decay value $d < 1$ that multiplies the search map; i.e., $z = \mathcal{S}(x, y, \hat{\theta})d$. Therefore, in the case of a dynamic environment, learning (which reduces uncertainty) competes with the increase in uncertainty due to a changing environment. In the third simulation experiment, shown in Fugure 1.12, we consider the same environment as in Figure1.11 with five UAAVs, but introduce a decay to represent a dynamic environment. The simulation shown is for three different levels of decay: (i) $d = 0.98$; (ii) $d = 0.995$; and (iii) $d = 1$ (no decay). This decay factor is applied every 5 steps. For different levels of decay, we compare the non-interleaved planning algorithm $r = 0$ with the interleaved search planning algorithm $r = 1$. As expected, the rate of reduction in percentage of uncertainty decreases as the decay rate d decreases. The plot also shows that, overall, the interleaved search planning algorithm performs slightly better than the corresponding non-interleaved.

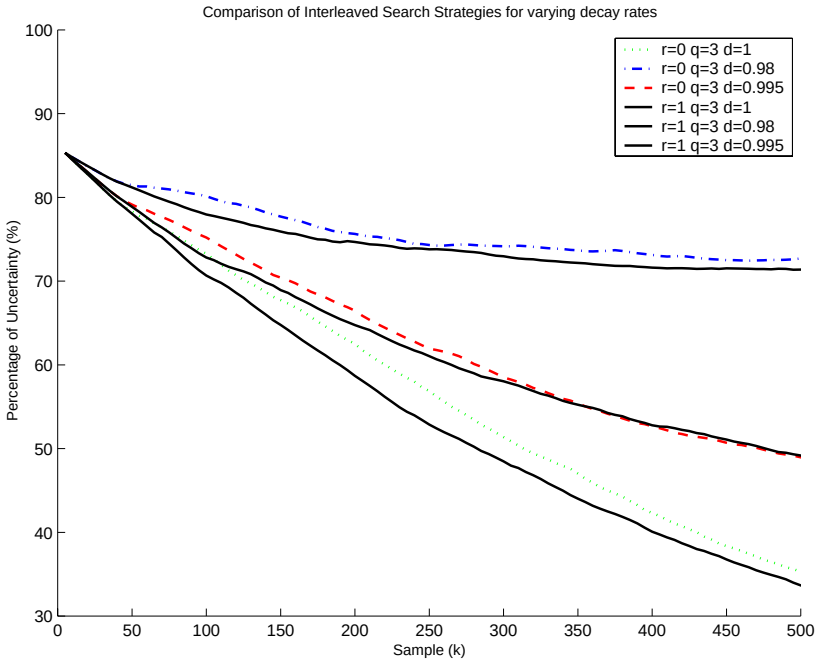

Figure 1.12. Comparison of interleaved search strategies for varying decay rates.

It is noted that in these simulations the path planning of the cooperative search algorithm is rather limited since at every sampled time each vehicle 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.

3.4. Related Research Work on Search Methods

Various types of 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 (Stone, 1975; Koopman, 1980), exhaustive geographic search (Spires and Goldsmith, 1998), obstacle avoidance (Cameron, 1994; Snorrason and Norris, 1999) and derivative-free optimization methods (Conn et al., 1997).

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 (Stone, 1975; Koopman, 1980). Search theory was initially formed 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 are given by (Stone, 1983; Richardson, 1987; Benkoski et al., 1991).

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 sec-

ond 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 (Stone, 1975). For the moving target problem, the emphasis in search theory has shifted from mathematical and analytical solutions to algorithmic solutions (Benkoski et al., 1991). A typical type of search problem, called the path constrain search problem (PCSP), that takes into account the movement of the searcher, was investigated by several researchers (Eagle and Yee, 1990; Stewart, 1980; Hohzaki and Iida, 1995b; Hohzaki and Iida, 1995a). 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 (Danskin, 1968; Hohzaki and Iida, 2000; Washburn, 1980). 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, 1996).

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 (Spires and Goldsmith, 1998). 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 (Spires and Goldsmith, 1998; Goldsmith and Robinett, 1998; Hert et al., 1996; Choset and Pignon, 1997).

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 (Cameron, 1994; Snorrason and Norris, 1999). 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 (Khatib, 1985). 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 (Conn et al., 1997). 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 (Torczon, 1997; Lucidi and Sciandrone, 1997), the Nelder and Mead Simplex Method (Nelder and Mead, 1965), the Parallel Direct Search Algorithm (Dennis and Torczon, 1991), and the Multi-directional Search Method (Torczon, 1991). In one way or another most of derivative free optimization methods use measurements of the cost function and form approximations to the gradient to decide which direction to move. (Passino, 2001) provides some ideas on how to extend non-gradient methods to team foraging.

4. Stable Vehicular Swarms

Groups of communicating vehicles, such as automobiles (e.g., in "platoons" in automated highway systems), robots, underwater vehicles, or aircraft formations have been studied for some time. In the current project we are conducting an analytical study where we are mathematically modeling "vehicular swarms" (groups of vehicles that behave as a single entity) where there are communication delays, and deriving conditions under which the swarms are stable (cohesive). The central focus of our work is not "swarming" per se, but the factors that drive a group of UAAVs to decide to aggregate (e.g., work together to search an area, or make a coordinated attack) or disintegrate (e.g., spread out to search, or scatter to evade a threat). Such factors include characteristics of the vehicle-to-vehicle communications, target/threat priorities and densities, and mission goals.

4.1. Stable Asynchronous Vehicular Swarms with Communication Delays

To briefly summarize our current work we first explain how stability relates to swarm cohesion, aggregation, and disintegration. Next, we overview the types of stability conditions we obtain for swarms, and show the output of a simulation testbed we have developed for studying swarm dynamics.

Stability, Cohesion, Aggregation, and Disintegration:. To achieve coordinated actions, a group of autonomous vehicles requires vehicle-to-vehicle communications. When should a swarm of UAAVs aggregate and when should it disintegrate? Some examples include:

- It could enhance mission effectiveness to aggregate for a coordinated attack on high priority target, or to protect one another from certain types of threats (e.g., vehicles on the edges of the swarms could protect ones in the middle). Or, it may be beneficial to work together to search an area.

- In some situations mission effectiveness could be enhanced if the swarm disintegrates (breaks up, disperses) for good search coverage (i.e., so search areas do not overlap) when there are many similar stationary targets with a uniform density over the search area. At other times it may be beneficial for a swarm to disintegrate in order to evade certain types of threats (e.g., a threat that has the ability to kill a whole group of UAAVs, in an especially effective way if they are all grouped together).

- Local UAAV goals and overall mission goals, which may compete with each other, affect the dynamics of vehicle swarm aggregation and disintegration. For instance, if the mission goals dictate that only the highest priority targets should be engaged, then for some coordination strategies it may be that there is a higher likelihood of aggregation near high priority targets for coordinated attacks on them.

A problem of critical importance is then to determine how characteristics of vehicle-to-vehicle communications affect the ability of a group of UAAVs to aggregate and disintegrate. Can aggregation/disintegration be achieved when the sensor ranges for each UAAV is limited? How does the sensor range impact formation of multiple swarms, and the ability to group vehicles for a coordinated attack? What are the effects of communication delays on the dynamic behavior of a swarm, and

its effectiveness? Moreover, it is clear that characteristics of the target/threat environment affect swarm aggregation/disintegration. Can we characterize a relationship between target density and priority and when it is good to swarm, or how this will drive a group of UAAVs to swarm? What threat patterns should drive the disintegration of a swarm of UAAVs? Finally, mission goals affect the overall dynamics of the vehicular swarm. How can we make sure that mission goals are properly specified so that detrimental aggregation/disintegration events do not occur (e.g., aggregation near a threat so that an entire group of UAAVs could be easily eliminated), and hence mission effectives is maintained.

In summary, stability analysis focuses on the underlying mechanisms that affect the dynamics of the coordinated behavior of the *group* of UAAVs. It is of fundamental importance to understand how to exploit the benefits of coordinated search and engagement for a group of UAAVs.

Stability of Swarms Modeled as Distributed Discrete Event Systems:. Here, our approach to begin to study such problems is to introduce a discrete event system (DES) framework based on the approach in (Passino and Burgess, 1998), characterize swarm cohesion characteristics via stability properties, then use relatively conventional stability analysis methods (e.g., Lyapunov methods) to provide conditions under which various cohesion properties can be achieved. The key theoretical difficulty is how to perform stability analysis for distributed DES with communication delays; however, the approaches in (Passino and Burgess, 1998; Tsitsiklis and Bertsekas, 1989) provide a basis to start, and this is the approach that we take.

A two-dimensional (2-D) swarm is formed by putting many single swarm members together on the (x_1, x_2)-plane. Assume that we have a two-dimensional asynchronous N-member swarm, where $N > 1$. Let $x^i(t)$ denote the position vector of swarm member i at time t. Let T^i be the set of time indices at which the i^{th} vehicle moves. We have $x^i(t) = [x_1^i(t), x_2^i(t)]^\top \in \Re^2$, $i = 1, 2, ..., N$, where $x_1^i(t)$ and $x_2^i(t)$ are member i's horizontal and vertical position coordinates respectively. The single swarm member model used here to discuss some of the basic ideas of our work includes only the simplest of dynamics (point-mass), neighbor position sensors, and proximity sensors. For a swarm member, we assume all other members which can be sensed by its neighbor position sensors are its "neighbors," and its sensed position information about its neighbors may be subjected to random but bounded delays. Define $\tau_j^i(t)$ to be the last time at which vehicle i received position information from vehicle j, where $i, j = 1, 2, ..., N$, $j \neq i$. Assume the proximity sensor of the swarm members has a circular-shaped sensing range with a radius

$\varepsilon > 0$, which can instantaneously indicate the positions of any other members inside this range. Swarm members like to be close to each other, but not too close. Assume each member has a "private" area, which is a circular-shaped area with a radius $d > 0$ around each member, where d is the desired comfortable distance between two adjacent swarm neighbors, which is known by every swarm member. For some $\gamma > 0$, assume $[d, d+\gamma]$ is a "comfortable distance neighborhood" relative to $x^i(t)$ and $x^j(t)$ for all i and j, where γ is the comfortable distance neighborhood size. In other words, swarm members do not want other members to enter its private area and they prefer their neighbors are at a comfortable distance neighborhood $[d, d+\gamma]$ from them. Here, for the sake of illustration we choose $\gamma = d = \varepsilon/2$. We assume that $|x| = \sqrt{x^\top x}$ and $|x^i(0) - x^j(0)| > d$, for $i, j = 1, 2, ..., N, i \neq j$ initially. We make certain assumptions about the times that all the vehicles will update their position and bounds on the length of the delays for obtaining neighbor's position information (similar to the "partial asynchronism" assumption in (Tsitsiklis and Bertsekas, 1989)).

Each swarm member i, $i = 1, 2, ..., N$, remains stationary at the beginning until some $t' \in T^i$, where it has sensed all its $N-1$ neighbors' position information and it then calculates the center position of them as its goal position. Assume the goal position $x_c^i(t)$ of member i at time $t \in T^i$ is defined as

$$x_c^i(t) = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} x^i(\tau_j^i(t)), \forall t \in T^i, i = 1, 2, ..., N \qquad (1)$$

Member i tries to approach this goal position since it wants to be adjacent to all its neighbors. Note that $x_c^i(t)$ may not be the real center of its neighbors since the swarm member's sensed information may include random delays. Assume swarm member i will move toward $x_c^i(t)$ with a step size d, which is the radius of its private area if it finds the distance between them is larger than or equal to d. It will move to $x_c^i(t)$ with one step if it finds the distance is less than d. It will remain stationary if it is already at $x_c^i(t)$. During movements it does not want any other member to enter into its private area because swarm members like to be close to each other, but not so close as to enter others' private area so that collisions can occur. Hence, assume that before swarm member i moves to a new position it will detect if there is any member inside the private area around this new position via its proximity sensors (note that we choose $\varepsilon = 2d$ so that the sensing range of proximity sensors is large enough to detect the new private area). It moves to this new position only when no member is found in the new private area. Otherwise, it remains stationary.

From the above moving rules, we assume the step vector $\phi^i(t)$ of member i at time $t \in T^i$ is such that

$$\begin{aligned} \phi^i(t) &= \min\{d, |x_c^i(t) - x^i(t)|\} \left[\frac{x_c^i(t) - x^i(t)}{|x_c^i(t) - x^i(t)|} \right], \\ & \forall t \in T^i, i = 1, 2, ..., N \end{aligned} \tag{2}$$

Here, the term in brackets is a unit vector which represents the moving direction, where $x_c^i(t)$ is the goal position of member i at time t defined in Equation (1), and the item in front of the brackets is a step size scalar, where "min" is used to model the rules of how to choose the step size. The step size is equal to d if $|x_c^i(t) - x^i(t)| \geq d$, and is equal to $|x_c^i(t) - x^i(t)|$ if $|x_c^i(t) - x^i(t)| < d$.

A mathematical model for the above asynchronous mobile swarm is given by

$$\begin{aligned} x^i(t+1) &= x^i(t) + \phi^i(t) \prod_{j=1, j \neq i}^{N} u(|x^i(t) + \phi^i(t) - x^j(t)| - d), \\ & \forall t \in T^i, i = 1, 2, ..., N \end{aligned} \tag{3}$$

where "u" is the step function, which is equal to one when the function variable is larger than or equal to zero, and is equal to zero when the function variable is less than zero. Clearly, swarm member i will remain stationary if it detects any neighbor in its new private area, and will move to a new position $x^i(t) + \phi^i(t)$ if no neighbor is found in its new private area.

4.2. Stability Analysis and Simulation Results

We have derived conditions involving bounds on communication delays, and characteristics of the asynchronous behavior of the communications, that are sufficient to guarantee the ability of a 1-D asynchronous vehicular swarm to aggregate. Our approach is based on defining a Lyapunov-like function for the inter-vehicle distance for two swarm members, showing properties for this case, and then using an induction method to extend it to the multiple-swarm member case. We show that in some cases intervehicle distances converge to a desired constant (i.e., asymptotic stability), while if communications are degraded such distances only converge to a pre-specified neighborhood (i.e., uniform ultiimate boundedness). We are currently extending these results to the two (and M) dimensional cases and investigating how the design of the coordination strategies affect stability conditions.

In addition, we have developed a simulation testbed for gaining insights into the dynamics of vehicular swarms. Below, in Figure 1.13 we show a sequence of plots to illustrate swarm dynamics for a 2-D swarm. Here, we show one lead vehicle (the diamond) moving across a plane, and there are several other vehicles that swarm together and follow the lead vehicle. Such simulation results show how: (i) aggregation is achieved, (ii) the dynamics of cohesion (e.g., when moving as a group do the communication delays cause unacceptable oscillations in inter-vehicle distance), and (ii) characteristics of motion of a mobile swarm (e.g., how motion in one direction "stretches" the swarm along the velocity vector of the group, how coordinated motion can slow UAAV velocities since adjustments made to maintain cohesion result in movements to avoid collisions that could be opposite to the velocity vector of the swarm).

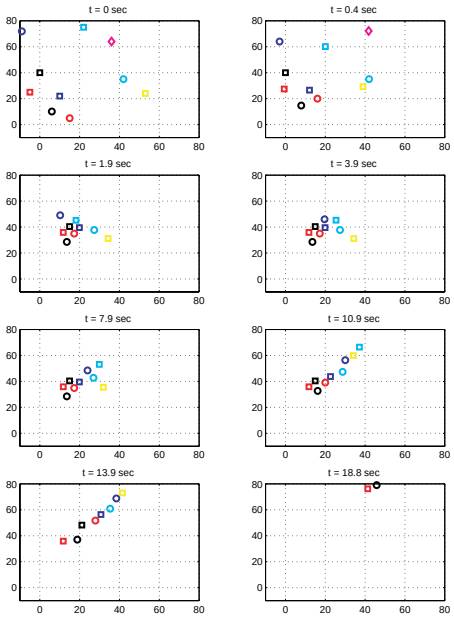

Figure 1.13. Mobile vehicular swarm (plots show a sequence of eight positions of the vehicles over time).

5. Biomimicry of Foraging for Cooperative Control

Animal species of all sorts have been involved in daily "warfare" for millions of years when they seek food (prey) and try to avoid noxious substances or predators (i.e., when they "forage"). What can we learn

from the evolved behaviors, especially in the case of "social foraging" where groups of animals work together to capture more prey (in our domain, kill targets) and at the same time avoid predators (in our domain, evade threats)? In this section we will briefly overview some initial work done to understand the relevance of social foraging behaviors of animals to the cooperative control problem for UAAVs. For more details see (Passino, 2001).

5.1. Optimal Decision-Making in Foraging

Animals search for and obtain nutrients in a way that maximizes $\frac{E}{T}$ where E is energy obtained, and T is time spent foraging (or, they maximize long-term average rate of energy intake) (Stephens and Krebs, 1986). Evolution optimizes foraging strategies since animals that have poor foraging performance do not survive. Generally, a foraging strategy involves finding a "patch" of food (e.g., group of bushes with berries), deciding whether to enter it and search for food (do you expect a better one?), and when to leave the patch. There are predators and risks, energy required for travel, and physiological constraints (sensing, memory, cognitive capabilities). Foraging scenarios can be mathematically modeled and optimal policies can be found using, for instance, dynamic programming (Stephens and Krebs, 1986).

Some animals forage as individuals, others forage as groups, and some will choose which approach to use depending on the current situation. While to perform social foraging an animal needs communication capabilities, it can gain advantages in that it can essentially exploit the sensing capabilities of the group, the group can "gang-up" on large prey, individuals can obtain protection from predators while in a group, and in a certain sense the group can forage with a type of collective intelligence. Social foragers include birds, bees, fish, ants, wildebeasts, and primates. Note that there is a type of "cognitive spectrum" where some foragers have little cognitive capability, and other higher life forms have significant capabilities (e.g., compare the capabilities of a single ant with those of a human). Generally, endowing each forager with more capabilities can help them succeed in foraging, both as an individual and as a group.

5.2. E. coli Bacterial Foraging Strategies: Useful for Cooperative Control of UAAVs?

Consider the foraging behavior ("chemotaxis") of *E. coli*, which is a common type of bacteria that takes the following foraging actions (these are "behavioral rules"):

1 If in neutral medium alternate tumbles and runs ⇒ Search

2 If swimming up nutrient gradient (or out of noxious substances) swim longer (climb up nutrient gradient or down noxious gradient) ⇒ Seek increasingly favorable environments

3 If swimming down nutrient gradient (or up noxious substance gradient), then search ⇒ Avoid unfavorable environments

What is the resulting emergent pattern of behavior for a whole group of *E. coli* bacteria? Generally, as a group they will try to find food and avoid harmful phenomena, and when viewed under a microscope you will get a sense that a type of intelligent behavior has emerged since they will seem to intentionally move as a group (analogous to how a swarm of bees moves). To help picture how the group dynamics of bacteria operate, consider Figure 1.14. Here, a "capillary technique" for studying chemotaxis in populations of bacteria is shown. A capillary containing an attractant is placed in a medium with a bacterial suspension in Figure 1.14(a) and the bacteria then accumulate in the capillary containing the attractant as shown in Figure 1.14(b). Figure 1.14(c) shows what happens when the capillary contains neither attractant or repellant (i.e., a neutral environment relative to the medium it is placed in) and Figure 1.14(d) shows what happens when it contains a repellant.

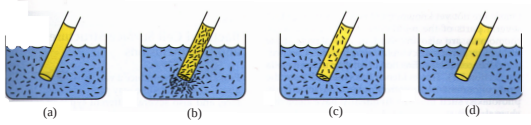

Figure 1.14. Experiment showing how *E. coli* swarm towards nutrients, and away from noxious substances (figure taken from Madigan et al., 1997).

It is interesting to note that *E. coli* and *S. typhimurium* can form intricate stable spatio-temporal patterns in certain semi-solid nutrient media. They can radially eat their way through a medium if placed together initially at its center. Moreover, under certain conditions they will secrete cell-to-cell attractant signals so that they will group and protect each other. These bacteria can "swarm." Other bacteria such as *M. Xanthus* exhibit relatively sophisticated social foraging behaviors involving several types of swarm behavior for protection, survival, and success in obtaining nutrients.

There are clear analogies with cooperative control of UAAVs:

- Animals, organisms = UAAVs

- Prey, nutrients = targets
- Predators, noxious substances = threats
- Environment = battlefield

Are these analogies useful? Biomimicry of social foraging of ants (Bonabeau et al., 1999) has provided some concepts and algorithms useful for the solution of some engineering problems (where the key contribution is new algorithms for combinatorial optimization). What the relevance of the decision-making strategies of social foraging animals to the development of cooperative control strategies for UAAVs?

To begin to answer this question we have developed a model of the bacterial foraging process and have shown how a computer algorithm that emulates the behavior of a group of bacteria can solve a type of distributed optimization problem (in particular, not for combinatorial optimization, but one for a continuous cost function where there is no explicit analytical knowledge of the gradient). In particular, consider Figure 1.15 where we show the trajectories chosen by bacteria in a certain environment. Notice that they are successful in finding the areas where there are nutrients or prey (in our domain, targets) and in avoiding areas where there are noxious substances or predators (in our domain, threats). We have also simulated social foraging (swarming) for *E. coli* and *M. Xanthus* and studied mechanisms of aggregation and disintegration for these.

6. 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 vehicle systems requires a framework that is based more on cooperation between vehicles, 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 vehicles 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

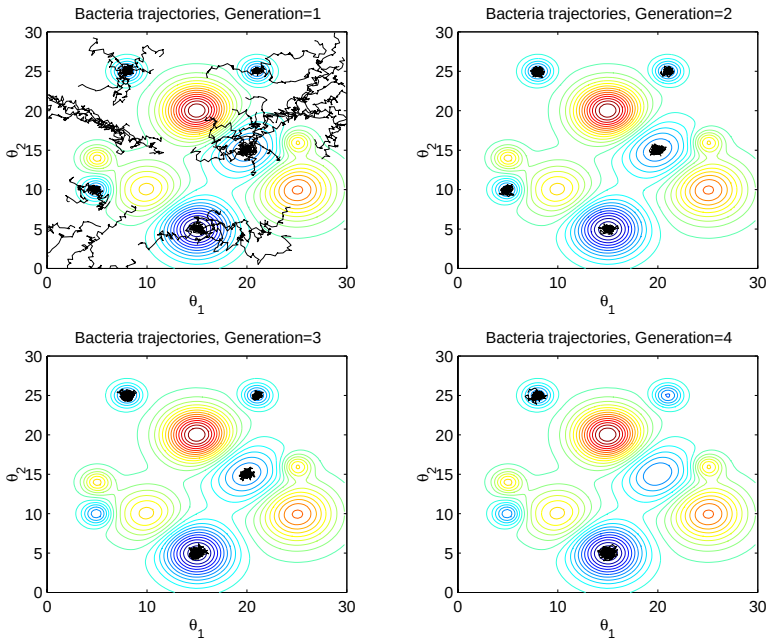

Figure 1.15. Bacterial motion trajectories, generations 1-4, on contour plot.

interleaved planning technique. These ideas were illustrated with simulation studies by comparing them to a restricted random search and a standard search pattern. Moreover, we studied the stability of vehicular swarms to try to understand what types of communications are needed to achieve cooperative search and engagement, and characteristics that affect swarm aggregation and disintegration. Finally, we explored the utility of using biomimicry of social foraging strategies to develop coordination strategies.

References

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

Benkoski, S., Monticino, M., and Weisinger, J. (1991). A survery of the search theory literature. *Naval Research Logistics*, 38:469–494.

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

Cameron, S. (1994). Obstacle avoidance and path planning. *Industrial Robot*, 21:9–14.

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

Conn, A., Scheinberg, K., and Toint, P. (1997). Recent progress in unconstrainted nonlinear optimization without derivatives. *Mathematical Programming*, 79:397–414.

Danskin, J. (1968). A helicopter versus submarines search game. *Operations Research*, 16:509–517.

Dell, R. and Eagle, J. (1996). Using multiple searchers in constrainted-path moving-targer search problems. *Naval Research Logistics*, 43:463–480.

Dennis, J. and Torczon, V. (1991). Direct search methods on parallel machines. *SIAM Journal Optimization*, 1:448–474.

Eagle, J. and Yee, J. (1990). An optimal branch-and-bound procedure for the constrained path moving target search problem. *Operations Research*, 38:11–114.

Gillen, D. and Jacques, D. (2000). Cooperative behavior schemes for improving the effectiveness of autonomous wide area search munitions. In *Proceedings of Workshop on Cooperative Control and Optimization*, University of Florida, Gainesville.

Goldsmith, S. and Robinett, R. (1998). Collective search by mobile robots using alpha-beta coordination. In Drogoul, A., Tambe, M., and Fukuda, T., editors, *Collective Robotics*, pages 136–146. Springer Verlag: Berlin.

Hert, S., Tiwari, S., and Lumelsky, V. (1996). A terrain-covering algorithm for auv. *Autonomous Robots*, 3:91–119.

Hohzaki, R. and Iida, K. (1995a). An optimal search plan for a moving target when a search path is given. *Mathematica Japonica*, 41:175–184.

Hohzaki, R. and Iida, K. (1995b). Path constrained search problem with reward criterion. *Journal of the Operations Research Society of Japan*, 38:254–264.

Hohzaki, R. and Iida, K. (2000). A search game when a search path is given. *European Journal of Operational Reasearch*, 124:114–124.

Jacques, D. and Leblanc, R. (1998). Effectiveness analysis for wide area search munitions. In *Proceedings of the AIAA Missile Sciences Conference*, Monterey, CA.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *International Conference on Robotics and Automation*, pages 500–505, St. Louis.

Koopman, B. (1980). *Search and Screening: General principles with Historical Application*. Pergarnon, New York.

Lucidi, S. and Sciandrone, M. (1997). On the global convergence of derivative free methods for unconstrained optimization. In *Technical Report*. Univ.di Roma.

Madigan, M., Martinko, J., and Parker, J. (1997). *Biology of Microorganisms*. Prentice Hall, NJ, 8 edition.

Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7:308–313.

Passino, K. and Burgess, K. (1998). *Stability Analysis of Discrete Event Systems*. John Wiley and Sons Pub., New York.

Passino, K. M. (2001). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine (to appear)*.

Richardson, H. (1987). Search theory. In *Center for Naval Analyses*. N00-014-83-C-0725.

Snorrason, M. and Norris, J. (1999). Vision based obstacle detection and path planetary rovers. In *Unmanned Ground Vehicle Technology II*, Orlanso, FL.

Spires, S. and Goldsmith, S. (1998). Exhaustive geographic search with mobile robots along space-filling curves. In Drogoul, A., Tambe, M., and Fukuda, T., editors, *Collective Robotics*, pages 1–12. Springer Verlag: Berlin.

Stephens, D. and Krebs, J. (1986). *Foraging Theory*. Princeton Univ. Press, Princeton, NJ.

Stewart, T. (1980). Experience with a branch-and-bound algorithm for constrained searcher motion. In Haley, K. and Stone, L., editors, *Search Theory and Applications*, pages 247–253. Plenum Press, New York.

Stone, L. (1975). *Theory of Optimal Search*. Acadamic Press, New York.

Stone, L. (1983). The process of search planning: Current approachs and the continuing problems. *Operational Research*, 31:207–233.

Torczon, V. (1991). On the convergence of the multidirectional search algorithm. *SIAM Journal Optimization*, 1:123–145.

Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal Optimization*, 7:1–25.

Tsitsiklis, J. and Bertsekas, D. (1989). *Parallel and Distributed Computation*. Prentice-Hall, Inc., Engelwood Cliffs, NJ.

Washburn, A. (1980). Search-evasion game in a fixed region. *Operations Research*, 28:1290–1298.

Weaver, S., Baird, L., and Polycarpou, M. (1998). An analytical framework for local feedforward networks. *IEEE Transactions on Neural Networks*, 9(3):473–482.