Coordination and Control of Multiple UAVs with Timing Constraints and Loitering

Mehdi Alighanbari, Yoshiaki Kuwata, and Jonathan P. How
Space Systems Laboratory,
Massachusetts Institute of Technology
{ mehdi_a, kuwata, jhow}@mit.edu

Abstract

This paper describes methods for optimizing the task allocation problem for a fleet of Unmanned Aerial Vehicles (UAVs) with tightly coupled tasks and rigid relative timing constraints. The overall objective is to minimize the mission completion time for the fleet, and the task assignment must account for differing UAV capabilities and no-fly zones. *Loitering times* are included as extra degrees of freedom in the problem to help meet the timing constraints. The overall problem is formulated using Mixed-integer Linear Programming (MILP), which gives the globally optimal solution. An approximate decomposition solution method is also used to overcome the computational issues that arise when using MILP for larger problems. The problem is also posed in a way that can be solved using Tabu search. This approach is demonstrated to provide good solutions in reasonable computation times for large problems that are very difficult to solve using the exact or approximate decomposition methods.

1 Introduction

The capabilities and roles of UAVs are evolving, and new methods in planning and execution are required to coordinate the operation of a fleet of UAVs [1]. 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 [1, 2], and the computation time increases very rapidly with the problem size. As discussed in this paper, the situation is further complicated if the tasks:

- Are strongly coupled – *e.g.,* a waypoint must be visited three times, first by a type 1 UAV, followed by a type 2 and then a type 3.
- Have tight relative timing constraints – *e.g.,* must assign three UAVs to strike a target from three different directions within 2 seconds of each other.

These tend to cause significant problems (*e.g.,* "churning" and/or infeasible solutions) for the approximate assignment algorithms based on "myopic algorithms" that have recently been developed [3] – especially towards the end of missions.

MILP provides a natural language for codifying these various mission objectives and constraints using a combination of binary and continuous variables [2, 4, 5]. Optimal solutions can be obtained to these problems using commercially available software such as CPLEX, but approximate techniques are required for real-time applications. This paper extends an approximate decomposition algorithm [2, 5] to include these relative timing constraints and adds extra degrees of freedom to the formulation that allow the UAVs to loiter during the mission. Impacts on the computational time by adding these timing constraints to the problem are demonstrated in a complex example with 6 UAVs and 12 waypoints. Tabu search techniques [6] are also investigated to solve the tightly coupled assignment problems for scenarios with a large number of UAVs and waypoints (and a large number of permutations and combinations) for which the decomposition method also becomes computationally intractable.

2 Problem Formulation

This section describes how the multiple vehicle routing problems with relative timing constraints and loitering can be written as a MILP. The algorithms assume that the team partitioning has already been performed, and that a set of tasks has been identified that must be performed by the team. The overall objective is to assign one set of ordered waypoints to each vehicle that is combined into the mission plan, and adjust the loiter times for the team such that the cost of the mission is minimized and the time of task execution at each waypoint satisfies the timing constraints.

2.1 Algorithm Overview

There are three main phases in our algorithm [2, 5]: (I) cost calculation, (II) planning and pruning, and (III) task assignment.

I-1. Find the visibility graph between the UAV starting positions, waypoints, and obstacle vertices.

I-2. Using the Dijkstra's algorithm, calculate the shortest length of the all feasible paths between waypoints, and form the cost table.

II-1. Obtain feasible `combinations` of waypoints, accounting for the capability and the maximum number of waypoints per UAV.

II-2. Enumerate all feasible `permutations` from these combinations, subject to the timing constraints.

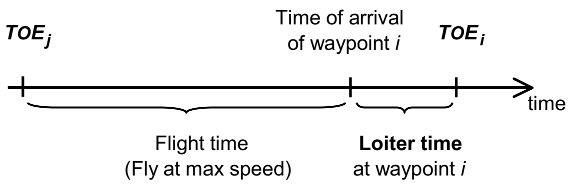


Fig. 1: Flight time, loiter time, time of arrival, and time of task execution.

II-3. Calculate cost for each permutation using the cost table obtained in phase I.
II-4. Select the n_p best permutations for each combination.
III-1. Solve the task allocation problem using an optimization solver.
III-2. Solve for the each UAV's trajectory (*e.g.* using straight line segments).

Let there be N_V UAVs and N_W waypoints. At the end of phase II, four matrices with the same column length N_M are produced whose j^{th} columns, taken together, fully describe one permutation of waypoints. These are the row vector $\mathbf{u}$, whose u_j entry identifies which UAV is involved in the j^{th} permutation; $N_W \times N_M$ matrix $\mathbf{V}$, whose V_{ij} entry is 1 if waypoint i is visited by permutation j, and 0 if not; $N_W \times N_M$ matrix $\mathbf{T}$, whose T_{ij} entry is the time at which waypoint i is visited by permutation j assuming there is no loitering, and 0 if waypoint i is not visited; and the row vector $\mathbf{c}$, whose c_j entry is the completion time for the j^{th} permutation, again, assuming there is no loitering.

2.2 Decision Variables

Selection of the Permutations: In order to assign one permutation to each vehicle, the $N_M \times 1$ binary decision vector $\mathbf{x}$ is introduced whose x_j equals 1 if permutation j is selected, and 0 otherwise. Each waypoint must be visited once, and each vehicle must be assigned to one permutation, so

$$\sum_{j=1}^{N_M} V_{ij} x_j = 1\,, \qquad i = 1, \dots, N_W \tag{1}$$

$$\sum_{j=N_p}^{N_{p+1}-1} x_j = 1\,, \qquad p = 1, \dots, N_V \tag{2}$$

where the permutations of p^{th} vehicle are numbered N_p to $N_{p+1}-1$, with $N_1 = 1$ and $N_{N_V+1} - 1 = N_M$.

Loitering time: As shown in Fig. 1, the loiter time at waypoint i is defined as the time difference between time of the task execution and the time of arrival at waypoint i. The UAVs are assumed to fly at the maximum speed between waypoints, and loiter before executing the task. Note that it can also be regarded as flying at a slower speed between the waypoints, or loitering at the previous waypoint, flying towards the waypoint at the maximum speed v_{max}, and executing the task.

Introduce the $N_W \times N_V$ loitering matrix $\mathbf{L}$, whose L_{ij} element expresses the loiter time at the i^{th} waypoint when visited by UAV j, as a set of new decision variables ($L_{ij} = 0$ if waypoint i is not visited by UAV j). The loitering matrix ensures that it is always possible to find a feasible solution as long as the timing constraints are consistent. In the MILP formulation, the time of the task execution at waypoint i, TOE_i, is written as

$$TOE_i = \sum_{j=1}^{N_M} T_{ij}\, x_j + LB_i\,, \qquad i = 1, \dots, N_W \tag{3}$$

where the first term expresses the flight time from the start point to waypoint i at $v_{\max}$, and LB_i is the sum of the loiter times before executing the task at waypoint i.

Define the set $\mathcal{W}$ such that $\mathcal{W}_i$ is the list of waypoints visited on the way to waypoint i (including i), so that

$$LB_i = \sum_{j \in \mathcal{W}_i} \sum_{k=1}^{N_V} L_{jk}\,, \qquad i = 1, \dots, N_W \tag{4}$$

Only one UAV is assigned to each waypoint, and each row of $\mathbf{L}$ has only one non-zero element. To express the logical statement "on the way to", we introduce a large number M, and convert the one equality constraint Eq. (4) into two inequality constraints

$$LB_i \le \sum_{j=1}^{N_W} \left(O_{ijp} \sum_{k=1}^{N_V} L_{jk} \right) + M \left(1 - \sum_{p=1}^{N_M} V_{ip} x_p \right) \tag{5}$$

and

$$LB_i \ge \sum_{j=1}^{N_W} \left(O_{ijp} \sum_{k=1}^{N_V} L_{jk} \right) - M \left(1 - \sum_{p=1}^{N_M} V_{ip} x_p \right) \tag{6}$$

where $\mathbf{O}$ is a three dimensional binary matrix that expresses waypoint orderings, and $O_{ijp} = 1$ if waypoint j is visited before waypoint i (including $i = j$) by permutation p, and 0 if not. When waypoint i is visited by permutation p, the second term on the right-hand side of the constraints in Eqs. 5 and 6 disappears, producing the equality constraint

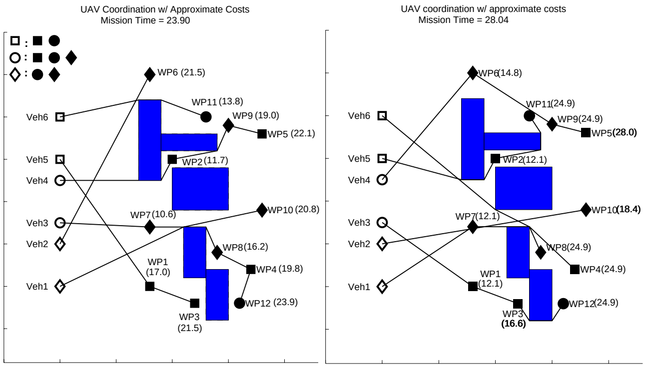
$$LB_i = \sum_{j=1}^{N_W} \left(O_{ijp} \sum_{k=1}^{N_V} L_{jk} \right) \tag{7}$$

which is the same as Eq. (4). Note that when waypoint i is not visited by permutation p, $O_{ijp} = 0$ for all j and $V_{ip} = 0$, so that both of the inequality constraints are relaxed and LB_i is not constrained.

2.3 Timing Constraints

The timing constraints of interest in this application are relative, as opposed to the absolute ones often considered [7, 8, 9], and they are written as

$$TOE_{C_{k2}} \ge TOE_{C_{k1}} + d_k\,, \qquad k = 1, \dots, N_C \tag{8}$$



(a) (b)

Fig. 2: Scenario with 6 heterogeneous UAVs & 12 waypoints. (a) No timing constraints. Solved in 2sec. (b) 11 timing constraints. Solved in 13sec.

where each row of matrix $\mathbf{C}$ and vector $\mathbf{d}$ represents a dependency between two waypoints. If the k^{th} row of $\mathbf{C}$ is $[\ i \quad j\ ]$, Eq.(8) becomes $TOE_j \geq TOE_i + d_k$. Note that d_k can also be negative. This formulation allows us to encode very general relative timing constraints. Although each waypoint i has only one time of execution TOE_i associated with it, this formulation can be used to describe several visits with timing constraints by putting multiple waypoints at that location.

2.4 Cost Function

The cost J to be minimized in the optimization is

$$J = \max_{k \in \{1,\ldots,N_V\}} t_{F_k} + \frac{\alpha}{N_V} \sum_{i=1}^{N_M} c_i x_i + \frac{\beta}{N_W} \sum_{j=1}^{N_V} \sum_{i=1}^{N_W} L_{ij} \quad (9)$$

where the first term gives the maximum completion time of the team, the second gives the average completion time, and the third gives the total loiter times. If the penalty $\alpha \geq 0$ on average flight time were omitted, the solution could assign unnecessarily long trajectories to all UAVs except for the last to complete its mission. Similarly, $\beta \geq 0$ can be used to include an extra penalty that avoids excessive loitering.

3 Simulation Results

This section presents results from several simulations using the formulation in Section 2. The problems were solved using CPLEX (v7.0) running on a 2.2GHz PC with 512MB RAM. The first result investigates how the timing constraints impact the solution times. The second considers the relationship between the complexity of timing constraints and the computation time.

3.1 Problems with and without timing constraints

A large scenario that includes a fleet of 6 UAVs of 3 different types and 12 waypoints is used as our baseline. The UAV capabilities are shown in Fig. 2(a) (top left). There are also several obstacles in the environment. The objective is to allocate waypoints to the team of UAVs in order to visit every waypoint once and only once in the minimum amount of time. For convenience, this problem without timing constraints will be referred to as the "original" problem. Fig. 2(a) shows the solution of this original problem. All waypoints are visited subject to the vehicle capabilities in 23.90 time units. Time of task execution of each waypoint is also shown beside the waypoint in the figure.

The problem with simultaneous arrival and ordered task was also considered. Timing constraints in this scenario are as follows:

1. Wpts 1, 7, 2 must be visited at same time.
2. Wpts 4, 8, 12 must be visited at same time.
3. Wpts 9, 11 must be visited at same time.
4. Wpts 4, 8, 12 must be visited before Wpts 9, 11.

The solution in this case is shown in Fig. 2(b), which is quite different from Fig. 2(a). In Fig. 2(a), UAV3 visits waypoints 4, 8, and 12, whereas in Fig. 2(b), three UAVs are assigned to these three waypoints since they must all be visited at the same time. More UAVs are assigned to the waypoints in the lower half of the figure in Fig. 2(a) than in Fig. 2(b) since the priority of waypoints 4, 8, 12 are higher than that of waypoints 9, 11 as a result of the timing constraints. Also, waypoint 4 is visited by UAV6, which is the farthest from it. The mission time for this scenario increased to 28.04 time units, and the computation time increased from 2 seconds to 13 seconds. To solve this problem in a reasonable time, the following approximations were made:

- Select only 1 best feasible permutation per combination.
- If there is a timing constraint $TOE_i \geq TOE_j + t_D$ $(t_D \geq 0)$, then the UAVs can loiter only at waypoint i.

In order to satisfy the many timing constraints, 4 UAVs loiter at 6 waypoints. UAV1 loiters on its way to waypoint 7 and 8, and UAV3 loiters on its way to waypoints 1 and 12. If time adjustment is allowed only on the initial position as in Ref. [2], a feasible solution cannot be found in this scenario. Since the loiter matrix $\mathbf{L}$ allows UAVs to loiter at any of the waypoints with timing constraints, problems with strongly coupled timing constraints are always solvable.

3.2 Complexity of Adding Timing Constraints

To investigate the impact of the timing constraints on the performance and computation time, we measured the computation time for the problem in Section 3.1, with these four timing constraints:

$$\begin{aligned}
&\text{Case}-1\text{:} \quad TOE_i \geq TOE_j \\
&\text{Case}-2\text{:} \quad TOE_i \geq TOE_j + 10 \\
&\text{Case}-3\text{:} \quad TOE_i \geq TOE_j \geq TOE_k \\
&\text{Case}-4\text{:} \quad TOE_i \geq TOE_j + 5 \geq TOE_k + 10
\end{aligned}$$

In each case, all feasible combinations of waypoints (i, j) or (i, j, k) were tested as the points associated with the timing constraints. The results are summarized in the histograms of Figures 3–6.

Figs. 3(a), 4(a), 5(a), and 6(a) show the results when all loitering times are included in the problem. Since there are 12 waypoints and 6 UAVs with different capabilities, there are 52 extra degrees of freedom in the

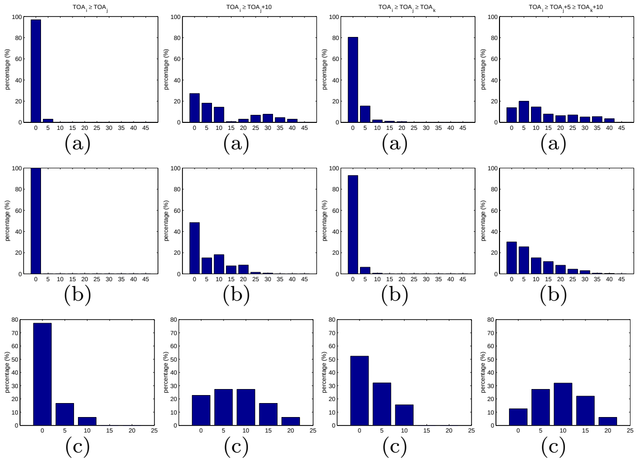


Fig. 3: **Fig. 4**: **Fig. 5**: **Fig. 6**:

Histograms of case 1–4 from left to right – (a) shows the computation time [sec] for the problem with no constraints on the loitering times; (b) shows the computation time [sec] for the problem with constrained loitering; and (c) shows the degree of "unnaturalness".

decision variable $\mathbf{L}$. Figs. 3(b), 4(b), 5(b), and 6(b) show the results when the constrained form of the loitering is used. This reduces the degrees of freedom in the loiter matrix $\mathbf{L}$ from 52 to 8–12, depending on the type of constraints.

Comparing Figs. 3(a), (b) with Figs. 4(a), (b), and, Figs. 5(a), (b) with Figs. 6(a), (b), it is clear that the computation time increases as more complicated timing constraints are imposed on the tasks (either by increasing the time gaps or by increasing the number of related tasks). With fewer degrees of freedom, the constrained loitering approach solves faster by a factor of two.

To approximately determine the complexity of these constraints, we introduce the concept of the "unnaturalness" of the timing constraints, which is a measure of the degree to which the timing constraints are violated by the solution of the original problem. Using the solution of the original problem to obtain the times associated with waypoints i and j ($TOE_i{}'$ and $TOE_j{}'$), define the unnaturalness of a timing constraint $TOE_i \geq TOE_j + t_D$ as

$$\max\left\{ TOE_j{}' + t_D - TOE_i{}', 0\right\} \qquad (10)$$

If the solution of the original problem happens to satisfy the timing constraint, the unnaturalness is 0. The sum of the unnaturalness of each timing constraint is used as a measure of the unnaturalness for the constrained problem. Note that other metrics such as the number of timing constraints, and the extent to which they are tightly coupled together can be misleading if the results are "naturally" satisfied by the solution to the unconstrained problem. The metric in Eq. (10) gives a direct (albeit approximate) measure of the extent to which the solution must be changed to satisfy the additional timing constraints.

Figs. 3(c), 4(c), 5(c), and 6(c) show four histograms that give the unnaturalness of the basic problem with timing constraints (cases 1 – 4). The shapes of the 4 histograms reflect the computation time required to solve these problem. In particular, as the distribution of the unnaturalness shifts towards the right (Fig. 3(c)→ 4(c) and 5(c)→ 6(c)), the distribution of the computation time also shifts to the right (Fig. 3(b)→ 4(b) and 5(b)→ 6(b)). Further observations include:

- If all of the timing constraints are natural, then the computation time does not increase significantly, even if there are many timing constraints.
- If all the timing constraints are natural, the best permutation is always feasible without pruning by timing constraints, but that is not the case if there are unnatural timing constraints.
- Additional permutations can be kept to account for unnatural timing constraints, but simulation results have shown that this can cause a combinatorial explosion and rapidly increase the computation time with a marginal improvement in performance.

The results show that this algorithm can solve the problem of UAV assignment with relative timing constraints. It was shown that increasing the number of timing constraints and the degree of "unnaturalness" makes the problem harder to solve, but the proposed algorithm can still be used to obtain the globally optimal solution.

4 Tabu Search for the UAV Problem

The UAV assignment problem discussed in the previous sections is a generalization of the vehicle routing problem, which is NP-hard. Thus finding the optimal solution to this problem using exact methods is computationally infeasible for large fleets, and even the decomposition methods discussed in Section 2 becomes intractable when the number of permutations and combinations increase. However, several researchers have demonstrated that the Tabu search method can be used to rapidly obtain sub-optimal solutions of the VRP problem [10]. This section formulates the UAV problem of Sections 2 and 3 as a VRP with relative timing constraints and uses modified Tabu algorithms to solve this problem.

4.1 Tabu Search Method

Glover first proposed the basic ideas of Tabu search [6] and the algorithm has been studied extensively because it has proven to be an effective heuristic for solving combinatorial optimization problems such as scheduling, telecommunications, transportation and network problems [6]. The Tabu method searches a neighborhood of a given solution for a better feasible solution, which is the basis for many solution algorithms. The *neighborhood of a solution* is defined to be all solutions that can be reached in a *single move*, where the definition of a move is problem specific (*i.e.,* changing one bit from 1 to 0, or swapping the position of two elements in a vector). The problem with most neighborhood methods of this type is that they can get trapped in a local minimum, and loop endlessly. To prevent this looping between the same solutions, Tabu search uses the concept of *memory* and a *Tabu list*. The detailed discussion on Tabu search method can be found in [6].

4.2 Problem Formulation

Suppose there are N_W waypoints and N_V UAVs. We represent a solution to the problem as a sequence of UAVs and waypoints, as shown in Fig. 7. The sequence consists of a list of UAV numbers followed by the waypoints that it visits (in order). If a UAV number is immediately followed by another UAV number, then the first vehicle does not visit any waypoints. The Tabu search algorithm in [7] is used in our implementation.

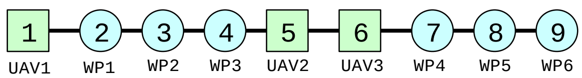


Fig. 7: Solution sequence when UAV1 visits WP1, WP2, WP3; UAV2 does not visit any waypoints; and UAV3 visits WP4, WP5, WP6.

The objective function to be minimized here is the fleet completion time:

$$J = \max_i \ (\texttt{finish}_i) + \frac{\alpha}{N_V} \sum_i \ (\texttt{finish}_i)$$

where, $\texttt{finish}_i$ is the time that UAV_i finishes its mission and $\alpha \ll 1$ scales the average completion time compared to maximum completion time.

4.3 Adding Side Constraints

The type of timing constraints implemented in this algorithm are the same as introduced in Section 2.3. There are two different ways to deal with these constraints, one is to treat them as hard constraints and exclude any solution that violates them. An alternative is to treat them as soft constraints and add a penalty to the objective functions for the solution that violate the constraints [7]. One advantage of using soft constraints is that the initial solution does not have to be a feasible solution. This is important in cases that finding a feasible solution itself is difficult. Another advantage of soft constraints is that the algorithm is not restricted to feasible regions and can move through an infeasible region to find a better feasible solution. Our algorithm treats the timing constraints as soft constraints. In this case, if there is a constraint on waypoint i being visited after j, but in a solution we have $\texttt{start}_j > \texttt{start}_i$ ($\texttt{start}_i$ represents the time that waypoint i is visited), then a penalty is added to the objective value of this solution. By increasing the magnitude of this penalty, we can move from soft to hard timing constraints.

The problem formulation also includes capability constraints on the different UAVs, which have a significant impact on the solution of the assignment problem. The UAV capabilities are given by the binary matrix K. Similar to the timing constraints, the UAV capabilities can be added in two ways. If, in a solution, waypoint x is in the list of missions for a UAV of type i that is not capable of visiting that waypoint (*i.e.,* $K_{ix} = 0$), then the solution is either rejected or kept as a solution with a penalty for violating the constraint.

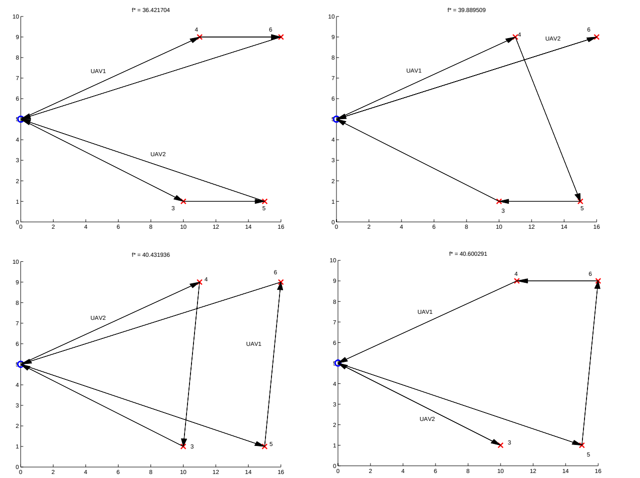


Fig. 8: Results of Tabu search applied to four simple problems with heterogeneous UAVs and relative timing constraints.

In the environments with obstacles, Tabu search can be applied with a slight change in the algorithm. A matrix that represents the distance between all pairs of points is given to the algorithm as an input. In the case that there is no obstacle, these distances are simply straight lines between the two points. But when there are obstacles, these distances can be calculated using visibility graph and Dijkstra's algorithm, as discussed earlier.

4.4 Simulation Results

To show different capabilities of the algorithm for this application, it is first applied to a small problem. Fig. 8 shows the result of a UAV assignment problem for a small fleet facing four different scenarios. There are two UAVs and four waypoints, and the objective is to minimize the mission completion time. The top left figure shows the result for the scenario in which the two UAVs have the same capabilities and there are no timing constraints. As expected, one UAV visits the waypoints at the top and the other UAV visits the ones at the bottom. The top right figure shows the solution of the same problem with timing constraints not satisfied by the solution to the first problem. In this scenario, waypoint 5 is constrained to be visited after waypoints 4 and 6. As shown, the cost has increased compared to the first scenario, but the constraints are now satisfied. The bottom left figure shows the scenario without the timing constraint, but with heterogeneous UAVs. In this scenario, UAV1 is capable of visiting all the waypoints, while UAV2 can just visit waypoints 3 and 4. Therefore UAV2 visits the waypoints that it is capable of visiting and UAV1 visits the rest. The bot-

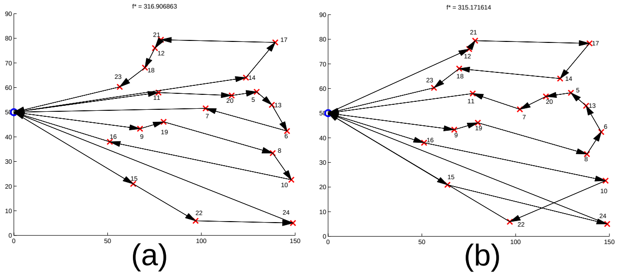


Fig. 9: (a) Result of Tabu search applied to a problem with 4 UAVs and 20 waypoints. (b) Result with timing constraints (waypoint 22 to be visited after waypoints 15 and 24)

tom right figure shows the result of the scenario with heterogeneous UAVs and relative timing constraints. In this scenario, again UAV1 can visit all waypoints and UAV2 can only visit waypoints 3 and 4. The timing constraints are that waypoint 4 should be visited after waypoints 5 and 6. As shown, the final result satisfies all of the constraints by having UAV2 just visit waypoint 3 and UAV1 taking care of the rest of waypoints.

To show the capability of the algorithm on larger problems, an example with 4 UAVs and 20 waypoints is shown in Figs. 9(a,b). Fig. 9(a) shows the solution with no constraints, while Fig. 9(b) has two timing constraints that force waypoint 22 to be visited after waypoints 15 and 24. As shown, the solution in Fig. 9(b) gives a cost (315.7) that is slightly lower than the cost (316.9) for the solution in Fig. 9(a). In the optimal case this should not be the case since Fig. 9(a) has no constraints. However, the Tabu search is a heuristic method and optimality of the solution is not guaranteed. Many studies have shown that it is possible to get very close to the optimal solution if the parameters in the problem are chosen correctly. The initial solution is another important factor in a Tabu search. The initial solution impacts the convergence rate for the algorithm, and it can also effects the final solution. However, in 2000 trials of this UAV problem with different initial conditions, the result shows that more than 95% of the solutions are within 3% of the best solution. The computation time for these trials using a non-optimized code in MATLAB, varies between 12 to 15 seconds. Better results can be achieved by using more efficient codes.

The Tabu solutions in this section meet the relative timing constraints by arranging the vehicle paths so that the UAVs arrive at the waypoints at the correct times. However, with complicated constraints of the type shown in Fig. 2(b), it is possible that this sub-optimal approach could result in a poor solution. The solution in Section 2 was to add loitering times to increase the number of degrees of freedom in the problem, thereby yielding better solutions. Loitering could be added to Tabu search by including (discrete) time increments to the solution sequence in Fig. 7. These times would then be used to separate the arrival and departure times of the preceding UAV at the preceding WP number.

5 Conclusions

This paper presents an extension of the multiple UAV task allocation problem that explicitly includes the relative timing constraints found in many mission scenarios. This not only allows us to determine which vehicle should go to each waypoint, but it also allows us to account for the required ordering and relative timing in the task assignment. The allocation problem was also extended to include loiter times as extra (continuous) degrees of freedom to ensure that, even with very complicated timing constraints, feasible solutions still exist. Simulation results clearly showed that adding these timing constraints to the problem increases the computational time when the constraints are active (*i.e.,* "unnatural"). The constrained allocation problem was also formulated in a way that can be solved using Tabu search, which was demonstrated to provide good solutions in reasonable computation times for large problems that are very difficult to solve using the exact or approximate decomposition methods.

Acknowledgments

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

References

[1] P. Chandler, M. Pachter, D. Swaroop, J. Fowler, *et al.* "Complexity in UAV cooperative control," *ACC* 2002. pp. 1831-1836

[2] J. Bellingham, M. Tillerson, A. Richards, and J. How, "Multi-Task Allocation and Path Planning for Cooperating UAVs," *Second Annual Conference on Cooperative Control and Optimization,* Nov 2001.

[3] C. Schumaker, P. Chandler, S. Rasmussen, "Task Allocation for Wide Area Search Munitions via Network Flow Optimization" AIAA *GNC,* Aug. 2001.

[4] A. Bemporad and M. Morari, "Control of Systems Integrating Logic, Dynamics, and Constraints," Automatica, Pergamon/Elsevier Science, Vol. 35, pp. 407-427, 1999.

[5] A. Richards, J. Bellingham, M. Tillerson, and J. P. How, "Co-ordination and Control of Multiple UAVs," AIAA *Guidance, Navigation, and Control Conference,* Monterey, CA, August 2002. AIAA Paper 2002-4588

[6] F. Glover and M. Laguna, *Tabu Search*, Kluwer Acad. Publ., 1997.

[7] K. P. O'Rourke, T. G. Bailey, R. Hill and W. B. Carlton, "Dynamic Routing of Unmanned Aerial Vehicles Using Reactive Tabu Search," *Military Operations Research Journal*, Vol.6, 2000.

[8] M. Gendreau, A. Hertz and G. Laporte , "A Tabu Search Heuristic for the Vehicle Routing Problem," *Management Science*, Vol. 40, pp. 1276-1289, 1994.

[9] E. D. Taillard, P. Badeau, M. Gendreau, F. Guertin, J. Y. Potvin, "A Tabu search heuristic for the vehicle routing problem with soft time windows," *Transportation science* Vol. 31, pp. 170-186, 1997.

[10] A. V. Breedam, "Comparing Descent Heuristics and Metaheuristics for the Vehicle Routing Problem", *Computers and Operations Research*, Vol.28, pp. 289-315, 2001.