Submitted to the 2003 IEEE Conference on Decision and Control

Multiple UAV Cooperative Search under Collision Avoidance and Limited Range Communication Constraints

Randal W. Beard[1]
Electrical and Computer Engineering
Brigham Young University
Provo, Utah 84602
beard@ee.byu.edu

Timothy W. McLain
Mechanical Engineering
Brigham Young University
Provo, Utah 84602
tmclain@et.byu.edu

Abstract

This paper focuses on the problem of cooperatively searching, using a team of unmanned air vehicles (UAVs), an area of interest that contains regions of opportunity and regions of potential hazard. The objective of the UAV team is to visit as many opportunities as possible, while avoiding as many hazards as possible. To enable cooperation, the UAVs are constrained to stay within communication range of one another. Collision avoidance is also required. Algorithms for team-optimal and individually-optimal/team-suboptimal solutions are developed and their computational complexity compared. Simulation results demonstrating the feasibility of the cooperative search algorithms are presented.

1 Introduction

Consider the problem where a team of unmanned air vehicles (UAVs) is given the task of searching a region with unknown opportunities and hazards. We assume that each UAV is equipped with sensing capability that identifies regions of opportunity and regions of hazard in the immediate look-ahead window of the UAV. We also assume that the team of UAVs is equipped with a communication network, and that the connectivity of the network depends upon the relative distance between neighboring UAVs. Therefore, maintaining network connectivity constrains the maximum allowable distance between UAVs. In addition, we assume that the problem is essentially two dimensional so collision avoidance must be accounted for explicitly. The control objective for the team is to maximize the regions of opportunity visited by the team, while minimizing the regions of hazard visited by the team, subject to two path constraints: (1) that the communication network remains connected at all times, and (2) that there are no collisions between UAVs.

In our previous work on cooperative control of UAVs [1, 2] we have used the path planning architecture shown in Figure 1. The cooperative waypoint

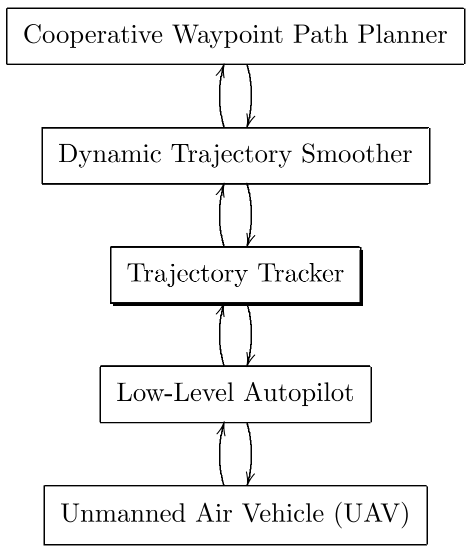
Cooperative Waypoint Path Planner
Dynamic Trajectory Smoother
Trajectory Tracker
Low-Level Autopilot
Unmanned Air Vehicle (UAV)

Figure 1: Path Planning Architecture.

path planner (CWPP) produces waypoint paths for the UAVs which satisfy the cooperation constraints. The dynamic trajectory smoother (DTS) smoothes through the waypoint paths producing time parameterized trajectories which maintain the cooperation constraints and also satisfy the kinematic constraints of the UAV [3, 4]. The trajectory tracker (TT) uses inertial position and heading information to track the trajectory produced by the DTS, and outputs desired altitude, velocity, and heading commands [5]. The physical UAV is controlled by a low-level autopilot which maintains commanded altitude, velocity and heading [6].

In our approach, cooperation constraints such as collision avoidance and maintaining communication connectivity, are handled by the cooperative waypoint path planner. The primary contribution of this paper is to describe several designs for the cooperative path planner given collision avoidance and limited communication constraints.

Recent work in trajectory generation for single UAVs

[1]Corresponding author.

includes probabilistic roadmap approaches [7, 8], differential flatness approaches [9, 10, 11], spline optimization approaches [12], and approaches that build on Dubin's circles [13, 14, 15, 4]. Any of these algorithms could be used in the dynamic trajectory smoothing block shown in Figure 1.

Cooperative path planning has been addressed in the robotics literature. Ref. [16] assumes that paths are designed myopically for N robots and then effects coordination by finding the optimal start times such that the robots do not collide with each other. A similar idea is presented in [17], which also allows robot velocities to be adjusted. Unfortunately, UAVs have minimum velocity constraints and must be in constant motion. Therefore the approaches in [16, 17] are not directly applicable to UAV scenarios.

This paper addresses the cooperative search problem for UAVs. Cooperative coverage of *a priori* unknown rectilinear environments using mobile robots is discussed in [18]. Ref [19], uses neural networks to direct robots in complex domains with dynamically moving obstacles.

In recent years, there has been a great deal of work on cooperative control for UAVs. The cooperative control problem that has received the most attention is formation flying [20, 21, 22]. In formation flying, the UAV trajectories are dynamically coupled through the physics of close flight. By exploiting the physical structure of the problem, path planning for formation flying applications can be reduced to path planning algorithms for single vehicles [23].

Unfortunately, there are many other cooperative control problems that do not admit solutions that are extensions of single vehicle solutions. These include cooperative rendezvous [24, 25], coordinated target assignment and intercept [26, 1], multiple task allocation [27, 28], and ISR scenarios [29].

The full solution to many of these cooperative control problems are NP-hard. While formation flying problems can be solved efficiently using numerical methods, there is a need to identify other classes of cooperative control problems that can also be solved efficiently. This paper is a step in that direction. In particular, we show that the class of cooperative search problems, where the vehicles are assumed to maintain a relative front temporally, and nearest neighbors spatially, can be solved efficiently using dynamic programming techniques.

The paper is organized as follows. In Section 2 we define the problem in mathematical terms. To facilitate the discussion, we also introduce two example problem scenarios. These examples will be used throughout the paper to illustrate the algorithms as they are introduced. In Section 3 we present an algorithm that solves the global optimization problem introduced in Section 2. Fortunately the algorithm is polynomial in the number of vehicles, but unfortunately it is exponential in the look-ahead window. In Section 4 we introduce an information reduction scheme based on individual best paths. The reduction scheme is used in Section 5 to produce a suboptimal search algorithm based on a best-leader approach and in Section 6 to derive a second cooperative search algorithm that resembles the global optimal. The computational complexity of both approaches is shown to be polynomial in the input data. Conclusions are given in Section 7.

2 Problem Definition and Notation

In this section we will introduce the notation and define two example problem scenarios that will be used to illustrate the ideas throughout the paper.

Consider a team of vehicles performing a cooperative search problems where the velocities of the vehicles are adjusted so that they move through the search domain maintaining a uniform longitudinal front. Since the motion in the longitudinal direction is uniform, the group dynamics are encapsulated in the lateral motion. Consider the discrete time dynamics

$$x_n[k+1] = f(x_n[k], u_n[k]), \quad n = 1, \ldots, N, \tag{1}$$

where $x_n[k] \in \mathcal{X}$ and $u_n[k] \in \mathcal{U}$. We will assume that $\mathcal{U}$ is a finite set of options. We will assume that the agents are initially ordered such that

$$x_1[0] < x_2[0] < \cdots < x_N[0],$$

and that it is desirable to maintain this ordering throughout the scenario. We will assume that the agents are at constant altitude and therefore have a collision avoidance constraint which can be quantified as

$$\underline{A} < x_{n+1}[k] - x_n[k], \quad n = 1, \ldots, N-1. \tag{2}$$

We will also assume that the vehicles need to maintain communication connectivity and that connectivity is a function of relative lateral distance. The communication connectivity constraint can be quantified as

$$x_{n+1}[k] - x_n[k] < \overline{A}, \quad n = 1, \ldots, N-1. \tag{3}$$

Let $\mathcal{R}(x, \ell)$ denote the set in $\mathcal{X}$ that is reachable from x after ℓ decisions. The reachable set can be constructed recursively as

$$\begin{aligned} &\mathcal{R}(x, 0) = \{x\} \\ &\mathcal{R}(x, \ell) = \{\xi \in \mathcal{X} : \xi = f(z, u), z \in \mathcal{R}(x, \ell-1), u \in \mathcal{U}\}. \end{aligned}$$

A path $\mathbf{p}(x, \ell)$ is a sequence of state-control pairs, i.e.,

$$\mathbf{p}(x, \ell) = \left(\begin{pmatrix} \xi[1] \\ \nu[0] \end{pmatrix}, \begin{pmatrix} \xi[2] \\ \nu[1] \end{pmatrix}, \cdots, \begin{pmatrix} \xi[\ell] \\ \nu[\ell-1] \end{pmatrix} \right),$$

where $\xi[1] = f(x, \nu[0])$ and $\xi[j+1] = \xi[j] + \nu[j]$ for $j = 1, \ldots, \ell - 1$. Let $\mathcal{P}(x, \ell)$ be the set of all paths of length ℓ starting at x. We will assume that $\mathcal{P}(x, \ell)$ is a finite set that can be enumerated as

$$\mathcal{P}(x, \ell) = \left\{ \mathbf{p}_1(x, \ell), \cdots, \mathbf{p}_{|\mathcal{P}|}(x, \ell) \right\}.$$

Two paths are called *feasible*, denoted $\left(\mathbf{p}^{(1)}(x, \ell), \mathbf{p}^{(2)}(y, \ell)\right) \in \mathcal{F}$, if

$$\begin{aligned} &\underline{A} < y - x < \overline{A} \\ &\underline{A} < \xi^{(2)}[j] - \xi^{(1)}[j] < \overline{A} \quad , j = 1, \ldots, \ell. \end{aligned}$$

In other words, two paths are feasible if they satisfy both the dynamics constraints (1) as well as the state constraints (2) and (3). Note that by definition, the order matters. In fact

$$\left(\mathbf{p}^{(1)}, \mathbf{p}^{(2)}\right) \in \mathcal{F} \implies \left(\mathbf{p}^{(2)}, \mathbf{p}^{(1)}\right) \notin \mathcal{F}.$$

Each possible state for the vehicles has a return value, where positive return indicates an opportunity and negative return represents a hazard. Let $R(\xi)$ represent the return of state ξ. We will assume that at each time instant k, the vehicles can sense the return value of each state of its reachable set, L decisions into the future. With a slight abuse of notation, we will denote the return of a path by

$$R(\mathbf{p}(x,\ell)) = \sum_{j=0}^{\ell-1} R(\xi_{j+1}).$$

Example 1. Following [7, 8], assume that each UAV is designed with five motion primitives designated as $u_{-2}, u_{-1}, u_0, u_1, u_2$, where u_q maintains a heading angle of $q\hat{\psi}$ radians. A discrete dynamic model of the transition in lateral position as the motion primitives are followed for T seconds can be written as

$$x_n[k+1] = x_n[k] + u_n[k], \quad n = 1, \ldots, N, \tag{4}$$

where $u_n[k] \in \mathcal{U} = \{-2, -1, 0, 1, 2\}$, and the initial states $x_n[0]$ are integers. It is assumed that the longitudinal motion is described by

$$y_n[k+1] = y_n[k] + 1.$$

Assuming a look-ahead window of $L = 5$, the set of possible paths is shown in Figure 2. The blue dot is the UAV, the red dots are opportunities and the green dots are hazards. The magenta lines are the possible paths. The size of $\mathcal{P}$ is $|\mathcal{P}| = |\mathcal{U}|^L = 5^5 = 3125$. ■

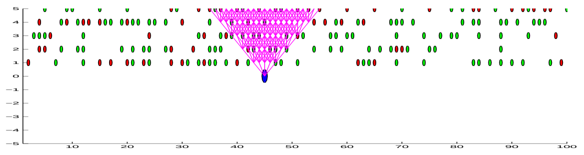

Figure 2: Look-ahead window for Example 1, and possible path set.

Example 2. As a second example, consider the cooperative search problem using a team of three UAVs flying over a field of targets and threats. Figure 3 shows an example scenario with threats shown as dots and targets depicted by diamonds. The cooperation objective for the team is to view as many targets as possible while simultaneously avoiding the threats. Each UAV has a downward looking sensor with a footprint of width w. If a UAV comes within a horizontal distance of $w/2$ of a target, the target is considered to have been viewed or sensed by the UAV. Again we

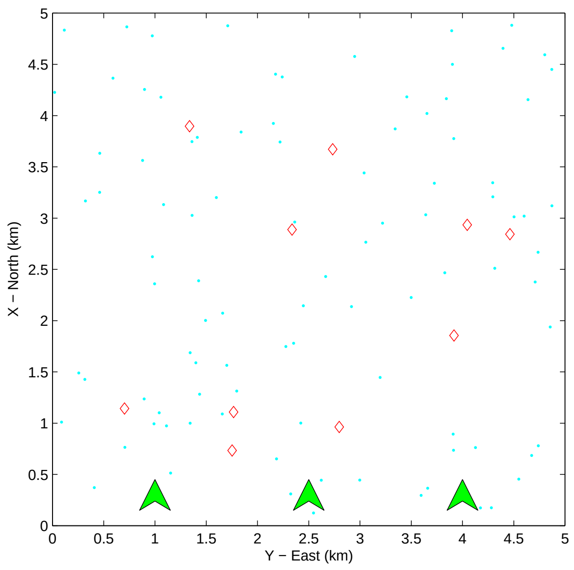

Figure 3: Example 2 problem scenario.

assume that the UAVs maintain relative longitudinal alignment, but are free to move laterally to maximize the number of targets sensed. A Voronoi graph is constructed from the threat and vehicle locations. Eppstein's k-best paths algorithm [30] is used to produce a set of candidate waypoint paths which is used as the decision set $\mathcal{U}$. Therefore $|\mathcal{U}| = k$, where k is a parameter that we can set. In this paper we use $k = 50$. In gauging the utility of a path, the primary objective is the number of targets sensed. A secondary objective is path length. In this example, if two paths result in the same number of targets sensed, the shorter of the two would be deemed the more optimal selection. The dynamics given by Eq. (1) represent the transition from the beginning of the first waypoint of $u[k]$ to the last waypoint in $u[k]$. For this example, the look-ahead window will be $L = 1$, meaning we look-ahead one path at a time. The path constraints (2) and (3) are checked by sampling the waypoint path along constant intervals in time. The set of possible paths for a single UAV are shown in Figure 4. ■

If the collision avoidance and communication constraints are ignored, then each agent could solve the myopic optimization problem represented by maximizing the value function

$$V_n(x_n[k]) = \max_{\mathbf{p} \in \mathcal{P}(x_n[k], L)} R(\mathbf{p}). \tag{5}$$

It is well known that the solution to this myopic optimization problem can be found efficiently using dynamic programming [31].

Letting $\mathbf{x}[k] = (x_1[k], \ldots, x_N[k])^T$, the team optimization problem can therefore be stated as

$$V(\mathbf{x}[k]) = \max_{\substack{\mathbf{p}^{(n)} \in \mathcal{P}^{(n)} \\ \left(\mathbf{p}^{(n+1)}, \mathbf{p}^{(n)}\right) \in \mathcal{F}}} \sum_{n=1}^{N} R(\mathbf{p}^{(n)}), \tag{6}$$

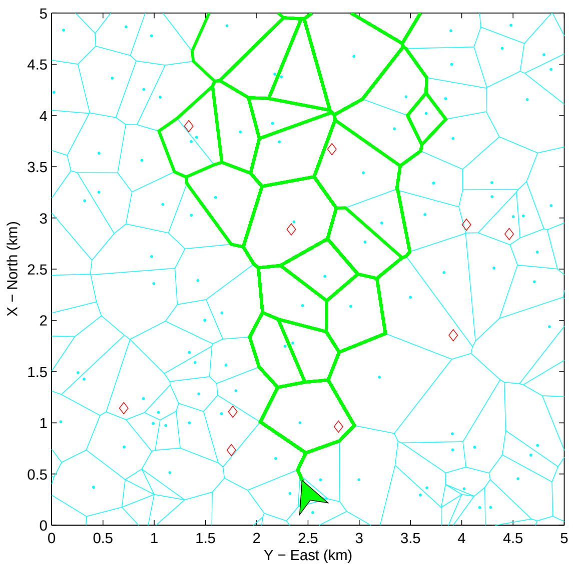

Figure 4: Possible path set for Example 2.

where $\mathcal{P}^{(n)} = \mathcal{P}(x_n[k], L)$ and $\mathbf{p}^{(n)} = \mathbf{p}(x_n[k], L)$.

3 Global Solution

This section presents a computational algorithm that computes the optimal solution to (6). Note that enumerating all possible options has computational complexity of $\bigcirc\left(|\mathcal{U}|^{LN}\right)$. We will show that the collision avoidance and communication constraints reduce to complexity of enumerating all solutions to polynomial in the number of vehicles N.

Consider the algorithm.

Algorithm 3.1 (Optimal for $N = 2$.)

Input. *$x_1[k]$, $x_2[k]$, $R(\xi)$ for each $\xi \in \mathcal{R}(x_n[k], \ell)$, $n = 1, 2$, $\ell = 1, \ldots, L$.*

Step 1. *Construct $\mathcal{P}^{(1)} = \mathcal{P}(x_1[k], L)$, $\mathcal{P}^{(2)} = \mathcal{P}(x_2[k], L)$.*

Step 2. *Compute the matrix $M = \{m_{ij}\}$ where*

$$m_{ij} = \begin{cases} R(\mathbf{p}_i^{(1)}) + R(\mathbf{p}_j^{(2)}), & \text{if } (\mathbf{p}_i^{(1)}, \mathbf{p}_j^{(2)}) \in \mathcal{F} \\ -\infty, & \text{otherwise,} \end{cases}$$

where $i = 1, \ldots, \left|\mathcal{P}^{(1)}\right|$, and $j = 1, \ldots, \left|\mathcal{P}^{(2)}\right|$.

Step 3. *Compute $(i^*, j^*) = \arg\max m_{ij}$.*

Return. $\mathbf{p}_{i^*}^{(1)}$ *and* $\mathbf{p}_{j^*}^{(2)}$.

Lemma 3.2 *If $N = 2$, Algorithm 3.1 returns the global optimal solution to problem (6). The computational complexity is $\bigcirc\left(L\,|\mathcal{U}|^{2L}\right)$.*

Proof: The sets $\mathcal{P}^{(n)}$, $n = 1, 2$ contain all possible paths for the two vehicles. Therefore M contains a finite return for each possible pair of feasible paths. Therefore Step 3 is simply a search over all feasible paths, and is therefore globally optimal.

The size of $\mathcal{P}^{(n)}$ is $|\mathcal{U}|^L$. Each path requires L computations. Therefore step 1 is $\bigcirc\left(2L\,|\mathcal{U}|^L\right)$. Each feasibility check requires L additions and $2L$ compares. This must be done for $|\mathcal{U}|^{2L}$ possible path combinations so step 2 is $\bigcirc\left(3L\,|\mathcal{U}|^{2L}\right)$. Step 3 is a search over $|\mathcal{U}|^{2L}$ elements and is therefore $\bigcirc\left(|\mathcal{U}|^{2L}\right)$. Therefore the total algorithm is

$$\bigcirc\left((3L+1)\,|\mathcal{U}|^{2L} + 2L\,|\mathcal{U}|^L\right) = \bigcirc\left(L\,|\mathcal{U}|^{2L}\right).$$

■

Each of the vehicles has a specific neighbor on its right and left. This structure can be exploited to find the global team optimal with teams of N agents, by N repeated applications of Algorithm 3.1.

Algorithm 3.3 (Optimal)

Input. $\mathbf{x}[k] = (x_1[k], x_2[k], \ldots, x_N[k])$, *$R(\xi)$ for each $\xi \in \mathcal{R}(x_n[k], \ell)$, $n = 1, \ldots, N$, $\ell = 1, \ldots, L$.*

Step 1. *Construct $\mathcal{P}^{(n)} = \mathcal{P}(x_n[k], L)$, $n = 1, \ldots, N$. Compute $\mu^{(1)} = \left(R(\mathbf{p}_1^{(1)}), \ \ldots, \ R(\mathbf{p}_{|\mathcal{P}^{(1)}|}^{(1)})\right)$.*

Step 2. *For n from 2 to N do*

2a. *Compute the matrix $M^{(n)} = \{m_{ij}^{(n)}\}$ where*

$$m_{ij}^{(n)} = \begin{cases} \mu_i^{(n-1)} + R(\mathbf{p}_j^{(n)}), & \text{if } (\mathbf{p}_i^{(n-1)}, \mathbf{p}_j^{(n)}) \in \mathcal{F} \\ -\infty, & \text{otherwise,} \end{cases}$$

where $i = 1, \ldots, \left|\mathcal{P}^{(n-1)}\right|$, and $j = 1, \ldots, \left|\mathcal{P}^{(n)}\right|$.

2b. *Compute*

$$\mu^{(n)} = \left(\max_i m_{i1}^{(n)}, \ \ldots, \ \max_i m_{i|\mathcal{P}^{(n)}|}^{(n)}\right)$$
$$I^{(n)} = \left(\arg\max_i m_{i1}^{(n)}, \ \ldots, \ \arg\max_i m_{i|\mathcal{P}^{(n)}|}^{(n)}.\right)$$

Step 3. *Compute $i^{(N)*} = \arg\max \mu^{(N)}$.*

Step 4. *For n from $N-1$ down to 1 do*

4a. $i^{(n)*} = I^{(n+1)}\left(i^{(n+1)*}\right)$.

Return. $\mathbf{p}_{i^{(n)*}}^{(n)}$, $\quad n = 1, \ldots, N$.

Lemma 3.4 *Algorithm 3.3 returns the global optimal solution to problem (6). The computational complexity is $\bigcirc\left(NL\,|\mathcal{U}|^{2L}\right)$.*

Proof: The key observation is that the paths of vehicle n are only constrained by vehicles $n-1$ and $n+1$. Therefore, path feasibility only needs to be checked between successive vehicles. The matrix $M^{(2)}$ in Step 2 is an enumeration of the return of all feasible paths of vehicles 1 and 2, without regard for the remaining $N-2$

vehicles. The i^{th} element of $\mu^{(2)}$ is the optimal return of the $(1,2)$-team given that vehicle 2 takes path $\mathbf{p}_i^{(2)}$. The i^{th} element of $I^{(2)}$ is the corresponding index of the optimal path for vehicle 1. Note that if a path for vehicle 2 is selected in the future, then $I^{(2)}$ returns the optimal path for vehicle 1.

The $(i,j)^{\text{th}}$ element of the matrix $M^{(3)}$ is the return for selecting path $\mathbf{p}_j^{(3)}$ for vehicle 3, as well as the return for selection $\mathbf{p}_i^{(2)}$ for vehicle 2 and the corresponding optimal path for vehicle 1, given that $(\mathbf{p}_j^{(3)}, \mathbf{p}_i^{(2)}) \in \mathcal{F}$. Note that only paths that are feasible for the $(1,2,3)$-team have finite values in $M^{(3)}$. Therefore the i^{th} element of $\mu^{(3)}$ is the optimal return for the $(1,2,3)$-team given that path $\mathbf{p}_i^{(3)}$ is selected for vehicles 3, and the i^{th} element of $I^{(3)}$ is the corresponding index for the optimal path for vehicle 2.

Continuing in this way we arrive at $M^{(N)}$ whose $(i,j)^{\text{th}}$ element is the return for selecting path $\mathbf{p}_j^{(N)}$ for vehicle N, as well as the return for selecting $\mathbf{p}_i^{(N-1)}$ for vehicle $N-1$, and the corresponding optimal paths for vehicles $1,\ldots,N-2$, given that $(\mathbf{p}_j^{(N)}, \mathbf{p}_i^{(N-1)}) \in \mathcal{F}$. Note that only paths that are feasible for the entire team have finite values in $M^{(N)}$. Therefore the i^{th} element of $\mu^{(N)}$ is the optimal return for the team given that path $\mathbf{p}_i^{(N)}$ is selected for vehicles N. Step 3 then picks the optimal path for vehicle N, given the effects of the group constraints (2) and (3). The optimal path indices for each vehicle can then be found using Step 4.

Following the proof of Lemma 3.2 the computation of $\mathcal{P}^{(n)}, n = 1,\ldots,N$ is $\bigcirc\left(2NL\,|\mathcal{U}|^L\right)$. Computation of $\mu^{(1)}$ is $\bigcirc\left(|\mathcal{U}|^L\right)$. The construction of $M^{(n)}, \quad n = 1,\ldots,N$ is $\bigcirc\left(3(N-1)L\,|\mathcal{U}|^{2L}\right)$. Construction of both $\mu^{(n)}$ and $I^{(n)}$ is $\bigcirc\left((N-1)\,|\mathcal{U}|^{2L}\right)$. Step 3 is $\bigcirc\left(|\mathcal{U}|^L\right)$ and step 4 is $\bigcirc(N-1)$. Therefore the total algorithm is

$$\bigcirc\Big((N-1)(3L+1)\,|\mathcal{U}|^{2L} + 2(N+1)\,|\mathcal{U}|^L + N - 1\Big) = \bigcirc\left(NL\,|\mathcal{U}|^{2L}\right).$$

■

Of course Algorithm 3.3 is only realizable for small $|\mathcal{U}|$ and small L. In Section 4 we will introduce an information reduction scheme that will be used to derive two (suboptimal) algorithms whose complexity is polynomial in N, L, and $|\mathcal{U}|$.

Example 2, (continued). Figure 5 shows results obtained using Algorithm (3.3). In this example, the sensor width was 1 km. The UAVs were required to stay within 2 km of each other to maintain communication and could come no closer than 0.2 km to avoid collision. Using the team-optimal algorithm, 10 targets were sensed by the UAVs. It should be pointed out that targets can be sensed by more than one UAV with each detection contributing to the team total. To produce these results, the 50 best paths for each UAV were considered. For three UAVs and 50 paths, execution of the algorithm required 13.3 seconds. In comparison, a brute force global search through the 50^3 possible path combinations required 522 seconds. Clearly, the need to consider feasibility only between successive vehicles reduces the computational burden significantly. ■

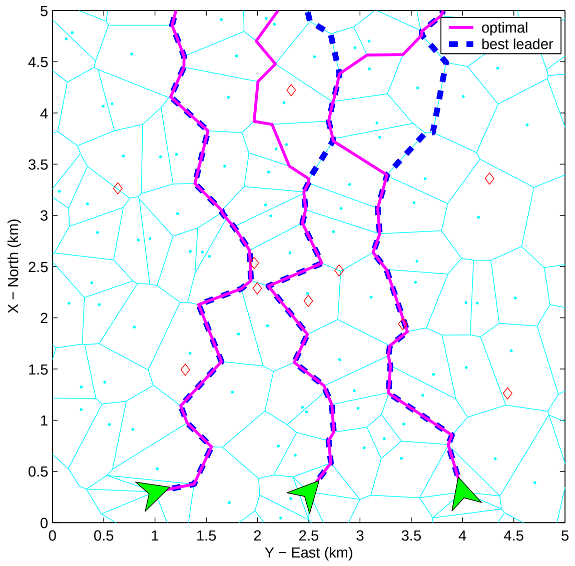

Figure 5: Example 2 problem scenario.

4 Best Path Reduction

The enumeration of feasible joint paths represented in step 2 in Algorithm 3.3 is computationally undesirable. In this section we present a (suboptimal) approach that results in feasible joint options without enumerating all possible paths.

The key observation is that the myopic optimization problem (5) can be solved efficiently using dynamic programming. Define

$$J^*(x[q], q) \overset{\triangle}{=} \max_{\substack{u[j]\in\mathcal{U} \\ j=q,\ldots,L-1}} \sum_{j=q}^{L-1} R\left(f(x[j], u[j])\right),$$

where $x[j+1] = f(x[j], u[j])$. Then, using standard dynamic programming arguments, it is straightforward to show that

$$V_n(x_n[k]) = J^*\left(x_n[k], 0\right)$$

and that J^* satisfies the recursion

$$J^*\left(x[q], q\right) = \max_{u[q]\in\mathcal{U}} \left\{R(f(x[q], u[q])) + J^*\left(f(x[q], u[q])\right)\right\},$$

with boundary constraint

$$J^*\left(x[L-1], L-1\right) = \max_{u[L-1]\in\mathcal{U}} R\left(f(x[L-1], u[L-1])\right).$$

Letting $\mathcal{C}(x)$ be the set of optimal paths from x to $\mathcal{R}(x, L)$, then $\mathcal{C}(x)$ can be found via the following algorithm.

Algorithm 4.1 (Best Myopic Paths)

Input. *$x[k]$, $R(\xi)$ for each $\xi \in \mathcal{R}(x[k], \ell)$, $\ell = 1, \ldots, L$.*

Step 1. *For ℓ from 0 to L do*

1a. *Compute $\mathcal{R}(x, \ell)$.*

Step 2. *For ℓ from $L-1$ down to 1 do*

2a. *Compute $J^*(\xi, \ell)$ at each $\xi \in \mathcal{R}(x, \ell)$ storing the associated decisions variables.*

Step 3. *Using the stored decision variables, construct the optimal paths from x to $\xi \in \mathcal{R}(x, L)$.*

Return. *$\mathcal{C}(x)$.*

Lemma 4.2 *Algorithm 4.1 returns the set of optimal paths from $x \in \mathcal{X}$ to each $\xi \in \mathcal{R}(x, L)$. The computational complexity is $\bigcirc\left(L\,|\mathcal{U}|^L\right)$. If*

$$A1: \quad \mathcal{X} \subset \mathbb{Z}, \quad \mathcal{U} \subset \mathbb{Z}, \quad \mathcal{R}(x, \ell) \subset \mathcal{R}(x, \ell+1),$$

where $\mathbb{Z}$ is the set of integers, then the computational complexity is $\bigcirc\left(L^2\,|\mathcal{U}|\right)$.

Proof: The first statement follows from standard dynamic programming arguments [31].

If no assumptions are made on $\mathcal{X}$ and $\mathcal{U}$, then each ξ generates $|\mathcal{U}|$ new states. Therefore there are $|\mathcal{U}|^L$ states in $\mathcal{R}(x, L)$, which implies that step 1 is $\bigcirc\left(|\mathcal{U}|^L\right)$. Under assumption A1, there are only $|\mathcal{U}| + (|\mathcal{U}| - 1)(\ell - 1)$ new states generated at stage ℓ, which implies that step 1 is $\bigcirc\left(L\,|\mathcal{U}|\right)$. Step 2 must be computed at each ξ in the look-ahead window. In general this requires $\bigcirc\left(\sum_{\ell=1}^{L} |\mathcal{U}|^\ell\right)$ computations. Under assumption A1, there are $\bigcirc\left(\sum_{\ell=1}^{L} \ell\right)$ computations. Step three requires L computations for each $\xi \in \mathcal{R}(x, L)$ therefore in general it is $\bigcirc\left(L\,|\mathcal{U}|^L\right)$, under assumption A1 it is $\bigcirc\left(L^2\,|\mathcal{U}|\right)$, ■

Example 1, (continued). Example 1 satisfies assumption A1. The best paths for a particular x are shown in Figure 6. ■

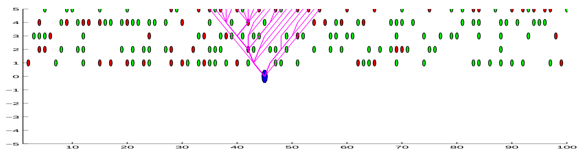

Figure 6: Best myopic paths for Example 1.

The best path from x is then

$$\mathbf{p}^*(x) = \max_{\mathbf{p} \in \mathcal{C}(x)} R(\mathbf{p}).$$

To address the team-optimization problem we need to introduce the notion of a *path constrained reachable set.* Suppose that $x \in \mathcal{X}$ and $y \in \mathcal{X}$ where $y < x$ satisfy constraints (2) and (3), and let $\mathbf{p}_y = \mathbf{p}(y, L)$ be a path from y, then the $\mathbf{p}_y$-path constrained reachable set from x is defined recursively as:

$$\begin{aligned}
&\mathcal{R}(x, 0|\mathbf{p}_y) = \{x\} \\
&\mathcal{R}(x, \ell|\mathbf{p}_y) = \{\xi \in \mathcal{X} : \xi = f(z, u),\ z \in \mathcal{R}(x, \ell - 1|\mathbf{p}_y), \\
&\qquad\qquad u \in \mathcal{U},\ \underline{A} < \xi - \xi_y[\ell] < \overline{A}\}\,.
\end{aligned}$$

Using Algorithm 4.1, with $\mathcal{R}(x[k], \ell)$ replaced by $\mathcal{R}(x[k], \ell|\mathbf{p}_y)$ we can compute the best myopic paths from $x[k]$ given the constraints imposed by $\mathbf{p}_y$.

Example 1, (continued). The best constrained paths for x are shown in Figure 7 given a particular path $\mathbf{p}_y$. ■

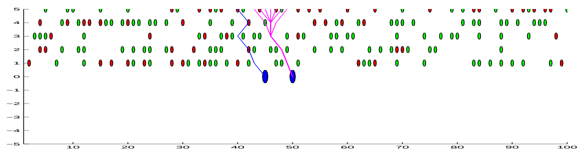

Figure 7: Best constrained myopic paths for Example 1.

Define $\mathcal{C}(x|\mathbf{p})$ be the set of constrained best paths from x to the set $\mathcal{R}(x, L|\mathbf{p})$. The key idea to the reductions schemes introduced in this paper is to use $\mathcal{C}(x|\mathbf{p})$ instead of the full enumeration of paths $\mathcal{P}(x, L)$ used in Algorithm 3.3. Since computation of $\mathcal{C}(x|\mathbf{p})$ will be used to construct our cooperative search schemes, we need the following algorithm and lemma.

Algorithm 4.3 (Best Constrained Paths)

Input. *$x[k]$, $\mathbf{p}(y, L)$, $R(\xi)$ for each $\xi \in \mathcal{R}(x[k], \ell)$, $\ell = 1, \ldots, L$.*

Step 1. *For ℓ from 0 to L do*

1a. *Compute $\mathcal{R}(x, \ell|\mathbf{p}(y, L))$.*

Step 2. *For ℓ from $L-1$ down to 1 do*

2a. *Compute $J^*(\xi, \ell)$ at each $\xi \in \mathcal{R}(x, \ell|\mathbf{p}(y, L))$ storing the associated decisions variables.*

Step 3. *Using the stored decision variables, construct the optimal paths from x to $\xi \in \mathcal{R}(x, L|\mathbf{p}(y, L))$.*

Return. *$\mathcal{C}(x|\mathbf{p}(y, L))$.*

Lemma 4.4 *Algorithm 4.3 computes the best constrained paths from x to $\mathcal{R}(x, L|\mathbf{p}(y, L))$. If assumption A1 holds then the computational complexity is $\bigcirc\left(L^2(\overline{A} - \underline{A})\right)$.*

Proof: The path constraints reduce the number of options at each stage from $|\mathcal{U}|$ to at most $\overline{A} - \underline{A}$. Summing over L stages implies the complexity of step 1 is $\bigcirc\left(L(\overline{A} - \underline{A})\right)$. The remainder of the proof follows the proof of 4.2 with $|\mathcal{U}|$ replaced by $\overline{A} - \underline{A}$. ■

5 Best Leader Cooperative Search

The most obvious cooperative search scheme using best constrained paths is a leader-following type search. For example, the first vehicle may plan its best myopic path, without consideration for the team. This path is passed to the second vehicle which plans its best path constrained to the first vehicles path. This is then repeated until the N^{th} vehicle plans its best path constrained to the path selected by vehicle $N-1$.

The selection of the first vehicle as leader was, of course, arbitrary. An alternative would be to select the second vehicle as the leader, and then to find the best paths for the first and third vehicles constrained to the second vehicle. The fourth vehicle then plans its best path constrained to the third vehicle and so on.

The following algorithm computes the team cost when each vehicle is acting as leader, and then selects the best leader.

Algorithm 5.1 (Best Leader Search)

```
Input. $\mathbf{x}[k]$, $\mathbf{p}(y,L)$, $R(\xi)$ for each $\xi \in \mathcal{R}(x_n[k],\ell)$,
    $\ell = 1,\dots,L$, $n = 1,\dots,N$.
Step 1. For $n$ from 1 to $N$ do
1a. Compute $\mathcal{C}(x_n[k])$.
1b. Determine $\mathbf{p}_n^* = \max_{\mathbf{p}\in\mathcal{C}(x_n[k])} R(\mathbf{p})$: the best myopic path for vehicle $n$.
1c. For $\hat{n}$ from $n+1$ to $N$ do
1c-i. Compute $\mathcal{C}(x_{\hat{n}}|\mathbf{p}_{\hat{n}-1})$.
1c-ii. Determine $\mathbf{p}_{\hat{n}}^* = \max_{\mathbf{p}\in\mathcal{C}(x_{\hat{n}}|\mathbf{p}_{\hat{n}-1})} R(\mathbf{p})$: the best constrained path for vehicle $\hat{n}$.
1d. For $\hat{n}$ from $n-1$ down to 1 do
1d-i. Compute $\mathcal{C}(x_{\hat{n}}|\mathbf{p}_{\hat{n}+1})$.
1d-ii. Determine $\mathbf{p}_{\hat{n}}^* = \max_{\mathbf{p}\in\mathcal{C}(x_{\hat{n}}|\mathbf{p}_{\hat{n}+1})} R(\mathbf{p})$: the best constrained path for vehicle $\hat{n}$.
1e. Compute $m_n = \sum_{\hat{n}=1}^{N} R(\mathbf{p}_{\hat{n}}^*)$.
Step 2. Determine the best leader: $n^* = \arg\max m_n$.
Return. Team paths associated with leader $n^*$.
```

Lemma 5.2 *Algorithm 5.1 computes the best leader approximation to* (6). *If assumption A1 holds, then the computational complexity is* $\bigcirc\left(N^2L^2\,|\mathcal{U}|\right)$.

Proof: From Lemmas 4.2 and 4.4, it is clear that steps 1a, 1c-i, and 1d-i, are $\bigcirc\left(L^2\,|\mathcal{U}|\right)$ each. Steps 1b, 1c-ii, and 1d-ii are searches over at most $L\,|\mathcal{U}|$ elements. Therefore step 1 is $\bigcirc\left(N\left[L^2\,|\mathcal{U}| + (N-1)L^2\,|\mathcal{U}|\right]\right) = \bigcirc\left(N^2L^2\,|\mathcal{U}|\right)$. Since step 2 is $\bigcirc(N)$, the lemma holds. ■

Example 1, (continued). Algorithm 5.1 was implemented in Example 1 for a seven UAV team. The results are shown in Figure 8. The paths below the UAVs are paths that have been flown, the paths above the UAVs are currently planned paths. As the UAVs make decisions and advance via Eq. (1) new opportunities and hazards present themselves on the horizon. Future paths are re-planned at each stage. Algorithm 8 requires approximately 1.6 seconds per stage. ■

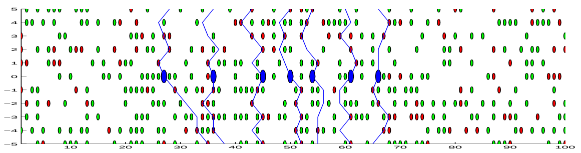

Figure 8: Best leader solution for Example 1.

Example 2, (continued). Figures 5 and 9 illustrate results obtained using the best leader cooperative search. Using this algorithm, 9 targets were sensed. Comparing the optimal result with the best leader result shown in Figure 5, the paths for UAVs 2 and 3 differ only slightly. In the optimal result, UAV 2 takes a longer (individually suboptimal) path to the left of the uppermost target. By doing so, UAV 3 is able to view this same target while satisfying the collision constraints. Using the leader-follower approach, the UAVs make myopic decisions that do not account for this coupling. Leader-follower solutions with each of

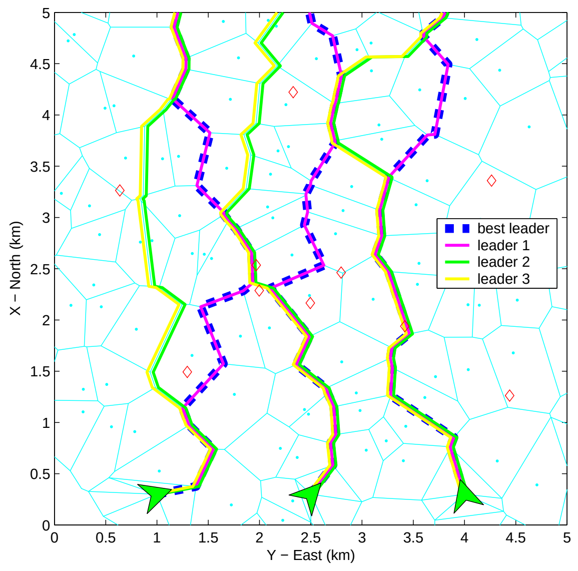

Figure 9: Example 2 leader-follower paths.

the UAVs as leader are shown in Figure 9. The best-leader solution results with UAV 1 as the leader. In this scenario, the solutions with UAV 2 and UAV 3 as leaders are identical. With UAV 2 as leader, it selfishly takes the path that yields the highest payoff. In doing so, it pushes UAV 1 to the left away from higher payoff paths that would better benefit the team.

With each UAV having 50 path options, the computational time for this algorithm is 1.2 seconds. This is a significant improvement over the optimal approach of Algorithm 3.3. Even greater advantages would be evident for greater numbers of vehicles or path options. For this particular scenario, repeated testing has shown that with randomly generated threat and target locations, the best-leader approach frequently finds the globally optimal solution. ■

6 Optimal Best Path Cooperative Search

The weakness of the best leader approach is that the team paths are based on the best myopic paths of the individuals. There may be circumstances where the team is best served having each individual choosing individually suboptimal paths. One idea is to use Algorithm 3.3, with the modification that instead of searching over all possible pairwise feasible paths in $\mathcal{P}$, we limit the search to (pairwise feasible by construction) paths in $\mathcal{C}$. In other words, instead of searching over all possible paths, we limit the search to optimal paths to the reachable set $\mathcal{R}(\cdot, L)$.

Algorithm 6.1 (Optimal Best Path Search)

```
Input. $\mathbf{x}[k]$, $R(\xi)$ for each $\xi \in \mathcal{R}(x_n[k], \ell)$, $n = 1, \ldots, N$, $\ell = 1, \ldots, L$.
Step 1. Construct $\mathcal{C}^{(1)} = \mathcal{C}(x_n[k], L)$.
    Compute $\mu^{(1)} = \left( R(\mathbf{p}_1^{(1)}), \quad \ldots, \quad R(\mathbf{p}_{|\mathcal{C}^{(1)}|}^{(1)}) \right)$.
Step 2. For $n$ from 2 to $N$ do
    2a. For $i$ from 1 to $\left|\mathcal{C}^{(n-1)}\right|$ do
        2a-i. Construct $\mathcal{C}_i^{(n)} \triangleq \mathcal{C}(x_n[k], L | \mathbf{p}_i^{(n-1)})$.
        2a-ii. For $j$ from 1 to $\left|\mathcal{C}_i^{(n)}\right|$ do
            2a-ii(1). Compute
                $m_{ij}^{(n)} = \begin{cases} \mu_i^{(n-1)} + R(\mathbf{p}_{j|i}^{(n)}), & \text{if } (\mathbf{p}_i^{(n-1)}, \mathbf{p}_{j|i}^{(n)}) \in \mathcal{F} \\ -\infty, & \text{otherwise}, \end{cases}$
                where $\mathbf{p}_{j|i}^{(n)}$ is the $j^{th}$ element of $\mathcal{C}_i^{(n)}$.
    2b. Compute
        $\mu^{(n)} = \left( \max_i m_{i1}^{(n)}, \quad \ldots, \quad \max_i m_{i|\mathcal{C}^{(n)}|}^{(n)} \right)$
        $I^{(n)} = \left( \arg\max_i m_{i1}^{(n)}, \quad \ldots, \quad \arg\max_i m_{i|\mathcal{C}^{(n)}|}^{(n)} \right)$
    2c. Let $\mathcal{C}^{(n)}$ be the set of paths for vehicle $n$ that correspond to the indices stored in $I^{(n)}$.
Step 3. Compute $i^{(N)*} = \arg\max \mu^{(N)}$.
Step 4. For $n$ from $N-1$ down to 1 do
    4a. $i^{(n)*} = I^{(n+1)}\left(i^{(n+1)*}\right)$.
Return. $\mathbf{p}_{i^{(n)*}}^{(n)}$,    $n = 1, \ldots, N$.
```

Lemma 6.2 *Let $J_{optimal}$ be the team return computed by Algorithm 3.3, $J_{best\text{-}leader}$ be the team return computed by Algorithm 5.1, and $J_{best\text{-}paths}$ be the team return computed by Algorithm 6.1, then*

$$J_{best\text{-}leader} \leq J_{best\text{-}paths} \leq J_{optimal}.$$

The computational complexity of Algorithm 6.1 is $\bigcirc\left(NL^2 \left|\mathcal{U}\right|^2\right)$.

Proof: Similar to the proofs of the other lemmas. ■

Example 1, (continued). Algorithm 6.1 was implemented in Example 1 for a seven UAV team. The results are shown in Figure 10. The paths below the UAVs are paths that have been flown, the paths above the UAVs are currently planned paths. As the UAVs make decisions and advance via Eq. (1) new opportunities and hazards present themselves on the horizon. Future paths are re-planned at each stage. Algorithm 10 requires approximately 4.6 seconds per stage. ■

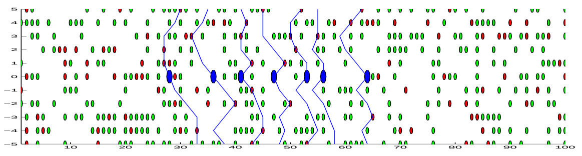

Figure 10: Optimal best paths solution for Example 1.

7 Conclusions

The problem of cooperative search by a team of UAVs with collision-avoidance and communication-range constraints has been considered. An algorithm for finding team-optimal paths by considering feasible paths for neighboring UAVs was developed. Two suboptimal, but computationally efficient approaches were developed: the best leader and optimal best path cooperative search algorithms. These algorithms were tested on two example cooperative search problems. Depending on the characteristics of a search problem (such as the number of vehicles, the number of stages, and the number of possible paths considered) and the computational resources available, these algorithms provide a spectrum of solutions with potential for real-time implementation.

Acknowledgments

This work was funded by AFOSR grants F49620-01-1-0091 and F49620-02-C-0094, and by DARPA grant NBCH1020013.

References

[1] R. W. Beard, T. W. McLain, M. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 911–922, December 2002.

[2] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative timing missions," *AIAA Journal of Guidance, Control and Dynamics*, (in review).

[3] E. P. Anderson and R. W. Beard, "An algorithmic implementation of constrained extremal control for UAVs," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, (Monterey, CA), August 2002. AIAA Paper No. 2002-4470.

[4] E. P. Anderson, R. W. Beard, and T. W. McLain, "Real time dynamic trajectory smoothing for uninhabited aerial vehicles," *IEEE Transactions on Control Systems Technology*, (in review).

[5] W. Ren and R. W. Beard, "CLF-based tracking control for UAV kinematic models with saturation constraints," in *Proceedings of the IEEE Conference on Decision and Control*, 2003. Submitted.

[6] R. C. Nelson, *Flight Stability and Automatic Control*. Boston, Massachusetts: McGraw-Hill, 2nd ed., 1998.

[7] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *Proceedings of the American Control Conference*, (Arlington VA), pp. 43–49, June 2001.

[8] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 25, pp. 116–129, January-February 2002.

[9] N. Faiz, S. K. Agrawal, and R. M. Murray, "Trajectory planning of differentially flat systems with dynamics and inequalities," *AIAA Journal of Guidance, Control and Dynamics*, vol. 24, pp. 219–227, March–April 2001.

[10] M. B. Milam, K. Mushambi, and R. M. Murray, "A computational approach to real-time trajectory generation for constrained mechanical systems," in *IEEE Conference on Decision and Control*, (Sydney, Australia), pp. 845–851, 2000.

[11] R. K. Prasanth, J. D. Boskovic, S.-M. Li, and R. K. Mehra, "Initial study of autonomous trajectory generation for unmanned aerial vehicles," in *Proceedings of the IEEE Conference on Decision and Control*, (Orlando, FL), pp. 640–645, December 2001.

[12] K. B. Judd and T. W. McLain, "Spline based path planning for unmanned air vehicles," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, (Montreal, CA), pp. Paper no. AIAA–2001–4238, AIAA, August 2001.

[13] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[14] P. Chandler, S. Rasumussen, and M. Pachter, "UAV cooperative path planning," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, (Denver, CO), August 2000. AIAA Paper No. AIAA-2000-4370.

[15] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Proceedings of the IEEE Conference on Decision and Control*, (Las Vegas, NV), pp. 1301–1306, 2002.

[16] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specific trajectories," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Washington DC), pp. 624–631, May 2002.

[17] S. Leroy, J. P. Laumond, and T. Simeon, "Multiple path coordination for mobile robots: A geometric algorithms," in *International Joint Conference on Artificial Intelligence*, (Stockholm, Sweden), pp. 1118–1123, 1999.

[18] Z. J. Butler, A. A. Rizzi, and R. L. Hollis, "Cooperative coverage of rectilinear environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (San Francisco, CA), pp. 2722–2727, April 2000.

[19] C. Luo, S. X. Yang, D. A. Stacey, and J. C. Jofriet, "A solution to vicinity problem of obstacles in complete coverage path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Washington DC), pp. 612–617, May 2002.

[20] F. Giulietti, L. Pollini, and M. Innocenti, "Autonomous formation flight," *IEEE Control Systems Magazine*, vol. 20, pp. 34–44, December 2000.

[21] M. Pachter, J. J. D'Azzo, and A. W. Proud, "Tight formation flight control," *AIAA Journal of Guidance, Control and Dynamics*, vol. 24, pp. 246–254, March–April 2001.

[22] C. Schumacher and S. N. Singh, "Nonlinear control of multiple UAVs in close-coupled formation flight," in *AIAA Guidance, Navigation, and Control Conference*, (Denver, CO), American Institute of Aeronautics and Astronautics, August 2000. Paper no. AIAA 2000-4373.

[23] S. T. Pledgie, Y. Hao, A. M. Ferreira, S. K. Agrawal, and R. Murphey, "Groups of unmanned vehicles: Differential flatness, trajectory planning, and control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Washington DC), pp. 3461–3466, May 2002.

[24] T. McLain and R. Beard, "Cooperative rendezvous of multiple unmanned air vehicles," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, (Denver, CO), August 2000. Paper no. AIAA-2000-4369.

[25] T. W. McLain, P. R. Chandler, S. Rasmussen, and M. Pachter, "Cooperative control of UAV rendezvous," in *Proceedings of the American Control Conference*, (Arlington, VA), pp. 2309–2314, June 2001.

[26] R. W. Beard, T. W. McLain, and M. Goodrich, "Coordinated target assignment and intercept for unmanned air vehicles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Washington DC), pp. 2581–2586, May 2002.

[27] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating UAVs," in *Cooperative Control: Models, Applications and Algorithms*, pp. 1–19, Conference on Coordination, Control and Optimization, November 2001.

[28] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and control of UAVs," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, (Monterey, CA), pp. AIAA–2002–4588, August 2002.

[29] P. R. Chandler, M. Pachter, D. Swaroop, J. M. Fowler, J. K. Howlett, S. Rasmussen, C. Schumacher, and K. Nygard, "Complexity in UAV cooperative control," in *Proceedings of the American Control Conference*, (Anchorage, AK), May 2002.

[30] D. Eppstein, "Finding the k shortest paths," *SIAM Journal of Computing*, vol. 28, no. 2, pp. 652–673, 1999.

[31] F. L. Lewis, *Optimal Control*. New York: John Wiley and Sons, 1986.