Coordinated Target Assignment and Intercept for Unmanned Air Vehicles

Randal W. Beard, *Member, IEEE*, Timothy W. McLain, *Member, IEEE*, Michael A. Goodrich, *Member, IEEE*, and Erik P. Anderson

***Abstract*—This paper presents an end-to-end solution to the cooperative control problem represented by the scenario where M unmanned air vehicles (UAVs) are assigned to transition through N known target locations in the presence of dynamic threats. The problem is decomposed into the subproblems of: 1) cooperative target assignment; 2) coordinated UAV intercept; 3) path planning; 4) feasible trajectory generation; and 5) asymptotic trajectory following. The design technique is based on a hierarchical approach to coordinated control. Simulation results are presented to demonstrate the effectiveness of the approach.**

***Index Terms*—Cooperation, multiagent coordination, path planning, trajectory generation, unmanned air vehicles.**

I. Introduction and Problem Statement

Consider a scenario where a group of M unmanned air vehicles (UAVs) are required to transition through N known target locations. In the region of interest, there are a number of threats, some known *a priori*, others "pop up," or become known only when a UAV maneuvers into their proximity. Suppose that to maximize the probability that the mission will succeed, it is desirable to have multiple UAVs arrive on the boundary of each target's radar detection region simultaneously.

The scenario can be decomposed into several subproblems.

1) Given M UAVs with N targets, assign each vehicle to a target such that each target has, if possible, multiple UAVs assigned to it, and such that preference is given to high-priority targets.
2) For each team of UAVs assigned to a single target, determine an estimated time over target (TOT) that ensures simultaneous intercept and that is feasible for each UAV on the team.
3) For each UAV with velocity constraints $v \in [V_{\min}, V_{\max}]$, determine a path (specified via waypoints), such that if the UAV were to fly along straight-line paths, it could complete the path in the specified TOT, while satisfying the given velocity constraints.
4) Transform each waypoint path into a feasible trajectory for the UAV. By feasible, we mean that the turning rate constraints and velocity constraints are not violated along the trajectory. Also, the trajectory should have the same TOT as the path specified by waypoints.
5) Develop globally asymptotically stable controllers for each UAV, such that the UAVs track their specified trajectory.

In this paper, we propose an approach to this problem, and demonstrate the effectiveness of our solution via a simulation study. Our solution is derived through a systematic hierarchical approach to multiple vehicle systems. We will assume that individual UAVs fly at different, preassigned altitudes, thereby ensuring collision avoidance.

Many of the subproblems outlined above have been previously addressed in the literature. The assignment problem, in particular, is a well-known optimization problem. In its simplest form, it is to assign items (jobs, missiles) to other items (machines, targets) [1]. Such problems frequently arise, especially in multiagent settings such as task-decomposition problems [2] and auction-based task assignment [3]. Although there are algorithms for solving the assignment problem subject to a performance criterion, the problem is NP-hard; thus, heuristic techniques are often used [4], [5].

In our treatment of the target assignment problem, we use satisficing decision theory [6] to balance the desires of individuals with the needs of the team. This theory, which has been applied in control settings [7], [8] as well as in multiagent interactions [9], has been applied to formation-initialization problems wherein agents (robots, satellites) are assigned to a location in the formation [10]. Since choosing which agents take which places in a formation is analogous to assigning targets, there is crossover from the formation problem to the one treated here. The principal difference is that in this paper, multiple UAVs can be assigned to a single target, and some targets can be unassigned. Our approach is to constrain the possible team assignment to the set of individually satisficing assignments, and then select the team assignment that maximizes group utility.

The simultaneous rendezvous problem has been addressed in [11] and [12]. A similar approach for simultaneous target intercept will be used in this paper.

Manuscript received November 20, 2001; revised May 29, 2002. This paper was recommended for publication by Associate Editor J. Leonard and Editor A. De Luca upon evaluation of the reviewers' comments. This work was supported in part by the Air Force Office of Scientific Research (AFOSR) under Award F49620-01-1-0091. This paper was presented in part at the IEEE International Conference on Robotics and Automation, Washington, DC, May 2002.

R. W. Beard is with the Electrical and Computer Engineering Department, Brigham Young University, Provo, UT 84602 USA (e-mail: beard@ee.byu.edu).

T. W. McLain is with the Mechanical Engineering Department, Brigham Young University, Provo, UT 84602 USA (e-mail: tmclain@et.byu.edu).

M. A. Goodrich is with the Computer Science Department, Brigham Young University, Provo, UT 84602 USA (e-mail: mike@cs.byu.edu).

E. P. Anderson was with the Electrical and Computer Engineering Department, Brigham Young University, Provo, UT 84602 USA. He is now with the Electrical Engineering Department, Stanford University, Palo Alto, CA 94305 USA (e-mail: eandersn@stanford.edu).

Digital Object Identifier 10.1109/TRA.2002.805653

The problem of planning waypoint paths through a cluttered environment has been addressed in several works. Reference [13] addresses the problem of moving a group of robots in a rigid formation, while minimizing the total distance traveled by all robots combined. To solve the problem, a grid is placed on the region, and a discrete dynamic programming problem is solved to find the shortest paths. The computational complexity is strongly dependent on the discretization, but only weakly dependent on the number of robots. The path planner developed in this paper uses a modified Voronoi diagram [14] to generate possible paths to the targets. The Voronoi diagram is then searched via Eppstein's k-best paths algorithm [15]. Path planners similar to the one described in this paper have been previously reported in [11] and [16].

Real time, feasible trajectory generation has also been the subject of numerous studies in the literature. The UAV dynamics that will be used in this paper can be shown to be differentially flat. An excellent introduction to differential flatness and its application to real-time trajectory generation is given in [17]. In [18], the notion of differential flatness is used to produce real-time trajectories for differentially driven mobile robots pulling a tractor.

Another common approach to trajectory generation is to represent the trajectory in terms of splines. In [19], B-splines characterize the desired path for the flat output of a nonlinear system subject to state and input constraints. In [20], splines are created that do not necessarily pass through the data points. The splines are created by casting the trajectory generation problem as an optimal control problem where the target set is the data points. In [21], a near-real-time algorithm for generating feasible trajectories for differentially flat systems is described. The approach is to first embed the state equations in the flat space. Second, the state and input constraints are converted to the flat space. Third, a semi-infinite optimization problem is solved to approximate the possibly nonlinear and nonconvex constraints using polytopes. After parameterizing the trajectory using basis functions, a collocation method converts the trajectory into a set of points that must lie within the constraint polytope. Finally, an optimization step finds the trajectory coefficients to satisfy the constraints.

Another approach to trajectory generation for UAVs is outlined in [16]. The basic idea is to plan a polygonal path through a set of threats using a Voronoi algorithm in connection with an A^* or Dijkstra algorithm. The polygonal paths are then made flyable by inserting fillets at the corners in the path. The algorithm works well when the turning radius is small compared to the path links. A related approach is reported in [22], which investigates the time-optimal trajectory generation problem for motor-driven robots, assuming that the trajectory consists of a straight-line section, followed by a constrained constant velocity arc, followed by a translational section. Given the current and velocity constraints of the robot motors, a time-optimal trajectory is generated.

The remainder of the paper is organized as follows. In Section II, we introduce the case study that will be used throughout the paper to motivate our approach. Section III outlines the proposed system architecture and describes our approach to target assignment, simultaneous strike, path planning, and trajectory generation. In Section IV, we describe a Matlab/Simulink simulation of several example scenarios. Finally, Section V offers some conclusions.

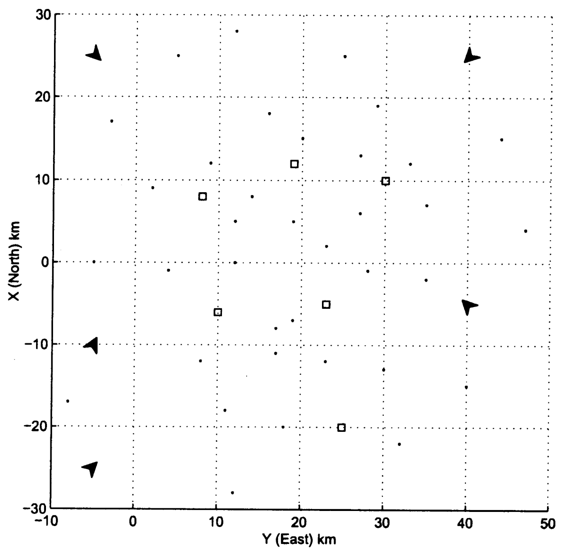


Fig. 1. Configuration for case study.

II. Case Study

We will illustrate our approach in a specific case study, which we introduce here to facilitate the discussion. Fig. 1 shows the region under consideration. The dots represent threats to be avoided. The squares represent targets that need to be prosecuted. The UAVs are represented by solid arrowheads, which are greatly exaggerated in size relative to the UAV. The mission objective is to visit all the targets, while minimizing the risk to each individual UAV. Risk is mitigated by maximizing the distance to threats and by simultaneously prosecuting targets with multiple UAVs to enhance the element of surprise. In the scenario depicted in Fig. 1, five UAVs are to visit six targets in the presence of 36 threats.

III. System Architecture

A detailed schematic of the system architecture for a single UAV is shown in Fig. 2. At the lowest level of the architecture is the physical UAV. In this paper, we assume that each UAV is equipped with a trajectory tracking controller.

In Fig. 2, the target manager, path planning, and intercept manager work together to generate waypoint paths for the UAV. The path planner generates a specified number of paths from the specified UAV to a specific target. The path planner returns information about each path, namely, the estimated fuel expenditure and the estimated threat exposure. Both the target manager and the intercept manager make calls to the path planner. The role of the target manager is to assign each UAV a target. The role of the intercept manager is to ensure that when the target manager assigns multiple UAVs to the same target, they arrive on the radar detection boundary of the target simultaneously. The design of the path planner, target manager, and intercept manager are discussed in Sections III-A–C, respectively.

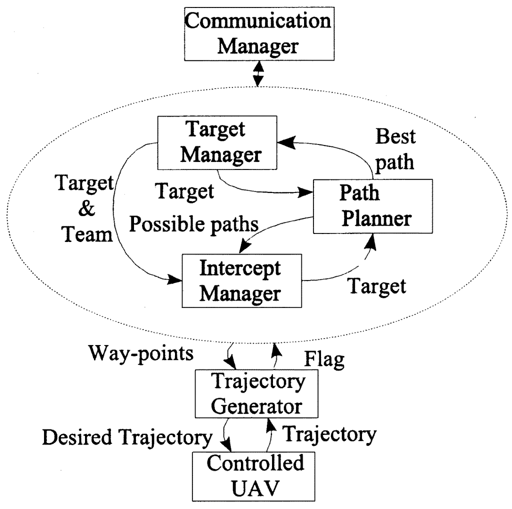


Fig. 2. System architecture for a single UAV.

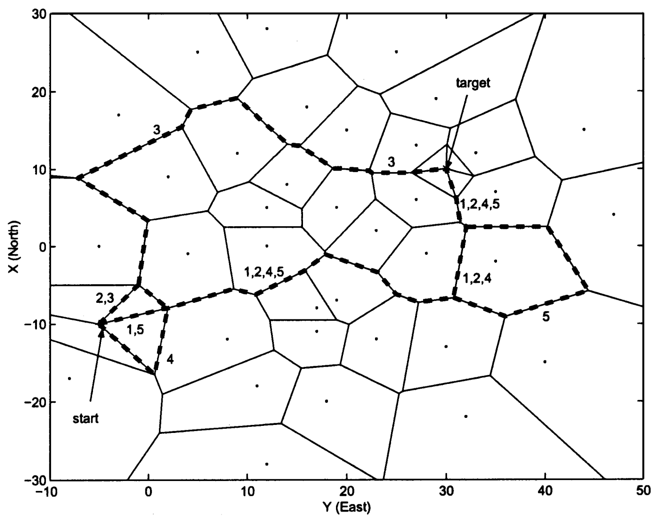


Fig. 3. Threat-based Voronoi diagram. The threats are represented by dots.

The trajectory generation block in Fig. 2 receives a set of waypoints, which specify the desired path of the UAV. The waypoints the trajectory generator receives are not parameterized in time, they are simply a set of X, Y, and h coordinates. The role of the trajectory generator is to generate a feasible time trajectory for the UAV. By feasible, we mean that in the absence of disturbances and modeling errors, an input trajectory causes the UAV to fly the trajectory without violating its velocity and heading rate constraints. The trajectory generator outputs a flag that indicates when a new waypoint path is needed. Several events may necessitate a new path. First, the UAV may successfully complete its path. Second, a pop-up threat may be detected in the environment. Third, because of disturbances, the tracking error may become unacceptably large. The design of the trajectory generator is discussed in Section III-D.

The communication manager shown in Fig. 2 facilitates communication between different UAVs. Each UAV implements a separate target manager, intercept manager, and path planner. Therefore, the decisions reached by these functional blocks must be synchronized among the different UAVs. The primary role of the communication manager is to ensure synchronization. The design of communication managers for multiple cooperating autonomous vehicles has been discussed in [23] for UAVs, and [24] for autonomous underwater vehicles.

A. Path Planner

The path planner shown in Fig. 2 is used by both the target manager and the intercept manager. The output of the path planner is a set of waypoints and commanded velocity for each vehicle. Paths from the initial vehicle location to the target location are derived from a k-best paths graph search [15] of a Voronoi diagram that is constructed from the known threat locations [25]. Creating the Voronoi diagram entails partitioning the region of interest with n threats into n convex polygons or cells. Each cell contains exactly one threat, and every location within a given cell is closer to its associated threat than to any other threat. By using threat locations to create the diagram, the resulting Voronoi polygon edges form a set of lines that are equidistant from the closest threats, thereby maximizing their distance from the closest threats. The initial and target locations are also contained within cells and are connected to the nodes forming the cell. Fig. 3 shows a Voronoi diagram created for a set of threats, UAV location, and target location.

Each edge of the Voronoi diagram is assigned two costs: a threat cost and a length cost. Threat costs are based on a UAV's exposure to a radar located at the threat. Assuming that the UAV radar signature is uniform in all directions and is proportional to $1/d^4$ (where d is the distance from the UAV to the threat), the threat cost for traveling along an edge is inversely proportional to the distance (to the threat) to the fourth power. An exact threat cost calculation would involve the integration of the cost along each edge. A computationally more efficient and acceptably accurate approximation to the exact solution is to calculate the threat cost at several locations along an edge and take the length of the edge into account. In this work, the threat cost was calculated at three points along each edge: $L_i/6$, $L_i/2$, and $5L_i/6$, where L_i is the length of edge i. The threat cost associated with the ith edge is given by the expression

$$J_{\text{threat},i} = \frac{\alpha L_i}{3} \sum_{j=1}^{N} \left(\frac{1}{d^4_{1/6,i,j}} + \frac{1}{d^4_{1/2,i,j}} + \frac{1}{d^4_{5/6,i,j}} \right) \quad (1)$$

where N is the total number of threats, $d_{1/2,i,j}$ is the distance from the 1/2 point on the ith edge to the jth threat, and α is a constant scale factor. Using this three-term approximation, errors were typically less than two percent for threats most closely situated to an edge, and on the order of tenths of a percent for threats located at a distance.

To include the path length as part of the cost, the length cost associated with each edge is

$$J_{\text{length},i} = L_i. \quad (2)$$

The total cost for traveling along an edge comes from a weighted sum of the threat and length costs

$$J_i = \kappa J_{\text{length},i} + (1-\kappa) J_{\text{threat},i}. \quad (3)$$

The choice of κ between 0 and 1 gives the designer flexibility to place weight on exposure to threats or fuel expenditure depending on the particular mission scenario. A κ value closer to 1 would result in the shortest paths, with little regard for the exposure to radar, while a value closer to 0 would result in paths that avoid threat exposure at the expense of longer path length. For this work, a value of 0.25 was found to produce paths that were balanced in terms of threat avoidance and path length.

With the cost determined for each of the Voronoi edges, the Voronoi diagram is searched to find the set of lowest-cost candidate paths between the initial UAV location and the location of the target. The graph search is carried out using a variation of Eppstein's k-best paths algorithm [15]. For the results presented here, ten candidate paths are calculated for each vehicle. In Fig. 3, the five best paths to the target are shown.

Taken in different combinations, the edges of the Voronoi diagram provide a rich set of paths from the starting point to the target. A great advantage of the Voronoi diagram approach is that it reduces the path-planning problem from an infinite-dimensional search, to a finite-dimensional graph search. This important abstraction makes the path-planning problem feasible in near-real time. Because the resulting Voronoi-based paths inherently avoid threats, the significant reduction in the solution space that occurs tends to eliminate paths that are undesirable from a threat-avoidance perspective.

B. Target Manager

Assume that there are N targets, M UAVs, and P threats. The task is to assign each vehicle to a target such that the overall group cost is mitigated, while maximizing the number of targets that are destroyed.

For this type of decision problem, the standard approach is to create an objective function that encodes the desired objectives of the decision problem. An optimal solution is one that maximizes this objective function. We can find the expected complexity for finding an optimal solution to the target assignment problem. For M agents with N targets, we must search through N^M possible assignments. Although for the world diagrammed in Fig. 1, where $M = 5$ and $N = 6$, an exhaustive search is possible; in worlds with a large number of vehicles and targets, an exhaustive search is not possible.

1) Individually Satisficing: In assigning targets to UAVs, there are four objectives.

1) Minimize the group path length to the target.
2) Minimize group threat exposure.
3) Maximize the number of vehicles prosecuting each target (to maximize survivability).
4) Maximize the number of targets visited.

We refer to these objectives as the *ShortPath*, *AvoidThreats*, *MaxForce*, and *MaxSpread* objectives, respectively.

The *ShortPath* and *AvoidThreats* objectives are myopic objectives, whereas, the *MaxForce* and *MaxSpread* objectives are team objectives. The proposed solution methodology is based on the satisficing paradigm [6], used to select a set of potential targets that are acceptable for each UAV. The set of selected targets is then used by each agent to negotiate an acceptable group assignment.

The first step is to use the path planner described in Section III-A to generate k-best paths to each target. Associated with these k-best paths is a median length cost, denoted as $\bar{J}_{\text{length}}(V_i, T_j)$, and a median threat exposure cost, denoted as $\bar{J}_{\text{threat}}(V_i, T_j)$, where V_i is the ith UAV and T_j is the jth target. The median operator throws away targets that have only one good path but keeps targets that have at least $k/2$ reasonable paths.

For an individual UAV, close targets are deemed *acceptable*, while targets with large threat exposure are deemed *rejectable*. Normalized measures of acceptability and rejectability can be defined as

$$\mu_{A_i}(T_j) = \frac{\max_{T_k}\left(\bar{J}_{\text{length}}(V_i,T_k)\right) - \left(\bar{J}_{\text{length}}(V_i,T_j)\right)}{\max_{T_k}\left(\bar{J}_{\text{length}}(V_i,T_k)\right) - \min_{T_k}\left(\bar{J}_{\text{length}}(V_i,T_k)\right)} \tag{4}$$

$$\mu_{R_i}(T_j) = \frac{\bar{J}_{\text{threat}}(V_i,T_k) - \min_{T_k}\left(\bar{J}_{\text{threat}}(V_i,T_j)\right)}{\max_{T_k}\left(\bar{J}_{\text{threat}}(V_i,T_k)\right) - \min_{T_k}\left(\bar{J}_{\text{threat}}(V_i,T_k)\right)}. \tag{5}$$

The acceptability function μ_A assigns the closest target the highest measure of acceptability (equal to one), the most distant target the lowest measure of acceptability (equal to zero), and all other targets some assignment in the range $[0, 1]$. The rejectability function μ_R assigns the target that requires the most threat exposure the highest measure of rejectability (equal to one), the target that requires the least threat exposure the lowest measure of rejectability (equal to zero), and all other targets some assignment in the range $[0, 1]$. Notice that both of these functions can be computed in order kNM time.

For each vehicle, we can identify the set of assignments that are individually satisficing. This set is given by

$$S_{V_i} = \{T_j \colon \mu_{A_i}(T_j) \geq b_i \mu_{R_i}(T_j)\} \tag{6}$$

where b_i is a selectivity index used to ensure that S_{V_i} is nonempty. Whenever the *ShortPath* objective has a more significant contribution than the *AvoidThreats* objective, the corresponding assignment is individually satisficing. In other words, a vehicle balances the desire to take short paths with the desire to avoid threats; the path to a target (relative to the path length to the closest target) must be short enough to justify the risks that arise by coming into proximity with threats (relative to the minimum threat proximity). Since each vehicle must accept an assignment, these heuristics are encoded relative to the best and worst cases, thereby causing actions to be evaluated with respect to the existing situation rather than with respect to an artificial standard. The value of b_i can be adjusted to ensure that S_{V_i} is nonempty, or to reduce the size of S_{V_i}.

We are now in a position to explore the time complexity of searching through the set of all individually satisficing options. Let $|S_{V_i}|$ denote the cardinality of the set S_{V_i}. The time complexity to find the optimal social welfare function over the set of satisficing options is given by $\prod_{i=1}^{M} |S_{V_i}|$, which, in general, is much less than N^M, since $|S_{V_i}| < N$.

To estimate how much complexity reduction is achieved, we ran 2000 trials with randomly selected positions for ten targets and six vehicles under various threat locations. In a simulated 100–km by 100–km world, targets and threats were randomly assigned locations within the world. The vehicles were assigned locations with uniform probability to a 20 km border around the world. For the optimal solution, $N^M = 1\,000\,000$ possible assignments must be examined. By contrast, when we restrict the search to the set of satisficing assignments, the median size of the search space over 2000 trials was 2400, the mean was 4437, the standard deviation of these complexities was 7214, and the maximum size was 122 472. This indicates that by far the vast majority of circumstances produce complexities that are several orders of magnitude less than the optimal assignment problem. However, there is a skew that causes some complexities to be only moderately better. However, only 9.3% of circumstances produce complexities greater than 10 000, most of these large search spaces occurring under conditions where more heuristics could be applied to further narrow the search. Although both approaches grow exponentially in the number of agents, the heuristic approach grows much more slowly than the optimal. For example, for the experiments conducted herein, the optimal approach must search through a search space 417 times larger than the search space of the heuristic approach.

2) Team Objectives: Two competing objectives affect how team behavior is evaluated: the *MaxForce* heuristic drives the team toward assigning every vehicle to the same target, but the *MaxSpread* heuristic drives the team toward visiting as many targets as possible. We will address these objectives in turn.

From the *MaxForce* perspective, the more vehicles assigned to a target, the better. We can represent this objective by assigning a value to the team size for each assigned target. In other words, we can identify a number that encodes how much better a larger team is than a smaller team. We use a monotonically increasing function that increases dramatically between one and three team members, indicating that a minimally acceptable team consists of two members. Let $\mathcal{G} = \{G_{j_1}, G_{j_2}, \ldots, G_{j_N}\}$ denote a set of targets assigned to each vehicle; G_{j_i} is the target assigned to vehicle V_i. The monotonic function that encodes the value of *MaxForce* is the sigmoid

$$U_{\text{MaxForce}}(G_j) = \frac{1}{1 + e^{-3(m(G_j)-2)}}$$

where $m(G_j) = |\{V_i : G_j = G_{j_i}\}|$ is the number of vehicles assigned to target G_j. We can use the value of the team size, encoded in $U_{\text{MaxForce}}(G_j)$, to estimate the value of an assignment. Let $\Gamma(\mathcal{G}) = \{G_j : \exists G_{j_i} \in \mathcal{G}$ for which $G_j = G_{j_i}\}$; $\Gamma(\mathcal{G})$ is the set of all targets that have vehicles assigned to them in the assignment $\mathcal{G}$. The value of this assignment from the perspective of the *MaxForce* heuristic is given by $V_{\text{MaxForce}}(\mathcal{G}) = \prod_{G_j \in \Gamma(\mathcal{G})} U_{\text{MaxForce}}(G_j)$. This value takes the product of each team-size value over the set of assigned target (the set of targets that have at least one vehicle prosecuting it). Using the product lets targets with small team sizes dramatically reduce the value of the overall assignment.

Turning attention to the *MaxSpread* heuristic, we note that the number of targets in the set $\Gamma(\mathcal{G})$ represents the spread of the assignment: $V_{\text{MaxSpread}}(\mathcal{G}) = |\Gamma(\mathcal{G})|$.

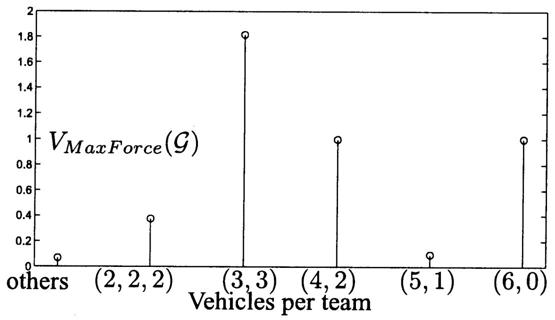


Fig. 4. Value of different team assignments. Numbers in parenthesis indicates the number of vehicles assigned to each target; e.g., (5,1) indicates that five vehicles are assigned to one target and one vehicle to another, and (2,2,2) indicates that three targets are assigned to three, two-vehicle teams. Any team configuration other than the ones shown [e.g., (4,1,1)], have values less than the value of the (5,1) assignment.

Taking the product of V_{MaxForce} and $V_{\text{MaxSpread}}$ yields the value of the assignment

$$V'(\mathcal{G}) = |\Gamma(\mathcal{G})| \times \prod_{G_j \in \Gamma(\mathcal{G})} U_{\text{MaxForce}}(G_j). \tag{7}$$

Multiplying by the number of targets assigned allows the preference for large team sizes (i.e., *MaxForce*) to be tempered by the preference for having multiple targets visited (i.e., *MaxSpread*). Since it is unlikely that all agents can feasibly visit the same target (it is unlikely that such an assignment is individually satisficing), the product prevents undesirable assignments from receiving high preference. Thus, multiple, nearly equal-sized teams are preferred to multiple, differently sized teams. $V_{\text{MaxForce}}(\mathcal{G})$ for various team assignments is shown in Fig. 4.

Note that a) $V'(\Gamma(\mathcal{G}))$ addresses the team centered need to assign a sufficient number of vehicles to each target while spreading the vehicles among multiple targets, and b) S_{v_i} addresses the individual centered need to restrict attention to safe and efficient paths for each vehicle. However, $V'(\Gamma(\mathcal{G}))$ does not explicitly account for the difficulties encountered by individual vehicles in fulfilling their assignment. Consequently, maximizing $V'(\Gamma(\mathcal{G}))$ may arbitrarily select a team assignment that is acceptable from a team perspective but difficult from an individual perspective over a nearly equivalent assignment that is also acceptable from a team perspective but much easier from an individual perspective. Fortunately, we can discriminate between these two assignments by letting the agent with the most difficult relative assignment act as a tie breaker when choosing which target a team of a given size pursues. This is done by updating the value of the assignment as follows:

$$V(\mathcal{G}) = V'(\Gamma(\mathcal{G})) + \epsilon \min_i \left[\mu_{A_i}(G_{j_i}) - \mu_{R_i}(G_{j_i})\right]$$

where the minimization is over the set of satisficing options. The multiplicative factor ϵ guarantees that team size dominates the utility of an assignment, but that the vehicle that experiences the longest path or most exposure has some influence over which target is assigned to its team. The assignment applied to the world is $\mathcal{G}^* = \arg\max V(\mathcal{G})$.

For the case study introduced in Section II, Fig. 5 shows the targets assigned to each UAV and its associated path as a solid

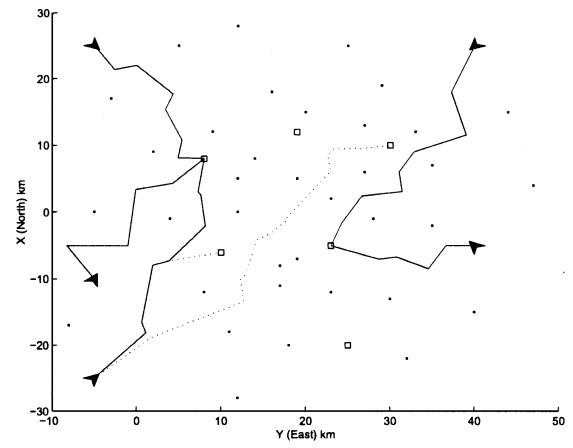


Fig. 5. Target assignment for case study scenario. Solid lines connect each vehicle to its assigned target. The dotted lines represent satisficing paths for the UAV in the bottom left corner.

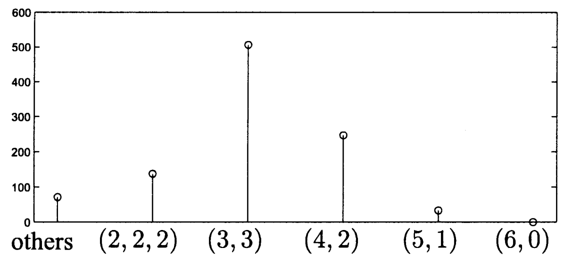


Fig. 6. Frequency of occurrence of each team size for 1000 experimental trials.

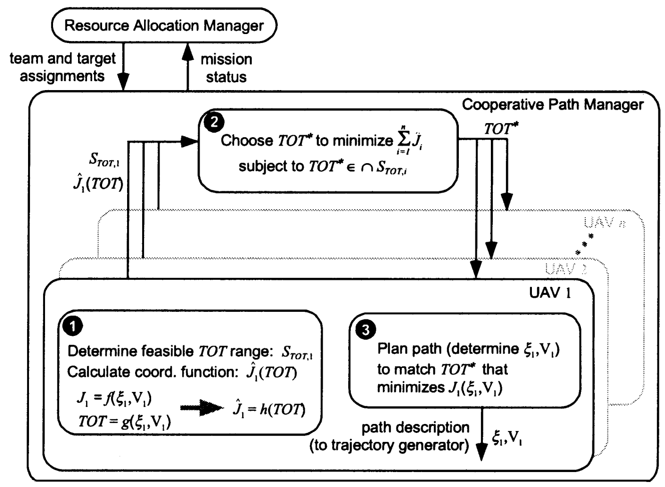

Resource Allocation Manager
team and target assignments
mission status
Cooperative Path Manager
Choose TOT* to minimize $\sum_{i=1}^{n} \hat{J}_i$ subject to TOT* ∈ ∩ $S_{TOT,i}$
$S_{TOT,i}$
$\hat{J}_i(TOT)$
TOT*
UAV n
UAV 2
UAV 1
Determine feasible TOT range: $S_{TOT,i}$
Calculate coord. function: $\hat{J}_i(TOT)$
$J_i = f(\xi_i, V_i)$
$TOT = g(\xi_i, V_i)$
$\hat{J}_i = h(TOT)$
Plan path (determine ξ_i, V_i) to match TOT* that minimizes $J_i(\xi_i, V_i)$
path description (to trajectory generator)
ξ_i, V_i

Fig. 7. Cooperative path planning algorithm.

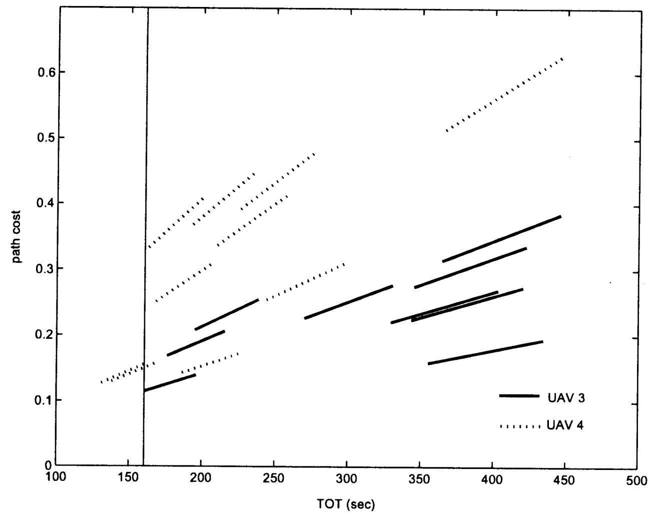


Fig. 8. Coordination functions for two UAVs.

line. In addition, the satisficing paths for the UAV in the bottom left-hand corner are plotted as dotted lines. Note the preference for nearby targets and targets that do not require passing too close to threats.

One of the more interesting ways to quantify the algorithm behavior is in the average chosen team size. Fig. 6 presents the results for 1000 experimental trials. This figure demonstrates that the algorithm tends to select teams with between two and four agents per team, and tends to avoid teams of one and teams greater than four. Thus, we can conclude that the team selection algorithm accomplishes our objective of maximizing the number of vehicles assigned to each target but also maximizing the spread of the prosecution. If vehicles are too far away to coordinate effectively (as enforced by the path length and threat exposure tradeoff), then they naturally form smaller strike forces to cover closer and more exposed targets.

C. Intercept Manager

Once teams have been formed and targets assigned to each team, it is necessary to plan the trajectories that each vehicle will take to its respective target. We will assume that coordinating the timing of target interception is critical. In the architecture developed here, the vehicles composing a team are required to prosecute their target simultaneously to enhance mission effectiveness. The challenge is to determine the best TOT specification for the team in light of the threat scenario and dynamic capabilities of the vehicles. The output of the intercept manager is a velocity and a set of waypoints for each vehicle. The corresponding waypoint paths avoid the threats and allow for simultaneous arrival of the vehicles at the assigned target.

Once candidate paths have been determined by searching the Voronoi graph, it remains to select which path each UAV should fly. Path selection should result in minimal collective threat exposure for the team and simultaneous arrival of team members at their target. The strategy for cooperatively planning UAV paths employed here is presented in detail in [12]. Fig. 7 depicts the steps of the cooperative path planning algorithm.

Candidate paths to the target for UAV i are parameterized by the waypoints ξ_i and the UAV forward speed V_i. TOT for each UAV is a function of the speed of the UAV along the selected path: $\mathrm{TOT}_i = f(\xi_i, V_i)$. Similarly, the cost (threat and fuel) to travel along any path to the target is a function of the path and the speed: $\mathcal{J}_i = g(\xi_i, V_i)$. To choose the best TOT for the team in a cooperative manner, some information regarding the candidate paths for the UAVs must be exchanged. Rather than passing ξ_i and V_i around among all of the UAVs, a more efficient parameterization of information, called the coordination function, is used. In this problem, the coordination function $\hat{\mathcal{J}}_i$ models the cost to UAV i of achieving a particular TOT: $\hat{\mathcal{J}}_i = h(\mathrm{TOT})$.

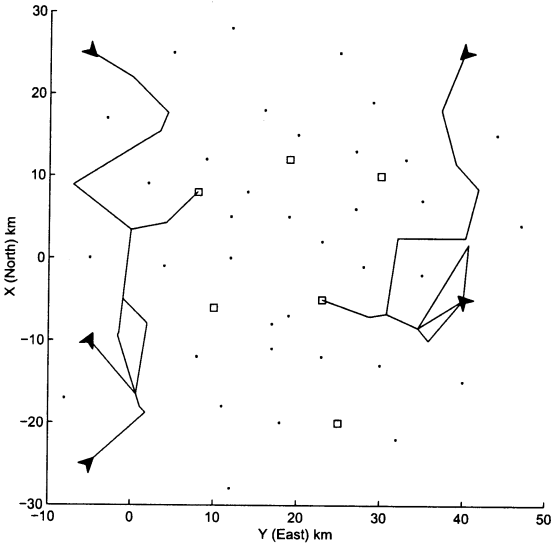


Fig. 9. Coordinated paths that have been planned to synchronize TOT for each team.

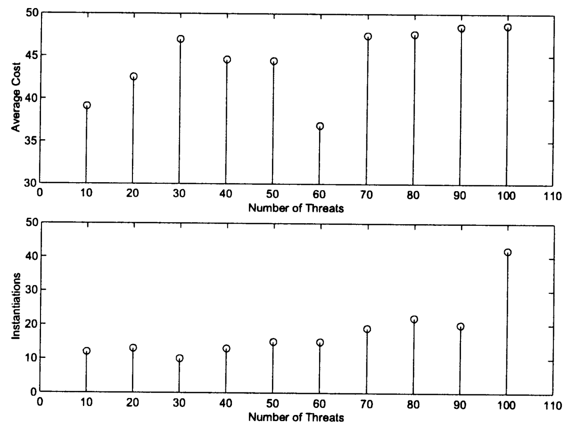


Fig. 10. Average threat cost, path length, and number of instantiations needed for ten successes, versus number of threats.

The coordination function is determined from the relations for TOT_i and $\mathcal{J}_i$. Based on the candidate paths determined from the Voronoi diagram and the feasible range of UAV air speeds, the feasible TOT range for each UAV on the team can also be determined. The set of feasible TOT ranges for UAV i is denoted by $S_{\mathrm{TOT},i}$. Coordination function information for two UAVs composing a team is shown in Fig. 8. Note that each candidate path produces a line segment in the plot.

Coordination function and feasible TOT range information are exchanged among vehicles to enable cooperative path planning. At the team level, the cooperative planning problem is simplified to finding TOT^* that minimizes the collective threat exposure of the team

$$\mathrm{TOT}^* = \arg\min \sum_{i=1}^{n} \hat{\mathcal{J}}_i$$

subject to

$$\mathrm{TOT}^* \in \cap S_{\mathrm{TOT},i}.$$

From Fig. 8, it can be seen that this optimization results in a team-optimal value of $\mathrm{TOT}^* = 160.5$ s. TOT is called the coordination variable. By requiring that individual UAV match the team-optimal value of the coordination variable TOT^*, cooperation is ensured. We note that this optimization problem has been greatly simplified by searching over the coordination variable instead of the original path parameters (ξ_i, V_i) of the system. For example, in the coordinated strike problem, the coordination variable is one dimensional (TOT), whereas the original path parameterization is of much higher dimension.

Once TOT^* has been determined, each UAV must determine which of the candidate paths ξ_i to take and the appropriate flight speed V_i so that its own threat exposure $\mathcal{J}_i = f(\xi_i, V_i)$ is minimized and the TOT^* value is matched. In implementation, the best path for a specified TOT was determined when the coordination function was computed. At this point, it is simply a matter of looking up the trajectory waypoints in memory and computing the appropriate velocity. Waypoint and velocity information is then supplied to the trajectory generator on the UAV.

For the target assignment determined in Section III-B, Fig. 9 shows the paths that have been generated to guarantee that the TOT for each team is synchronized.

To quantify the limitations of the intercept manager, we ran a number of experiments with three UAVs assigned to simultaneously intercept a single target. The initial conditions of the UAVs were fixed, as was the target location. The number of threats was varied from $N = 10$ to $N = 100$ in increments of ten. For a given number of threats, random threat configurations were generated, and an instantiation of the intercept manager was run on each configuration. For each instantiation, the success or failure of the algorithm was recorded, as well as the total cost $J = \kappa J_{\mathrm{length}} + (1-\kappa)J_{\mathrm{threat}}$ This process was repeated until ten successful instantiations were recorded. Fig. 10 shows the total cost and the number of instantiations needed to generate ten successes, as a function of the number of threats N.

Total cost increases as the number of threats increase. This is expected since the UAVs will be required to make more deviations to avoid the threats. It is interesting that cost levels off as the threat density is increased.

The rendezvous manager fails when the intersection of feasible TOT values is empty. Failure may occur when a sufficient number of candidate paths have not been generated. In that case, the number of candidate paths k must be increased when calling Eppstein's algorithm. As the threat density increases, the path length of the k-best paths may be very similar, and hence, may cause the intersection of feasible TOT values to become empty. Again, increasing k will solve this problem. For the results shown in Fig. 10, k was fixed at ten. Alternatively, the algorithm could be modified so that if the algorithm fails, k is increased until the algorithm succeeds. Another interpretation of Fig. 10 is that it qualitatively describes the required increase in k as a function of N, to guarantee success of the intercept manager. There appears to be a dramatic increase in failures at about $N = 100$ threats.

D. Trajectory Generator

Given a set of waypoints that define a coarse path, the objective of the trajectory generator (TG) is to generate time-

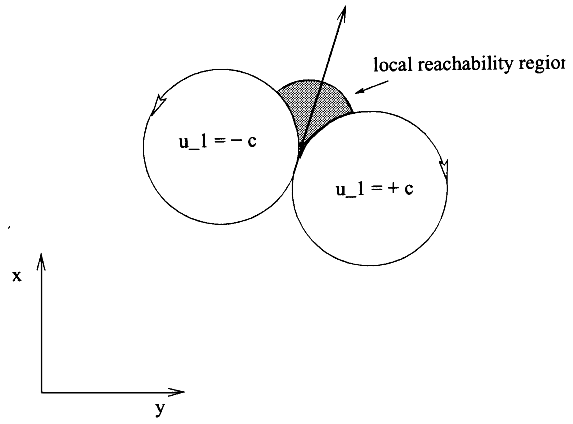


Fig. 11. Local reachability region of the trajectory generator.

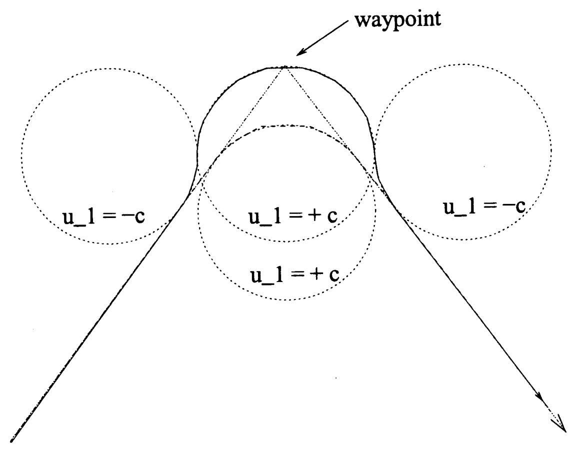


Fig. 12. Time-optimal transitions between path segments. The dashed line is not constrained to go through the waypoint. The solid line is constrained to pass through the waypoint.

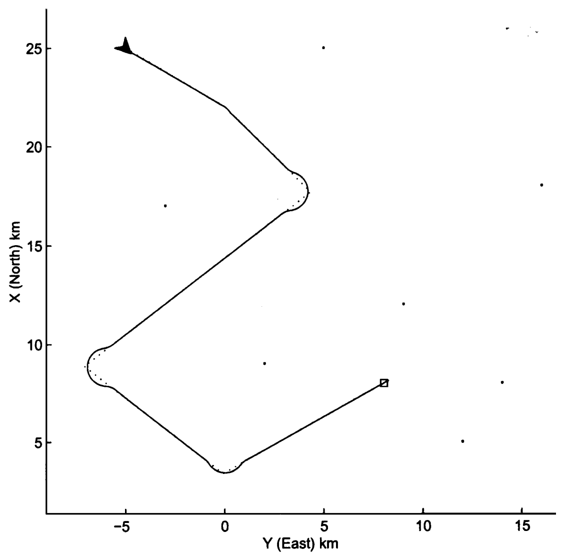


Fig. 13. Trajectory generated by TG to smooth through the Voronoi points such that the length of the trajectory equals that of the Voronoi path.

parameterized trajectories that are feasible within the dynamic constraints of the vehicle. The essential idea is to use a nonlinear filter that has a mathematical structure similar to the kinematics

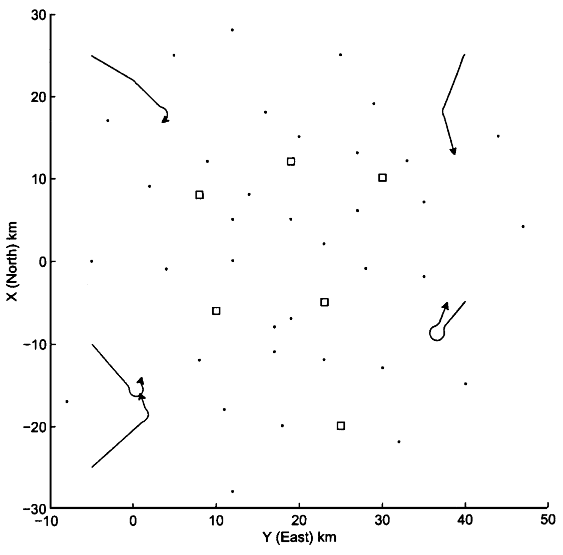


Fig. 14. Mission scenario: one fourth of the way through the mission.

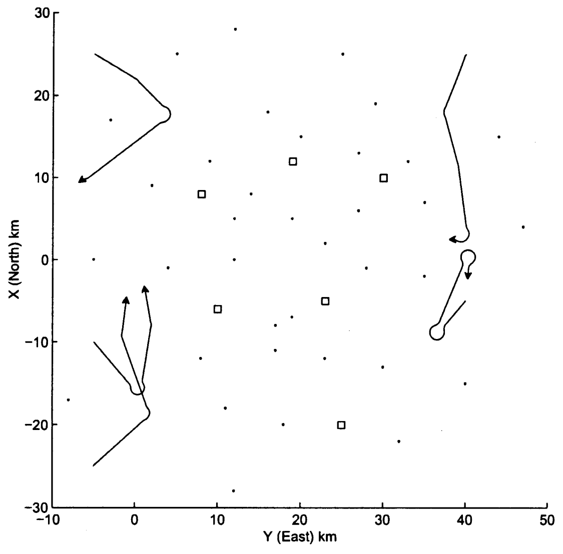


Fig. 15. Mission scenario: one half of the way through the mission.

of the vehicle to generate trajectories that smooth through the waypoints in real time. The UAV dynamics listed in [26] suggest the following structure for the TG for constant altitude maneuvers:

$$\begin{aligned}\dot{X}_i^d &= V_i^d \cos \psi_i^d \\ \dot{Y}_i^d &= V_i^d \sin \psi_i^d \\ \dot{\psi}_i^d &= u_1 \\ \dot{V}_i^d &= u_2 \\ \dot{h}_i^d &= 0\end{aligned} \tag{8}$$

where (X_i^d, Y_i^d) is the desired inertial position of the ith UAV, ψ_i^d is the desired heading, V_i^d is the desired velocity, h_i^d is the desired altitude, and u_1 and u_2 are constrained by the dynamic capability of the UAVs, namely, the heading rate constraint and the acceleration constraint, respectively.

Assuming that the desired velocity is held constant by letting $u_2 = 0$. The turning rate constraint can be expressed as $-c \leq$

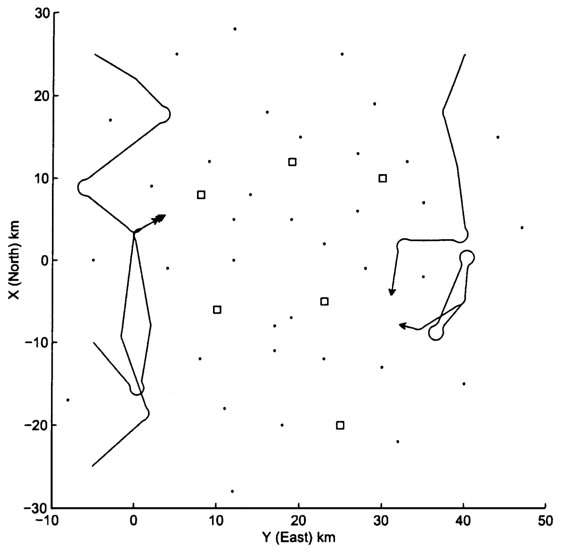


Fig. 16. Mission scenario: three quarters through the mission.

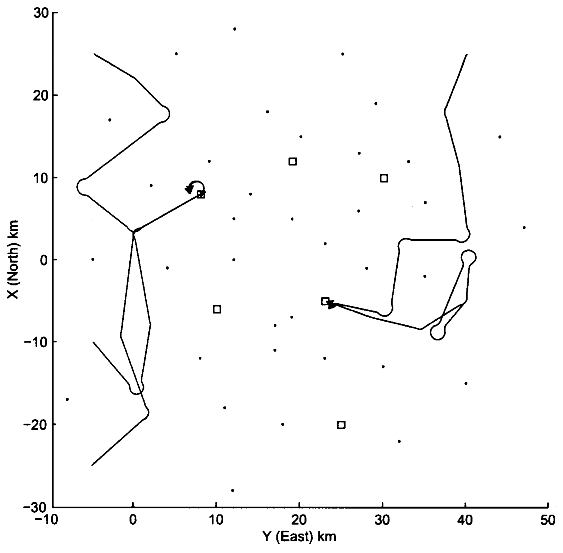


Fig. 18. Mission scenario: the second team has intercepted their target.

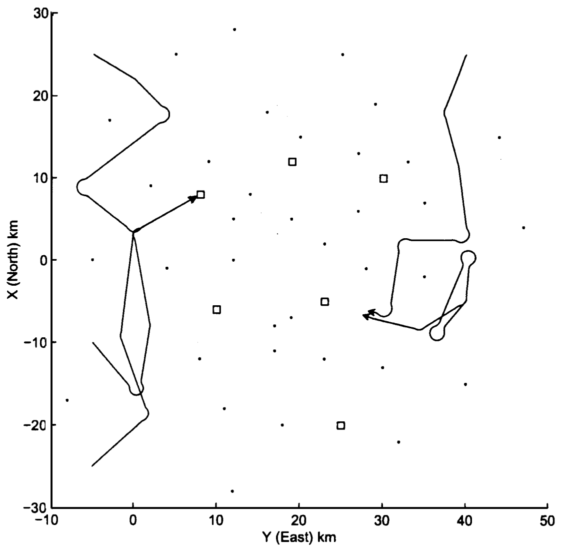


Fig. 17. Mission scenario: the first team has intercepted their target.

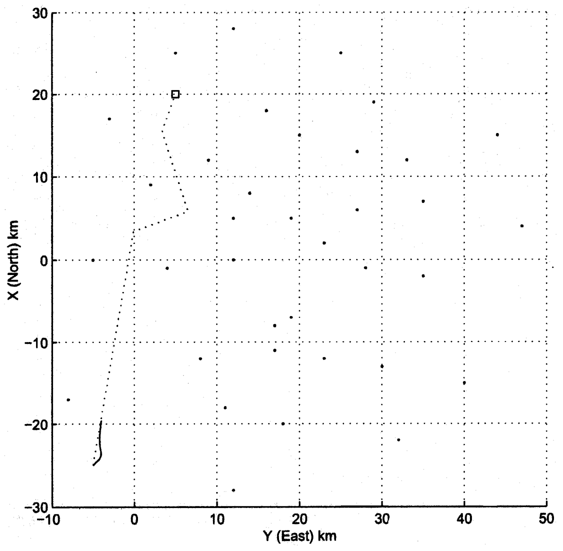


Fig. 19. Pop-up scenario: First pop-up threat is about to be detected.

$u_1 \le c$. Note that if $u_1 = c$, then the TG given in (8) traces out a right-hand circle, as shown in Fig. 11. Similarly, if $u_1 = -c$, then the TG traces out a left-hand circle. As Fig. 11 shows, the local reachability region of the TG is bounded by these two circles. The radius of the circles defining the local reachability region is given by

$$R_i = \frac{V_i^d}{c}.$$

Note that as the desired velocity increases, the minimum turning radius increases.

Consider the problem of switching from one straight-line segment to another in minimum time at a constant velocity. If the trajectory is not constrained to pass through the waypoint connecting the two lines, then a time-optimal trajectory is shown as a dashed line in Fig. 12.

If $\left(X_i^d, Y_i^d, \psi_i^d, V_i^d\right)$ is the configuration of the TG, then the centers of the two circles that bound the reachability region are given by

$$z_{\text{right}} = \begin{pmatrix} X_i^d \\ Y_i^d \end{pmatrix} + \begin{pmatrix} -\frac{V_i^d}{c}\sin\left(\psi_i^d\right) \\ \frac{V_i^d}{c}\cos\left(\psi_i^d\right) \end{pmatrix}$$

$$z_{\text{left}} = \begin{pmatrix} X_i^d \\ Y_i^d \end{pmatrix} + \begin{pmatrix} \frac{V_i^d}{c}\sin\left(\psi_i^d\right) \\ -\frac{V_i^d}{c}\cos\left(\psi_i^d\right) \end{pmatrix}.$$

Since the boundaries of the reachability region are given by circles with known centers and radii, the intersection of the reachability region with lines and circles can be found in a computationally efficient way. A simple, real-time algorithm for trajectory generation is, therefore, suggested by Fig. 12.

The following algorithm is for minimum-time transition between path segments where the trajectory is not constrained to go through the waypoint.

Algorithm 1:

```
Step 1) While traversing the current path
        segment, set $u_1 = 0$ and monitor the
        distance from the boundaries of the
        reachability set to the next path
        segment.
Step 2) When the right (resp. left)
        boundary intersects the next path
        segment, set $u_1 = +c$, (resp. $-c$).
Step 3) When the left (resp. right)
        boundary intersects the next path
        segment, set $u_1 = 0$, and go to Step
        1.
```

The next algorithm is for minimum-time transition between path segments where the trajectory is constrained to go through the waypoint.

Algorithm 2:

```
Step 1) Inscribe a circle whose center
        lies on the bisector of the angle
        formed by the current line segment
        and the next, and that intersects
        the waypoint.
Step 2) While traversing the current line
        segment, set $u_1 = 0$ and monitor the
        distance from the boundaries of the
        reachability set to the next line
        segment.
Step 3) When the left (resp. right)
        boundary intersects the circle
        passing through the waypoint, set
        $u_1 = -c$ (resp. $+c$).
Step 4) When the trajectory intersects the
        inscribed circle, set $u_1 = +c$ (resp.
        $-c$).
Step 5) When the left (resp. right)
        boundary intersects the next line
        segment, set $u_1 = -c$ (resp. $+c$).
Step 6) When the right (resp. left)
        boundary intersects the next line
        segment, set $u_1 = 0$, and go to Step
        1.
```

One of the disadvantages of both minimum time transitions and transitions constrained to go through the waypoint is that trajectories that are generated will have a different path length than the original Voronoi path. Since the Voronoi path is used to determine intercept times, we would like to transition between path segments, such that the path length of the feasible trajectory is equal to the path length of the original Voronoi path. From Fig. 12, it is obvious that the path length of a minimum time trajectory will be less than than the path length of the Voronoi path, and that the path length of the constrained trajectory will be greater than the path length of the Voronoi path. Let $\kappa \in [0, 1]$ be a dimensionless parameterization of the distance d from the waypoint to an inscribed circle C whose center lies along the bisector of the angle shown in Fig. 12, such that the distance d is no larger than the distance from the waypoint to the inscribed circle of *Algorithm 1*. Modify *Algorithm 2* so that it uses one of these C's. *Algorithm 1* corresponds to $\kappa = 1$; *Algorithm 2* corresponds to $\kappa \in [0, 1)$. *Algorithm 2* can be modified with a preliminary step to find κ such that the path lengths are equal [27], [28].

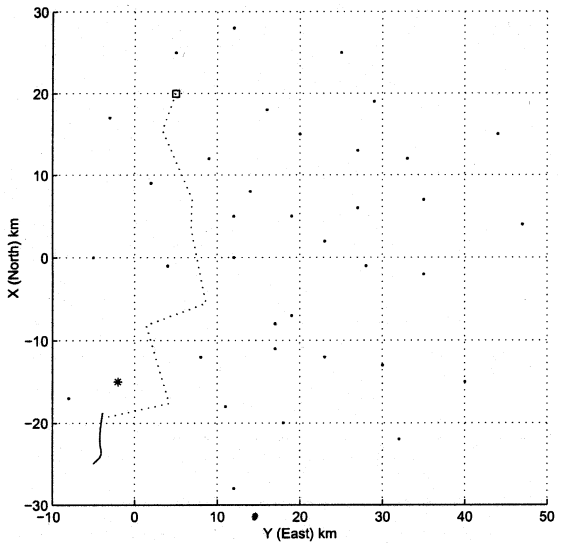


Fig. 20. Pop-up scenario: The first pop-up threat is detected.

The above algorithms are easily modified to account for short path segments that are smaller than the radius R_i. To account for those cases, the TG must look ahead several path segments.

There are several advantages to our approach. First, it couples nicely with the Voronoi search algorithms described in Section III-C. Second, the approach has low computational overhead. In fact, trajectories are generated in real time, as the vehicles move. Third, it can be shown by using optimal control techniques that the approach minimizes the time that the vehicle deviates from the Voronoi path. Finally, it can be adapted to allow low-level deconfliction, and to allow threat avoidance for dynamic threats. The algorithm described above can be modified using behavior-based approaches [29] to allow local adjustments to the path in response to dynamically moving threats.

Fig. 13 shows the trajectory generated for a single UAV and target from Fig. 9. The trajectory has been planned such that the lengths of the Voronoi path and the trajectory are equal.

IV. Simulated Mission Scenario

In this section, we will demonstrate the effectiveness of our approach for the mission scenario described in Section II and discussed throughout the paper. Figs. 14–18 show five different stages of the mission.

In Fig. 17, the first team has intercepted their target. Note the simultaneous arrival times. Throughout the paper, we have assumed that individual UAVs fly at different, preassigned altitudes, thereby ensuring collision avoidance. In Fig. 18, the second UAV team has intercepted their target.

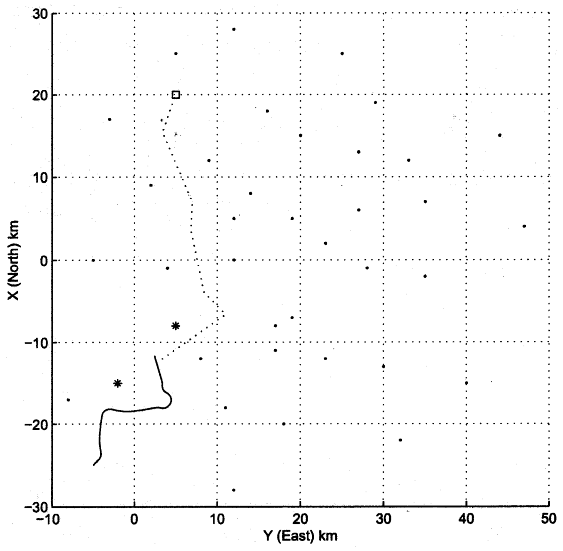


Fig. 21. Pop-up scenario: The second pop-up threat is detected.

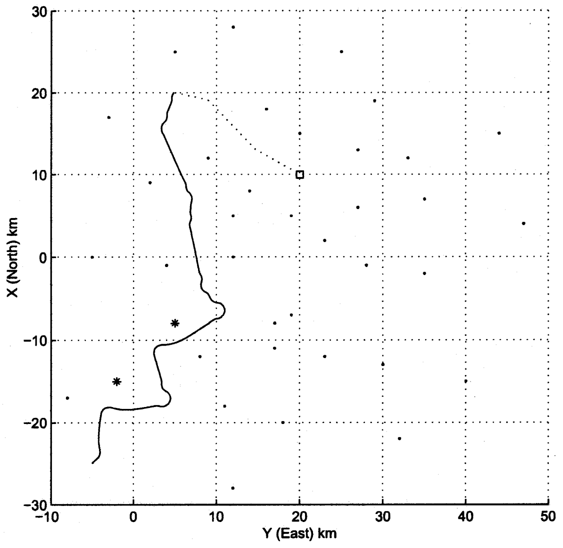


Fig. 22. Pop-up scenario: The first target is reached.

Pop-Up Threats: When pop-up threats are encountered the target assignment algorithm is reinstantiated with the new threats list. Subsequently, the rendezvous manager and path planner are also reinstantiated. For clarity, the response to pop-up threats is shown with a single UAV, assigned to a single target. The dotted line in Fig. 19 is the initial waypoint path. The solid line shows the trajectory produced by the TG from the initial time until the instant immediately before the detection of the first pop-up threat. In Fig. 20, a new threat at location $(-15$ km, -2 km) has been detected by the UAV. The new waypoint path is shown in the dotted line. Fig. 21 shows the state of the path planner/TG immediately after a second pop-up threat has been detected at position $(-8$ km, 5 km). Finally, Fig. 22 shows the state of the path planner/TG at the instant of time when the first target location has been reached and a new waypoint path, represented by the dotted line, to the next target has been planned.

V. Conclusion and Discussion

In this paper, we have developed a complete system architecture for the target assignment and coordinated intercept problem. Coordination between UAVs takes place at two levels. At the highest level, the UAVs must negotiate a target assignment vector that assigns UAVs to targets. Once a target assignment has been made, the UAVs that compose each team must coordinate to find a feasible team TOT.

The target assignment problem was solved using the satisficing and social welfare paradigms. The TOT coordination problem was solved by encapsulating the essential myopic information in coordination functions that were then used to optimize the team TOT. The path-planning problem was solved via a Voronoi diagram and Eppstein's k-best paths algorithm. The TG problem was solved via a novel real-time nonlinear filter that explicitly accounts for the dynamic constraints of the vehicle. The decomposition of the motion-planning problem into a waypoint path planner and a dynamic trajectory generator has the advantage of decomposing a nonpolynomial optimization problem into two subproblems that can be computed in near-real time. The disadvantage is that the resulting solution is suboptimal.

Acknowledgment

The technical guidance of Dr. P. Chandler at AFLR is gratefully acknowledged. The authors would like to thank K. Judd for assistance with developing the path planner.

References

[1] R. E. Burkard, "Selected topics in assignment problems," Inst. Math. B, Tech. Univ. Graz, Graz, Austria, 1999.
[2] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *Artif. Intell.*, vol. 110, no. 2, pp. 241–273, June 1999.
[3] F. Brandt, W. Brauer, and G. Weiß, "Task assignment in multiagent systems based on Vickrey-type auctioning and leveled commitment contracting," in *Proc. Cooperative Information Agents (CIA-2000) Workshop*, Boston, MA, 2000, pp. 95–106.
[4] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin, "Frequency assignment in cellular phone networks," in *Proc. Int. Symp. Mathematical Programming*, Lausanne, Switzerland, Aug. 24–29, 1997.
[5] Y. Li, P. M. Pardalos, and M. G. C. Resende, "A greedy randomized adaptive search procedure for the quadratic assignment problem," *DIMACS Series on Discrete Math. and Theoretical Comp. Sci.*, vol. 16, pp. 237–261, 1994.
[6] W. C. Stirling and M. A. Goodrich, "Conditional preferences for social systems," in *Proc. IEEE Int. Conf. Systems, Man, Cybernetics*, 2001, pp. 995–1000.
[7] M. A. Goodrich, W. C. Stirling, and R. L. Frost, "A theory of satisficing decisions and control," *IEEE Trans. Syst., Man, and Cybern. A*, vol. 28, pp. 763–779, Nov. 1998.
[8] ——, "Model predictive satisficing fuzzy logic control," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 319–332, June 1999.
[9] W. C. Stirling and M. A. Goodrich, "Satisficing games," *Inform. Sci.*, vol. 114, pp. 255–280, Mar. 1999.
[10] R. Beard, R. Frost, and W. Stirling, "A hierarchical coordination scheme for satellite formation initialization," in *Proc. AIAA Guidance, Navigation and Control Conf.*, Boston, MA, Aug. 1998, AIAA Paper 98-4225, pp. 677–685.
[11] T. McLain and R. Beard, "Cooperative rendezvous of multiple unmanned air vehicles," in *Proc. AIAA Guidance, Navigation and Control Conf.*, Denver, CO, Aug. 2000, AIAA Paper 2000-4369 (CD-ROM).

[12] T. McLain, P. Chandler, S. Rasmussen, and M. Pachter, "Cooperative control of UAV rendezvous," in *Proc. ACC*, June 2001, pp. 2309–2314.

[13] F. Gentili and F. Martinelli, "Robot group formations: A dynamic programming approach for a shortest path computation," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 2000, pp. 3152–3157.

[14] R. Sedgewick, *Algorithms*, 2nd ed. Reading, MA: Addison-Wesley, 1988.

[15] D. Eppstein, "Finding the k shortest paths," *SIAM J. Comput.*, vol. 28, no. 2, pp. 652–673, 1999.

[16] P. Chandler, S. Rasumussen, and M. Pachter, "UAV cooperative path planning," in *Proc. AIAA Guidance, Navigation, and Control Conf.*, Denver, CO, Aug. 2000, AIAA Paper 2000-4370.

[17] M. J. Van Nieuwstadt and R. M. Murray, "Real time trajectory generation for differentially flat systems," *Int. J. Robust Nonlinear Contr.*, vol. 8, no. 11, pp. 995–1020, 1998.

[18] F. Lamiraux, S. Sekhavat, and J.-P. Laumond, "Motion planning and control for Hilare pulling a trailer," *IEEE Trans. Robot. Automat.*, vol. 15, pp. 640–652, Aug. 1999.

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

[20] S. Sun, M. B. Egerstedt, and C. F. Martin, "Control theoretic smoothing splines," *IEEE Trans. Automat. Contr.*, vol. 45, pp. 2271–2279, Dec. 2000.

[21] N. Faiz, S. K. Agrawal, and R. M. Murray, "Trajectory planning of differentially flat systems with dynamics and inequalities," *AIAA J. Guidance, Contr., Dynam.*, vol. 24, pp. 219–227, Mar.–Apr. 2001.

[22] J.-S. Choi and B. K. Kim, "Near-time-optimal trajectory planning for wheeled mobile robots with translational and rotational sections," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 85–90, Feb. 2001.

[23] F. Giulietti, L. Pollini, and M. Innocenti, "Autonomous formation flight," *IEEE Contr. Syst. Mag.*, vol. 20, pp. 34–44, Dec. 2000.

[24] D. J. Stilwell and B. E. Bishop, "Platoons of underwater vehicles," *IEEE Contr. Syst. Mag.*, vol. 20, pp. 45–52, Dec. 2000.

[25] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. New York: Springer-Verlag, 2000.

[26] A. W. Proud, M. Pachter, and J. J. D'Azzo, "Close formation flight control," in *Proc. AIAA Guidance, Navigation, and Control Conf.*, Portland, OR, Aug. 1999, AIAA Paper 99-4207, pp. 1231–1246.

[27] E. P. Anderson, "Constrained extremal trajectories and unmanned air vehicle trajectory generation," Master's thesis, Brigham Young Univ., Provo, UT, 2002.

[28] E. P. Anderson and R. W. Beard, "Constrained extremal trajectories and UAV path planning," in *Proc. AIAA Guidance, Navigation and Control Conf.*, Aug. 2002, AIAA Paper 2002–4470.

[29] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 926–939, Dec. 1998.

Randal W. Beard (S'91–M'92) received the B.S. degree in electrical engineering from the University of Utah, Salt Lake City, in 1991, and the M.S. degree in electrical engineering in 1993, the M.S. degree in mathematics in 1994, and the Ph.D. degree in electrical engineering in 1995, all from Rensselaer Polytechnic Institute, Troy, NY.

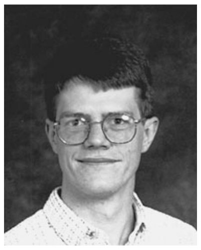
Since 1996, he has been with the Electrical and Computer Engineering Department at Brigham Young University, Provo, UT, where he is currently an Associate Professor. In 1997 and 1998, he was a Summer Faculty Fellow at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. His research interests include coordinated control of multiple vehicle systems and nonlinear and optimal control.

Dr. Beard is a member of the AIAA, and Tau Beta Pi, and is currently an Associate Editor for the IEEE Control Systems Society Conference Editorial Board.

Timothy W. McLain (S'91–M'93) received the B.S. and the M.S. degrees in mechanical engineering from Brigham Young University, Provo, UT, in 1986 and 1987, respectively. In 1995, he received the Ph.D. degree in mechanical engineering from Stanford University, Stanford, CA.

From 1987 to 1989, he was employed at the Center for Engineering Design at the University of Utah, Salt Lake City. Since 1995, he has been with the Department of Mechanical Engineering at Brigham Young University, Provo, UT, where he is currently an Associate Professor. In the summers of 1999 and 2000, he was a Visiting Scientist at the Air Force Research Laboratory, Air Vehicles Directorate. His research interests include cooperative control of multiple vehicle systems and modeling and control of microelectromechanical systems.

Dr. McLain is a member of the AIAA.



Michael A. Goodrich (S'92–M'92) received the B.S., M.S., and Ph.D. degrees from the Electrical and Computer Engineering Department, Brigham Young University, Provo, UT, in 1992, 1995, and 1996, respectively.

From 1996 to 1998, he was a Postdoctoral Research Associate at Nissan Cambridge Basic Research, Cambridge, MA. Since 1998, he has been with the Computer Science Department at Brigham Young University, Provo, UT, where he is currently an Assistant Professor. His research interests include human–robot interaction, decision theory, and multiagent choice.

Erik P. Anderson received the B.S. degree in electrical engineering, the B.S. degree in mathematics, and the M.S. degree in electrical engineering, all from Brigham Young University, Provo, UT, in 2002.

He currently is a graduate student in electrical engineering at Stanford University, Palo Alto, CA, where he is working toward the Ph.D. degree.