Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms

Eugene Edison, Tal Shima *

Technion – Israel Institute of Technology, Department of Aerospace Engineering, Haifa 32000, Israel

1. Introduction

Autonomous operation of uninhabited aerial vehicles (UAVs) is a problem of much recent interest. Having this capability will allow UAVs to perform missions with minimal or no human intervention. Such military and civilian missions may include intelligence gathering, target tracking, and rescue missions. Moreover, it will enable new operational paradigms for employing numerous such vehicles in groups, performing complex scenarios with robust endurance. The main motivation for team cooperation stems from the possible synergy, as the group performance is expected to exceed the sum of the performance of the individual agents. However, some basic challenges arise during simultaneous operation of autonomous vehicles. The interdependency of the cooperative multiple task assignment and the problem of trajectory optimization in multi-vehicle systems is one of them. Usually, task assignment is performed according to a cost that refers to vehicle path length, which depends on the assigned tasks. To ensure that mutual dependencies are properly managed, the agents in the system have to coordinate motion planning and task assignment with each other in real-time.

Complexity is one of the main features of cooperative problems involving vehicles with kinematic constraints. Beyond the common factors that may induce the cooperative problem complexity, such as the size of the problem (e.g., number of vehicles, tasks, and constraints), the complexity is affected by the coupling between the performance of different tasks [1]. For example, the cooperative operation of multiple autonomous vehicles, which perform a set of common missions, such as combat intelligence, surveillance, and reconnaissance (ISR), introduces strong interdependency in their trajectories. This coupling between the trajectories of different team members greatly increases the problem complexity and requires multi-vehicle path coordination.

Motion planning [2,3] is a long studied problem in robotics that can be tackled for example by using graph and optimal control theories. It involves the search for a collision free path and trajectory optimization by taking into consideration the geometry of the vehicle and its surroundings, the vehicle's kinematic and dynamic constraints, and any other external constraints that may affect the resulting path. These restrictions on a planned path are considered since the majority of practical applications involving physical machines require smooth feasible paths. The feasible routes of minimal length for a planar nonholonomic vehicle, constrained to move at a constant speed along paths of bounded...
curvature without reversing direction, were introduced by Dubins [4]. A feasible trajectory for such a vehicle, denoted as a Dubins car, should be defined as a twice differentiable curve (almost everywhere), such that the magnitude of its curvature is bounded above by $1/r$, where $r > 0$ is the vehicle’s minimum turn radius.

Task assignment is one of the fundamental combinatorial optimization problems in the branch of operations research in mathematics [5]. The general problem refers to assign a number of agents to perform a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment algorithm. It is required to perform all tasks in such a way that the total cost of the assignment is minimized. An assignment task might be presented as a problem in graph theory [6]. Many of the problems in graph theory can be solved using search algorithms. The data in the graph is represented by vertices and edges. Basically, a vertex is taken from a data structure, its neighbors are examined, and possibly added to the data structure. Then, the search is conducted by exploring the data structure level by level (breadth-first search) or reaching leaf node first and backtracking (depth-first search). Other examples of deterministic graph search algorithms include minimum weighted path search like Dijkstra [7], and Bellman-Ford [8].

The major difficulty in solving the general assignment problem [5] is that it belongs to a class of NP-hard combinatorial optimization problems and therefore could not be solved in polynomial time by deterministic methods. So, due to the prohibitive computational complexity of the problem, the traditional deterministic search algorithms provide an optimal solution only for small-sized problems. For large sized problems it may provide a feasible solution and low/high bounds on complexity. In contrast, randomized search algorithms [9] (or stochastic algorithms) are algorithms which employ a degree of randomness as part of their logic. An algorithm of this kind typically uses a random input to guide its behavior in the hope of achieving good performance in the “average case” and get a good solution in the expected runtime. One of the efficient ways to solve such a problem by an evolution inspired stochastic approach is genetic algorithm (GA) [10,11]. The evolution usually starts from a population of randomly generated solutions and evolves in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are randomly selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population. This population is then used in the next iteration of the algorithm.

The cooperative multiple task assignment problem (CMTAP) [12], associated with the autonomous operation of a group of heterogeneous fixed wing UAVs performing multiple tasks on multiple heterogenous stationary ground targets, is the focus of the current study. Several combinatorial optimization methods have been customized for solving CMTAP, including mixed integer linear programming (MILP) [13,14], the capacitated transhipment network solver [15,16], tree search [17], and genetic algorithm [18]. Most of these methods involve integer assignment decision variables and continuous controllers that are responsible for specific guidance of each vehicle. The main characteristic of the proposed optimization techniques is hierarchical decoupling of the problem of task assignment from that of trajectory optimization, which stems from the need to find a tractable solution for this computationally prohibitive problem. This approach always results in a suboptimal solution. For example, the task assignment algorithm in Ref. [15], is optimal only for the current tasks, without considering the later tasks; similarly, the MILP based algorithm presented in Ref. [14] obtains piecewise optimal trajectories by planning motion only for one task at a time. The iterative network flow model of Ref. [19] is designed to plan multiple task assignments, but also utilizes a decoupling approach and therefore optimality is not achieved. The deterministic solution method proposed in Ref. [13] solves the entire problem without decoupling it, but uses Euclidean distances instead of computing flyable trajectories. In Ref. [17] a tree search algorithm was developed that produces the suboptimal solution to the assignment problem based on piecewise optimal trajectories. These trajectories are obtained by setting the initial heading angle condition on the flight direction of a UAV for each segment based on the final direction obtained in the previous segment. This results in trajectories that are optimal only from one task to the next, but the entire trajectory is not necessarily optimal. In another work, Ref. [18], the same piecewise optimal trajectories methodology was used and a GA was proposed for the stochastic search of the space of feasible assignments.

In this paper, unlike the aforementioned works, we concentrate on the cooperative multiple task assignment coupled with the problem of trajectory optimization for the UAV team servicing a set of heterogeneous targets. The GAs, derived here, are based on discretization of possible heading angles of a vehicle while flying over the target. This new approach enables to obtain good fast feasible solutions of coupled CMTAP that improve monotonically towards optimality and constitutes the main contribution of the current work.

The remainder of this paper is organized as follows: in the next section the formulation of the CMTAP is given. It includes denotation of the problem components and is followed by its graph representation. In the following section deterministic search is studied. Then, a GA is proposed for solving the CMTAP with homogeneous vehicles. Computational complexity and examples are also provided. The GA is extended in the next section, to cover scenarios that involve a group of heterogeneous UAVs. Monte Carlo simulation runs are described in the sequel. Concluding remarks are offered in the last section.

2. Problem formulation

In this section a formulation of the coupled cooperative task assignment and path planning problem is presented. The CMTAP is considered for scenarios involving a group of heterogeneous UAVs that performs a set of predefined missions on known ground targets. The heterogeneity of the UAVs is expressed in terms of their operational capabilities and kinematic constraints. The CMTAP needs to be solved each time a UAV enters or leaves the group and when new targets are found or a task fails.

2.1. Targets and tasks

Let

$$T = \{T_1, T_2, \ldots, T_{N_t}\} \quad (1)$$

be the set of $N_t$ stationary ground targets, with known positions, designated to the UAV group, possibly by a human operator.

We define the proper mission termination as the completion of all predefined tasks on all the targets, while each target has different tasks that need to be performed on it. In this assignment problem the UAVs are required to visit each target (i.e. to fly over the target) for completion of a set of $N_T$ tasks

$$M_T \subseteq M = \{C, A, V\} \quad (2)$$

where $C, A,$ and $V$ denote classify, attack, and verify, respectively. Moreover, there is a precedence requirement, as for example a
target cannot be attacked prior to being classified and cannot be verified before being attacked.

Let \( N_m \) denote the number of tasks that needs to be performed on target \( T \in \mathbf{T} \), such that \( N_m = ||M_T|| \), where each task \( k \in M_T \) must be performed only once. Hence, the number of tasks that all the UAVs are required to perform throughout the scenario is given by

\[
N_c = \sum_{k=1}^{N_T} N_m_k
\]  

(3)

For the simplicity of presentation, when all the targets need to be serviced an identical number of times, we denote the number of tasks on each target \( N_m \), as \( N_m \). In this case the number of all tasks throughout the scenario, see Eq. (3), simplifies to \( N_c = N_T N_m \).

2.2. Vehicles

Let

\[
U = \{U_1, U_2, \ldots, U_{N_U}\}
\]  

be the set of \( N_U \) cooperating heterogeneous fixed wing UAVs. In this problem, UAVs are required to perform a number of tasks under their specialty \( S \). We discuss three vehicle specialties, called combat UAV, surveillance UAV, and munition UAV, but other specialty types can be easily defined. Each type defines a vehicle capable of performing a different set of tasks. For example, the UAV of the first type, combat, can perform all tasks from the set \( M \); the UAV of the second type can perform only surveillance missions, i.e. classification and verification tasks, but has no attacking ability; and the munition UAV is only able to attack targets, see Table 1.

For the sake of simplicity we will assume that (1) the UAVs flight can be confined to a plane at a given altitude; (2) the involved UAVs have collision free paths achieved by altitude layering; (3) no wind disturbances; (4) each UAV airspeed is constant (5) there are no constraints on fuel consumption and required weapons.

We also assume that the vehicles spatial configuration can be defined by three states

\[
q = (x, y, \psi)
\]  

with the following equations of motion:

\[
x = v_U \cos \psi
\]  

(6a)

\[
y = v_U \sin \psi
\]  

(6b)

\[
\psi = c \Omega_{\text{max}}
\]  

(6c)

where \( x \) and \( y \) are a UAV’s horizontal coordinates in a Cartesian inertial reference frame; \( v_U \) is the constant speed; \( \Omega_{\text{max}} \) is the maximum turning rate of the vehicle; \( c \) is the steering command such that \(|c| \leq 1\); and \( \psi \) is the azimuth flight path angle, namely a heading, such that the reference plane is true north, considered 0 azimuth. Moving clockwise, a point due east has an azimuth of 90°, south 180°, and west 270°. The above model for representing the kinematics of a UAV is commonly denoted as a Dubins car model [2–4]. Following Ref. [4], the minimum length feasible curve for a Dubins car, from any arbitrary initial configuration, \((x_{\text{init}}, y_{\text{init}}, \psi_{\text{init}})\) to any arbitrary final configuration, \((x_{\text{final}}, y_{\text{final}}, \psi_{\text{final}})\), is either: (i) an arc of a circle of radius \( r_{\text{min}} \)

\[
r_{\text{min}} = v_U / \Omega_{\text{max}}
\]  

(7)

followed by a line segment, and then an arc of a circle of radius \( r_{\text{min}} \), or (ii) a sequence of three arcs of circles of radius \( r_{\text{min}} \), or (iii) a sub-path of a path of type (i) or (ii).

To specify minimum length feasible curves that represent a path of the Dubins car, we follow the same notations used in Ref. [4]. Three elementary motions are considered: turning to the left (denoted \( L \)), turning to the right (denoted \( R \)), and straight line motion (denoted \( S \)). Note that both \( L \) and \( R \) turns are along a circle of radius \( r_{\text{min}} \) that equals the UAV’s minimum turning radius. We define the Dubins set \( D \)

\[
D = \{LSL, RSR, RSL, LSR, RLR, LRL\}
\]  

(8)

which is the domain of the minimum length feasible curves between given initial and final configurations. For example, the notation \( LSL \) denotes a Dubins car trajectory composed of \( L, S, \) and \( L \) segments.

2.3. Graph representation

As it was mentioned before, the CMTAP is a computationally intractable problem. To enable an optimization process, we make an approximation of the CMTAP by representing it as a graph. This graph representation is based on heading angle discretization. The heading angle discretization set is defined as

\[
H = \{\psi; \psi_1 = 2 \pi i/N_{\psi}, i = 0, 1, \ldots, N_{\psi} - 1\}
\]  

(9)

where \( N_{\psi} > 0 \) is an integer defining the desired heading angle resolution. For example, for \( N_{\psi} = 8 \) the discretization set is \( H = \{0, 45°, 90°, 135°, 180°, 225°, 270°, 315°\} \) (illustrated in Fig. 1).

The heading of the presented UAV in this figure is equal to \( \psi = 45° \).

Using this heading angle discretization set we will now define the graph.

Let

\[
V_T = \{(T_1, \psi_1), \ldots, (T_k, \psi_k), \ldots, (T_N, \psi_N)\}
\]  

(10)

be the set of vertices in the graph, where each node consists of a target \( T \) identified by its position and heading angle \( \psi \in H \), i.e. a UAV’s spatial configuration \( q \) when it flies over the target. Thus, \( |V_T| = N_T N_{\psi} \).

---

**Table 1**

UAV capability of performing a task according to its specialty.

<table>
<thead>
<tr>
<th>Type</th>
<th>UAV</th>
<th>Specialty, ( S )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Combat</td>
<td>(C, A, V)</td>
</tr>
<tr>
<td>2</td>
<td>Surveillance</td>
<td>(C, V)</td>
</tr>
<tr>
<td>3</td>
<td>Munition</td>
<td>(A)</td>
</tr>
</tbody>
</table>

---

**Fig. 1.** Discretization set \( H = \{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\} \), and the UAV with heading angle \( \psi = 45° \).
Scenario: $U = \{U_1, U_2\}$
$T = \{T_1, T_2\}$
$H = \{0, 120^\circ, 240^\circ\}$
$N_{\text{ini}} = 3$
$N_T = N_{\text{ini}} = 6$

The set of vertices that designate initial positions and headings of the UAVs is

$$V_U = \{(U_1, \psi_{U_1}), (U_2, \psi_{U_2}), \ldots, (U_{N_U}, \psi_{U_{N_U}})\}$$

(11)

where $|V_U| = N_U$.

Thus, the set of all vertices in the graph is given by

$$V = V_T \cup V_U$$

(12)

where

$$|V| = N_T = N_U + N_T N_\psi$$

(13)

A directed edge connects any two nodes in $V_T$ and every node $V_U$ to all nodes in $V_T$. A path in the graph $V$ is intended to symbolize a route of a UAV starting from an initial configuration $\mathbf{q}_1 = (x_{\text{ini}}, y_{\text{ini}}, \psi_{\text{ini}})$ (a node in $V_U$) and ending in a final configuration $\mathbf{q}_f = (x_{\text{final}}, y_{\text{final}}, \psi_{\text{final}})$ (a node in $V_T$). The weight of each edge $e = (v_i, v_f)$, defined as $w_{(v_i, v_f)}$, equals to the Dubins curve length between two vehicle configurations contained in connected vertices, and is calculated by using the publicly released trajectory optimization subroutine of the MultiUAV2 simulation [12,20]. The set of edges in such multiple targets Dubins paths optimization graph is given by

$$E = \{(v_i, v_f) | v_i \in V, v_f \in V_T\}$$

(14)

where

$$|E| = N_T N_{\psi} (N_U + N_T N_\psi)$$

(15)

The above result is obtained by the summation of $N_T N_{\psi} (N_U + N_T N_\psi)$ edges that connect the nodes of $V_T$ with $N_T N_{\psi}$ self-loops of all vertices in $V_T$, and with $N_T N_{\psi} N_U$ edges connecting the initial configuration node of each UAV from $V_U$ to all nodes in $V_T$.

It should be noted that a trajectory obtained for a vehicle transition from any initial configuration, $\mathbf{q}_i = (x_{\text{ini}}, y_{\text{ini}}, \psi_{\text{ini}})$, to any final configuration, $\mathbf{q}_f = (x_{\text{final}}, y_{\text{final}}, \psi_{\text{final}})$, is identical to the inverse travel trajectory from $(x_{\text{final}}, y_{\text{final}}, \psi_{\text{final}} + \pi)$ to $(x_{\text{ini}}, y_{\text{ini}}, \psi_{\text{ini}} + \pi)$. This feature enables reducing the amount of edge weights precalculations by approximately half.

Fig. 2 presents an example of a graph representation of the CMTAP, along with a feasible solution, for a problem with two vehicles $\{U = \{U_1, U_2\}\}$ having to visit two targets $\{T = \{T_1, T_2\}\}$ and performing three tasks ($N_T = 3$) on each target. The green path indicates the flight route of the first UAV, $U_1$, that starts from its initial configuration, visits the first target, $T_1$, with heading $\psi = 120^\circ$, and completes its multiple assignment by visiting the second target, $T_2$, with the same heading. The purple path represents the flight route of the second vehicle, $U_2$, that visits the second target, $T_2$, with zero heading $\psi = 0^\circ$, proceeds to the first target, $T_1$, with the same heading angle, returns to $T_2$ with heading of $\psi = 240^\circ$ and flies back to $T_1$ to visit it with $\psi = 120^\circ$.

Note that the task precedence order is not important in the scenarios that involve only homogeneous UAVs because each UAV can perform any of the required tasks.

### 2.4. Combinatorial optimization problem

We choose the cumulative distance that all the UAVs travel, in order to fulfill the tasks of the coupled problem, as the cost function to be minimized

$$J = \sum_{i=1}^{N_U} \sum_{t=1}^{N_T} \sum_{i=1}^{N_T} \sum_{j=1}^{N_T} X_{(v_i, v_f)} w_{(v_i, v_f)}$$

(16)

where $X_{(v_i, v_f)}$ is a binary decision variable

$$X_{(v_i, v_f)} \in \{0, 1\}$$

(17)

The decision variable equals 1 if the edge $e = (v_i, v_f)$ is used to indicate the movement of the vehicle $U_i \in U$ from any configuration defined in $v_i$ to any other $v_f$, and is 0 otherwise. Note that only weights $w_{(v_i, v_f)}$ of the existing edges $e = (v_i, v_f) \in E$ in the graph have to be counted in the above formulation of $J$.

There are three constraints that must be satisfied throughout the solution process. The first

$$\sum_{i=1}^{N_U} \sum_{t=1}^{N_T} \sum_{i=1}^{N_T} X_{(v_i, v_f)} = N_{\text{ini}}, \quad \forall \ t \in T$$

(18)

refers to the exact number of tasks that have to be performed on each target during the scenario.

The second

The set $P_a = \{(v_i, v_f) | x^a_{(v_i, v_f)} = 1, v_i \in V, v_f \in V_T\}$

composes a path in the graph with start node from $V_U \forall U_i \in U$.

$$\sum_{i=1}^{N_U} \sum_{t=1}^{N_T} \sum_{i=1}^{N_T} X_{(v_i, v_f)} w_{(v_i, v_f)} = N_{\text{fin}}$$

(19)

is posed to verify that the solution constitutes a set of connected traceable vehicle paths in the multiple targets Dubins trajectories graph, see Fig. 2.

Finally, the third constraint

$$t_c \leq t_m \leq t_f, \quad \forall \ t \in T$$

$$t_m = \sum_{i=1}^{N_U} \sum_{t=1}^{N_T} t_{m_{(v_i, v_f)}} X_{(v_i, v_f)} w_{(v_i, v_f)}$$

(20)

is posed to preserve the precedence order for execution of tasks on each target. The task execution time is noted as $t_m$, where $m$ is one of the needed tasks from the set $M$. It is defined as the time required for a vehicle to arrive from its initial configuration, noted as $v_i = (U_i, \psi_{U_i})$ to the target for completing the task. This time is obtained through division of the distance, traversed by the vehicle by its constant speed, $v_0$. The distance the UAV travels until it arrives to perform a task $m \in M_T$ on a target $T \in T$ is calculated by the summation of weights of the edges that compose the UAV’s sub-path from its initial configuration until the $k$-th visitation in one of the nodes designated by the target

---

1. This a full directed sub-graph.

---

2. A list of graph vertices, where each vertex connected by an edge to the next vertex, defines a path in a graph.
3. Deterministic search

To illustrate the solution methodology of the coupled problem we present in this section some simplified subproblems and discuss how deterministic graph search techniques can be used to solve them. First, using the heading angle discretization set that was defined in the previous section, we represent the path optimization problem for a single UAV visiting a number of targets with predefined visitation order as a directed acyclic graph. After that, the shortest-path search algorithm is used for the problem solution, i.e. finding a Dubins path via multiple targets under the predetermined level of discretization.

3.1. Given visitation ordering

We will assume here that for each UAV, the targets and their visitation order are given, and are not a part of the problem. Thus, for each vehicle we are concerned only with finding the shortest path and not with task assignment. Although this problem is significantly simpler than CMTAP, obtaining a closed form solution for calculating the shortest path of a Dubins car that is traversing multiple waypoints is an open problem. Several numerical solutions of this problem were obtained using optimal control methods [21], without approximations. These methods endure computational inefficiency and do not allow one to achieve real-time solutions. In Ref. [22] the two-point optimal Dubins path solution was extended to consider three consecutive waypoints, with several variations. The resulting three-point paths are restricted to having a free terminal orientation.

3.1.1. Upper bound

Solving the path optimization problem, with a given level of discretization, is actually an upper bound for the Dubins path optimization problem involving one vehicle and more than two targets.

Consider a UAV with minimum turning radius of \( r_{\text{min}} = 200 \text{ m} \), initially positioned in the origin and headed to 0°. It has to once visit three targets \( \{T_1, T_2, T_3\} \) with known positions. The visitation order is given by the sequence \( T_1 \rightarrow T_2 \rightarrow T_3 \) and the discretization set is \( H = \{0°, 120°, 240°\} \). To obtain the path, we build a directed acyclic graph (DAG). Every vertex of the constructed DAG will be identified by the target position (or alternatively by the target ID number) and by the heading angle of the vehicle when it flies over the target. The root of the DAG denotes the UAV initial configuration. Every directed edge in the graph connects between an ancestor, that is the previous position and heading of the vehicle, to a descendant, that is the next position and heading. The edge weight is the Dubins distance between two configurations. In such a way, a path from the initial configuration of the vehicle to the last node describing the target position and heading, is a multiple-points Dubins path under the provided heading angle discretization. The solution of the described problem, i.e. the Dubins path of minimal length, can be achieved by the use of some shortest-path search algorithm, such as the Dijkstra algorithm. The graph solution can be easily converted to UAV’s physical trajectory that is illustrated in Fig. 3. The presented trajectory consists of point-to-point Dubins paths between every pair of positions, and under the given discretization its shortest length is 1927 m.

![UAV's trajectory](image)

Fig. 3. UAV’s trajectory obtained by using Dijkstra algorithm on DAG of a simplified problem when the visitation order is given; \( r_{\text{min}} = 200 \text{ m} \), \( N_p = 3 \), \( N_m = 1 \). Length = 1927 m.
Providing a finer discretization enables a more precise solution. For example, in Fig. 4 the same scenario is investigated with a level of discretization of $N_c = 36$. The obtained trajectory is of length 1667 m.

However, there is a diminishing return for such an effort. The number of edges grows not only as a function of the number of targets, but also with the level of discretization. So, the finer the discretization is, the more complex the problem becomes, and therefore, the runtime required to solve the problem is longer, occasionally without much benefit. For example a tenfold increase of the heading angle discretization accuracy to $N_c = 360$ (corresponding to the discretization of $1^\circ$) yields a similar trajectory to that of Fig. 4, with small changes of the UAV’s heading at each target. This has a minor effect on the length of the trajectory, reducing it by only 8 m (making it 1659 m).

3.1.2. Lower bound

Loosening the kinematic constraints, i.e. setting $r_{\text{min}} = 0$, gives a vehicle the ability to turn on the spot and results in the Euclidian shortest path through the targets with the given order of visitation ($T_1 \rightarrow T_2 \rightarrow T_3$). Moreover, the absence of kinematic constraints eliminates the need for heading angle discretization. Thus, the obtained trajectory of 668 m, see Fig. 5, forms the lower bound on all feasible paths for vehicles with $r_{\text{min}} \geq 0$ m.

3.2. Undetermined visitation order

The path optimization of a Dubins car that passes through multiple destinations without a predetermined visitation order is a more complex problem. It might be abstracted as a single traveling salesman problem with kinematic constraints. The significant difference of the graph for this problem from the one discussed in the previous subsection is that the number of target visits is not fixed a priori, requiring an additional step in the solution process to determine the order of visitation.
additional edges, indicating all possible feasible routes, grows significantly. For instance, routes exist from the initial position of the vehicle to each of the target positions with all possible headings.

To obtain the optimal minimum length path for the discussed scenario (one vehicle, three targets, one task per target, and \( N_c = 36 \)) the comparison table, Table 2, for all of the possible assignments is presented. The optimal path yielded by the graph search method is obtained for the visitation order of \( T_1 \rightarrow T_3 \rightarrow T_2 \). The trajectory and the visitation angles are the ones depicted in Fig. 6.

Although the deterministic approach results in an optimal solution of the approximate problem (that is actually an upper bound for the Dubins path optimization problem involving more than two targets), it can be used only for small-sized scenarios, such as finding the shortest path for a single vehicle between a small number of targets (two or three). A large number of targets and high degree of discretization make the problem computationally prohibitive, even for a single vehicle. Therefore, the need to obtain solutions for varied kinds of scenarios, which involve a large number of targets and vehicles, leads us to seek a stochastic search method. Such an algorithm is presented in the next section.

4. GA for homogeneous CMTAP

In this section we present a GA [10,11] for efficiently searching the space of solutions. Being a global search stochastic method it is used for obtaining a good solution to the problem, not necessarily the optimal one.

4.1. Methodology

At first, many individual solutions are randomly generated to form an initial population, covering the range of possible solutions (the search space). For the simple GA the population is constructed as an array of individuals where each individual, called a chromosome, is a solution encoded naturally or by a bit string. The population size depends on the nature of the problem. Individual solutions are selected from the population through a fitness-based process. The selection process has a random element, and there is a larger probability to choose more fit solutions. Nonetheless, even less fit solutions have a chance of being selected. This helps in keeping the diversity of the population large, preventing premature convergence to poor solutions and getting trapped into a local minimum.

The next step is to use the selected solutions to generate a second generation of solutions. For this process we use the operators: crossover (also called recombination) on each set of two selected solutions, denoted as parents, and/or mutation. A new child solution, produced using the above methods of crossover and mutation, typically shares many of the characteristics of its parents. New parents are selected for each child, and the process continues until a new population of solutions of appropriate size is generated. This generational process is repeated until a termination condition is reached.

4.2. Encoding

The most critical part of a GA is the solution chromosome encoding. For the problem where all involved UAVs are homogeneous (homogeneous CMTAP), the encoding of the solution is based on the target visitation order, by each vehicle, with a heading angle selected from the set \( H \). Thus, every gene corresponds to a specific configuration of an assigned vehicle.

<table>
<thead>
<tr>
<th>Visit order</th>
<th>Path length (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1→2→3</td>
<td>1667</td>
</tr>
<tr>
<td>1→3→2</td>
<td>1483</td>
</tr>
<tr>
<td>2→1→3</td>
<td>2553</td>
</tr>
<tr>
<td>2→3→1</td>
<td>2512</td>
</tr>
<tr>
<td>3→2→1</td>
<td>1583</td>
</tr>
<tr>
<td>3→1→2</td>
<td>1720</td>
</tr>
</tbody>
</table>

Table 2
Graph search results for all assignment paths; \( r_{\text{min}}=200 \text{ m}, N_c = 36, N_m=1. \)

Fig. 6. Optimal assignment Dubins trajectory; \( r_{\text{min}}=200 \text{ m}, N_c = 36, N_m=1. \) Length = 1483 m.

Please cite this article as: Edison E, Shima T. Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. Computers and Operations Research (2010), doi:10.1016/j.cor.2010.06.001
therefore will be dismissed as an unnecessary one.

some obtained by switching the third gene with the fourth one in an arrangement we achieve unique representation of each for the second vehicle activity are designated, and so on. By such process, there is a unique possibility to order the sub-chromosome symbolize the assigned tasks. Furthermore, during the encoding process, there is a unique possibility to order the sub-chromosome genes of a chromosome. Each target \( T \) in the chromosome orders the UAV to visit target \( T_1 \) (that is found at (50, 300)) with \( \psi = 70^\circ \), to proceed to the next target, \( T_2 \) (located at (100, 150)) and fly over it with \( \psi = 140^\circ \), and finish the mission by visiting the last target, \( T_3 \) (located at (150, 350)), with \( \psi = 10^\circ \).

4.2.2. Two vehicles and two targets

An example chromosome for the same sample scenario discussed in the previous section is presented in Table 3. The scenario involves one vehicle (identified as \( U_1 \) in the third row of the chromosome) performing a single task on each of three targets, identified by their ID in the first row of the chromosome. The initial configuration of the UAV is \((0, 0, 330)\) and the heading discretization parameter is \( N_\psi = 36 \). The candidate solution that is encoded in the chromosome orders the UAV to visit target \( T_1 \) (that is found at (50, 300)) with \( \psi = 70^\circ \), to proceed to the next target, \( T_2 \) (located at (100, 150)) and fly over it with \( \psi = 140^\circ \), and finish the mission by visiting the last target, \( T_3 \) (located at (150, 350)), with \( \psi = 10^\circ \).

4.2.2. Two vehicles and two targets

Another example chromosome that describes a potential solution of the coupled multiple task assignment and trajectory optimization problem for two cooperating homogeneous UAVs is presented in Table 4. Now, the scenario involves the vehicles identified as \( U_1 \) and \( U_2 \), performing \( N_m = 2 \) tasks on each of two targets starting from their initial configurations at \((0, 0, 315)\) and \((200, 0, 45)\), respectively, with \( N_\psi = 36 \).

The chromosome corresponds to (i) the path of the first UAV from its initial configuration to target \( T_1 \) (located now at (150, 350)), visiting it with a heading of \( \psi = 70^\circ \), moving on to target \( T_2 \) (found at (300, 450)) and visiting it with \( \psi = 50^\circ \) and then revisiting the last target with \( \psi = 260^\circ \); (ii) the visitation of target \( T_1 \) (at (150,350)) by the second vehicle with heading angle \( \psi = 230^\circ \).

Remark 1. Attention should be given that, due to UAV homogeneity, the chromosome provides neither information about arrival times of any vehicle to any target nor precedence order of task execution. Thus, during each visitation of a target by a vehicle, the currently needed task is executed.

Remark 2. Note that the \( i \in \{0, \ldots, N_T \} \) genes representing operation of the first UAV are presented, after that \( j \in \{0, \ldots, N_T - i \} \) genes for the second vehicle activity are designated, and so on. By such an arrangement we achieve unique representation of each candidate in the solutions search space. For instance, a chromosome obtained by switching the third gene with the fourth one in the second example, see Table 4, represents the same solution and therefore will be dismissed as an unnecessary one.

### Table 3
Feasible solution encoded as chromosome; \( N_c = 36, N_m = 1 \).

<table>
<thead>
<tr>
<th>( T_1 )</th>
<th>( T_2 )</th>
<th>( T_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>70°</td>
<td>140°</td>
<td>10°</td>
</tr>
<tr>
<td>( U_2 )</td>
<td>( U_1 )</td>
<td>( U_1 )</td>
</tr>
</tbody>
</table>

### Table 4
Feasible solution encoded as chromosome; \( N_c = 36, N_m = 2 \).

<table>
<thead>
<tr>
<th>( T_1 )</th>
<th>( T_2 )</th>
<th>( T_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>70°</td>
<td>50°</td>
<td>260°</td>
</tr>
<tr>
<td>230°</td>
<td>( U_1 )</td>
<td>( U_1 )</td>
</tr>
</tbody>
</table>

4.3. Computation complexity

To analyze the computational complexity of the homogeneous CMTAP, we compute the cardinality of the feasible solutions set. Thus the number of feasible chromosomes, obtained through the combinatorial analysis, defines the complexity of the problem.

Each of the \( N_T \) targets requires \( N_m \) visits of assigned UAVs. As a result, the length of a chromosome is equal to the number of needed tasks throughout a scenario, as given by Eq. (3). Now, we shall calculate the number of task permutations with repetitions in the \( N_c \) genes of a chromosome. Each target \( T \) in the chromosome is presented exactly \( N_m \) times in the chromosome genes and the order of repetitions is not important. The number of such generalized permutations [23] is defined by Eq. (21). To obtain this equation, first, we set \( N_m \) genes for the tasks on the first target by computing \( \left( \begin{array}{c} N_c \\ N_m \end{array} \right) \) possibilities. Next, we set \( N_m \) tasks for the second target, out of the remaining \( (N_c - N_m) \) genes, resulting in \( \left( \begin{array}{c} N_c - N_m \\ N_m \end{array} \right) \) possibilities. We then continue the process for all the remaining targets and obtain

\[
P = P(N_c; N_m_1, N_m_2, \ldots, N_m_{N_T}) = \frac{N_c!}{N_m_1!N_m_2!\ldotsN_m_{N_T}!}.
\]

4.4. Fitness calculation and selection process

A chromosome fitness calculation is based on summation of edge weights in the multiple targets Dubins path optimization graph (Fig. 2). Each edge weight denotes the Dubins distance between two locations with initial and final heading angles. The calculation of these distances is accomplished by using the path optimization subroutine of the MultiUAV2 simulation [12,20]. A reduction in the number of edge weight calculations is achieved by efficient use of repeated data, as explained in Section 2.3. The resulting weight of each edge is stored in a data table and is pulled out during the evaluation process.

Every path in the graph, encoded in a chromosome solution, denotes a vehicle’s trajectory. The length of each trajectory is included in the cost function Eq. (16) and equals the sum of the edge weights representing it in the graph. Thus, chromosome fitness is inversely proportional to the total vehicle trajectories.
lengths, i.e. \( 1/j \), where \( j \) is given by Eq. (16). To speed up fitness computation, the process is based on simple mathematical operations, such as addition, and fast access to memory. Therefore, during the evaluation process, the presented algorithm only extracts required edge weights from the data table and sums them.

After the fitness of all the chromosomes in a population is computed, the selection of parental chromosomes for generating the offspring population is initiated. This process is based on the selection of the best fit solutions and is carried out through a simple roulette wheel method. So the fitter the chromosome becomes, the higher its probability to be included into the parental population. This process continues till the next parental population is assembled and has the same size as the current population.

4.5. Genetic operators

4.5.1. Crossover

The single site crossover method has been chosen to create a pair of child chromosomes from a pair of parent chromosomes. The idea behind the crossover operator is that an offspring chromosome can have better fitness than both of the parents if it takes the best characteristics from each of the parents.

The crossover example is illustrated in Fig. 7. Firstly, primary parent \( A \) is randomly chosen from a pair of involved progenitors. Then, the crossover site \( s \) is uniformly selected from \([1,N_c]\), and \( s-1 \) first genes of the primary parent are copied to the generated offspring. In order to force the child chromosome to be feasible in terms of task assignments such that each target \( T \in T \) will be visited exactly \( N_{mT} \) times, see Eq. (18), all the targets are copied to the offspring in the same order that they appeared in the primary parent.

Next, one of the genes from the secondary parent chromosome, which encodes any task execution on the same target as the presently recombined gene, is randomly chosen and copied to the child’s current gene. This process is repeated for each of the remaining \([s,N_c]\) child’s genes, i.e. until the offspring construction is completed. Thus, the crossover operator affects a vehicle allocation and a visitation heading angle in a particular gene without the need for testing the feasibility of an offspring chromosome.

To finalize the creation of the first child, the offspring is sorted according to the UAV identity. The creation of a second child is performed through changing the secondary parent to be the primary one and initiation of the presented crossover process.

4.5.2. Mutation

Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. Mutation is an important part of the genetic search as it helps the convergence of the GA towards an optimal solution by preventing the population from stagnating at a local minimum.

In our study, the mutation is performed at randomly chosen sites of each chromosome in the population at each successive generation. We apply three alternative ways to mutate a gene: (1) heading angle mutation; (2) mutation in UAV allocation; (3) mutation in order of required tasks.

A mutation of a heading angle is performed by a random choice of a different heading angle \( \psi_{\text{new}} \in H \setminus \{\psi_{\text{old}}\} \) and inserting it into the mutated gene instead of \( \psi_{\text{old}} \). A mutation in vehicle allocation is achieved by substituting vehicle \( U_{\text{old}} \) with vehicle \( U_{\text{new}} \in U \setminus \{U_{\text{old}}\} \). Swapping of two tasks is done by mutual interchange of two different targets from two randomly chosen different genes.

Each of the three mutation alternatives is applied with an empirically found probability \( p_{m,i} \in \{1,2,3\} \) that is about 0.05, i.e. 5%, per gene in a chromosome. We choose this relatively high probability despite the fact that increased mutation probability may reduce the algorithm rate of convergence because of numerous random changes in the obtained child population. This choice is explained by a need to increase the diversity of solution sampling. Moreover, we allow all three mutations to be applied on each chromosome concurrently.

4.5.3. Elitism

The elitism operator is employed to preserve the best solution obtained during each generation. A solution that was found best through the fitness evaluation process is declared as an elite and is kept in a predefined space of a chromosome population. This elite solution is substituted in subsequent generations with one of the newly generated chromosomes only if the new one was found to have a higher fitness.

4.6. Examples

In this subsection we examine the performance of the GA in different scenarios. The first scenario involves a single vehicle and three targets and is identical to the one used in the previous section of deterministic search. The second includes two vehicles and two targets and necessitates solving a cooperative problem.
In the third scenario, we examine the applicability of the GA to solve a Euclidean TSP.

4.6.1. Single vehicle and three targets

The GA simulation parameters, for the scenario involving three targets and a single vehicle, are presented in Table 5.

A sample of an obtained trajectory is plotted in Fig. 8 with the relevant heading angles. The trajectory defines also the target visitation order. After the allotted 50 generations, the path length of the achieved solution is 1583 m compared with the minimal length path of 1483 m, obtained by using a deterministic graph search method. Also, the target visitation order and the heading angles are different from those presented in the optimal solution. As an aside, the solution is the same as the one obtained by the deterministic method for the given visitation order of \( T_3 \rightarrow T_2 \rightarrow T_1 \), as can be seen from Table 2.

Raising the population size and/or the number of iterations produces better results. As an example we raised the number of iterations from 50 to 200 and obtained the optimal solution under the given discretization level (the one shown in Fig. 6).

4.6.2. Two vehicles and two targets

We now extend the problem to examine a more complex scenario. The scenario consists of two homogeneous vehicles \( U = \{ U_1, U_2 \} \) with initial configurations \( (0, 0, 315) \) and \( (200, 0, 45) \), respectively, and two targets \( T = \{ T_1, T_2 \} \), located at \( (150, 350) \) and \( (300, 450) \), respectively. Each target requires two visits from a vehicle, i.e. \( N_m = 2 \). We apply the GA with the parameters presented in Table 5, except that the number of iterations was chosen as 200.

The obtained trajectories of the cooperating vehicles are plotted in Fig. 9 and the related heading angles of each vehicle when flying over a target are presented. The cumulative path length of the solution achieved by the GA run is 1710 m.

A single-source deterministic graph search method, like Dijkstra’s algorithm, cannot deal with the case of cooperative multiple task execution because of the need for complex multi-source task coordinated graph search. Thus, an exhaustive search technique was used to obtain the optimal total path length. The optimal solution (under the given discretization) was found to be identical to the one obtained by the GA.

4.6.3. Euclidean TSP

Assuming a single vehicle, no kinematic constraints (i.e. \( r_{\text{min}} = 0 \)), multiple targets, one visit per target, and that the vehicle must return to its initial location, degenerates the CMTAP examined in this work to the Euclidean TSP [5].

To verify the correctness of the presented GA and to examine its performance once again, we use it to solve a Euclidean TSP. In the examined scenario we define 11 different cities that the agent (i.e. the vehicle) has to visit. That is similar to a definition of 10 targets, \( T = \{ T_1, \ldots, T_{10} \} \), and an initial position of the vehicle. The only difference is that the vehicle has to return to its initial location at the end of the travel. Therefore, we slightly change our objective function by adding a terminal cost that is equal to the Euclidean distance from the last visited site to the initial location of the UAV. In this way the GA will obtain solutions for the defined TSP.

We compare the solution obtained by the GA (352 m), see Fig. 10, with the output of the acknowledged Concorde TSP Solver, one of the most advanced and fastest TSP solvers [24]. Selecting the greedy search technique for Concorde and giving enough running time results in a path of optimal length, which is identical to the one plotted for GA in Fig. 10.

Table 5: GA simulation parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>200</td>
</tr>
<tr>
<td>Number of iterations</td>
<td>50</td>
</tr>
<tr>
<td>Minimum turn radius (m)</td>
<td>200</td>
</tr>
<tr>
<td>Discretization (deg)</td>
<td>10</td>
</tr>
</tbody>
</table>

![Fig. 8. Resulting trajectory from GA; \( r_{\text{min}} = 200 \text{ m} \), \( N_c = 36 \), \( N_m = 1 \). Length = 1583 m.](image)
Remark 3. In all the above three scenarios the optimal trajectories were found by the GA because of the comparatively small search space of candidate solutions. In the first two scenarios the small search space is due to the small number of vehicles and discretization level, while in the third scenario the number of targets is a bit larger but we use only a single discrete heading, as $r_{\text{min}}=0$. 

Please cite this article as: Edison E, Shima T. Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. Computers and Operations Research (2010), doi:10.1016/j.cor.2010.06.001
5. GA for heterogeneous CMTAP

In this section we discuss scenarios involving a group of heterogeneous UAVs performing a set of predefined tasks on a number of known targets. The heterogeneous CMTAP may appear more challenging than the one involving only homogeneous UAVs. However, in fact, it is simpler in terms of size of the search space. The computational complexity of this problem is directly connected to the search space size and bounded above by the complexity of homogeneous CMTAP (presented in Section 4.3). This bound is derived from the fact that heterogeneous UAVs have different capabilities to perform tasks. Thus, there are fewer possibilities to assign UAVs than in the homogeneous CMTAP.

5.1. Encoding

The homogenous CMTAP solution, presented in the previous section, requires that all UAVs be able to perform any task and that they are identical in terms of kinematic constraints, i.e. minimum turning radius and constant flight speed. In order to avoid this limitation, the genetic algorithm from the previous chapter is expanded here to cover scenarios where non-identical UAVs cooperatively perform a set of different tasks on various targets. For example, a scenario involving two different types of UAVs—a combat UAV that is capable of performing all kinds of tasks from the set $M$ and a surveillance UAV which specializes in terrain observation, i.e. able to classify and verify stationary targets, $M_s = \{C, V\}$. These UAVs may also have different kinematic constraints. The development of a GA for CMTAP with heterogeneous UAVs forces us to introduce a new encoding of a solution, considering all the kinematic constraints and operational abilities of each UAV.

We present a new chromosome that consists of two parts. First, three bottom rows, which are identical to those presented in the chromosome of the homogeneous CMTAP and similarly describe UAV allocations to the targets and heading angles of target visitations. The second part, the upper row, defines the kind of task (task assignment) that a UAV must perform on the specific target. Obviously, each UAV is only assigned to tasks which it is capable of performing. In contrast to the homogeneous CMTAP, the new chromosome should consider the task execution times in addition to the multiple task assignment and the path planning. For example, a UAV that flies faster can reach a target location earlier than another that tracks the same trajectory with lower speed. This may result in a situation where a UAV reaches the target to perform a subsequent task before the prior task has been completed. Therefore, the UAV is forced to lengthen its path upon the execution of the previous task.

An example chromosome for a scenario that involves two heterogeneous UAVs is presented in Table 6. The specialities of each UAV can be seen at the bottom row of the chromosome. Thus, $U^1_T$ means the UAV is of type 2 (see Table 1) so it can perform only surveillance tasks, while $U^2_T$ is a combat UAV (type 1) with a full range of operational capabilities. The solution encoded in the chromosome is interpreted by (i) classification of target $T_2$ by $U_1$ with heading of $140^\circ$ and then proceeding to verification of $T_1$ with $60^\circ$ heading, and (ii) attacking target $T_1$ and target $T_2$ by $U_2$ with heading angles of $130^\circ$ and $100^\circ$, respectively.

The tasks precedence constraint, see Eq. (20), is especially important in the scenarios involving a group of heterogeneous UAVs. To satisfy this constraint the algorithm has to elongate the path of a UAV that arrived to perform the subsequent task before the prior task was executed, otherwise the solution will become unfeasible.

5.2. Path elongation

There are several path elongation methods that can be used for this purpose. For example, feasible path segments, like arcs and straight lines, may be included in the UAV trajectory obliging it to arrive immediately after the completion of the previous task [16]. Another possibility to lengthen the path is to use a longer path from the Dubins set that will satisfy the posed constraint [25].

A path elongation method used in this study is based on appending busy-wait circles. This technique obliges the UAV whose path has to be lengthened to perform circular flight with minimum turning radius around the target till the prior task is completed, thus satisfying the task precedence constraint. Note that once started, a busy-wait circle has to be finished even if the previous task has been already accomplished. The chromosome example that encodes such a path elongation is given in Table 7, while the respective UAV trajectories with included busy-wait circles are presented in Fig. 11. In this scenario each of two targets, $T = \{T_1, T_2\}$, has to be visited twice; $T_1$ has to be attacked first and then its destruction must be verified, while $T_2$ needs classification followed by an attack. Additionally, let us define vehicle characteristics as follows: (i) $U_1$ is a surveillance UAV flying at speed $v_{U_1} = 50$ m/s and turning with $r_{min_1} = 180$ m, and (ii) $U_2$ is a munition UAV with a speed of $v_{U_2} = 70$ m/s, and $r_{min_2} = 200$ m; it can only attack the targets (specialty type is 3).

At the beginning $U_2$ is commanded to attack target $T_2$; it arrives first at the target, before $U_1$ that has to classify this target. Therefore $U_2$ is forced to wait by flying in a single busy-wait circle around the target till the classification task is completed. After $U_1$ finishes its first task it proceeds to verify the damage to target $T_1$, but this target was not attacked yet by $U_2$. This makes $U_1$ utilize another busy-wait circle in its path and complete the verification only after the attack. The resulting total path length is $3511$ m.

Another way to elongate the UAV path without the combination of busy-wait circles is made by the GA via giving a better score to the chromosome that encodes different heading angles of the UAV in the start and/or the end of point-to-point trajectory between two targets than the chromosome which forces an addition of busy-wait circles. Obviously, to obtain higher fitness during the evaluation process, the path has to be shorter than a path that includes busy-wait circles. Furthermore, this method of path elongation prevents deadlock, i.e. the state that occurs when the chromosome defines a path with an infinite number of busy-wait circles. Deadlock is discussed next.

| Table 6 | Feasible solution of heterogeneous CMTAP encoded as a chromosome: $N_g = 36$, $M_t = \{A, V\}$, $M_s = \{C, A\}$. |
| C | V | A |
| T2 | 140° | U1^T |
| T1 | 60° | U1^T |
| A1 | 130° | U2^T |
| A2 | 100° | U2^T |

| Table 7 | Feasible solution of heterogeneous CMTAP encoded as a chromosome with two busy-wait circles: $N_g = 36$, $M_t = \{A, V\}$, $M_s = \{C, A\}$. |
| C | V | A |
| T2 | 140° | U1^T |
| T1 | 60° | U1^T |
| A1 | 130° | U2^T |
| A2 | 100° | U2^T |

Please cite this article as: Edison E, Shima T. Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. Computers and Operations Research (2010), doi:10.1016/j.cor.2010.06.001
5.3. Deadlock

Generally, deadlock is a situation wherein two or more competing processes are waiting for each other to release a resource. Also, deadlock occurs when none of the processes meet the condition to move to another state. Moreover, there is no known general solution to avoid deadlocks.

In our case deadlock is a condition when the encoded task execution order forces all the UAVs to wait by combining an infinite number of busy-wait circles. This state causes endless waiting that is translated into an infinite path length for the UAV group. Thus, the chromosome that encodes the deadlock situation is infeasible.

Example of the chromosome, which encodes deadlock, is presented in Table 8 and Fig. 12. Here, the targets and the vehicles are identical to the ones presented in the previous scenario. \( U_1 \) visits target \( T_1 \) to verify the damage, but this target was not attacked yet by \( U_2 \). Therefore, \( U_1 \) waits until \( U_2 \) finishes the prior task on \( T_1 \). Meanwhile, \( U_2 \), which arrived to attack \( T_2 \), waits until \( U_1 \) will first classify it. This forces \( U_2 \) to fly in busy-wait circles also. Thus, both of the vehicles are waiting and no one can proceed to the next task.

To avoid the possibility that chromosomes which encode deadlock will be included in the solution population, we set such a chromosome’s fitness to an infinitely small value. More details regarding the path elongation algorithm and deadlock can be found in Ref. [26].

5.4. Fitness calculation

The fitness evaluation process is based on preliminary calculation of all possible Dubins distances tables as presented earlier in Section 4. In contrast to the homogeneous CMTAP, here the vehicles have different minimum turning radii. Therefore, we have to consider this difference while creating the Dubins distance tables. Thus, in heterogeneous CMTAP each UAV has its own Dubins distances table.

In the heterogeneous CMTAP it is critical to compute the time for each task execution for all UAVs. This is performed by dividing the path length of the UAV performing a specific task by the constant speed of this UAV. During fitness evaluation, the chromosome is split into a number of sub-chromosomes according to the identities of the involved UAVs. Then the calculation of task execution times takes place according to their time precedence. If needed, path elongation is performed, as described earlier.

5.5. Genetic operators

The crossover process is applied similarly to the GA for homogeneous CMTAP. To ensure child chromosome feasibility in terms of correct number of tasks per target and task types needed to be completed on each target, the first two lines (the row of task types and the row of targets) of the first progenitor are replicated in the offspring. Next, to complete the offspring construction, genes that encode the same task type, which is presented after the crossover site in the child, are randomly chosen from the second parent. Thus, it is verified that only UAVs with appropriate operational capabilities are chosen. Gene parts, which consist of a UAV ID and heading angle are copied to the offspring chromosome.

To avoid the possibility that chromosomes which encode deadlock will be included in the solution population, we set such a chromosome’s fitness to an infinitely small value. More details regarding the path elongation algorithm and deadlock can be found in Ref. [26].
5.6. Example

An example run for a scenario that involves a pair of heterogeneous UAVs, $U_1$ and $U_2$, and a pair of targets, $T = \{T_1, T_2\}$, is presented in Fig. 13. The first UAV has a minimum turn radius of 180 m and flies at a speed of 50 m/s, while the second vehicle has a turn radius and speed of 200 m and 70 m/s, respectively. Both of the targets have to be visited twice, where $M_{T_1} = \{A, V\}$, $M_{T_2} = \{C, A\}$. The presented paths do not include busy-wait circles, but the satisfaction of the task precedence constraint is evident (see the task execution times presented in the figure). Despite the absence of busy-wait circles, the paths are indeed elongated. The elongation is performed implicitly within the GA by optimizing the heading angles that make a point-to-point path longer and causing a vehicle to arrive after the completion time of the prior task. The total path length obtained in this run equals 3045 m, which is 466 m less than the path obtained by the insertion of busy-wait circles, see Fig. 11. The obtained result compares favorably with the shortest path of 2925 m computed without considering the task precedence constraint.

Remark 4. Due to one time calculation of Dubins distances between every two targets and elimination of the need for a chromosome feasibility test during the crossover, the algorithm time consumption is significantly reduced. Thus, an approximate total time needed for computation of Dubins distances and for the GA runtime (when both are implemented by non-optimized native code) for heterogeneous CMTAP is about $\frac{1}{60}$ of the entire scenario duration. For example, the solution for a scenario that lasts about 2 min is derived within approximately 2 s. This result makes it possible to use the developed technique for real-time applications. Moreover, it is possible to replan fast the UAV paths by rerunning the algorithm in a case of new data acquisition or mission failure.

6. Monte Carlo study

We use a Monte Carlo study, consisting of 100 runs for each set of randomly chosen parameters, to provide an extended analysis of the GA algorithm.
6.1. Homogeneous case

The investigated scenario involves three homogeneous UAVs, four targets, three tasks per target, and minimum turn radius \( r_{\text{min}} \in \{50,100,200\} \) m. Fig. 14 demonstrates sample trajectories in one of these complex scenarios. Other GA parameters are provided in Table 9. Note that the defined cost criterion of minimum cumulative path length and the assumption of unlimited resources cause one of the vehicles (vehicle 1, path represented by blue solid line) to execute significantly more tasks than the other vehicles.

In Fig. 15 the average over the set of 100 runs of the normalized costs \( J_i / J_1 \) is presented as a function of the genetic generation, where \( i \in \{1,2,\ldots,300\} \) designates the generation number. Here, the progressive improvement of an achieved solution for each case of different turn radius is evident. Note that reducing the turn radius reduces the affect of the kinematic constraints on the resulting point-to-point Dubins trajectories. Thus, an assumption of \( r_{\text{min}}=0 \) m causes all of the Dubins path lengths between one configuration to another to be identical, i.e. the metric becomes Euclidean. Therefore, the diversity of the search space lessens with the reduction of the turn radius and the convergence is faster.

6.2. Heterogeneous case

For an additional analysis of the techniques that were developed in this work we perform a comparison between the GA and random search in a scenario involving heterogeneous vehicles. The comparison is performed by running a Monte Carlo simulation for 100 large-scale scenarios per algorithm. Each scenario involves \( N_U=15 \) heterogeneous vehicles with different kinematic constraints that are chosen in a random manner in each scenario. Minimum turning radii are chosen from the range of \( r_{\text{min}} \in \{100,300\} \) m and the constant speed of each vehicle is picked from \( v_U \in \{30,100\} \) m/s. In each scenario it is required to perform all \( N_m=3 \) tasks from the set \( M \) on each of \( N_T=10 \) targets. The positions of the targets are selected randomly in the square area of 10,000 m\(^2\). All UAVs are able to perform all the required tasks, i.e. the heterogeneity is expressed only in terms of different kinematics. The population size and the number of generations that are chosen for the optimization process are both 100.

Fig. 16 presents the simulation results. As in Fig. 15, the vertical axis designates the mean cost value of 100 runs at each generation (i.e. the total path lengths) normalized by the mean cost of the first random generation. The horizontal axis designates the generation number in the GA optimization process. The improvement in the rates of convergence when using the GA optimization scheme is obvious in Fig. 16. Additional analysis and results can be found in Ref. [26].

7. Conclusions

A new approach for solving integrated task assignment and motion planning, for a problem where multiple autonomous UAVs cooperatively perform multiple tasks on multiple targets, was presented. The method enables solving the coupled problem by using a discretization of the possible visitation angle of a vehicle.

Table 9

<table>
<thead>
<tr>
<th>GA simulation parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
</tr>
<tr>
<td>Number of GA generations</td>
</tr>
<tr>
<td>Minimum turn radius (m)</td>
</tr>
<tr>
<td>Discretization (deg)</td>
</tr>
</tbody>
</table>
over its target. Using a smaller discretization variable enables obtaining a solution that is closer to the optimal one. However, this enlarges the search space and can render the use of deterministic search methods infeasible. For such cases a GA was proposed.

First, the GA for scenarios that involve only homogeneous UAVs was proposed. It was developed to use simple encoding and a very fast fitness evaluation process. After an examination of the obtained solutions and comparison with the results derived by deterministic methods, it was concluded that the method works well, and for small sized problems finds the optimal solution.

Notwithstanding, an assumption of using the homogeneous UAVs complicates the use of such an algorithm in real world scenarios. Therefore, this algorithm was extended to solve the scenarios that involve heterogeneous UAVs. This extension was done by applying mild changes to the chromosome encoding and the fitness calculation process while preserving simplicity of implementation and rapidity of computation. We managed new

Please cite this article as: Edison E, Shima T. Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. Computers and Operations Research (2010), doi:10.1016/j.cor.2010.06.001
challenges that were raised during the development of the algorithm, such as path elongation and deadlock. The new GA for heterogeneous CMTAP kept all the good characteristics of the previously proposed GA.

Performance analysis using a Monte Carlo study was also presented. Results show that the rate of a solution improvement is affected by severity of the kinematic constraints. Even for scenarios that involve UAVs with strict kinematic constraints, the GA provides good feasible solutions. Additionally, the improvement in the GA’s convergence rate compared to random search was demonstrated via a Monte-Carlo study.

The techniques presented in this study were developed with careful attention to runtime requirements. This enables the use of the algorithms in real-time applications. Moreover, it makes possible to replan fast the UAVs paths by rerunning the algorithm in the case of new data acquisition or mission failure.

References


Please cite this article as: Edison E, Shima T. Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. Computers and Operations Research (2010), doi:10.1016/j.cor.2010.06.001