Chapter from the book
Downloaded from: http://www.intechopen.com/books/

Interested in publishing with InTechOpen?
Contact us at book.department@intechopen.com
Employing Multiple Unmanned Aerial Vehicles for Co-Operative Path Planning

Durdana Habib1,*, Habibullah Jamal2 and Shoab A. Khan3

1 Dept. of Electrical Engineering, National University of Computer & Emerging Sciences, Islamabad, Pakistan
2 Dept. of Electrical Engineering, University of Engineering & Technology, Taxila, Pakistan
3 Dept. of Computer Systems Engineering, National University of Sciences & Technology, Rawalpindi, Pakistan

* Corresponding author E-mail: durdana.habib@nu.edu.pk

Received 20 Jul 2012; Accepted 1 Mar 2013

DOI: 10.5772/56286

© 2013 Habib et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract In this paper, we work to develop a path planning solution for a group of Unmanned Aerial Vehicles (UAVs) using a Mixed Integer Linear Programming (MILP) approach. Co-operation among team members not only helps reduce mission time, it makes the execution more robust in dynamic environments. However, the problem becomes more challenging as it requires optimal resource allocation and is NP-hard. Since UAVs may be lost or may suffer significant damage during the course of the mission, plans may need to be modified in real-time as the mission proceeds. Therefore, multiple UAVs have a better chance of completing a mission in the face of failures. Such military operations can be treated as a variant of the Multiple Depot Vehicle Routing Problem (MDVRP). The proposed solution must be such that $m$ UAVs start from multiple source locations to visit $n$ targets and return to a set of destination locations such that (1) each target is visited exactly by one of the chosen UAVs (2) the total distance travelled by the group is minimized and (3) the number of targets that each UAV visits may not be less than $K$ or greater than $L$.

Keywords Multiple Depot Vehicle Routing Problem, Cooperative Path Planning, Optimal Resource Allocation, Mixed Integer Linear Programming

1. Introduction

Small autonomous vehicles have paved the way for numerous military and civil applications, which are currently an active area of research. Military applications may involve the use of UAVs, where operations may not be restricted to just one depot or launch site and so the scenario may require several different launch and landing sites. Unmanned Aerial Vehicles (UAVs) not only eliminate threat to pilot’s life, they require less power, leading to longer mission times and more flexible manoeuvring. To utilize the full potential of a team of UAVs, efficient methods of decentralized sensing and path planning must be employed.

The UAV task assignment problem has been explored from many different angles. Auction-based approaches have been used for task allocation to multiple robots to find feasible or even guaranteed quality solutions easily using PRIM’s algorithm [1]. Sariel et al. [2] propose a framework that employs an incremental task allocation method, dynamically adapting to the changes in the environment. Gao et al. [3] present a decentralized allocation algorithm, which combines a sequential single-task auction with a task transfer among robots of the
same sub-team. The tasks allocated to the sub-team are clustered using an orthogonal genetic algorithm, which further improves the efficiency of task transfer. In the absence of threats, the multivehicle target assignment problem is related to the dual or multiple vehicle problem where the sum of the distances travelled by the vehicles is sought to be minimized. This approach is presented in [4], where seven vehicles are used to visit fifty targets. However, this approach does not incorporate load balancing. Methods for solving the multivehicle, target assignment problem in the presence of threats are presented by Maddula et al. in [5]. The objective, in this case, is to minimize the maximum path length.

Use of a centralized planner such as GRAMMPS [6] has also been made for constrained task allocation and planning. Centralized algorithms have been used to solve the path planning problem. Cui et al. [7] propose a search method based on Probabilistic Road Maps (PRM) to obtain the Pareto-optimal coordination solution for multiple agents. A decoupled approach is used, consisting of two parts, building the roadmap in a coordination space and searching for Pareto-optimal solutions in the co-ordination roadmap. Decoupled planning does not increase the dimensionality of the configuration space and is quite practical [8].

Desaraju et al. [9] present the Decentralized Multi-Agent Rapidly exploring Random Tree algorithm by extending the Closed Loop Rapidly exploring Tree (CL-RRT) strategy to the multi-agent case. It gives priority to re-plan to agents with greater potential to improve the path and allows an agent to modify its teammates’ plan in order to select paths that reduce their combined cost. The strategy suits scenarios where a large number of agents operate in cluttered environments. However, it requires that the agents have emergency nodes in their trajectory where they can halt to avoid collision/deadlocks. The approach needs to be modified for air vehicles which cannot have zero forward speed.

Though many different approaches have been used to solve the task allocation problem, an efficient integer programming formulation gives flexibility, whereby additional constraints can be easily added to obtain a realistic, optimal and feasible solution to the path planning problem. For example, the UAV may have a constraint on the turning radius. Avoidance of travel legs with a risk factor above a certain threshold may be modelled as a constraint. An attempt at balancing loads across different UAVs can be incorporated. Thus, a variety of constraints can be included in the mathematical programming model presented here.

Prior to this work, MILP has been combined with clustering techniques [10] to reduce the problem size. An attempt to use MILP for computing fuel-optimal paths for multiple vehicles was made by Schauwenaars et al. [11]. In this work, MILP is applied in a receding horizon framework to design a series of short trajectories to the goal instead of one long trajectory. The Receding Horizon (RH) approach was adopted as opposed to a fixed arrival time to reduce solution time. Simulations of moving a single vehicle around ten obstacles demonstrate its feasibility using different horizon lengths. The approach was extended to calculate the trajectories of Autonomous Aerial Vehicles by Bellingham et al. [12, 13].

In this paper, we propose a mixed integer linear programming model to be used for $m$ multiple vehicles. The vehicles carry sensors that have footprints several times greater than 2$r$, where $r$ is the minimum turn radius of the vehicle. Assuming the targets to be sufficiently spread, reasonably flyable, optimal paths may be computed. Path planning is carried out prior to mission execution using the information available a-priori. However, as the real-time assessments, carried out using on-board sensors, reveal obstacles and/or high risk zones, more informed actions are needed. The team may suffer the loss of one or more of its members. Paths need to be re-computed in real-time to cater for such contingencies. Each UAV must have on-board computational resources to compute a new plan using the information acquired. The communication of solutions is implicit, that is, it is accomplished by all vehicles running the same model in parallel. Information collected by on-board sensors is shared among all the vehicles. The vehicle travelling longest determines the duration of the mission. Thus load-balancing among team members plays a crucial role in reducing mission execution time.

The problem of resource allocation is well known as being NP-hard. An important point to be mentioned here is that NP-classification is defined in terms of worst-case behaviour, not average case behaviour. For most practical situations, the main interest is in average case behaviour. Aside from this application, lower bounds can still be derived and used in branch and bound solvers of commercial software to get faster results. Rathinam et al. [14] have extended the lower bound of Held-Karp [15], the best-known result for the single Travelling Salesman Problem (TSP), to the multiple depot, multiple TSP.

2. Related Work

Over the past two decades the MDVRP has attracted greater attention due to its applicability in many real-life scheduling and routing problems. Bektas et al. [16] gives a comprehensive review of these applications as well as the known approaches. MILP based methods have been gaining in popularity as a means of solving a wide variety of planning problems because they are provably
optimal and can handle a wide variety of constraints. The main drawback is that when they are applied to uncertain environments, the solutions must be obtained quickly. Furthermore, the problem may need to be resolved quite frequently. The work presented in this paper tries to overcome these issues through combining MILP with less optimal methods to improve the computation time for large problems.

Specific cases of resource allocation problems can also be formulated as a MILP problem. Related versions of this problem can be found in [17-19]. Kara et al. [20] propose a polynomial-size integer linear programming model for this problem, with improvements in subtour elimination constraints (SECs). Thus, a practical formulation that is solvable in real-time with a commercial code will pave the way for real-time implementations.

In previous work by Bellingham [12], paths are optimized to compensate for UAV loss and to exploit the coupling effects of cooperation between UAVs to improve survival probability. Since MILP uses discrete points along the trajectory, the computational complexity grows with the length of the horizon and the number of obstacles. Simulation results using 4 vehicles and 8 targets are given. It takes 60 minutes to find a feasible solution, yet the problem is not solved to completion. Results in [13] show that for fixed-wing autonomous vehicles, with 200 discrete steps and eight obstacles, the computation takes over 600 seconds. In our work, UAV attrition is taken into account and risky paths are avoided. Since our work deals with small UAVs in sparse environments and targets are scattered, the paths computed are flyable. For small and medium-sized instances, MILP solutions are computed quickly, qualifying for real-time applications.

Two recent papers, Kim et al. [21] and Alidaee et al. [22] have considered the problem of m identical UAVs that are to serve n targets. The problem is to find a sequence of visiting targets for each UAV, starting from a depot, such that all targets are visited only once by a UAV. The formulations also have an additional bound on the maximum number of targets each UAV can visit. In our implementation, we limit both the minimum and maximum number of nodes that a UAV should visit which allows the decision maker to assign appropriate work-loads to each vehicle. Post-optimality analysis can be conducted simply by varying the parameters of the problem and observing the changes in the solution values.

Julia et al. [23] provide some insight into the path planning strategies to be used for different applications. It is shown that coordination improves if agents use techniques that provide close to global optimal solutions.

The main contributions of this work are: (i) a different approach to catering for real-time path planning for moderate-sized problems; (ii) plans are computed in a decentralized manner yet solutions are optimal since all vehicles share the same information; (iii) use of a strategy to avoid collisions and static obstacles; (iv) plans are reconfigurable in real-time in case of UAV attrition.

Computational analysis shows that moderate-sized instances can be solved to optimality via the proposed formulations. Since the problem considered in [22] is closely related to the one being considered in this work, the two formulations will be examined for performance in Section 3. In Section 4, simulation and experimental results will be presented. The computed path plans will also be compared to those computed in [3] and to those produced by a variant of the open-source Matlab code available at www.matlabcentral.com by Joseph Kirk.

3. Problem Description

Let \( T = \{ t_1, t_2, \ldots, t_m \} \) be the set of targets to be visited. There are \( m \) UAVs initially located at source vertices \( S = \{ s_1, s_2, \ldots, s_m \} \). The cardinality of \( S \) is \( m \) and that of \( T \) is \( n \). Each depot sends out one vehicle so that the number of depots is equal to the number of UAVs. Each UAV is required to visit at least \( K \) and at most \( L \) targets, and then reach a final location. There are \( m \) final locations denoted by the set \( F = \{ f_1, f_2, \ldots, f_m \} \). The problem is to assign a sequence of targets to each vehicle such that its path contains one source vertex, a subset of target vertices and one final location. Given a set of \( m \) UAVs such that \( m > n \) where \( n \) is the number of targets to be monitored. The objective is to minimize the total distance travelled by all the vehicles.

\[ \text{Minimize} \quad \sum_{i \in S, j \in T} c_{ij} x_{ij} + \sum_{i \in T, j \in T, (i,j)} c_{ij} x_{ij} + \sum_{j \in F} c_{ij} x_{ij} \quad (1) \]

In this case \( c_{ij} \) is the cost between targets \( i \) and \( j \), \( x_{ij} \) is a binary variable, which is ‘1’ if the link between \( i \) and \( j \) is part of the optimal solution and ‘0’ otherwise. The vehicles share information acquired by their sensors. Using the updated information, each UAV can find the shortest paths in real-time which may need to be modified whenever obstacles are detected.

Our problem is an Open Multiple Depot Vehicle Routing Problem (OMDVRP) as UAVs end their journey at one of the specified final \( m \) locations and do not return to the starting depots. The problem is formulated as a variant of the model given by Kara et al. [20], where each source vertex has one UAV. The total number of UAVs would thus be the same as the number of source vertices. The model is solved subject to the following constraints.
subject to

\[
\sum_{j \in T} x_{ij} = 1, \quad \forall i \in S \tag{i} \\
\sum_{i \in F} x_{ij} = 1, \quad \forall j \in F \tag{ii} \\
\sum_{i \in T} x_{ij} = 1, \quad \forall j \in T \tag{iii} \\
\sum_{i \in T} x_{ij} = 1, \quad \forall j \in T \tag{iv} \\
u_i + (L - 2)\sum_{x_{ik} = 1} - \sum_{x_{ik} = 1} x_{ik} \leq L - 1, \quad \forall i \in T \tag{v} \\
u_i + \sum_{x_{ik} = 1} + (2 - K)\sum_{x_{ik} = 1} \geq 2, \quad \forall i \in T \tag{vi} \\
x_{ki} + x_{ik} \leq 1, \quad k \in S, i \in T \tag{vii} \\
u_i - u_j + Lx_{ij} + (L - 2)x_{ij} \leq L - 1, \quad i \neq j; i, j \in T \tag{viii} \\
x_{ij} \in \{0,1\} \quad \forall i, j \in S \cup T \cup F \tag{ix} \\
x_{ij}r_{ij} \leq r_{\text{threshold}} \quad \forall i \in S \cup T, j \in T \cup F \tag{x}
\]

It is to be noted that for each \( i \in S \) and \( j \in F \), one inward and one outward edge is guaranteed by constraints (i) and (ii). This is true as one UAV starts from each source vertex and terminates its flight at one of the final locations. The degree of each of these vertices is one. Equations (iii) and (iv) are the degree constraints for the target node ensuring that each node entered must also be left. Thus, each target node has a degree of two. Constraints (v) and (vi) impose bounds on the number of target nodes a UAV visits together with initializing the value of the \( u_i \) as 1 if \( i \) is the first target visited on the mission. For any UAV, \( u_i \) is the sequence number of targets visited on that vehicle’s path from the source vertex up to the final location (i.e. visit number of the \( i \)th target), \( L \) is the maximum number of targets an agent may visit; thus \( 1 \leq u_i \leq L \) for all \( i \geq k \). In addition, if \( K \) is the minimum number of nodes an agent must visit, i.e., if \( x_{ik} = 1 \), then \( K \leq u_i \leq L \) must be satisfied.

Constraint (vi) prevents a UAV path consisting of only one vertex from \( S \) and one vertex from \( F \), since such a tour does not visit any target. Finally, constraint (viii) is the Sub Elimination Constraints (SECs) that break all sub-tours between target nodes. Each path contains exactly one vertex from \( S \), one vertex from \( F \) and a subset of vertices from \( T \). Constraint (x) allows only links with risk factor \( r_{ij} \leq r_{\text{threshold}} \) to be used in the solution. The risk factor assigned to a particular link will be based on information collected by the sensors. \( r_{ij} \) is the risk factor of the link joining nodes \( i \) and \( j \). It is to be assigned a value between 1 and 10 based on how risky traversal along that arc is considered. In this formulation, a link will not be used for traversal if the risk associated with it is greater than 9. Table 1 provides the details of variables and constraints for the formulation in [22], referred to as AWL, as well as our formulation. The working is shown for the case where UAVs do not return to their depots. However, they must end their journey at one of the designated end point locations. Table 2 shows a comparison of the two formulations in terms of the total variables and constraints for the benchmarks of VRP [24] and TSP [25] used in this work. As can be seen from the table, there is a significant reduction in the number of variables and constraints.

| Variables/ Constraints | AWL formulation | Our formulation |
|-------------------------|----------------|-----------------|
| Binary variables        | \( mn^2 + 2mn \) | \((n-m)^2\) |
| Continuous variables    | \( mn \)         | \((n-m)^2 + (n-m)\) |
| Total variables         | \( mn^2 + 3mn \) | \(2(n-m)^2 + (n-m)\) |
| Total constraints       | \( mn(n-1)+2n+n \) | \(2n(n-2)m[2+2m+n(m-3)]\) |

Table 1. List of variables and constraints for AWH and our formulation.

| UAVs (m) | Targets (n) | Benchmark |
|----------|-------------|-----------|
| 4        | 32          | A-n32-45  |
| 5        | 38          | A-n38-45  |
| 5        | 45          | A-n45-46  |
| 5        | 54          | A-n54-45  |
| 6        | 64          | A-n64-49  |
| 6        | 69          | A-n69-49  |
| 5        | 80          | A-n80-410 |
| 4        | 22          | E-n22-44  |
| 4        | 33          | E-n33-44  |
| 5        | 51          | E-n51-45  |
| 5        | 76          | E-n76-47  |
| 5        | 101         | E-n101-48 |

Table 2. Variables and constraints for the benchmark instances used.
4. Simulation Results

An integer programming model based on the problem formulated in Section 3 was simulated using Lingo 11.0 optimization software to study the behaviour of UAV teams and the time required to solve various instances to optimality. To the best of our knowledge no specific benchmarks have been proposed for open vehicle routing problems. VRP benchmarks without any hard constraints have been used to verify the performance of the MILP model.

4.1 Static Environments

Experiments were carried out to study the different aspects of the path planning problem. A team of identical UAVs starts from the source vertices, information relating to the environment is known a-priori and it is thus static. Vehicles complete the mission without suffering any losses and their journey terminates when the UAVs reach the final locations. The optimal plan generated using the MILP formulation is checked for cross-over using a Path Analyzer in Matlab. If there are no intersections between the paths of any vehicles, the mission is executed. However, in case vehicle paths are found to cross-over, the time interval between crossings is computed using the heading angle between successive targets on a given path and the UAV speed. If the interval is found to be greater than $t_{\text{threshold}}$ seconds, no change is needed. However, if the interval between crossings is less than $t_{\text{threshold}}$ seconds, the velocity of one of the UAVs is reduced such that the time difference becomes equal to or greater than $t_{\text{threshold}}$. For example, it would not be harmful if the vehicles cross-over the same point with some time difference, but it could cause a collision of the vehicles if the cross-over occurs at the same time. If UAV$_1$ crosses the point $x_i, y_i$ at time $t_i$ while UAV$_2$ crosses the same point at time $t_2$, it would cause no harm if $t_2 - t_1 \geq t_{\text{threshold}}$. For example, a value of 0.5 for $t_{\text{threshold}}$ would be sufficient as UAV$_1$ and UAV$_2$ would be 50 meters apart if both are moving with a velocity of 100 meters/second. The paths thus obtained are collision-free and avoid known threat zones by using only those links which have an associated risk factor $r_{ij} \leq t_{\text{threshold}}$.

Figure 1 shows the allocation results of a VRP benchmark a-n32-k5 among four vehicles using Ortho Cluster Algorithm Allocation (OCA_Alloc) [3]. The start locations are the first four coordinates in the benchmark respectively and are marked as ‘0’ and the maximum path length is 176 units. A modified version of the Matlab code by Joseph Kirk available at www.mathworks.com/matlabcentral [26] was also used, which computes the path using a genetic algorithm. It will be referred to as JK implementation in our work. This gives a maximum path length of 134, when evolved over 5000 generations. A plot corresponding to this is shown in Figure 2.

However, the maximum path length is further reduced to 121 units when the MILP model is solved using Lingo11.0 as shown in Figure 3. This indicates an improvement of 31% over OCA_Alloc [3]. The results in Table 3 show the average value of 30 simulations for each instance. All except the first benchmark used five UAVs. A comparison of the path length with an auction-based representative algorithm, OCA_Alloc [3] indicates that a performance enhancement of up to 34% is achieved. The last column shows the time taken to solve the benchmark to optimality. For the instances kroa100 and kroa200, ‘*’ in the MILP solution time column shows that an optimal solution could not be obtained within a time limit of three hours.
| Benchmark    | Maximum Path Length | Ratio | time |
|--------------|---------------------|-------|------|
|              | OCA-Alloc (a)       | MILP (c) | (a-c)/a | (sec) |
| a-n32-k5     | 176                 | 121    | 31%  | 0.23 |
| a-n45-k6     | 180                 | 119    | 34%  | 10   |
| a-n64-k9     | 176                 | 127    | 28%  | 44   |
| a-n80-k10    | 177                 | 153    | 13%  | 5    |
| kroa100      | 6953                | 5409   | 22%  | *    |
| kroa150      | 7567                | 6619   | 13%  | 397  |
| kroa200      | 8209                | 6197   | 25%  | *    |

Table 3. Performance of OCA_Alloc versus MILP.

| Benchmark    | Maximum Path Length | Ratio | time |
|--------------|---------------------|-------|------|
|              | JK (b)              | MILP (c) | (b-c)/b | (sec) |
| A-n32-k5     | 134                 | 121    | 10%  | 0.23 |
| A-n38-k5     | 139                 | 105    | 24%  | 5    |
| A-n45-k6     | 164                 | 119    | 27%  | 10   |
| A-n54-k5     | 194                 | 141    | 27%  | 1.1  |
| A-n64-k9     | 189                 | 159    | 16%  | 44   |
| A-n90-k9     | 283                 | 263    | 7%   | 5    |
| A-n80-k10    | 232                 | 153    | 34%  | 5    |
| E-n22-k4     | 87                  | 81     | 7%   | 0.1  |
| E-n33-k4     | 154                 | 133    | 14%  | 4    |
| E-n51-k5     | 173                 | 104    | 40%  | 6    |
| E-n76-k7     | 183                 | 169    | 8%   | 21   |
| E-n101-k8    | 205                 | 156    | 24%  | 94   |
| kroa100      | 7225                | 5409   | 25%  | *    |
| kroa150      | 9233                | 6619   | 28%  | 397  |
| kroa200      | 13020               | 6197   | 52%  | *    |

Table 4. Comparison of JK and MILP.

The optimization software Lingo 11.0 has been used to solve each formulation on a Core2 Duo 2.20 GHz processor running Windows 7.0. The results indicate our approach can be used to solve moderate-size path planning problems in real-time situations. Table 4 shows the performance of the JK implementation and the MILP model. Each instance was solved 30 times and the average value obtained is shown.

More experiments were carried out to compare the performance of solutions obtained using the MILP model with the benchmarks given in [3]. Figure 4 depicts the paths for a-n80-k10, indicating the maximum distance travelled as 153 units. The paths are conflict-free with reasonably balanced loads. The start locations are the first five coordinates of the benchmark.

However for large-sized problems, it sometimes becomes difficult to find the best solution in a limited time. The feasible path found for the instance of kroa100 at the end of three hours has been plotted in Figure 5. Though intersections are seen in the paths, these cross-overs are allowed after analysis by the Path Analyzer. An alternate path-planning strategy for large-sized problems was to partition the entire region of the benchmark into reasonably small-sized chunks. The path for each partition is then planned which takes a comparatively smaller time, as can be seen from the last column in Table 5. These results indicate that the solution, though not optimal, may still be better than those obtained with the use of other methods such as OCA-Alloc and JK implementation.

| Benchmark    | Without partitioning | With partitions (2-4 chunks) |
|--------------|---------------------|----------------------------|
|              | OCA-Alloc [J/K] MILP | MILP solve time |
|              | Max Path length     | Max Path length | Solve time |
| kroa100      | 6953                | 7225            | 5409 | *     | 6362   | 71+29 |
| kroa150      | -                   | 7472            | 5839 | 270   | 6716   | 86+31 |
| kroa200      | -                   | 7580            | 6323 | 315   | 6880   | 90+123 |
| kroa150      | -                   | 7684            | 5194 | 684   | 6852   | 130+124 |
| kroa150      | 7567                | 9233            | 6619 | 397   | 6973   | 90+30+52 |
| kroa200      | 8209                | 13020           | 6197 | *     | 7805   | 74+30+42+85 |

Table 5. Reduced solution times for MILP with partitioning.
Figure 6. Path planned for partition 1 of kroa100 benchmark

The entire region for the benchmark kroa100 is partitioned into two parts. Path planning for both can be done simultaneously. Each partition shows three paths. The start (4,1) and end (15,22) nodes for two vehicles are contained in partition 1 as can be seen from Figure 6.

Figure 7. Path planned for partition 2 of kroa100 benchmark.

However, for the third vehicle, the start node (2) lies in partition 2, while the end node (6) lies in partition 1. So the third vehicle will travel a length of 4342 (node 2-23) plus 2020 (node 23-6), total distance of 6362 units. Figure 7 shows that the fourth and fifth vehicles have their start (3,5) and end (71,52) nodes in partition 2 and hence travel a shorter distance. Thus, the total number of vehicles used is five. The traversal path of each can be seen in Figures 6 & 7.

4.2 Uncertain Environments

In the next set of experiments, teams of UAVs start from source vertices and move along their paths a certain distance. Since UAVs used for surveillance missions may be lost or significantly damaged during operations, employing multiple UAVs can increase the chances of success. However, their efficiency depends on the extent of their collaboration. In such cases, the surviving vehicles must collaborate to share the load of the destroyed vehicle.

Figure 8. Path re-computed after the loss of one UAV.

Figure 8 depicts this scenario for the benchmark a-n32-k5, where one UAV has been lost and the remaining three collaborate to complete the mission. The graph originally used to solve the model is then modified. The target nodes already visited by the team are deleted along with their links. This is evident from Figure 8, where the four start nodes as well as the visited target nodes (9,13,20,23,27) have been deleted. The present positions of the vehicles are taken as the depot nodes and the path plan is recomputed in order for them to visit the remaining targets. However, the complexity of the problem is reduced as the problem size is smaller than the original one and consequently the time required to find a new solution is reduced.

Figure 9. Path re-computed after the loss of one UAV.

Figure 9 shows the loss of one UAV being shared by the remaining four for the benchmark kroa100. Re-computing the path took thirty-six seconds.

Lastly, during the mission, changes in the environment may be detected. Ellipsoidal obstacles may be revealed along the traversal path and must be avoided. For obstacle avoidance, the path is slightly changed by adding two way-points tangential to the ellipse. The targets on either side of the ellipsoidal obstacle are now connected through one of these way-points. The modified path will consist of two line segments replacing the single line connecting the two consecutive targets representing...
the UAVs’ traversal path. In this way, only a slight modification of the path occurs which does not affect the optimality of the solution in any major way. Figure 10 illustrates the changes in the planned path as obstacles pop up. The mission now takes a little longer to complete. Figure 11 illustrates the avoidance of obstacles that are detected along the travel path for kroa100.

![Figure 10. Altered path plan to avoid obstacles.](image1)

![Figure 11. Obstacle avoidance for larger instances.](image2)

5. Conclusions

A real-time optimization solution for UAV path planning in dynamic situations is discussed. A MILP model which can be readily solved with commercially available tools is proposed. The vehicles carry on-board sensors which can collect information, as well as communication devices to transmit the data.

Initially an optimal path is computed by the computing resources on-board each UAV. The vehicles traverse multiple target regions and avoid collision with obstacles and other team members in a dynamic hostile environment. Most of the computational processing and communication must be carried out by each vehicle in real-time on-board. These UAVs operate in situations where the available information about the environment is known with a limited level of certainty. Failures of system components are likely to occur during missions, e.g. failures in communication links or the loss of a team member. The planning-system must be able to adapt to unexpected failures and reconfigure the mission plan. This will enhance the chances of achieving success in completing a particular mission.

6. Acknowledgments

We are grateful to Professor Ping-an Gao and Professor Fanggeng Zhao for providing us with their test problems and answering our queries regarding their approach. We would also like to thank Joseph Kirk for help with the open source Matlab code on matlabcentral. The research was funded and supported by the University of Engineering and Technology, Taxila, Pakistan.

7. References

[1] M. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak and A. Kleywegt, “Simple auctions with performance guarantees for multi-robot task allocation,” in proc. IEEE/RSJ. Intl. Conf on Intelligent Robots and Systems, 2004, pp. 698-705.

[2] S. Sariel, T. R. Balch and N. Ergoden, “Multiple Travelling Robot Problem: A Solution Based on Dynamic Task Selection and Robust Execution,” IEEE Transactions on Mechatronics, vol. 14, no. 2, 2009, pp. 198-206.

[3] P. Gao, Z. Ca and L. Yu, “Evolutionary Computation Approach to Decentralized Multi-Robot Task Allocation,” in proc. the 5th International Conference on Natural Computing, 2009, pp. 415-419.

[4] S. Rathinam, R. Sengupta and S. Darbha, “A Resource Allocation Algorithm for Multi-Vehicle Systems with Nonholonomic Constraints,” IEEE Transactions on Automation Science and Engineering, vol 4, no. 1, 2007, pp. 98-104.

[5] T. Maddula, A. A. Minai and M. N. Polycarpou, “Multi-Target Assignment and Path Planning for Groups of UAVs,” in Cooperative Control and Optimization, S. Batenkhu, R. Murphy and P. Parolac, editors. Kluwer Academic Publishers, 2004, pp. 261-272.

[6] B. Brumitt and A. Stenz, “GRAMMPS: a Generalized Mission Planner for Multiple Mobile Robots” in proc. IEEE International Conference on Robotics and Automation, vol. 2, 1998, pp. 1564-1571.

[7] R. Cui, B. Gao and J. Guo, “Pareto-optimal Coordination of Multiple Robots with Safety Guarantees,” Autonomous Robots, vol. 32, no. 3, 2012, pp. 189-205.

[8] S. M. LaValle, Planning algorithms. Cambridge : Cambridge University Press. 2006.

[9] V. R. Desaraju and J. P. How, “Decentralized Path Planning for Multi-agent teams with Complex Constraints,” Autonomous Robots, vol. 33, no. 4, 2012, pp. 385-403.

[10] T. Ousingsawat and M.E. Campbell, “Multiple Vehicle Team Tasking for Cooperative Estimation” in proc. American Control Conference. 2004.[Online]. Available:citeseer.ist.psu.edu doi :10.1.1.5.4025.
[11] T. Schouwenaars, B. De Moore, E. Feron and J. P. How, “Mixed Integer Programming for Multi-Vehicle Path Planning,” in proc. the European Control Conference, 2001, pp. 2603-2608.

[12] J. S. Bellingham, M. Tillerson, M. Alighanbari and J. P. How, “Cooperative Path Planning for multiple UAVs in Dynamic and Uncertain Environments,” in proc. 41st IEEE Conference on Decision and Control, vol. 3, 2002, pp. 2816-2822.

[13] J. S. Bellingham, A. Richards and J. P. How, “Receding Horizon Control of Autonomous Aerial Vehicles,” in proc. the American Control Conference, vol. 5, 2002, pp. 3741-3746.

[14] S. Rathinam and R. Sengupta, “Lower and Upper Bounds for a Symmetric Multiple Depot, Multiple Travelling Salesman Problem,” Research Report, Institute of Transportation Studies, University of California, Berkeley, 2006.

[15] M. Held, and R. P. Karp, “The Travelling-Salesman Problem and Minimum Spanning Trees,” Operations Research, vol. 18, 1970, pp. 1138-1162.

[16] T. Bektas, “The Multiple Traveling Salesman Problem: an overview of formulation and solution procedures,” International Journal of Management Science, vol. 34, no. 3, 2006, pp. 209-219.

[17] C. J. Schumacher, P. R. Chandler and S. R. Rasmussen, “Task Allocation for Wide Area Search Munitions via Network Flow Optimization,” in proc. AIAA Guidance, Navigation, Control Conference and Exhibit, Paper 2001-4147, 2001.

[18] P. R. Chandler, M. Pachter, D. Swaroop, J. M. Fowler, J.K. Howlett, S. R. Rasmussen, C. J. Schumacher and K. Nygard, “Complexity in UAV co-operative control,” in proc. the Amer. Control Conference, 2002, pp. 1831-1836.

[19] J. S. Bellingham, M. Tillerson, A. Richards and J. P. How, “Multi-task allocation and trajectory design for co-operating UAVs,” presented at the Co-operative Control: Models, Applications and Algorithms at the Conference on Coordination, Control and Optimization, 2001, pp. 1-19.

[20] I. Kara and T. Bektas, “Integer Linear Formulations of Multiple Salesman Problems and its Variations,” European Journal of Operational Research, vol. 174, no. 3, 2006, pp. 1449-1458.

[21] Y. Kim, D-W. Gu and I. Postlethwaite, “Real-time Optimal Mission Scheduling and Flight Path Selection,” IEEE Trans. Autom. Control, vol. 52, no. 6, 2007, pp. 1110-1123.

[22] B. Alidaee, H. Wan, and F. Landram, “A Note on Integer Programming Formulations of the Real-time optimal scheduling and flight path selection,” IEEE Trans. On Control System Tech., vol.17, no. 4, 2009, pp. 839-843.

[23] M. Julia, A. Gil and O. Reinoso, “A Comparison of Path-planning Strategies for Autonomous Exploration and Mapping of Unknown Environments,” Autonomous Robots, vol. 33, no. 4, 2012, pp. 427-444.

[24] http://www.branchandcut.org/VRP/data

[25] http://elib.zib.de/pub/mp-testdata/tsplib/ tsplib.html

[26] http://www.mathworks.com/matlabcentral/fileexchange /21301.

[27] Lingo 11 User’s Guide, LINDO Systems Inc. 2008