Co-ordination and Control of Multiple UAVs

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

Space Systems Laboratory

Massachusetts Institute of Technology

Abstract

This paper addresses the key problems of autonomous task allocation and trajectory planning for a fleet of UAVs. Task assignment must account for differing UAV capabilities and types of waypoints. Timing constraints, no-fly-zones, and different vehicle dynamics must also be included in the trajectory planning. The overall objective is to minimize the mission completion time for the fleet. The resulting optimization is a highly-coupled combination of the task assignment problem, involving logical constraints, and the trajectory design problem, involving non-convex obstacle and collision avoidance and dynamics constraints. Two methods are presented for solving this problem. One expresses the entire problem as a single mixed-integer linear program, which can then be solved using commercially available software. This method is guaranteed to find the globally-optimal solution to the problem, but is computationally intensive. It therefore offers a benchmark against which heuristic methods can be evaluated. The second method employs an approximation for the trajectory-planning part, allowing 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 two methods are compared to evaluate the performance degradation incurred due to the approximation. Several examples are presented to show that, since the approximate method finds the optimized assignments much faster, it can be applied to very large fleet assignment problems.

[*]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**
MIT 37-391, 77 Massachusetts Ave., Cambridge, MA 02139

1 Introduction

The capabilities and roles of Unmanned Aerial Vehicles (UAVs) are evolving, and require new concepts for their control [1]. In particular, new methods in planning and execution are required to coordinate the operation of a fleet of UAVs. Today's UAVs typically require several operators for control, but future UAVs will be designed to make their own tactical decisions autonomously and will be integrated into teams that coordinate to achieve high-level goals, thereby allowing one operator to control an entire fleet [1]. With these goals in mind, this paper presents results on the guidance and control of fleets of cooperating UAVs. A key challenge for these systems is to develop an overall control system architecture 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.

The focus of this paper is fleet coordination, which includes team composition and goal assignment, resource allocation, and trajectory optimization. For many vehicles, obstacles, and targets, fleet coordination is a very complicated optimization problem [1, 2, 3, 8]. A general problem is posed involving heterogenous vehicles and waypoints subject to obstacle avoidance, vehicle dynamics, and timing constraints. 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. The timing constraints result from the fact that some tasks must be performed in a specific order. For example, a typical sequence in a Suppression of Enemy Air Defenses (SEAD) mission would require UAVs to fly to a specified waypoint to perform reconnaissance, then electronic suppression, followed by ordnance delivery, and finally, damage assessment. The fleet coordination problem is to allocate the vehicles with the appropriate capabilities (*e.g.,* sensors, payloads) to the waypoint and ensure that they can arrive with the correct temporal separation. A typical scenario is illustrated in Fig. 1. Five vehicles of three types, 1, 2 and 3, are to be assigned among six waypoints of three types, A, B and C. The vehicles must avoid the hatched regions, or "no-fly-zones". Table 1 shows a typical set of capabilities for the fleet shown in Fig. 1.

The problem is to design a trajectory for each vehicle such that each waypoint is visited by a suitably capable vehicle, at times permitted by the temporal 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 costs could be calculated for all permutations of visits, the assignment could be done quite easily, but this cost calculation is impractical due to the extremely large number of possible assignments. Even for the relatively small problem in Fig. 1, there are 1296 valid assignment *combinations.* For each of these, the order of the visits must still be chosen, so the number of possible *permutations* is even higher. Additionally, the

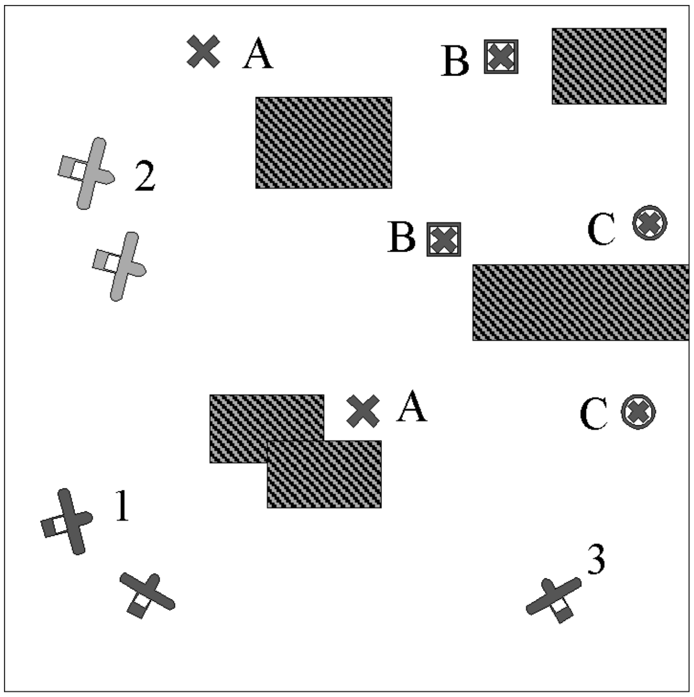

Figure 1: Example Scenario

numbers of permutations and combinations grow at a non-polynomial rate with the number of waypoints.

This paper presents two methods for solving this problem. The first captures the coupling of the problems by combining both assignment and trajectory design into a single *mixed-integer linear program* (MILP) optimization. While previous work treats the two problems separately [2, 3], the combination of the two that is presented here is guaranteed to find the globally optimal solution. This provides a benchmark against which all other methods, which may be faster but based on heuristics, can be evaluated. However, the computation can be intensive and, using current optimization techniques, the method is not well-suited for real-time operation. The combined optimization method uses a linearization 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 assignment of vehicles to waypoints, subject to the constraints described in the following.

The second method presented simplifies the coupling between the assignment and trajectory design problems by calculating and communicating only the key information that connects the two [8]. It estimates the cost of various trajectory options by using straight line path approximations. 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 vertices of obstacle polygons, and the waypoints. The assignment is then

Table 1: Example Capabilities

Waypoint Type	UAV Type 1	2	3
A		X	X
B	X		X
C	X	X	

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. This simplification of the coupling presents a more computationally tractable approach, and maintains the advantages over previous methods that treat the two problems in isolation.

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

2 Encoding the Problem

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. No fly zones are modeled as rectangular regions 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 $\mathbf{t}$ contains the visiting times for each waypoint, the constraint is 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

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

3 Combined Assignment and Trajectory Design

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 pre-assigned list of waypoints can be written and solved as a MILP [4]. Here, the waypoint selection constraints will be modified to include the assignment and capability constraints for a group of heterogenous vehicles. The effect of the assignment constraints are shown in a series of examples in Section 3.4.

3.1 Dynamics in the Combined Formulation

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

The maximum speed v_{max_p} is enforced by an approximation to a circular region in the velocity plane

$$\begin{gathered}\forall t \in [1 \ldots N_T] \, \forall p \in [1 \ldots N_V] \, \forall m \in [1 \ldots N_C] \\ \dot{x}_{\text{tp}} \sin\left(\frac{2\pi m}{N_C}\right) + \dot{y}_{\text{tp}} \cos\left(\frac{2\pi m}{N_C}\right) \leq v_{\text{max}_\text{p}}\end{gathered} \tag{2}$$

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

$$\begin{gathered}\forall t \in [0 \ldots N_T - 1] \, \forall p \in [1 \ldots N_V] \, \forall m \in [1 \ldots N_C] \\ f_{\text{x}_\text{tp}} \sin\left(\frac{2\pi m}{N_C}\right) + f_{\text{y}_\text{tp}} \cos\left(\frac{2\pi m}{N_C}\right) \leq f_{\text{p}}\end{gathered} \tag{3}$$

where f_{p} is related to the maximum turn rate by

$$\omega_{\mathrm{p}} = \frac{f_{\mathrm{p}}}{v_{\mathrm{max_p}}} \tag{4}$$

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

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

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 described in Section 2, $\mathbf{s}_{\mathrm{0p}} = \mathbf{S}_{\mathrm{p}}$, where $\mathbf{S}_{\mathrm{p}}$ is the p^{th} row of the initial state matrix $\mathbf{S}$.

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

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

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 $c_{jptz} = 0$, 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 $c_{jpkz} = 1$, the constraint is relaxed. The final inequality ensures that no more than three of the constraints are relaxed at any time step, so there must always be safe separation in at least one direction.

3.2 Assignment Logic in the Combined Formulation

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

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

where b_{ipt} is a binary decision variable, $\mathbf{W}$ is the waypoint location matrix from Section 2, and R is the same large, positive number used in Eq. 6. It can be seen that $b_{\text{ipt}} = 1$ 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 constraint enforces that each waypoint is visited exactly once by a vehicle with suitable capabilities.

$$\forall i \in [1 \dots N_W] \; \sum_{t=1}^{N_T} \sum_{p=1}^{N_V} K_{\text{pi}} b_{\text{ipt}} = 1 \tag{8}$$

Time dependencies are enforced by the following constraint

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

in which the summations $\sum_{t=1}^{N_T} \sum_{p=1}^{N_V} t \; b_{\text{ipt}}$ 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.

3.3 Cost Function for the Combined Method

This section develops the cost function for the centralized method. The primary aim 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 at which it visits its last waypoint

$$\forall p \in [1 \dots N_V] \; \forall i \in [1 \dots N_W] \; t_{\text{p}} \geq \sum_{t=1}^{N_T} t \; b_{\text{ipt}} \tag{10}$$

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

$$\forall p \in [1 \dots N_V] \; \bar{t} \geq t_{\text{p}} \tag{11}$$

The complete cost function is

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

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

The weighting factors ϵ_1 and ϵ_2 are small positive numbers and are included to help the solution process. 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 modeling 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 [11]. A series of scripts in Matlab and AMPL allow the entire path-planning problem to be invoked by a single command.

3.4 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 all 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. However, for the third problem, the scenario was modified to require that point D must be visited *before* point A. Clearly the previous trajectory is no longer feasible. 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 incapability of vehicle 1. In the final variation on this problem, an obstacle is added to block the path from C to D taken by vehicle 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.

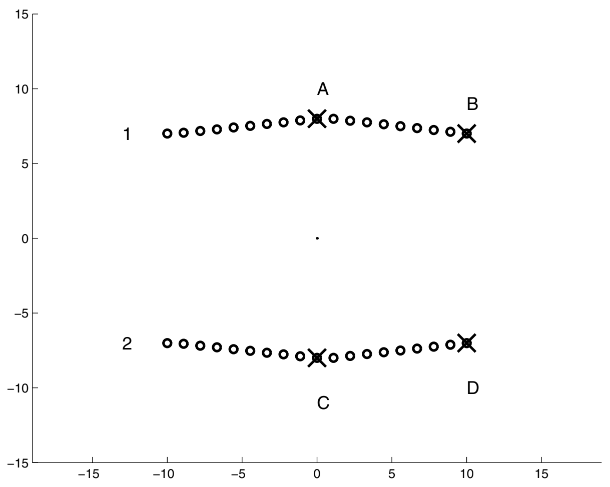

Fig. 2: Centralized assignment for two vehicles among four waypoints. Both vehicles may visit all four waypoints.

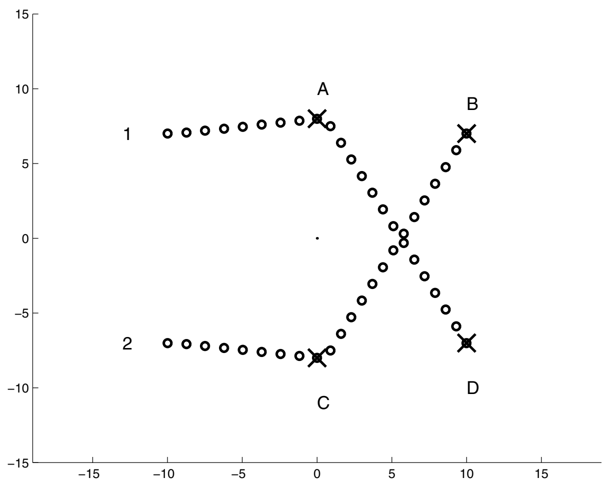

Fig. 3: Centralized assignment with full capabilities *except* vehicle 1 cannot visit waypoint B

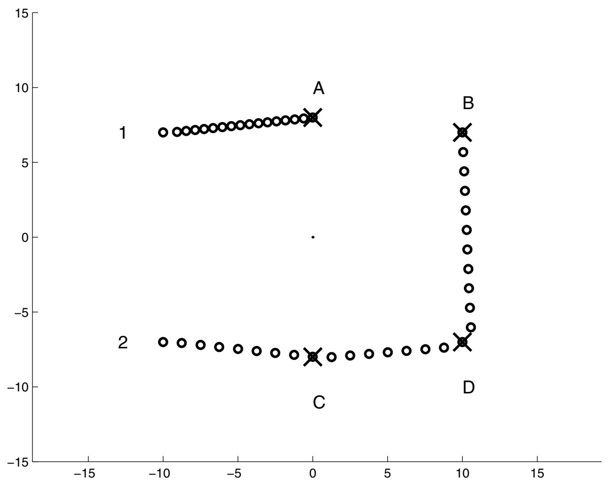

Fig. 4: Centralized assignment problem from Fig. 3 with extra time dependency: waypoint D must be visited before A.

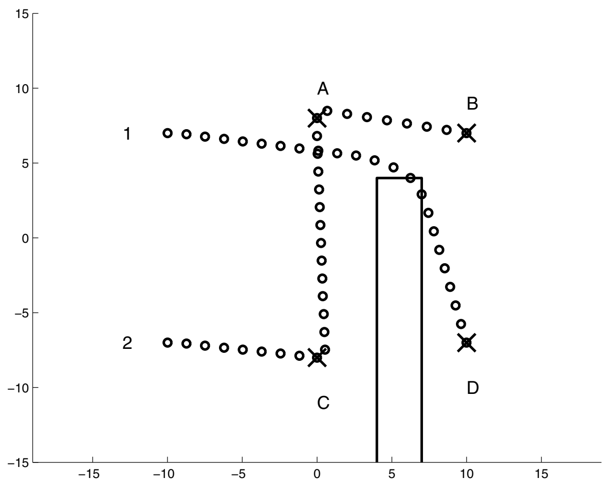

Fig. 5: Centralized assignment problem from Fig. 4 with additional obstacle.

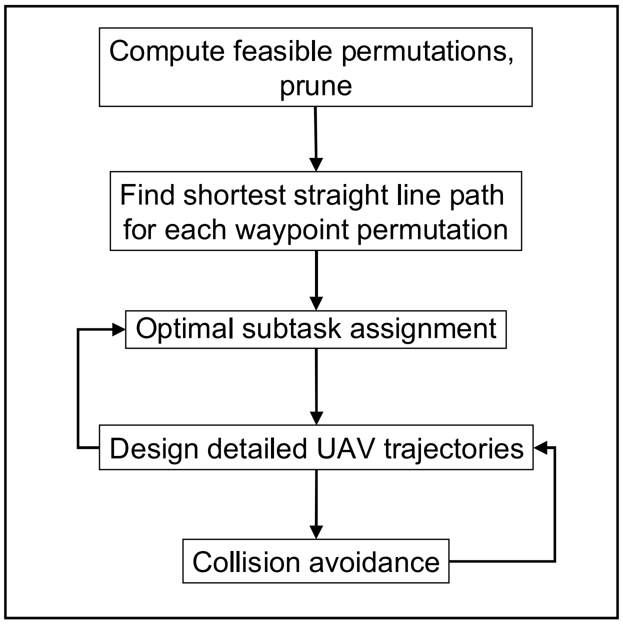
Compute feasible permutations, prune
Find shortest straight line path for each waypoint permutation
Optimal subtask assignment
Design detailed UAV trajectories
Collision avoidance

Fig. 6: Steps in the subtask assignment and detailed trajectory planning algorithm

These examples have shown 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 complexity of the problem at hand: small changes in capability, timing constraints or obstacles can lead to completely different vehicle assignments, and a wide selection of permutations are used in even this simple example. 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.

4 Approximate Method

This section describes an approximate method [8] for solving the UAV coordination and control problems. The approximate method offers much faster solution times, but could yield sub-optimal results. However, the globally-optimal solution from the combined method can be used to evaluate the degradation in the performance associated with using this approximation (see Section 5). The cost function used in the approximate method is similar to that of the combined method, and is the overall mission completion time, plus a small weighting on the individual vehicle finishing times. However, this method evaluates the cost as a function of estimated finishing times found using straight line path approximations. Given the estimated finishing times for each UAV t_p, this cost can be evaluated as

$$\bar{t} = \max_p \; t_p \tag{13}$$

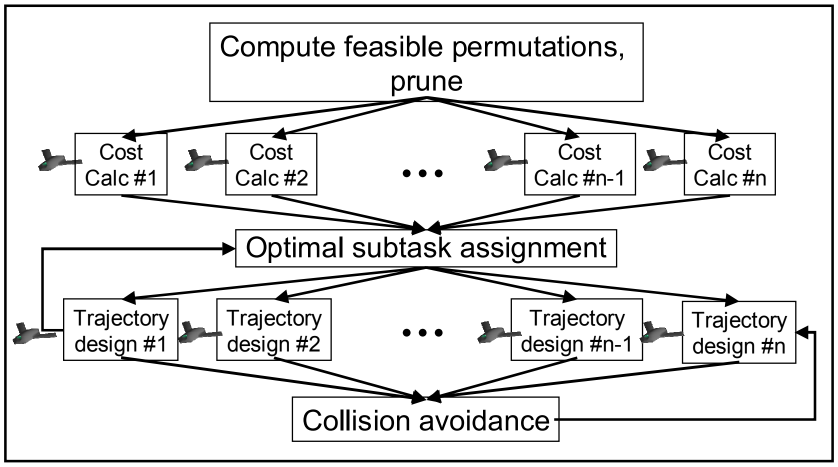

Fig. 7: Steps in the distributed task assignment and trajectory planning algorithm

$$J_1(\bar{t}, \mathbf{t}) = \bar{t} + \frac{\alpha}{N_V} \sum_{p=1}^{N_V} t_p \tag{14}$$

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

4.1 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].

The first step is to enumerate a list of all feasible assignments and orderings, subject to vehicle capabilities. Assignments, such as those involving a large number of tasks for a single vehicle, can also be eliminated at this step. These are predicted with confidence to have long completion times and will therefore be unfavorable.

Next, the approximate finishing time associated with each assignment and ordering is calculated using the straight-line approximation method (see Section 4.2). With these approximate finishing times available, the task assignment problem can be performed to find the minimum of the approximate cost, which is performed by a MILP optimization (see Section 4.3). The final step uses fixed-assignment MILP methods [4, 5] to plan detailed trajectory commands for each vehicle. These account for dynamics and inter-vehicle collision avoidance.

4.2 Finding Permutation Costs in the Approximate Method

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

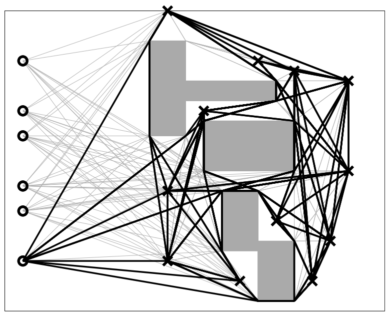

Fig. 8: Visibility graph and shortest paths between UAV 6, all waypoints

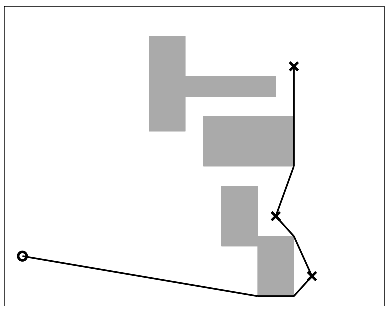

Fig. 9: Shortest path for UAV 6 over one combination of waypoints

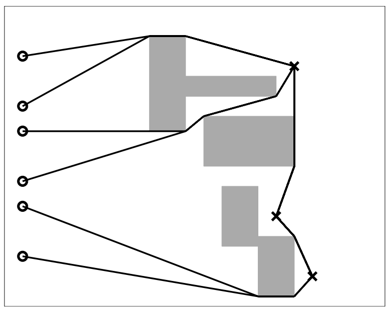

Fig. 10: Shortest paths for all UAVs over same combination of waypoints

list. This algorithm accepts the aircraft starting states $\mathbf{S}_0$, capabilities $\mathbf{K}$, obstacle vertex positions $\mathbf{Z}$, and waypoint positions $\mathbf{W}$. The algorithm also accepts two upper bounds to decrease computational effort: $n_{\max}$ which specifies the maximum number of waypoints that a UAV may visit on its mission, and $t_{\max}$ which specifies the maximum time that any UAV can take to complete its mission. From this information, this algorithm finds for each UAV the shortest permutation of every combination of fewer than $n_{\max}$ waypoints.

The steps in this algorithm are depicted in Figures 8–10, 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. 8 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. 8 shows the result for UAV 6 with black lines). In Fig. 9, one combination of waypoints has been chosen, and the shortest path from UAV 6's starting position to visit them all is shown. The order of arrival for this path is found by forming all possible ordered permutations of the unordered combination of waypoints, then summing the distance over the path associated with each order of arrival from UAV 6's starting point. In Fig. 10, the shortest path to visit the same combination of waypoints is shown for each vehicle. Note that, as expected, the best permutation of these waypoints is not the same for all vehicles.

The algorithm produces four matrices whose j^{th} columns, taken together, fully describe one permutation of waypoints. These are the row vector $\mathbf{u}$, whose element u_j identifies which UAV participates in the j^{th} permutation; $\mathbf{P}$, 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; $\mathbf{T}$, 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 $\mathbf{C}$, 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

```
1: Find shortest distances between all waypoint pairs $(i, j)$ as $D(i, j)$ using $\mathbf{S}_0$, $\mathbf{Z}$, and $\mathbf{W}$.
2: for all UAVs $p$ do
3:   Find shortest distances $d(i)$ between start points of UAV $p$, and all waypoints $i$ using
     $\mathbf{S}_0$, $\mathbf{Z}$, and $\mathbf{W}$.
4:   for all combinations of $n_C$ waypoints that $p$ is capable of visiting, $n_C = 1 \dots n_{\max}$ do
5:     for $j = 1 \dots n_C P n_C$ do
6:       Make next unique permutation $P'_{1j} \dots P'_{n_C j}$ of waypoints in the combination
7:       $C'_j = \frac{d(P'_{1j})}{v_{\max,p}}$
8:       $T'_{P'_{1j} j} = C'_j$
9:       for $i = 2 \dots n_C$ do
10:        if $C'_j > t_{\max}$ then
11:          go to next permutation
12:        end if
13:        $C'_j \leftarrow C'_j + \frac{D(P'_{(i-1)j}, P'_{ij})}{v_{\max,p}}$
14:        $T'_{P'_{ij} j} = C'_j$
15:      end for
16:    end for
17:    Append $p$ to $\mathbf{u}$
18:    Append a column to $\mathbf{V}$, whose $i^{\text{th}}$ element is 1 if waypoint $i$ is visited, 0 if not.
19:    $j_{\min} = \text{minarg}_j \; C'_j$
20:    Append column $j_{\min}$ of $\mathbf{T}'$ to $\mathbf{T}$
21:    Append column $j_{\min}$ of $\mathbf{P}'$ to $\mathbf{P}$
22:  end for
23: end for
```

Algorithm 1: Algorithm for finding shortest paths between waypoints

shortest path algorithm such as the Bellman-Ford All-Pairs Shortest Path algorithm [10]. 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. 7. The parallel platforms could be processors onboard the UAVs, or could be several computers at a centralized command and control facility.

4.3 Task Allocation in the Approximate Method

Section 4.2 outlined a method for determining the approximate costs for a UAV to visit a set of waypoints. This section presents a mathematical 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) [9]. 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 problem is given by

$$\begin{aligned} \min\ J_2 &= \sum_{j=1}^{N_M} C_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{N_M} V_{ij} x_j \geq w_i \\ & \sum_{j=N_p}^{N_{p+1}-1} x_j = 1 \end{aligned} \tag{15}$$

where the permutations of vehicle p are numbered N_p to $N_{p+1}-1$, with $N_1 = 1$ and $N_{N_v+1} = N_m + 1$ and the indices have the ranges $i \in \{1, \dots, N_W\}$, $j \in \{1, \dots, N_m\}$, $p \in \{1, \dots, N_V\}$. The binary decision variable $x_j = 1$ if permutation j is selected, and 0 otherwise. The cost in this problem formulation minimizes the sum of the costs to perform each selected permutation. The first constraint enforces that waypoint i is visited at least w_i times, usually $w_i = 1$. The second constraint prevents more than one permutation being assigned to each vehicle.

The MMKP formulation forms the base of the task allocation algorithm, however, 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 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 cost in Eq. 14 is a weighted combination of the sum of the individual mission times (as in the MMKP problem) but also the total mission time. In order to include the total mission in the cost, a new continuous variable, $\bar{t}$, is introduced and the following constraint is added to the original formulation in Eq. 15

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

The constraint forces $\bar{t} = \max_p\ t_p$ and allows the total mission time to be included in the cost. The new cost is as follows,

$$J_3 = \bar{t} + \frac{\alpha}{N_V} \sum_{i=1}^{N_m} C_i x_i \tag{17}$$

The problem is now a mixed-integer linear programming problem that can be solved using commercially available software such as CPLEX [11]. 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 [8].

4.4 Example of Approximate Method

This section shows the application of the approximate method to the example scenarios shown in Figs. 2–5. The results are shown in Fig. 11. These show the assignments and corresponding straight-line path approximations. The problems are in the same order as earlier, reading left-to-right, top-to-bottom. In each case, the assignment and route are the same as those found by the combined optimization method. In the last two 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 used in the two methods, but lead to equivalent results.

5 Performance Comparison

This section compares the performance of the two methods (combined optimization and approximation) in solving two sample problems. The combined method provides a benchmark evaluation of the globally-optimal solution to the problem by solving a single, highly complex optimization. The comparison evaluates what computational savings are available with the approximate method, and what cost is incurred in the resulting solution.

The first sample problem is small, involving two vehicles, four waypoints and two obstacles. This was solved by both methods and allows a performance comparison between the two. The second problem is much larger and is beyond the capabilities of the combined method using current solution techniques. Thus the first example illustrates the performance of the method and the second example shows its speed and potential for large problems.

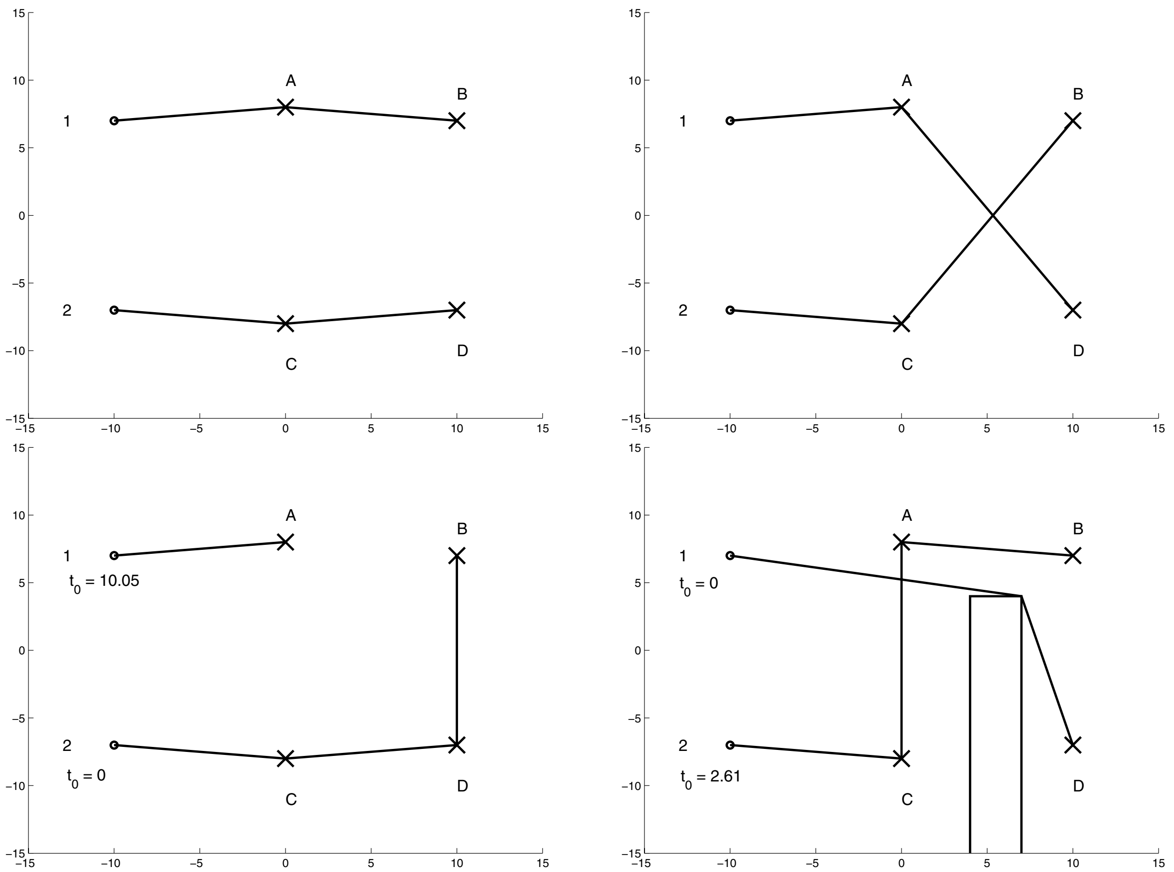

Fig. 11: Solutions to example problem in Figs. 2–5 using the approximate method.

5.1 Smaller Evaluation Problem

Fig. 12 shows the designed trajectories for the smaller problem, found using the combined method. The vehicle capabilities are shown in Table 2. There are no time dependencies in this problem. The computation took just over eight minutes, solved by CPLEX software running on a 1GHz PC with 256MB RAM. The total mission time for the designed solution is 23 time steps (in this case, each step is four units: the scaling of this problem is arbitrary).

Fig. 13 shows the solution for the same problem from the approximate method. 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 to compute the result, using CPLEX and MATLAB on a 1GHz PC with 256MB RAM. While the cost computation could be distributed, it was performed sequentially on a single PC in this example.

Table 2: Vehicle Capabilities in Comparison Problem

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

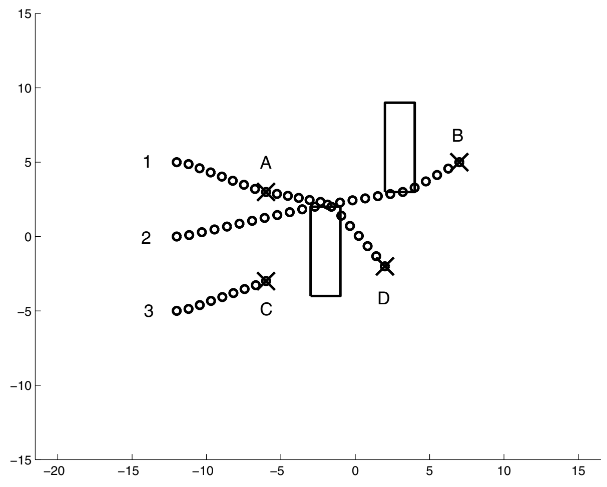

Fig. 12: Solution of the comparison problem by centralized method

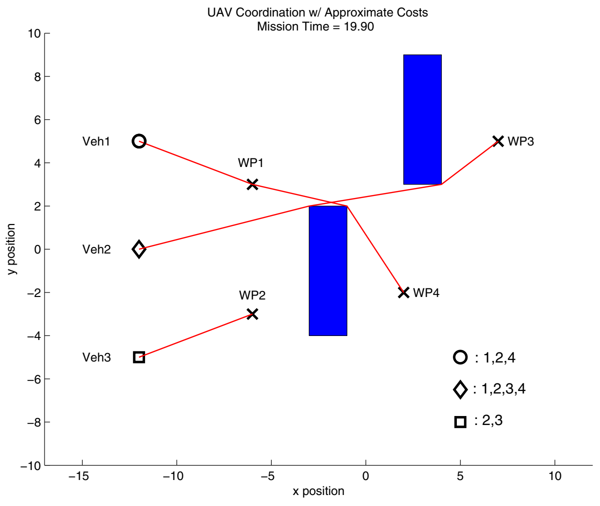

Fig. 13: Solution of the comparison problem by approximate method

5.2 Large Evaluation Problem

Fig. 14 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, due to the extremely large number of permutations involved. However, the approximate method found the solution in 27 seconds, running on the same PC used for the previous examples. Note that the capabilities of the various vehicles are denoted by the table at the top-left of Fig. 14.

6 Conclusions

This paper presents two solutions to the coupled problems of task allocation and trajectory planning for a group of UAVs. Both approaches minimize the mission completion time and

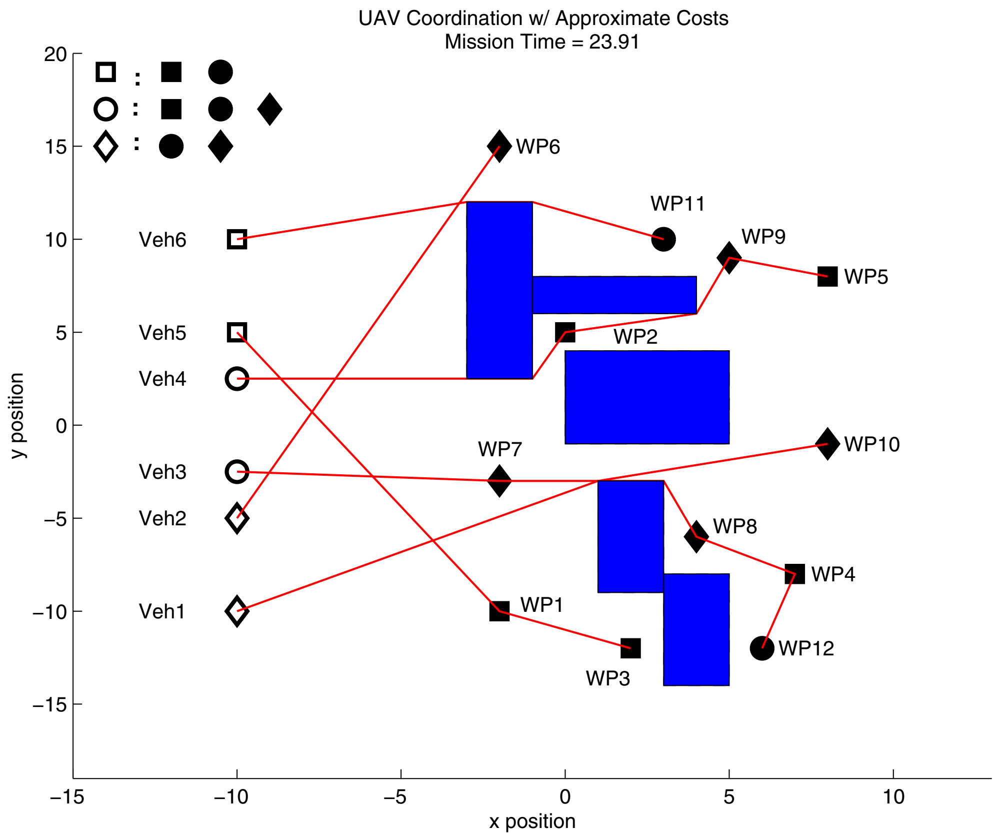

Fig. 14: Solution of large problem by approximate method

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 for the trajectory-planning part, 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. Several examples are presented in the paper to evaluate the cost degradation incurred due to the approximation. Examples are also presented to show that, since the approximate method finds the optimized assignments much faster, it can be applied to very large UAV fleet assignment problems.

Acknowledgments

The research was funded in part under Air Force grant # F49620-01-1-0453 and DARPA contract # N6601-01-C-8075.

References

[1] S. A. Heise, “DARPA Industry Day Briefing,” available on-line at www.darpa.mil/ito/research/mica/MICA01mayagenda.html

[2] C. Schumaker, P. Chandler, S. Rasmussen, “Task Allocation for Wide Area Search Munitions via Network Flow Optimization” proceedings of the *AIAA Guidance, Navigation, and Control Conference and Exhibit,* Montreal, Canada, Aug. 6-9, 2001.

[3] P. Chandler, M. Pachter, “Hierarchical Control for Autonomous Teams” proceedings of the *AIAA Guidance, Navigation, and Control Conference and Exhibit,* Montreal, Canada, Aug. 6-9, 2001.

[4] A. G. Richards, J. P. How, “Aircraft Trajectory Planning with Collision Avoidance using Mixed Integer Linear Programming,” submitted to the *American Control Conference,* 2002.

[5] T. Schouwenaars, B. DeMoor, E. Feron and J. How, “Mixed integer programming for safe multi-vehicle cooperative path planning,” presented at the *ECC*, 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] Robert Fourer, David M. Gay, and Brian 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,” to appear in the proceedings of the *Conference on Cooperative Control and Optimization,* 2001.

[9] 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.

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

[11] *ILOG CPLEX User’s guide*, ILOG, 1999.