AIAA Guidance, Navigation, and Control Conference and Exhibit
5-8 August 2002, Monterey, California

AIAA 2002-4588

COORDINATION AND CONTROL OF MULTIPLE UAVs

Arthur Richards,* John Bellingham,† Michael Tillerson,‡ and Jonathan How§

ABSTRACT

This paper addresses the problems of autonomous task allocation and trajectory planning for a fleet of UAVs. Two methods are compared for solving the optimization that combines task assignment, subjected to UAV capability constraints, and path planning, subjected to dynamics, avoidance and timing constraints. Both sub-problems are non-convex and the two are strongly-coupled. The first method expresses the entire problem as a single mixed-integer linear program (MILP) that can be solved using available software. This method is guaranteed to find the globally-optimal solution to the problem, but is computationally intensive. The second method employs an approximation for rapid computation of the cost of many different trajectories. This enables the assignment and trajectory problems to be decoupled and partially distributed, offering much faster computation. The paper presents several examples to compare the performance and computational results from these two algorithms.

INTRODUCTION

The capabilities and roles of Unmanned Aerial Vehicles (UAVs) are evolving, and require new concepts for their control.[1] For example, today's UAVs typically require several operators, but future UAVs will be designed to make tactical decisions autonomously and will be integrated into teams that coordinate to achieve high-level goals, thereby allowing one operator to control a group of vehicles. New methods in planning and execution are required to coordinate the operation of a fleet of UAVs. An overall control system architecture must also be developed that can perform optimal coordination of the vehicles, evaluate the overall system performance in real-time, and quickly reconfigure to account for changes in the environment or the fleet. This paper presents results on guidance and control of fleets of cooperating UAVs. This includes the goal assignment, resource allocation, and trajectory optimization problems. For many vehicles, obstacles, and targets, fleet coordination is a very complicated optimization problem.[2,3,8]

A general problem statement is posed involving heterogenous vehicles and waypoints subject to obstacle avoidance, vehicle dynamics, and timing constraints. A typical scenario is illustrated in Fig. 1. Five vehicles are to be assigned among six waypoints, but the vehicles must avoid the hatched regions, or "no-fly-zones". Each task must be performed at a corresponding waypoint and each vehicle has the capabilities to do some, but not necessarily all, of the tasks, *e.g.,* vehicle 1 can only do the tasks at points A, C and F. Timing constraints are also included to represent the need to perform some tasks in a specific order, such as waypoint D to be visited at least one minute before point B. Subject to these constraints, the trajectories and assignments are designed to minimize some cost function, in this case the overall mission completion time.

The complexity of this class of problems arises from the inherent coupling between the assignment of tasks and the trajectory generation. The cost for a vehicle to visit a particular waypoint is strongly dependent on the path taken and hence on other waypoints visited on the way. If trajectory optimization could be performed for all permutations of visits, the assignment could be done quite easily, resulting in the globally-optimal, dynamically-feasible solution. However, the number of possible permuations grows combinatorially with the number of vehicles and waypoints, and each trajectory optimization is highly complex, so this method is impractical for

* Research Assistant, MIT Dept. of Aeronautics and Astronautics, **arthurr@mit.edu**
† Research Assistant, MIT Dept. of Aeronautics and Astronautics, **john_b@mit.edu**
‡ Research Assistant, MIT Dept. of Aeronautics and Astronautics, **mike_t@mit.edu**
§ Associate Professor, MIT Dept. of Aeronautics and Astronautics, **jhow@mit.edu**. Room 33-328, 77 Mass. Ave., Cambridge, MA 02139.

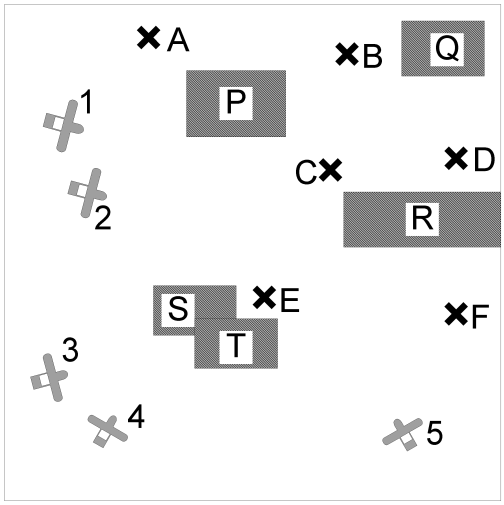

Figure 1: Example Scenario

all but the smallest problems. Previous work treats the two sub-problems separately,[2,3] using hierarchical approaches and auction-based assignment.

This paper presents two methods for solving this problem. The first captures the coupling of the assignment and trajectory design by combining both into a single *mixed-integer linear program* (MILP) optimization. This method uses a linearized form of aircraft dynamics previously applied to UAV avoidance problems.[4] The logical constraints for collision and obstacle avoidance can be encoded as constraints upon binary variables,[6] resulting in a MILP problem. Further logical constraints are added in this paper to include the general form of assignment of vehicles to waypoints, subject to the constraints described above. The combined method is guaranteed to find the globally-optimal solution, but the computation can be intensive. Using current optimization techniques, this method is not well-suited for real-time operation. However, it provides a benchmark against which all other methods, which may be faster but based on heuristics, can be evaluated for speed and performance.

The second method presented simplifies the coupling between the assignment and trajectory design problems by calculating and communicating only the key information that connects them.[8] It estimates the cost of various trajectory options by using straight line path approximations.[9] This method takes advantage of the fact that, for typical missions, the shortest paths for the UAVs tend to resemble straight lines that connect the UAVs starting position, the corners of obstacles, and the waypoints. The assignment is then performed using these approximated costs. A more detailed trajectory generator is then used to plan exact, dynamically feasible paths for the given assignments. The cost estimation and detailed trajectory design steps can both be performed on distributed platforms for faster execution.

The following sections present the problem formulation and these two solution methods. The performance of these algorithms is then compared on several examples.

PROBLEM STATEMENT

This section describes a matrix formulation for encoding the statement of the problem described in the previous section. Let there be N_V vehicles, N_W waypoints and N_Z no-fly-zones. A total of N_T time steps are used for planning, although the mission will typically not require all of this time. The UAVs are modeled as vehicles moving in two dimensions with limited speed and turning rate. Therefore, the vehicle dynamics are completely specified by the vectors $\mathbf{v}_{\max}$ and ω, where $v_{\max_{\mathrm{p}}}$ and ω_{p} are the maximum speed and turning rate of the p^{th} vehicle respectively. Both $\mathbf{v}_{\max}$ and ω are N_V-element vectors. Initial states are also included in a matrix $\mathbf{S}$ such that the p^{th} row is the initial state vector $(x, y, \dot{x}, \dot{y})$ of the p^{th} vehicle, forming a $N_V \times 4$ matrix.

The waypoints are specified by the $N_W \times 2$ matrix $\mathbf{W}$ where $(W_{\mathrm{i}1}, W_{\mathrm{i}2})$ is the position of the i^{th} waypoint. The no-fly-zones can be any convex polygons, but are modeled as rectangles here for simplicity. They are defined by the $N_Z \times 4$ matrix $\mathbf{Z}$ where $(Z_{\mathrm{j}1}, Z_{\mathrm{j}2})$ is the bottom left vertex of the j^{th} no-fly-zone and $(Z_{\mathrm{j}3}, Z_{\mathrm{j}4})$ is the top right vertex. The vehicle capabilities are included in the matrix $\mathbf{K}$, in which $K_{pi} = 1$ if vehicle p can visit the i^{th} waypoint and 0 otherwise. The matrix $\mathbf{K}$ has size $N_V \times N_W$.

Time dependencies, forcing one waypoint to be visited after another, separated by some interval, are included in the matrix $\mathbf{\Delta}$. Each row of the matrix represents a time dependency and it has a column for each waypoint. Thus if there are N_D time dependencies, the matrix is $N_D \times N_W$. A dependency is encoded by -1 in the column corresponding to the first waypoint and $+1$ in the column for the second. The corresponding element in the vector $\mathbf{t}_D$ is the interval between the two visits. Thus, if the vector $\mathbf{t}$ contained the visiting times for each waypoint, the constraint would be implemented as

$$\mathbf{\Delta t} \geq \mathbf{t}_D \tag{1}$$

Therefore, in summary, the problem can be completely specified by the vectors and matrices

$$\mathbf{v}_{\max}, \omega, \mathbf{S}, \mathbf{W}, \mathbf{Z}, \mathbf{K}, \mathbf{\Delta}, \mathbf{t}_D$$

COMBINED METHOD

This section describes the solution of the global optimal assignment and trajectory problem as a single MILP. It has been shown that the problem of designing a trajectory for an aircraft to visit a preassigned list of waypoints can be written and solved as a MILP.[4] A summary of that method is given here, with extended waypoint selection constraints to include the assignment and capability constraints for a group of heterogenous vehicles. The effect of these assignment constraints are then shown in a series of examples.

Dynamics in the Combined Method

This section presents the model of aircraft dynamics used in the combined method (see Ref. [4] for details). Each aircraft is modeled as a point mass moving in 2-D. Let the position of aircraft p at time-step t be given by $(x_{\mathrm{tp}}, y_{\mathrm{tp}})$ and its velocity by $(\dot{x}_{\mathrm{tp}}, \dot{y}_{\mathrm{tp}})$, forming the elements of the state vector $\mathbf{s}_{\mathrm{tp}}$. The aircraft is assumed to be acted upon by control forces $(f_{x_{\mathrm{tp}}}, f_{y_{\mathrm{tp}}})$ in the X- and Y-directions respectively, forming the force vector $\mathbf{f}_{\mathrm{tp}}$.

The maximum speed $v_{\max_{\mathrm{p}}}$ is enforced by an approximation to a circular region in the velocity plane

$$\forall t \in [1 \ldots N_T] \, \forall p \in [1 \ldots N_V] \, \forall m \in [1 \ldots N_C]$$
$$\dot{x}_{\mathrm{tp}} \sin\left(\frac{2\pi m}{N_C}\right) + \dot{y}_{\mathrm{tp}} \cos\left(\frac{2\pi m}{N_C}\right) \leq v_{\max_{\mathrm{p}}} \quad (2)$$

where N_C is the order of discretization of the circle. The maximum turning rate is enforced by limiting the force magnitude, using another circular region approximation

$$\forall t \in [0 \ldots N_T - 1] \, \forall p \in [1 \ldots N_V] \, \forall m \in [1 \ldots N_C]$$
$$f_{\mathrm{x_{tp}}} \sin\left(\frac{2\pi m}{N_C}\right) + f_{\mathrm{y_{tp}}} \cos\left(\frac{2\pi m}{N_C}\right) \leq f_{\mathrm{p}} \quad (3)$$

where f_{p} is related to the maximum turn rate, for travel at constant speed $v_{\max_{\mathrm{p}}}$, by

$$\omega_{\mathrm{p}} = \frac{f_{\mathrm{p}}}{v_{\max_{\mathrm{p}}}} \quad (4)$$

The discretized dynamics of the overall system, applied to all N_V vehicles up to N_T time-steps, can be written in the linear form

$$\forall p \in [1 \ldots N_V] \, \forall t \in [0 \ldots N_T - 1]$$
$$\mathbf{s}_{(\mathrm{t+1})\mathrm{p}} = \mathbf{A}\mathbf{s}_{\mathrm{tp}} + \mathbf{B}\mathbf{f}_{\mathrm{tp}} \quad (5)$$

where $\mathbf{A}$ and $\mathbf{B}$ are the system dynamics matrices for a unit point mass. In all cases, the initial conditions are specified from the initial condition matrix $\mathbf{S}$.

The constraints for avoiding a rectangular obstacle were developed in Ref. [5] and can be written as

$$\forall t \in [1 \ldots N_T] \; \forall p \in [1 \ldots N_V] \; \forall j \in [1 \ldots N_Z]$$
$$\begin{array}{lrcl} & x_{\mathrm{tp}} - Z_{\mathrm{j3}} & \geq & -Rc_{\mathrm{jpt1}} \\ \text{and} & Z_{\mathrm{j1}} - x_{\mathrm{tp}} & \geq & -Rc_{\mathrm{jpt2}} \\ \text{and} & y_{\mathrm{tp}} - Z_{\mathrm{j4}} & \geq & -Rc_{\mathrm{jpt3}} \\ \text{and} & Z_{\mathrm{j2}} - y_{\mathrm{tp}} & \geq & -Rc_{\mathrm{jpt4}} \\ \text{and} & \sum_{z=1}^{4} c_{\mathrm{jptz}} & \leq & 3 \end{array} \quad (6)$$

where c_{jptz} are a set of binary decision variables and R is a positive number that is much larger than any position to be encountered in the problem. If $c_{\mathrm{jptz}} = 0$, the vehicle p is clear of the no-fly-zone j in the z^{th} direction (of the four directions $+X$, $-X$, $+Y$, $-Y$) at the t^{th} time step. If $c_{\mathrm{jpkz}} = 1$, the constraint is relaxed. The final inequality ensures that no more than three of the constraints are relaxed at any time-step, so the vehicle must be clear of the obstacle in at least one direction.

Assignment Logic in the Combined Method

The set of constraints to detect if a vehicle visits a waypoint can be written as

$$\forall p \in [1 \ldots N_V] \; \forall t \in [1 \ldots N_T] \; \forall i \in [1 \ldots N_W]$$
$$\begin{array}{lrcl} & x_{\mathrm{tp}} - W_{\mathrm{i1}} & \leq & R(1 - b_{\mathrm{ipt}}) \\ \text{and} & x_{\mathrm{tp}} - W_{\mathrm{i1}} & \geq & -R(1 - b_{\mathrm{ipt}}) \\ \text{and} & y_{\mathrm{tp}} - W_{\mathrm{i2}} & \leq & R(1 - b_{\mathrm{ipt}}) \\ \text{and} & y_{\mathrm{tp}} - W_{\mathrm{i2}} & \geq & -R(1 - b_{\mathrm{ipt}}) \end{array} \quad (7)$$

where b_{ipt} is a binary decision variable, $\mathbf{W}$ is the waypoint location matrix, and R is the same large, positive number used in Eq. 6. It can be seen that $b_{\mathrm{ipt}} = 1$ implies that vehicle p visits waypoint i at time-step t. This binary variable can then be used in logical constraints for the assignment. This formulation can easily be relaxed so that a vehicle "visits" a waypoint if it passes within a specified distance of that point. The following logical constraint en-

forces that each waypoint is visited exactly once by a vehicle with suitable capabilities.

$$\forall i \in [1 \dots N_W] \ \sum_{t=1}^{N_T} \sum_{p=1}^{N_V} K_{\mathrm{pi}} b_{\mathrm{ipt}} = 1 \qquad (8)$$

Time dependencies are enforced by the following constraint

$$\forall k \in [1 \dots N_C] \ \sum_{i=1}^{N_W} \Delta_{\mathrm{ki}} \sum_{t=1}^{N_T} \sum_{p=1}^{N_V} t \ b_{\mathrm{ipt}} \geq t_{D_{\mathrm{k}}} \qquad (9)$$

in which the summations $\sum_{t=1}^{N_T} \sum_{p=1}^{N_V} t \ b_{\mathrm{ipt}}$ extract the time of visit for the i^{th} waypoint. Note that the time is in units of time-steps, since the index t is used as the measure of time at each step. This is equivalent to the form shown in Eq. 1.

Cost Function for the Combined Method

This section develops the cost function for the combined method. The primary aim of the current work is to minimize the mission completion time. Small penalty weightings are included to help the numerical conditioning and accelerate the solution process. The first step is to extract the flight completion time t_{p} for the p^{th} vehicle, which is the time it visits its last waypoint

$$\forall p \in [1 \dots N_V] \ \forall i \in [1 \dots N_W] \ t_{\mathrm{p}} \geq \sum_{t=1}^{N_T} t \ b_{\mathrm{ipt}} \qquad (10)$$

A similar set of constraints finds the overall mission completion time $\bar{t}$

$$\forall p \in [1 \dots N_V] \ \bar{t} \geq t_{\mathrm{p}} \qquad (11)$$

The complete cost function is

$$J_1 = \min_{\mathbf{s},\mathbf{f},\mathbf{b},\mathbf{c}} \bar{t} + \epsilon_1 \sum_{p=1}^{N_V} [t_{\mathrm{p}} + \epsilon_2 \sum_{t=0}^{N_T - 1} (|f_{x_{\mathrm{tp}}}| + |f_{y_{\mathrm{tp}}}|)] \qquad (12)$$

where the decision variables are the forces $\mathbf{f}$, the state vectors $\mathbf{s}$, and the binary variables $\mathbf{b}$ and $\mathbf{c}$, for waypoint visit and no-fly-zone logic respectively.

The weighting factors ϵ_1 and ϵ_2 are small positive numbers that are included to help the solution process.[11] The first weighting ensures that the minimum time path is chosen for all aircraft. If it were omitted, only the aircraft that finished last would be explicitly minimized, and those finishing earlier could select multiple paths without affecting the cost. The force weighting has a similar role for each aircraft: many paths may lead to flight completion at the same discrete time-step, but the additional force weighting means there is a unique solution for each vehicle. Together, the weightings force the problem to have a unique solution. Experience has shown that this greatly reduces the solution time for the problem.

The optimization problems shown here can be easily translated into the AMPL language.[7] An AMPL model file contains the constraint forms for all cases, while the data is written to an AMPL data file by a MATLAB script. CPLEX optimization software is used to solve the problem.[13] Using MATLAB and AMPL scripts, the entire path-planning problem is invoked by a single command.

Example of Combined Method

This section demonstrates the effect of vehicle assignment constraints on a simple scenario. Fig. 2 shows the designed trajectories for two vehicles visiting four waypoints. Both vehicles have the capability to visit all the waypoints, so every entry of the capability matrix is 1. There are no timing dependencies. As expected, each vehicle travels in a nearly straight path to the two nearest waypoints. In Fig. 3, the scenario has been changed by removing the capability of vehicle 1 to perform the task at waypoint B, as it does in the solution of the first problem. Vehicle 2 is now the only vehicle with that capability, so it is required to visit point B. It would be feasible for vehicle 2 to follow the same trajectory as in the previous example, then visit point B at the end. However, by assigning vehicle 1 to point D, vehicle 2 can proceed straight from C to B, leading to an earlier mission completion.

Note that, in the plan shown in Fig. 3, vehicle 1 visits point A then point D. For the third problem, the scenario was modified to require that point D must be visited *before* point A, excluding the previous solution. In the optimal solution shown in Fig. 4, vehicle 2 goes almost directly to point D. Point C is on the way so it is visited in passing. Vehicle 1 moves slowly in order to arrive at point A just after vehicle 2 arrives at D. Finally, vehicle 2 is still required to visit point B due to the lack of capability of vehicle 1. In the final variation on this problem, an obstacle is added to block the path from C to D taken by ve-

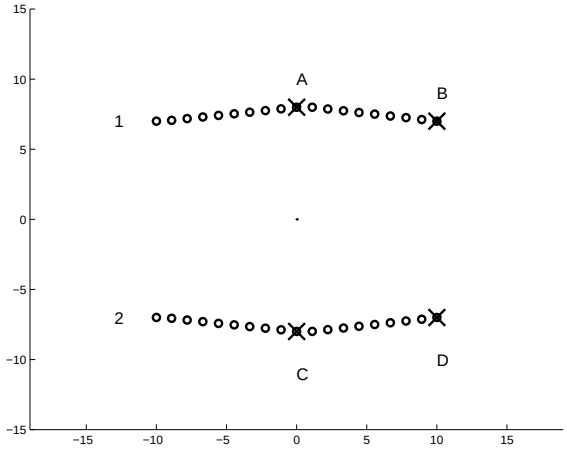

Fig. 2: Centralized assignment for two vehicles among four waypoints. Both vehicles may visit all four waypoints.

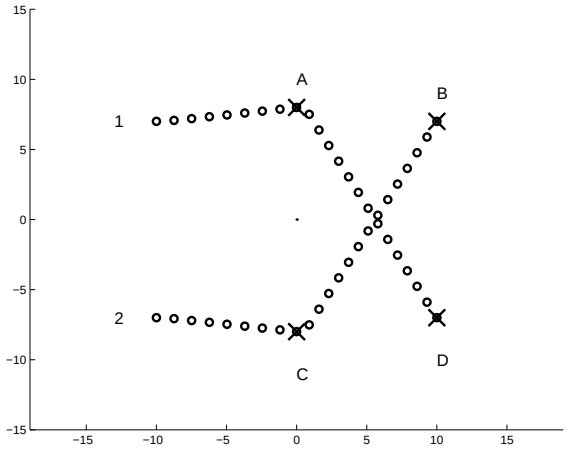

Fig. 3: Centralized assignment with full capabilities *except* vehicle 1 cannot visit waypoint B

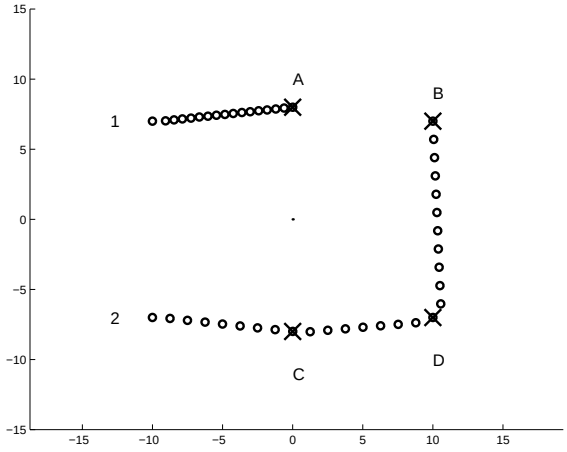

Fig. 4: Centralized assignment problem from Fig. 3 with extra time dependency: waypoint D must be visited before A.

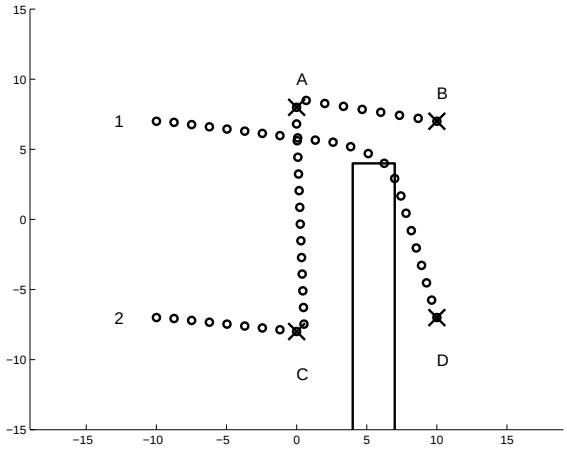

Fig. 5: Centralized assignment problem from Fig. 4 with additional obstacle.

hicle 2 in the previous design. Fig. 5 shows the new assignment and trajectories. It is still necessary that vehicle 2 visits point B, due to the lack of capability of vehicle 1, and that point D must be visited before point A. Therefore, vehicle 1 is sent directly to point D, while point A is visited by vehicle 2 on its way from C to B.

These simple examples show that the constraints developed in this section have the intended effects. It can be seen that the design in each case satisfies the mission requirements. The examples also demonstrate the coupling between the assignment and path-planning problems. Adding a path constraint, such as an obstacle, leads to changes in the assignments. Similarly, changing assignment constraints, such as capabilities, leads to different trajectories. These lessons are important in the development of any approximate approach for real-time implementation and illustrate the importance of comparison with the globally optimizing benchmark method.

APPROXIMATE METHOD

This section describes an approximate method[8] for solving the UAV coordination and control problem. The approximate method offers much faster solution times, but could yield sub-optimal results. The cost function used in the approximate method is similar to that of the combined method, minimizing the overall mission completion time, plus a small weighting on the individual vehicle finishing times. However, this method estimates finishing times using straight line path approximations.[9] Given the

estimated finishing times for each UAV t_p, this cost can be evaluated as

$$\bar{t} \;=\; \max_{p} \; t_p \tag{13}$$

$$J_2 \;=\; \min_{\bar{t},\mathbf{t}} \bar{t} + \frac{\alpha}{N_V} \sum_{p=1}^{N_V} t_p \tag{14}$$

where $\alpha \ll 1$ adds a small extra penalty on the average completion time, similar to Eq. 12.

Overview of the Approximate Algorithm

The five main steps in the algorithm are shown in Fig. 6. This section gives a brief description of each step. Some steps are discussed further in later sections, and the details are in Ref. [8].

1. Enumerate a list of all feasible assignments and orderings, subject to vehicle capabilities. Some assignments, such as those involving a large number of tasks for a single vehicle, can also be eliminated (pruned) at this step. These are predicted with confidence to have long completion times and will therefore be unfavorable.
2. Use the straight-line approximation method to estimate finishing times for each assignment and ordering.
3. Using these approximate finishing times, assign tasks to achieve the minimum of the approximate cost. This is performed by a MILP optimization.
4. Use fixed-assignment MILP methods[4, 5] to plan detailed trajectories for each vehicle. These account for dynamics and inter-vehicle collision avoidance. In some cases, this step may indicate that a revised assignment is required, and an iteration back to step 3 is required.

Steps 1–3 are discussed further in the following subsection. Step 4 is similar to techniques presented in the first sections of the paper and is not considered further. Examples are then shown to validate this method.

Permutation Costs in Approximate Algorithm

This section presents the process for developing a list of feasible subtask assignments, finding approximate finishing time information for each subtask assignment, and pruning the list. This algorithm accepts the aircraft starting states **S**, capabilities **K**, obstacle vertex positions **Z**, and waypoint positions **W**.

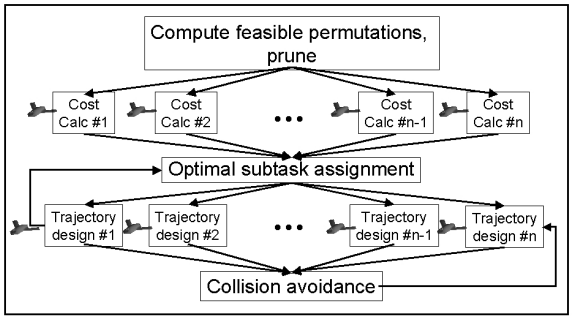

Fig. 6: Steps in the distributed task assignment and trajectory planning algorithm

The algorithm also accepts an upper bound $n_{\max}$, which specifies the maximum number of waypoints that a UAV may visit on its mission. Assignments involving many waypoints are likely to be unfavorable due to their long completion times. Therefore, they are "pruned" from the assignment options at this stage to reduce redundant computation. The algorithm then finds for each UAV the shortest permutation of every combination of fewer than $n_{\max}$ waypoints.

The steps in this algorithm are depicted in Figs. 7–9, in which a fleet of UAVs (shown with $\circ$) must visit a set of waypoints (shown with $\times$). First, the visibility graph between the UAV starting positions, waypoints, and obstacle vertices is found. The visibility graph is shown in Fig. 7 as grey lines. Next, this graph is searched to find the shortest paths between all pairs of waypoints, and between the starting position of each UAV and all waypoints (Fig. 7 shows the result for UAV 6 with black lines).

Using this graph, the optimal path for each UAV to visit each combination of waypoints is found. Only those points for which the UAV has the necessary capabilities are considered. Fig. 8 shows the shortest path for UAV 6 to visit a particular three points. This is found by forming the paths for the UAV to visit those points in all possible orders and choosing the shortest. This result is referred to as a *permutation*. It includes an ordered list of waypoints and a UAV. Fig. 9 shows the permutations for the same three waypoints and each of the vehicles. Note that, as expected, the best ordering of the visits is not the same for all vehicles.

The algorithm produces five matrices whose j^{th} columns, taken together, fully describe one *permutation*. These are the row vector **u**, whose elements u_j identify which UAV participates in the j^{th} per-

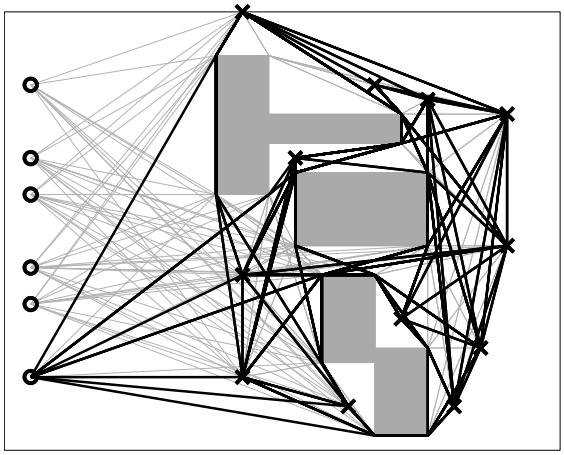

Fig. 7: Visibility graph and shortest paths between UAV 6, all waypoints

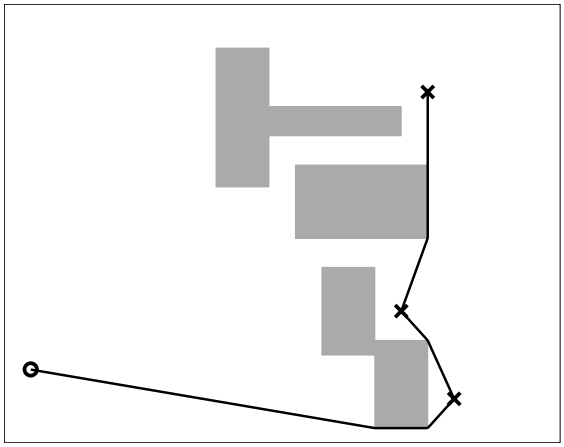

Fig. 8: Shortest path for UAV 6 over one combination of waypoints

mutation; the ordered lists stored in $\mathbf{P}$, whose P_{ij} entry identifies the i^{th} waypoint visited by permutation j; $\mathbf{V}$, whose V_{ij} entry is 1 if waypoint i is visited by permutation j and 0 if not (this matrix is used in the assignment logic in the next subsection); $\mathbf{T}$, whose T_{ij} entry is the time at which waypoint i is visited by permutation j, and 0 if waypoint i is not visited; and the cost matrix $\mathbf{C}$, whose element C_j is the time at which permutation j is completed. This procedure is described in detail in Algorithm 1.

In this algorithm, finding the shortest distance between a set of points is performed by finding the visibility graph between the points and vertices of obstacles, then applying an appropriate shortest path algorithm such as the Bellman-Ford All-Pairs Shortest Path algorithm.[12] Note that the iterations through the "**for loop**" between lines 2 and 23 of Algorithm 1 are independent, and can each be distributed to parallel processors. The corresponding matrices from each processor can then be combined and passed onto the next stage in the algorithm, the task assignment problem. Once the tasks assigned to each vehicle are known, trajectories that visit the required waypoints can be designed on distributed platforms. The distributed form of the approximate algorithm is shown in Fig. 6. The parallel platforms could be processors onboard the UAVs, or could be several computers at a centralized command and control facility.

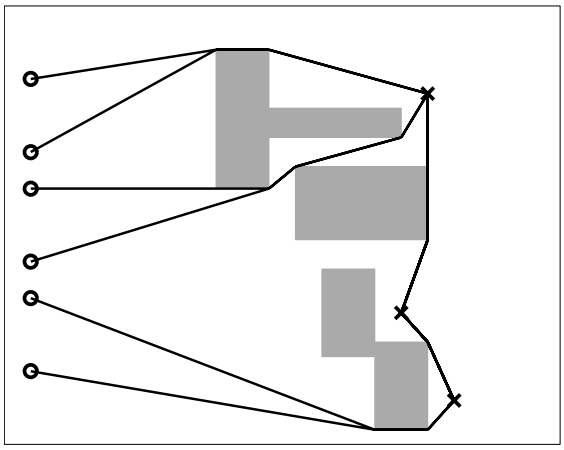

Fig. 9: Shortest paths for all UAVs over same combination of waypoints

Task Allocation in Approximate Algorithm

The previous subsection outlined a method for determining the approximate costs for a UAV to visit a set of waypoints. This section presents a method of allocating the waypoints to each UAV based on these costs and other constraints. The base of the task allocation problem is formulated as a Multidimensional Multiple Choice Knapsack Problem (MMKP).[10] The "knapsack" in this case is the complete mission plan. The "multi-dimensional" aspect refers to the N_W waypoints (N_W-dimensions) that can be grouped in N_M different permutations (elements). The list of waypoints visited in a permutation make up the weight of that permutation. The "multiple-choice" comes from choosing which waypoints to assign to each of the N_V different UAVs (sets). The objective is to assign one permutation (element) to each vehicle (set) that is combined into the mission plan (knapsack), such that the cost of the mission (knapsack) is minimized and the waypoints visited (weight) meets the constraints for each dimension N_W.

The standard MMKP formulation forms the base

Find shortest distances between all waypoint pairs (i, j) as $D(i, j)$ using $\mathbf{S}_0$, $\mathbf{Z}$, and $\mathbf{W}$.
for all UAVs p **do**
  Find shortest distances $d(i)$ between start points of UAV p, and all waypoints i using $\mathbf{S}_0$, $\mathbf{Z}$, and $\mathbf{W}$.
  for all combinations of n_C waypoints that p is capable of visiting, $n_C = 1 \dots n_{\max}$ **do**
    for $j = 1 \dots n_C P n_C$ **do**
      Make next unique permutation $P'_{1j} \dots P'_{n_C j}$ of waypoints in the combination
      $C'_j = \frac{d(P'_{1j})}{v_{\max,p}}$
      $T'_{P'_{1j}j} = C'_j$
      for $i = 2 \dots n_C$ **do**
        if $C'_j > t_{\max}$ **then**
          go to next permutation
        end if
        $C'_j \leftarrow C'_j + \frac{D(P'_{(i-1)j}, P'_{ij})}{v_{\max,p}}$
        $T'_{P'_{ij}j} = C'_j$
      end for
    end for
    Append p to $\mathbf{u}$
    Append a column to $\mathbf{V}$, whose i^{th} element is 1 if waypoint i is visited, 0 if not.
    $j_{\min} = \text{minarg}_j\ C'_j$
    Append column $j_{\min}$ of $\mathbf{T}'$ to $\mathbf{T}$
    Append column $j_{\min}$ of $\mathbf{P}'$ to $\mathbf{P}$
  end for
end for

Algorithm 1: Algorithm for finding shortest paths between waypoints

of the task allocation algorithm. Modifications are made to the basic problem statement to include additional cost considerations and constraints. The solution to the MMKP selects a permutation for each vehicle. The cost in Eq. 14 is a weighted combination of the sum of the individual mission times and the total mission time. In the assignment problem, this is given by

$$J_3 = \min_{\bar{t},\mathbf{x}} \bar{t} + \frac{\alpha}{N_V} \sum_{i=j}^{N_M} C_j x_j \tag{15}$$

where N_M is the total number of permutations for which the costs have been evaluated. The binary decision variable $x_j = 1$ if permutation j is selected, and 0 otherwise. A new continuous variable, $\bar{t}$, is introduced to represent the total mission time. The following constraint is added

$$\sum_{j=1}^{N_M} C_j x_j \leq \bar{t} \tag{16}$$

This ensures that $\bar{t}$ is the maximum finishing time of all the permuations, equivalent to the overall finishing time. The other constraints are

$$\begin{aligned} \sum_{j=1}^{N_M} V_{ij} x_j &= 1 \\ \sum_{j=N_p}^{N_{p+1}-1} x_j &= 1 \end{aligned} \tag{17}$$

where the permutations of vehicle p are numbered N_p to $N_{p+1} - 1$. The first constraint enforces that waypoint i is visited exactly once. The second constraint prevents more than one permutation being assigned to each vehicle. This formulation differs from the standard MMKP problem by the inclusion of the overall mission time in the cost function.

The problem is now a mixed-integer linear programming problem that can be solved using commercially available software such as CPLEX.[13] The ordered set of waypoints for each vehicle is then determined from the column in $\mathbf{P}$ corresponding to the selected permutation for each vehicle. The solution to the task allocation problem is a set of ordered sequences of waypoints for each vehicle which ensure that each waypoint is visited the correct number of times while minimizing the desired cost (mission completion time). Additional constraint formulations for heterogenous vehicles and timing constraints are available in Ref. [8].

Examples using Approximate Algorithm

This section shows the application of the approximate method to the example scenarios shown in Figs. 2–5. The results are shown in Figs. 10–13. These show the assignments and corresponding path approximations. In each case, the assignment and route are the same as those found by the combined optimization method. In the third and fourth cases, in which a timing constraint was added, the approximate method has specified a delayed starting time for one of the vehicles. In the same circumstances, the combined planner used the same start time for both vehicles but made one move slower. These differences arise from the different dynamics models

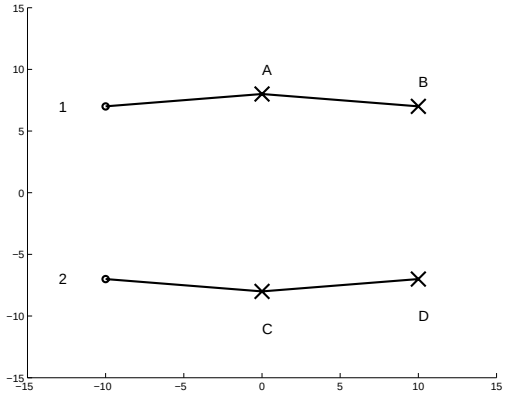

Fig. 10: Solutions to example problem in Fig. 2 using the approximate method.

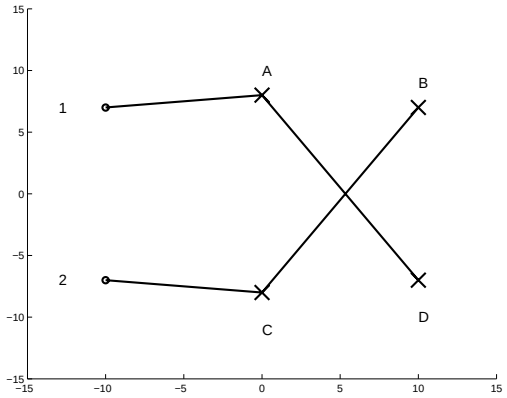

Fig. 11: Solutions to example problem in Fig. 3 using the approximate method.

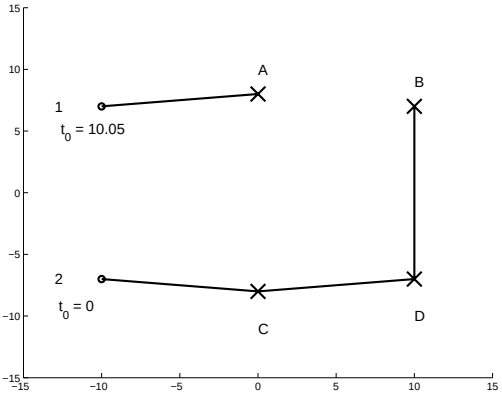

Fig. 12: Solutions to example problem in Fig. 4 using the approximate method.

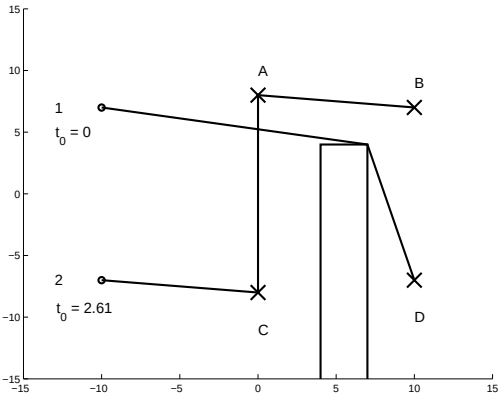

Fig. 13: Solutions to example problem in Fig. 5 using the approximate method.

used in the two methods, but lead to equivalent results.

PERFORMANCE COMPARISON

This section compares the performance of the two methods (combined and approximate). The combined method provides a benchmark evaluation of the globally-optimal solution to the problem by solving a single, very complex optimization. The comparison evaluates what computational savings are available with the approximate method, and what cost is incurred in the resulting solution. The problems were solved using CPLEX software running on a 2.2GHz PC with 512MB RAM.

First, a relatively small example is presented, involving two vehicles, four waypoints and two obstacles. This was solved by both methods and the solutions are presented for comparison. Computation and cost results from many similar, randomly-generated problems are then compared for further evaluation of the approximate method. The final problem is much larger and is beyond practical solution using the combined method. Thus the smaller examples illustrate the performance of the method and the large example shows its potential for larger problems.

Smaller Evaluation Problem

Fig. 14 shows the designed trajectories for the smaller example, found using the combined method. The vehicle capabilities are shown in Table 1. There are 24 possible assignments for this problem, not counting the different orderings of each. No time dependencies are included. The computation took just over 25 seconds. The total mission time for the designed solution is 18 time-steps.

Fig. 15 shows the solution for the same problem using the approximate algorithm. Comparing the figures, it is evident that both methods generate the same result, indicating that the approximate method has successfully found the globally-optimal solution. The approximate method took under five seconds

Table 1: Vehicle Capabilities in Comparison Problem

Waypoint	Vehicle		
	1	2	3
A	X	X	
B	X	X	X
C		X	X
D	X	X	

Table 2: Computation Times for Random Comparison Problems

Method	Computation Time (s)	
	Mean	Max.
Combined	165	577
Approximate		
Assignment	0.54	0.61
Trajectories	3.19	6.51

to compute the result. While the cost computation could be distributed, it was performed sequentially on a single PC in this example.

Random Comparison Problems

Table 2 compares the computation times for random problems, similar to the comparison problem in Fig. 14 but with randomly-generated targets, capabilities, and obstacle positions. Of 50 problems attempted, the combined method found the globally-optimal solutions for 37 within ten minutes of computation each. These same problems were solved by the approximate algorithm, including the step using MILP with fixed assignments to find the dynamically-feasible trajectories for each vehicle. The table divides the results between "assignment", including cost approximation and the assignment optimization, and "trajectories", representing the time to find dynamically-feasible paths for all three vehicles. The results show that the approximate method, including all steps of the algorithm, was over 40 times faster than the combined method on average. Furthermore, in all but one of the 37 problems, the mission completion time found by the approximate algorithm was identical to the global optimal found using the combined method. The remaining problem took one time-step more than the optimal solution. These results show that the approximate algorithm offers substantial computation improvements with a negligible degradation in performance.

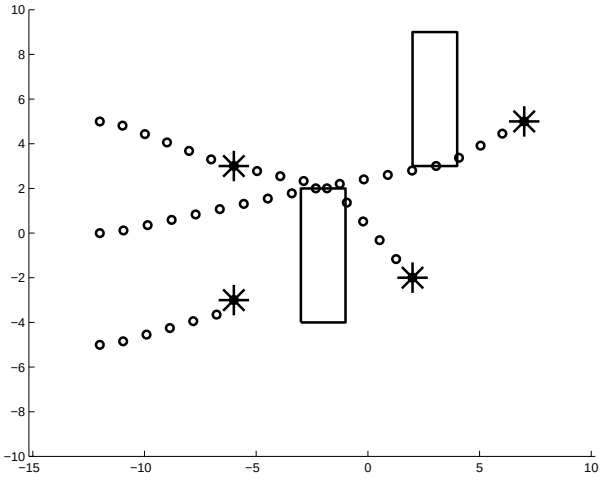

Fig. 14: Solution of the comparison problem by centralized method

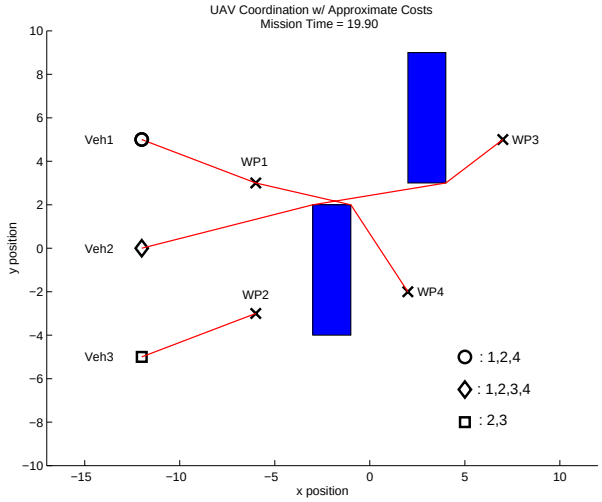

Fig. 15: Solution of the comparison problem by approximate method

Large Evaluation Problem

Fig. 16 shows the design for a large fleet assignment problem found using the approximate method. With six vehicles, twelve waypoints, and numerous no-fly-zones (obstacles), this problem is too large to be solved by the combined method in any practical computation time. There are just under 38 million assignment combinations, each with various orderings. However, the approximate method found the solution in 27 seconds. Note that the capabilities of the various types of vehicles are denoted by the table at the top-left of Fig. 16.

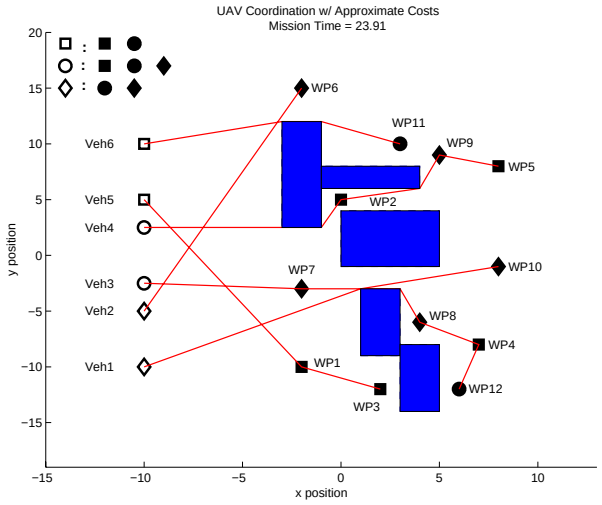

Fig. 16: Solution of large problem by approximate method

CONCLUSIONS

This paper presents two solutions to the coupled problems of multi-task allocation and trajectory planning for a fleet of UAVs. Both approaches minimize the mission completion time and account for differing UAV capabilities and types of waypoints, timing constraints, and no-fly-zones. One approach (the combined method) solves these coupled optimization problems as a single mixed-integer linear program. This method is guaranteed to find the globally-optimal solution to the problem, but is computationally intensive. The second method employs an approximation to the initial path-planning, allowing rapid computation of many different trajectory costs. This enables the assignment and trajectory problems to be decoupled and partially distributed, offering much faster computation. The results of various example problems demonstrate that the approximate method offers significant computational savings over the combined method with negligible degradation to the solutions found. In addition, the greater computational efficiency of the approximate method allows it to be applied to very large UAV fleet assignment problems.

ACKNOWLEDGMENTS

Research funded in part by AFOSR grant # F49620-01-1-0453.

REFERENCES

[1] P. R. Chandler, S. Rasmussen, "UAV Cooperative Path-Planning," proc. of the *AIAA GNC,* Aug. 14–17, Paper No. AIAA-2000-4370, 2000.

[2] C. Schumaker, P. Chandler, S. Rasmussen, "Task Allocation for Wide Area Search Munitions via Network Flow Optimization" proceedings of the *AIAA GNC,* Montreal, Canada, Aug. 6-9, 2001.

[3] P. Chandler, M. Pachter, "Hierarchical Control for Autonomous Teams" proceedings of the *AIAA GNC,* Montreal, Canada, Aug. 6-9, 2001.

[4] A. G. Richards, J. P. How, "Aircraft Trajectory Planning with Collision Avoidance using Mixed Integer Linear Programming," *ACC,* May 2002.

[5] T. Schouwenaars, B. DeMoor, E. Feron and J. How, "Mixed integer programming for safe multi-vehicle cooperative path planning," *ECC*, Porto, Portugal, September 2001.

[6] A. Bemporad and M. Morari, "Control of Systems Integrating Logic, Dynamics, and Constraints," in *Automatica,* Pergamon / Elsevier Science, New York NY, Vol. 35, pp. 407–427, 1999.

[7] R. Fourer, D. M. Gay, and B. W. Kernighar, *AMPL, A modeling language for mathematical programming,* The Scientific Press, 1993.

[8] J. S. Bellingham, M. J. Tillerson, A. G. Richards, J. P. How, "Multi-Task Assignment and Path Planning for Cooperating UAVs," *Conference on Cooperative Control and Optimization,* Nov. 2001.

[9] J. S. Bellingham, A. G. Richards and J. P. How, "Receding Horizon Control of Autonomous Aerial Vehicles", *ACC,* Anchorage, AK, May 2002.

[10] M. Moser, D. Jokanovic, N. Shiratori, " An Algorithm for the Multidimensional Multiple-Choice Knapsack Problem" *IEICE Trans. Fundamentals,* Vol. E80-A, No.3 March 1997.

[11] H. P. Rothwangl, "Numerical Synthesis of the Time Optimal Nonlinear State Controller via Mixed Integer Programming," *ACC,* 2001.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to Algorithms,* MIT Press, 1990.

[13] *ILOG CPLEX User's guide*, ILOG, 1999.