Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 3252–3269
computers & operations research
www.elsevier.com/locate/cor

Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms

Tal Shima[a,*], Steven J. Rasmussen[a], Andrew G. Sparks[a], Kevin M. Passino[b]

[a] *Control Theory and Optimization Branch, Air Force Research Laboratory, Bldg. 146, 2210 Eighth St., Wright-Patterson AFB, OH 45433, USA*
[b] *Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210, USA*

Available online 2 April 2005

Abstract

A problem of assigning cooperating uninhabited aerial vehicles to perform multiple tasks on multiple targets is posed as a new combinatorial optimization problem. A genetic algorithm for solving such a problem is proposed. The algorithm allows us to efficiently solve this NP-hard problem that has prohibitive computational complexity for classical combinatorial optimization methods. It also allows us to take into account the unique requirements of the scenario such as task precedence and coordination, timing constraints, and trajectory limitations. A matrix representation of the genetic algorithm chromosomes simplifies the encoding process and the application of the genetic operators. The performance of the algorithm is compared to that of deterministic branch and bound search and stochastic random search methods. Monte Carlo simulations demonstrate the viability of the genetic algorithm by showing that it consistently and quickly provides good feasible solutions. This makes the real time implementation for high-dimensional problems feasible.

Keywords: Task assignment; Multiple tasks; Cooperating agents; Genetic algorithm; Uninhabited aerial vehicles

1. Introduction

The use of uninhabited aerial vehicles (UAVs) for various military missions has received a growing attention in the last decade. Apart from the obvious advantage of not putting human life in harm's way,

* Corresponding author. Fax: +1 937 656 4000.
E-mail address: shima_tal@yahoo.com (T. Shima).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.02.039

the lack of a human pilot enables significant weight savings, lower costs, and gives an opportunity for new operational paradigms.

Historically, UAVs have been employed as intelligence assets monitored and controlled from the ground. For helping the operator in on-line generation of trajectories between numerous targets several routing methods have been recently proposed [1,2]. In future scenarios the UAVs are expected to have a high level of autonomy and preferably work in groups. In this context, an intensive research effort has been conducted in recent years on the development of cooperative decision and control algorithms. Scenarios of particular interest are the wide area search and destroy (WASD) [3] and combat intelligence, surveillance, and reconnaissance (ISR) missions, which have similar characteristics except that the combat ISR scenario typically has a longer duration. In such scenarios powered vehicles are released in the target area and are independently capable of searching, classifying, and attacking targets, along with subsequent battle damage verification. Exchange of information within the group can improve the group's capability to meet performance requirements related to fast and reliable execution of such tasks. While cooperation between the UAVs is desirable, it can be very complicated to implement. For the cooperation, sophisticated optimization problems must be solved, in real time, taking into account the need for task precedence and coordination, timing constraints, and flyable trajectories.

One of the main challenges in cooperative decision and control problems is computational complexity. The sheer size of the problem (e.g. number of vehicles, targets, and threats) is one form of complexity. However, in scenarios such as WASD and combat ISR, coupling between the completion of the four different tasks (search, classification, attack, and verification) and coupling between the assignment process and trajectory optimization have the most significant impact on complexity. For example, if the vehicles each have a default task of searching then performing it cooperatively introduces extensive coupling in their search trajectories. Once a target has been found it needs be classified, attacked, and verified by the UAV team which further imposes coupling between the trajectories of different team members [4].

Emerging cooperative decision and control algorithms of different classes have been proposed for solving such problems. These algorithms are based on customized combinatorial optimization methods including: mixed integer linear programming (MILP) [5,6], the capacitated transhipment problem [3], and the iterative capacitated transhipment problem [7]. Due to the special characteristics of the problem and the requirement for a tractable solution, all of these proposed algorithms are suboptimal in some sense. For example, the MILP algorithm of [5] uses Euclidean distances while that of [6] uses piecewise UAV trajectories between targets; and hence both do not fully take into account the need for flyable trajectories. The single task assignment algorithm [3] is only optimal for the current tasks and does not take into account tasks that will be required when the current tasks are completed. Although performing iterations on the single task assignment algorithm (that utilize in essence a greedy solver) [8] provides a solution to the multiple task assignment requirement, it is heuristic in nature and therefore optimality is not achieved. Note also that for most problems of realistic complexity the algorithms in [5,6] take a long time to set up and to execute.

In a recent paper [9], a tree generation algorithm was developed that produces the optimal solution to the assignment problem based on piecewise optimal trajectories. This algorithm generates a tree of feasible assignments and then performs exhaustive search to find the optimal assignment. During the generation of the tree all of the requirements of the mission are met. However, since it requires an enumeration of all the feasible assignments, direct use of this approach is only reasonable for relatively low-dimensional scenarios and off-line applications. For an on-line application, a best first search (BFS) algorithm that

uses the branch and bound concept has been proposed [10]. This deterministic greedy search method has desirable qualities such as providing immediately a feasible solution, that improves monotonically and, eventually, converges to the optimal solution.

Stochastic and non-gradient search methods [11] might be considered in order to avoid the computational complexity of the combinatorial optimization methods described above and thus speed up the convergence to a good feasible solution. The genetic algorithm (GA) is such an approach that does not require explicit computation of the gradients in the problem [12]. Assuming that the search space is not extremely rugged [13], the GA will often quickly yield good feasible solutions and will not be trapped in any local minimum; however, it may not yield the optimal solution. Another key attribute of the method is the possibility of parallel implementation.

In this paper the GA methodology is used to solve the UAV cooperative multiple task assignment problem (CMTAP). The remainder of this manuscript is organized as follows: In the next section, the GA optimization method and its previous use for solving classical combinatorial optimization problems is discussed. Then, the UAV CMTAP is posed. This is followed by an analysis of the problem's computational complexity and the design of the GA. A performance analysis is then presented and concluding remarks are offered in the last section.

2. Methodology

In this section, the GA methodology and its previous use in solving classical combinatorial optimization problems is reviewed.

2.1. GA—brief review

GAs are described in many papers and textbooks including [12,14], and will only briefly be reviewed in this section. It enables efficient search of a decision state space of possible solutions, as long as the search space is not extremely rugged. The method involves iteratively manipulating a population of solutions, called chromosomes, to obtain a population that includes better solutions. The encoding of the GA chromosome is a major part of the solution process. After that stage has been overcome, the algorithm consists of the following steps: (1) initialization—generation of an initial population, (2) fitness—evaluation of the fitness of each chromosome in the population, (3) test—stopping if an end condition is satisfied and continuing if not, (4) candidate new solutions—creating new candidate chromosomes by applying genetic operators, thus simulating the evolution process, (5) replacement—replacement of old chromosomes by new ones, (6) loop—going back to step 2.

The genetic operators mentioned above are: selection, crossover, mutation, and elitism. These operators are performed on the chromosome solutions consisting each of N_c genes. In the selection stage two parent chromosomes are chosen from the population based on their fitness. Several selection methods are commonly used, including: roulette wheel, rank, and binary tournament. In all of these methods the better the fitness, the better the chance of being selected. Crossover is performed in single or multiple points across the chromosome, location of which is selected randomly. For example, in the simple one-point crossover operator on chromosomes with N_c genes, the child solution consists of the first g genes from the first parent and N_c-g genes from the second parent; and vice versa for the second child. The mutation operator involves randomly changing one of the genes in the child chromosome. This operator reduces

the probability of the algorithm getting trapped in any local minimum. The Elitism operator is used to save the best solutions of the generation.

For parallel implementation of the GA several methods can be used. The master-slave GAs are probably the simplest type of parallel GAs. The master process stores the population and executes the genetic operators discussed earlier; at the same time the slave process evaluates different subsets of the population [15]. Another type of parallelization is achieved via the multiple population parallel GAs. In such a method multiple population GAs are run in parallel, infrequently exchanging individuals [15].

2.2. GA in classical combinatorial optimization problems

Many combinatorial optimization problems are concerned with pairing between agents and tasks. The simplest one is the classical assignment problem consisting of such optimal pairing, without assigning an agent more than once and ensuring that all tasks are completed. Such a problem can be easily solved by the Hungarian algorithm [16]. When an agent can be assigned to more than one task, or there are resource limitations to process a given task by an agent, the problem becomes much more complicated and cannot be solved in polynomial time. Such problems include the travelling salesman problem (TSP), the generalized assignment problem (GAP), and the vehicle routing problem (VRP). In all of these classical problems the minimum cost assignment is sought where: in the TSP the tour is of one agent between a finite number of cities; in the GAP m agents need to perform n jobs, such that each job is assigned to exactly one agent and the resource of each agent is limited; and in the VRP m vehicles, with a given capacity, are dispatched from a single depot to deliver to n customers each requiring a specified weight of goods, and then return to the depot.

The application of GAs to these problems has been widely studied. Much work in applying GAs to the TSP [14] is concerned with the encoding of the chromosomes and the use of special crossover operators to preserve the feasibility of the solution representation. In [17] a GA was used to solve the GAP. Using simulations it was shown that on average the GA finds solutions that are within 0.01% from the optimal one. The VRP was solved in [18] using a pure GA and a hybrid GA with neighborhood search methods showing promising results compared to simulated annealing and Tabu search, with respect to solution time and quality. In all of these studies the assignments solved require one tour/service per target and there are no precedence requirements as in the UAV task assignment problem, discussed in detail next.

3. Cooperative multiple task assignment problem

Based on [3] a generic CMTAP is defined in this section. First, the tasks precedence and requirements are discussed and then a tree representation is given. Next, different performance measures are reviewed and finally the problem is posed as a new combinatorial optimization problem.

3.1. Tasks

It is assumed that the terrain has already been searched (by other UAVs or sensor assets) and N_t targets have been found. Let $T = \{1, 2, \ldots, N_t\}$ be the set of targets found and let $V = \{1, 2, \ldots, N_v\}$ be a set of UAVs performing tasks on these targets. The set of tasks that need be performed by the UAV team on each target is $M = \{Classify, Attack, Verify\}$ and we let N_m be the number of such tasks. Each of

these tasks has requirements governing its execution. Target classification, consisting of maximizing the correct target recognition under given observation ability, can be performed only if the vehicle follows a trajectory that places its sensor footprint on the target. After a target has been successfully classified from a standoff distance, one or more UAVs attack it by releasing appropriate weapons. Following target attack, cooperative damage verification is performed. For the sake of simplicity, we assume that the probability of accomplishing a task given the physical requirements have been met (e.g. for the classification task, the target is in the UAV sensor foot print) is one. If this assumption is found not to be true, then the assignment algorithms must be re-evaluated.

3.2. Assignment requirements

The requirements from the assignment algorithm for a feasible solution include taking into account task precedence and coordination, timing constraints, and flyable trajectories.

The precedence requirement states that tasks on each target must be accomplished in order; i.e. a target can be attacked only after it has been classified, and verified only after an attack on it has been performed. For group coordination each task should be accomplished once, e.g. UAVs should not be assigned to attack a target twice, unless the target is verified alive after an attack or there is a predefined need for multiple attacks. The timing constraints require that a certain task be performed within a given time frame. Such a requirement is of importance when engaging time critical targets, e.g. surface to air missile sites. The requirement of flyable trajectories is a prerequisite for an assignment that can be performed by the UAV team members. Otherwise, assigned tasks may not be executed and the team coordination could collapse.

3.3. Tree representation

In [9] it was demonstrated that the UAV CMTAP can be represented by a tree. This tree not only spans the decision space of the problem, but it also incorporates the state of the problem in its nodes. The tree is constructed by generating nodes that represent the assignment of a vehicle $i \in V$ to a task $k \in M$ on a target $j \in T$ at a specific time. The child nodes are found by enumerating all of the possible assignments that can be made, based on the remaining tasks and requirements of the UAV CMTAP. Nodes are constructed until all combinations of vehicles, targets, and tasks have been taken into account. Note that a branch of the tree, from a root node to a leaf node, represents a feasible set of assignments for the UAV group.

Fig. 1 shows an example of the tree in a scenario consisting of just two vehicles performing two tasks on two targets. Note that this figure shows four subtrees. The top node of each of these subtrees is connected to the root node of the entire tree. Each node represents one assignment of a vehicle to a task where the notations Cij and Aij denote that vehicle i performs on target j classification or attack, respectively.

In the case of the UAV CMTAP, the tree is wide and shallow. The depth of the tree, i.e. the number of nodes from the root node to the leaf nodes is equal to

$$N_c = N_t N_m, \tag{1}$$

and in the investigated problem $N_m = 3$ (the number of tasks). Since, as noted above, traversing the tree from a root node to a leaf node produces a feasible assignment it is possible to compute such an assignment in a known time, which is the node processing rate times N_c.

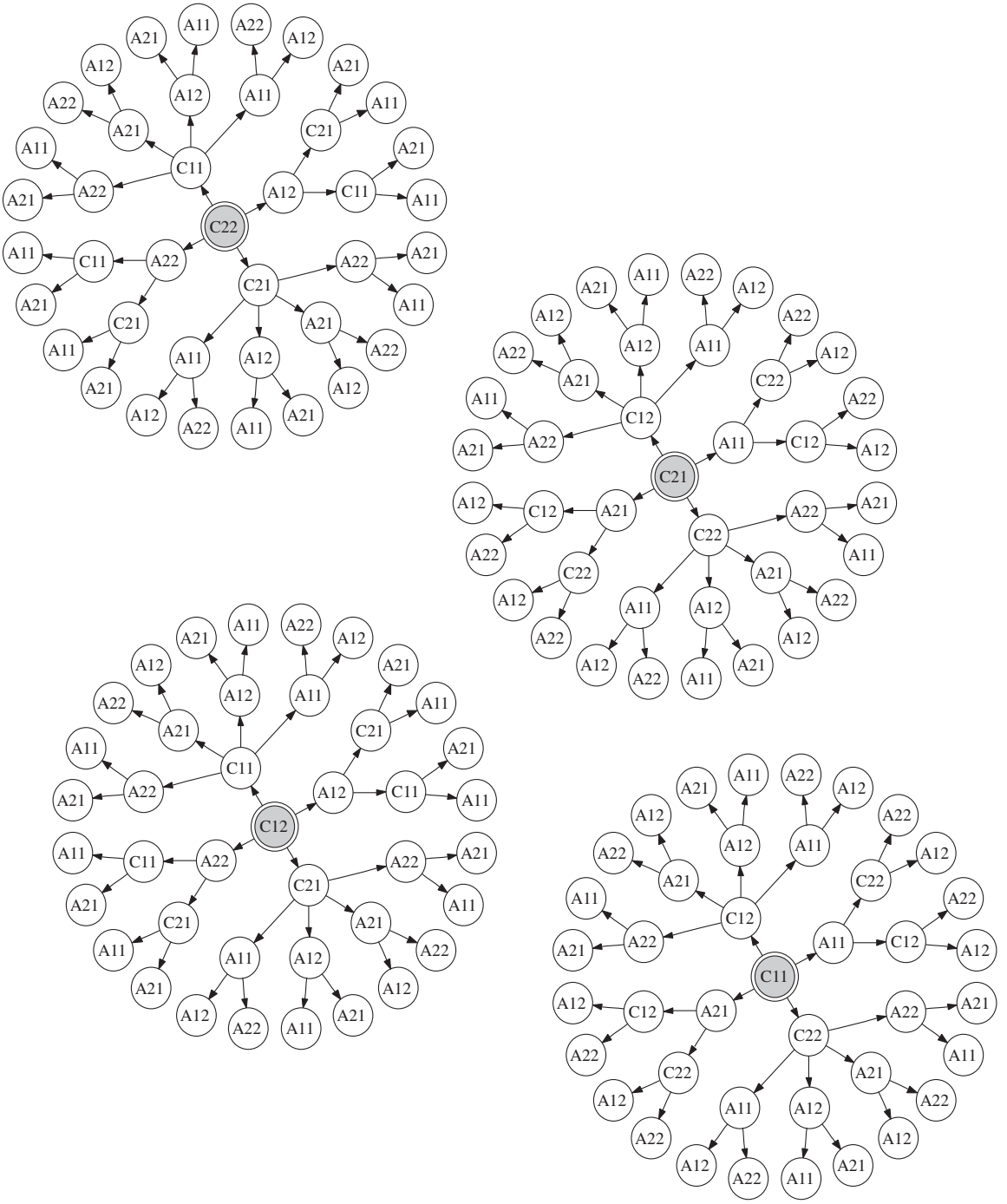

Fig. 1. UAV CMTAP tree, $N_v = N_t = N_m = 2$.

3.4. Performance requirements

Different performance indexes can be chosen for the UAV CMTAP. Two such criteria are discussed next.

One performance criterion that can be used is the cumulative distance travelled by the vehicles to perform all required tasks,

$$J_1 = \sum_{i=1}^{N_v} R_i > 0, \tag{2}$$

where R_i is the distance travelled by UAV $i \in V$ from the beginning of the mission until finishing its part in the group task plan. The group objective is to minimize J subject to the assignment requirements of Section 3.2. By minimizing this cost function the use of the overall group resources (i.e. flight time, assuming constant fuel consumption) is optimized. After each UAV fulfills its group tasks it can then resume a default task, such as searching for new targets. This cost function is appropriate mainly for problems involving low-value stationary targets.

Another possible performance criterion is the minimum time for the team to accomplish the required tasks on all targets. Assuming constant equal speeds for all UAVs, this performance criterion can be replaced by

$$J_2 = \max_{i \in V} R_i > 0. \tag{3}$$

Thus, the group objective is to minimize the longest distance travelled by one (or more) of its members subject to the assignment requirements of Section 3.2. This cost function can be used in a scenario where it is critical to service targets as quickly as possible, e.g. hunting mobile ballistic missile launchers.

3.5. Combinatorial optimization problem

The problem is to minimize the cost function J_1 or J_2 of Eq. (2) or (3) by optimizing the N_c assignments of vehicles to targets. Let $S = \{1, 2, \ldots, N_c\}$ be the set of stages in which the assignment is made, where each stage $l \in S$ corresponds to a layer in the tree representation plotted in Fig. 1. Let $x_{l,i,j} \in \{0, 1\}$ be a decision variable that is 1 if at stage $l \in S$ vehicle $i \in V$ performs a task on target $j \in T$ and is 0 otherwise; and $X_l = \{x_{1,i,j}, x_{2,i,j}, \ldots, x_{l,i,j}\}$ be the set of assignments up to, and including, stage l. Let $c_{l,i,j}^{X_{l-1}}$ be the distance to be travelled by vehicle $i \in V$ to perform a task on target $j \in T$ at stage $l \in S$, given the prior assignment history X_{l-1}; $r_{l,i,j}^{X_{l-1}}$ be the resource (fuel) required to perform the task; and b_i as the resource availability of vehicle $i \in V$ (fuel capacity of each vehicle).

The mathematical formulation of the CMTAP with the two different cost functions is given next, where Eq. (4) corresponds to the cost function of Eq. (2) and Eq. (5) to that of Eq. (3). Thus, the problem is

$$\text{Min} \quad J_1 = \sum_{l=1}^{N_c} \sum_{i=1}^{N_v} \sum_{j=1}^{N_t} c_{l,i,j}^{X_{l-1}} x_{l,i,j} \tag{4}$$

or

$$\text{Min} \quad J_2 = \max \sum_{l=1}^{N_c} \sum_{j=1}^{N_t} c_{l,i,j}^{X_{l-1}} x_{l,i,j}, \quad i \in V \tag{5}$$

$$s.t. \quad x_{l,i,j} \in \{0,1\}, \quad l \in S, \ \ i \in V, \ \ j \in T, \tag{6}$$

$$\sum_{i=1}^{N_v} \sum_{j=1}^{N_t} x_{l,i,j} = 1, \quad l \in S, \tag{7}$$

$$\sum_{l=1}^{N_c} \sum_{i=1}^{N_v} x_{l,i,j} = N_m, \quad j \in T, \tag{8}$$

$$\sum_{l=1}^{N_c} \sum_{j=1}^{N_t} r_{l,i,j}^{X_{l-1}} x_{l,i,j} \leqslant b_i, \quad i \in V. \tag{9}$$

Eq. (6) is the binary constraint on the decision variables; Eq. (7) ensures that at each stage exactly one task on a target $j \in T$ is assigned to exactly one vehicle $i \in V$; Eq. (8) ensures that on each target $j \in T$ exactly N_m tasks are performed; and Eq. (9) ensures that the total resource requirement of the tasks from each vehicle $i \in V$ does not exceed its capacity.

4. GA synthesis

The UAV CMTAP discussed in the previous section is a very complex combinatorial optimization problem, even for relatively small numbers of vehicles and targets. Due to the need for on-line implementation, an algorithm is sought that has the following desirable attributes: quick feasible solution, monotonically improving solution over time, and convergence near to the optimal solution. As discussed in Section 2 it has been shown that GAs can be used to solve combinatorial optimization problems such as the TSP, GAP, and VRP. In this section we derive a GA, having the desirable attributes sought, for solving the UAV CMTAP with its special characteristics.

4.1. Encoding

The most critical part of deriving a GA is the chromosome encoding. For this problem the encoding of the GA chromosomes is based on the UAV CMTAP tree. It is customary to use a string for the chromosome encoding. However, for simplifying the encoding process and the application of the genetic operators, for our problem we use a matrix. Each chromosome matrix is composed of two rows and N_c columns. The columns of the chromosome matrix correspond to genes. We define the set of genes as $G = \{1, 2, \ldots, N_c\}$. The first row presents the assignment of vehicle $i \in V$ to perform a task on target $j \in T$ appearing in the second row. The ordering of appearance of the target determines which task $k \in M$ is being performed; e.g. the first time a target appears in the second row (from left to right) means it has been assigned to be classified. Thus, we avoid the need to explicitly specify the task being performed and are able to simplify

Vehicle	1	1	2	2	1	2	2	2	1
Target	2	3	1	3	1	1	2	2	3

Fig. 2. Example of chromosome representation.

significantly the chromosome encoding of the assignments. We denote a feasible chromosome as defining an assignment for the cooperating vehicles performing exactly N_m tasks on each of the N_t targets.

An example chromosome for a problem of two vehicles ($N_v = 2$) performing the three tasks ($N_m = 3$) on three targets ($N_t = 3$) is shown in Fig. 2. It can be seen that the chromosome is of length $N_c = 9$ and the assignment is as follows: Vehicle 1 classifies target 2 and then classifies target 3. In the meanwhile vehicle 2 classifies target 1 and then attacks target 3 (after it has been classified by vehicle 1). After target 1 has been classified (by vehicle 2) it is attacked by vehicle 1 and then verified by vehicle 2. Vehicle 2 then attacks target 2 and verifies its kill. In the meantime vehicle 1 verifies the kill of target 3.

4.2. Computational complexity

Finding good solutions requires searching the space of feasible solutions. Thus, the number of feasible solutions is a measure of the problem's computational complexity. In the investigated problem the number of different feasible chromosomes N_f is given by

$$N_f = \frac{(N_t N_m)!}{(N_m!)^{N_t}} N_v^{N_t N_m}. \tag{10}$$

Note that N_f is an upper bound on the number of different feasible assignments (see the appendix for relevant proofs).

From Eq. (10) it is apparent that the number of feasible chromosomes explodes as the number of targets and/or vehicles is raised. Note that the dependency on the number of targets is larger than on the number of vehicles. This can be explained by noting that the addition of a target adds N_m genes to the chromosome while the addition of a vehicle just provides more possibilities for the current chromosome structure.

The fitness f of each of the solutions coded in the chromosomes will be based on computing the value of Eqs. (2) or (3) where

$$f = 1/J_i, \quad i = 1, 2. \tag{11}$$

This computation is accomplished by using the trajectory optimization subroutine of the MultiU*AV*2 simulation [19] that calculates the relevant $c_{l,i,j}^{X_{l-1}}$ for Eqs. (4) and (5). The calculation is performed using the Dubin's car model [20] and it enables enforcing flyable trajectories as well as timing constraints. The function also optimizes the path planning for each single assignment. Note that the computation of $c_{l,i,j}^{X_{l-1}}$ for $l \geqslant 2$ is dependent on the tasks performed by vehicle i prior to step l, since those tasks affect its trajectory. Thus, $c_{l,i,j}^{X_{l-1}}$ cannot be computed a priori and is different for every assignment.

The MultiU*AV*2 subroutine mentioned above allows only to perform the trajectory optimization independently for each stage $l \in S$. Thus, using it the optimization of the path planning is decoupled from the assignment problem. This greatly simplifies the computational complexity of the problem on the expense

of obtaining a suboptimal solution. Note that only when the turn radius of the aerial vehicle is very small compared to the distance between the targets then this decoupling may not have a negative effect on the overall optimality of the solution.

4.3. The solution process

The initial population is obtained by employing a random search method over the tree. In order to comply with the requirement of a feasible chromosome we impose that each target appears exactly N_m times in the bottom row of each chromosome. We denote the population size as N_s.

Producing children/offspring chromosomes from the parent ones, in each of the next generations, is composed of three stages:

4.3.1. Selection

We select randomly, using the roulette wheel method, two parents based on their fitness.

Let $C = \{c_1, c_2, \ldots, c_{N_s}\}$ be the set of chromosomes and using the mapping of Eq. (11) let $F = \{f_1, f_2, \ldots, f_{N_s}\}$ be the set of corresponding fitness of the chromosomes, that does not have to be ordered based on fitness. We now calculate

$$Fs = \sum_{\chi=1}^{N_s} f_\chi \tag{12}$$

and define the set $Fc = \{Fc_1, Fc_2, \ldots, Fc_{N_s}\}$ where

$$Fc_\xi = \sum_{\chi=1}^{\xi} f_\chi; \quad \xi = 1, 2, \ldots, N_s. \tag{13}$$

Note that $Fc_1 = f_1$ and $Fc_{N_s} = Fs$.

We then generate a uniformly distributed random number $\mu \sim U(0, Fs)$ and select as the first parent the chromosome that satisfies

$$parent = \underset{c_\xi \in C}{arg\ min}(Fc_\xi - \mu \geqslant 0) \tag{14}$$

and repeat the process for the second parent.

4.3.2. Crossover

We apply the crossover operator with a high probability p_c. Thus, there is a high probability that the parents will reproduce between themselves and a low probability $(1 - p_c)$ that the offsprings will be exact replicas of their parents.

In this study the single point crossover method has been chosen. This point is selected randomly based on a uniform distribution. When applying this operator, the first child solution consists of the first $g \in G$ genes (columns) from the first parent and N_c-g genes from the second parent; and vice versa for the second child.

In order for the children chromosome solutions to be feasible each target $j \in T$ has to appear exactly N_m times in the second row. We perform a check on the number of times each target appears in the N_c-g genes

switched between the parents. We start checking the second row of the switched part from its beginning (from left to right) and whenever a target appears more times than is required than that gene is changed randomly so as to reference a target that does not appear the required number of times. Thus, we enforce that each child chromosome is feasible. Note that using this crossover mechanism we do not change the first row of the switched part and hence the ordering of the vehicles performing tasks remains the same.

The application of this operator actually utilizes the implicit gradients in the problem. A crossover at one of the genes corresponds to a perturbation in the direction encoded in the other parent's gene. Note that, generally speaking, a crossover at one of the first genes corresponds to a larger perturbation than in one of the last genes since it has a higher probability of affecting the rest of the group plan.

4.3.3. Mutation

In this stage a mutation operator is applied with a small probability p_m to each gene (column) of the chromosome. We mutate only the identity of the vehicle $i_{old} \in V$ (first row) performing the task, so as not to affect the integrity of the assignment. The identity of the new vehicle is selected randomly such that $i_{new} \in V$ and $i_{new} \neq i_{old}$. The application of this operator prevents the algorithm from getting stuck in one branch of the tree and thus tries to avoid getting trapped in a local minimum. Generally speaking, as for the crossover operator, mutating the first genes has a larger effect on J than mutating the last genes, since it has a higher probability of affecting the rest of the group plan.

Note that whenever the first part of each offspring chromosome is identical to one of its parents (usually up to the crossover point, unless mutation has been performed) then there is no need in re-computing the cost of the sub-assignment encoded in that part of the chromosome. The cost can be obtained from the computations already performed, using the MultiU*AV*2 path optimization subroutine, for the parent and thus save computations.

4.4. Generations

We produce an entire generation at each step and keep a constant population size. In order to avoid the possibility of loosing the best solutions, when propagating to the new generation, we employ the *elitism* genetic operator and keep the best N_e chromosomes from the previous generation. The rest of the new chromosomes ($N_s - N_e$) are obtained by repeatedly producing children, by the methods described in the previous subsection, until the new population is filled. At this stage the new population replaces the entire previous generation of chromosomes. Off-line, the process of producing new generations can be repeated until some stopping criterion is met. For an online implementation the algorithm can be run for a specific allotted time. In this study the generations have been progressed for a given number of times, denoted N_g.

Since for each generation numerous candidate solutions are evaluated independently, the most straightforward parallel implementation would be the slave-master method. It will allow us to utilize possibly multiple on board processors. And, under a perfect communication assumption, it will allow us to distribute the evaluation process of the chromosomes between the different UAVs in the group.

5. Performance analysis

The performance of the proposed GA is analyzed in this section using simulation. We utilized the graphics and path optimization subroutines of the, publicly released, MultiU*AV*2 simulation [19]. Assuming a

Table 1
Simulation parameters

GA	Scenario
$N_s = 200$	$N_t \in \{3, 10\}$
$N_e = 6$	$N_v \in \{4, 8\}$
$p_m = 0.01$	$N_m = 3$
$p_c = 0.94$	$v = 105\,\text{m/s}$
$N_g = 100$	$\Omega_{max} = 0.05\,\text{rad/s}$

vehicle constant speed v and maximum turning rate Ω_{max}, the corresponding optimal trajectories (plotted in Figs. 7 and 8) consist of straight lines and arcs with radius $R_{min} = v/\Omega_{max}$ [20]. We assume in our study that the vehicles' resource limit is not reached, i.e. they each have enough fuel to accomplish any assignment (corresponding, from Eq. (9) to $b_i \to \infty\ \forall i$). The parameters used for the simulation and the GA are summarized in Table 1.

5.1. Monte Carlo study

A Monte Carlo study, consisting of 100 runs, is used in this section to compare the performance of the GA to the BFS and random search algorithms for the two different cost functions of Eqs. (4) and (5). Note that the random search algorithm explores only feasible solutions and the best solution is kept. Two problems of different sizes are examined: four UAVs on three targets and eight UAVs on 10 targets. The random variables are the initial location of the targets and the location and heading of the members of the UAV team.

To enable a comparison that is independent of the computer platform on which the different algorithms are run, we chose the independent variable in the following figures as the number of nodes (vehicle-target pairs) that need to be processed by the Multi*UAV*2 path optimization subroutine to compute the individual assignment cost $c_{l,i,j}^{X_{l-1}}$. This computation, that involves optimizing the UAV trajectories for a given vehicle–target pair, is the main computational burden in running the different algorithms (between 70% and 85% in the investigated cases). For the sake of completeness the corresponding approximate run time on a personal computer (Pentium IV—2400 MHz) is given as an additional top axis in each figure.

First, the small size scenario of four UAVs on three targets is analyzed. In Figs. 3 and 4 the convergence of the measures of interest $J_{1\,\min}/J_1$ and $J_{2\,\min}/J_2$ are plotted where $J_{1\,\min}$ and $J_{2\,\min}$ are the costs of the a posteriori best assignments for the cost functions of Eqs. (4) and (5), respectively (found using the BFS algorithm). For the cost function J_1 it is apparent that, in the domain of interest of short run time (enabling on-board implementation), the GA outperforms the other methods. However, for J_2 while the GA still performs better than random search, the best performance is obtained using the BFS algorithm. The improved performance of the BFS algorithm can be explained by noting that for the cost function J_2 the BFS algorithm prunes the tree considerably faster than for the cost function J_1, since it is dependent on the trajectory of only one vehicle. Note that on average the BFS algorithm finishes exhaustively searching the tree after 8×10^5 nodes (corresponding to a run time of approximately 114 s) for J_2 compared to 3.3×10^6 nodes (corresponding to a run time of approximately 471 s) for J_1.

Performing a complete search of the tree using the BFS algorithm in a problem of higher dimension ($N_v = 8$, $N_t = 10$) proved computationally infeasible on a personal computer (Pentium IV—2400 MHz);

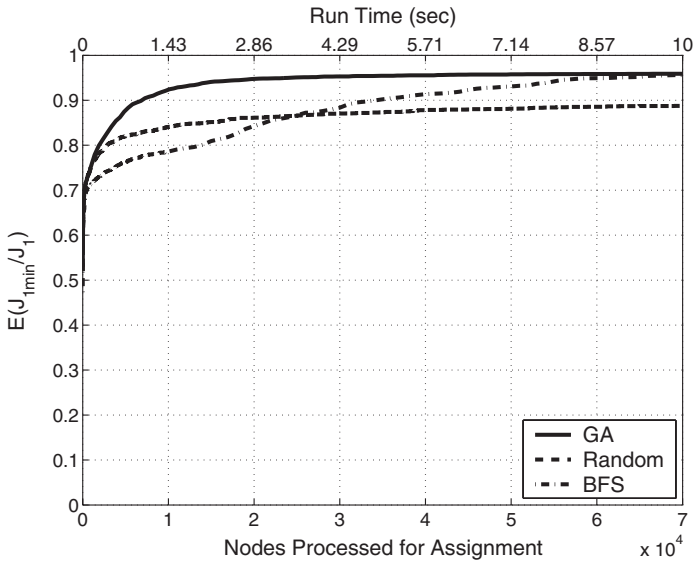

Fig. 3. Mean of Monte Carlo runs: $4 \times 3 \times 3$ scenario, J_1 cost function.

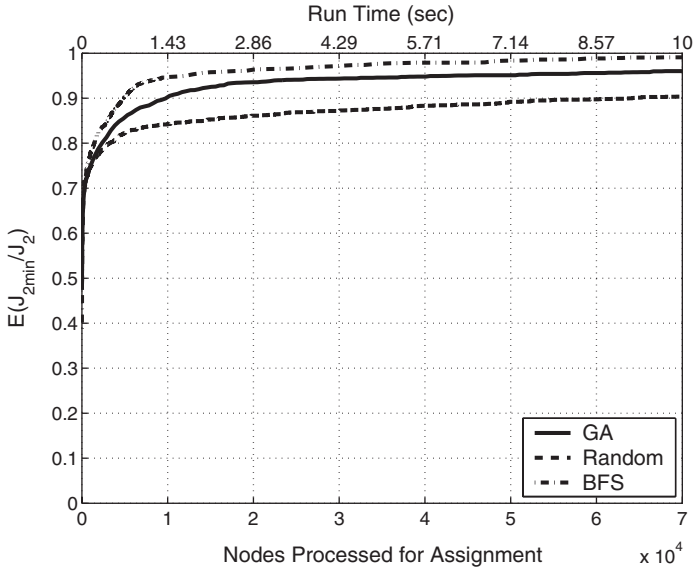

Fig. 4. Mean of Monte Carlo runs: $4 \times 3 \times 3$ scenario, J_2 cost function.

since one run time of the BFS algorithm took more than 3 weeks. Thus, the optimal assignments were not found and the normalizing factors $J_{1\ \min}$ and $J_{2\ \min}$ are the costs of the best solutions found in the given time by all the methods (by GA in these cases). In Figs. 5 and 6 the average of $J_{1\ \min}/J_1$ and $J_{2\ \min}/J_2$ as a function of the number of nodes computed and run time is plotted for the different algorithms. The superior performance of the GA is clearly evident for both cost functions.

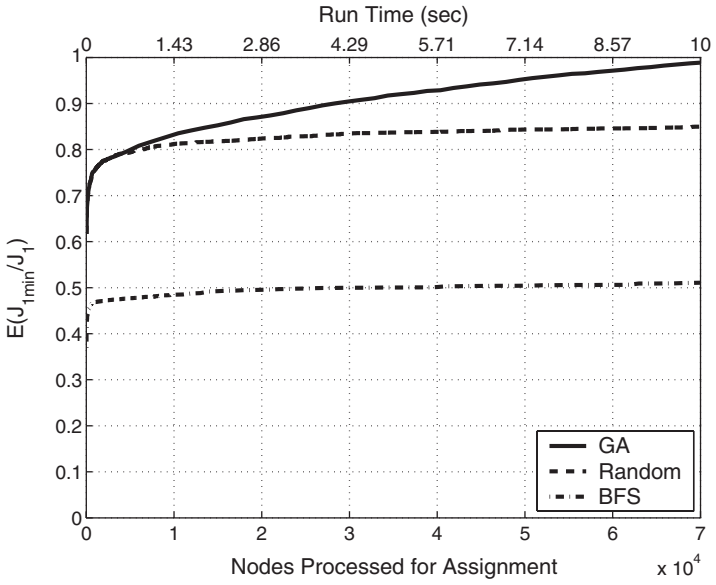

Fig. 5. Mean of Monte Carlo runs: $8 \times 10 \times 3$ scenario, J_1 cost function.

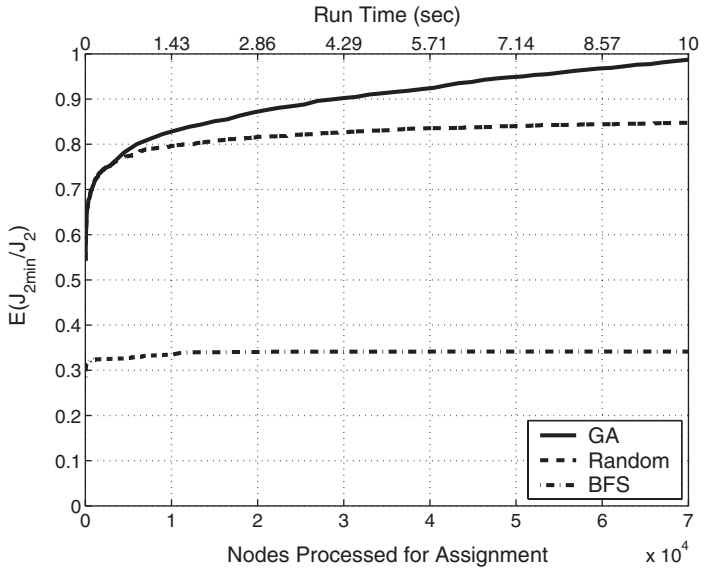

Fig. 6. Mean of Monte Carlo runs: $8 \times 10 \times 3$ scenario, J_2 cost function.

Note that in all the cases investigated the standard deviation of the solution using the GA was of the same order as the other methods. Also, the value of the standard deviation when using the three methods was small (< 0.12), thus validating that enough Monte Carlo runs have been performed.

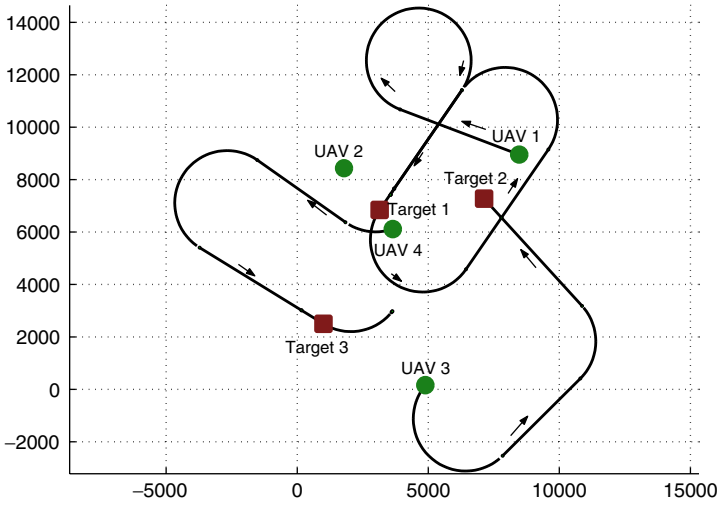

Fig. 7. Example trajectories, for a $4 \times 3 \times 3$ scenario, produced by the genetic algorithm, J_1 cost function.

5.2. Sample runs comparison

In this section the UAV trajectories that correspond to sample representative solutions found by the GA algorithm for a scenario of four UAVs on three targets are plotted. The cost functions being optimized are J_1 and J_2 and the solutions have been obtained after computing 7×10^4 nodes (corresponding to approximately 10 s of run time). In both cases the initial location of the targets and the location and heading of the members of the UAV team are identical. Note that the trajectories of each UAV are plotted from its initial location until completion of its part of the task assignment. Thus, when a UAV does not participate is executing the group task assignment only its initial location is shown.

The corresponding routes for the GA solution providing a cost $J_1 = 92,128$ m are plotted in Fig. 7. A circle represents a vehicle and a square represents a target. The cooperative assignment is as follows: vehicle 3 classifies and attacks target 2. Vehicle 4 classifies and attacks target 3 and then verifies the kill of target 2. Vehicle 1 classifies and attacks target 1 and then verifies the kill of targets 1 and 3. Vehicle 2 is not involved in the assignment and thus its trajectory, for conducting the default task of searching for new targets, is not plotted.

The UAVs' trajectories, corresponding to the GA solution providing $J_2 = 29,358$ m are plotted in Fig. 8. The cooperative assignment is as follows: vehicle 1 classifies target 2. Vehicle 3 classifies and attacks target 1. It then attacks target 2 and verifies the kill of target 3, after it has been classified and attacked by UAV 2. To finish the assignment, UAVs 2 and 4 verify the kill of targets 2 and 1, respectively.

The enhanced cooperation in Fig. 8 compared to Fig. 7 is evident since all 4 vehicles participate in executing the group tasks. Note that the enhanced cooperation when using J_2 instead of J_1 is typical over the Monte Carlo simulations. As expected, the enhanced cooperation results in a shorter maximum path for finishing all the tasks (29, 358 m compared to 47, 814 m for the trajectories given in Fig. 7). This results with only a marginal expense on longer overall trajectories (92, 342 m compared to 92, 128 m for the trajectories given in Fig. 7). Thus, when it is critical to service targets as quickly as

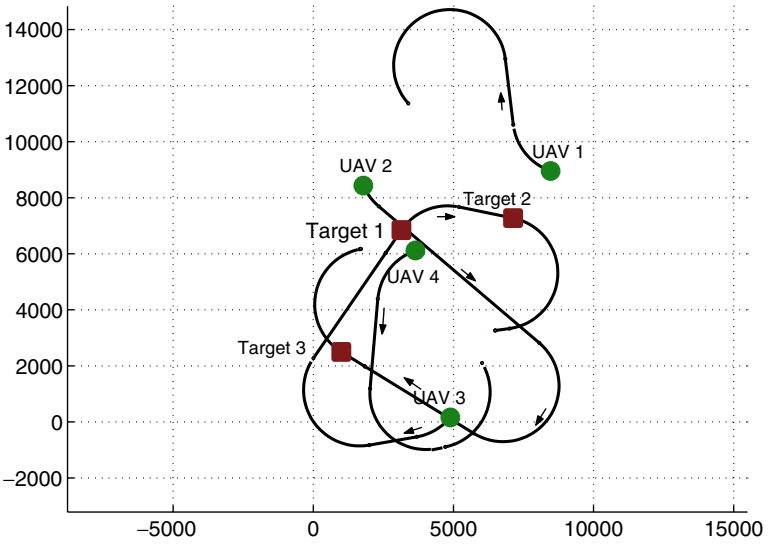

Fig. 8. Example trajectories, for a $4 \times 3 \times 3$ scenario, produced by the genetic algorithm, J_2 cost function.

possible, e.g. hunting mobile ballistic missile launchers, the penalty is in somewhat increased overall team trajectories.

6. Conclusions

A new cooperative multiple task assignment problem has been posed and its computational complexity has been analyzed. The scenario of interest was of assigning multiple uninhabited aerial vehicles to perform multiple tasks on multiple targets. A GA has been proposed for solving this NP-hard problem that has prohibitive computational complexity for classical combinatorial optimization methods. A matrix representation of the GA chromosomes simplifies the encoding process and the application of the genetic operators.

Using simulations the performance of the algorithm was compared to stochastic random search and deterministic branch and bound search methods for two different cost functions. The selection of the cost function has a considerable effect on the level of the group cooperation and consequently on their trajectories. The main advantage of using the genetic algorithm has been established in large size problems by providing good solutions considerably faster than the other search methods. This enables real-time implementation for high dimensional problems.

Acknowledgements

This research was performed while the first author held a National Research Council Research Associateship Award at the U.S. Air Force Research Laboratory (AFRL). The authors wish to express their

gratitude to the anonymous reviewers for their constructive and insightful comments that helped improve the paper.

Appendix

Theorem 1. *The number of different feasible chromosomes is given by*

$$\frac{(N_c)!}{(N_m!)^{N_t}} N_v^{N_c}. \tag{15}$$

Proof. Assigning a vehicle $i_1 \in V$ to perform a task on a given target $j_1 \in T$ appearing in column $g_1 \in G$ is independent from assigning a vehicle $i_2 \in V$ to perform a task on a given target $j_2 \in T$ appearing in column $g_2 \in G$ for $g_1 \neq g_2$. Hence, the number of possible assignments of vehicles to a given target/task sequence (given bottom row) is $N_v^{N_c}$.

Assigning a target $j_1 \in T$ to a given vehicle $i_1 \in V$ appearing in column $g_1 \in G$ is dependent on the assignment of a target $j_2 \in T$ to a given vehicle $i_2 \in V$, since each target should be serviced by the UAV team exactly N_m times. Using the combinatorial relationship from [21], the number of possible target assignments to a given vehicle sequence (given top row) is $(N_c)!/(N_m!)^{N_t}$.

Since assigning a specific vehicle $i \in V$ to perform a specific task on target $j \in T$ are mutually independent events the two rows are independent and the theorem is proved. □

Theorem 2. *The number of different feasible chromosomes is an upper bound on the number of different feasible assignments.*

Proof. Let c_1 be a feasible chromosome where in column $g_1 \in G$, vehicle $i_1 \in V$ is assigned to target $j_1 \in T$; and in the nearby column $g_2 \in G$, vehicle $i_2 \in V$ is assigned to target $j_2 \in T$ where $i_1 \neq i_2$ and $j_1 \neq j_2$. Now let c_2 be a similar chromosome having the same genes $g_a \in G\ \forall a > 2$ and the two remaining genes are switched. It is apparent that the assignments encoded in chromosomes c_1 and c_2 are identical. Thus, different chromosomes may encode the same assignments and the theorem is proved. □

References

[1] Ryan JL, Bailey TG, Moore JT, Carlton WB. Unmanned aerial vehicles (UAV) route selection using reactive Tabu search. Military Operations Research 1999;4:5–24.
[2] Kinney Jr GW, Hill RR, Moore JT. Devising a quick-running heuristic for an unmanned aerial vehicle (UAV) routing system. Journal of Operational Research Society, 2004.
[3] Schumacher CJ, Chandler PR, Rasmussen SJ. Task allocation for wide area search munitions. In: Proceedings of the American control conference, Anchorage, AK. 2002.
[4] Chandler PR, Pachter M, Swaroop D, Fowler JM, Howlet JK, Rasmussen S, Schumacher C, Nygard K. Complexity in UAV cooperative control. In: Proceedings of the American control conference, Anchorage, AK. 2002.
[5] Richards A, Bellingham J, Tillerson M, How JP. Coordination and control of multiple UAVs. In: Proceedings of the AIAA guidance, navigation, and control conference, Monterey, CA. 2002.
[6] Schumacher CJ, Chandler PR, Pachter M, Pachter L. Constrained optimization for UAV task assignment. In: Proceedings of the AIAA guidance, navigation, and control conference, Providence, RI. 2004.

[7] Chandler PR, Pachter M, Rasmussen S, Schumacher C. Multiple task assignment for a UAV team. In: Proceedings of the AIAA guidance, navigation, and control conference, Monterey, CA. 2002.
[8] Schumacher CJ, Chandler PR, Rasmussen SJ. Task allocation for wide area search munitions via iterative network flow optimization. In: Proceedings of the AIAA guidance, navigation, and control conference, Monterey, CA. 2002.
[9] Rasmussen SJ, Chandler PR, Mitchell JW, Schumacher CJ, Sparks AG. Optimal vs. heuristic assignment of cooperative autonomous unmanned air vehicles. In: Proceedings of the AIAA guidance, navigation, and control conference, Austin, TX. 2003.
[10] Rasmussen SJ, Shima T, Mitchell JW, Sparks A, Chandler PR. State-space search for improved autonomous UAVs assignment algorithm. In: Proceedings of the IEEE conference on decision and control, Paradise Island, Bahamas. 2004.
[11] Spall CJ. Introduction to stochastic search and optimization, 1st ed. Wiley Interscience Series in Discrete Mathematics and Optimization. Hoboken, NJ: Wiley; 2003.
[12] Mitchell M. An introduction to genetic algorithms. Cambridge, MA: MIT; 1996.
[13] Bornholdt S. Genetic algorithm dynamics on a rugged landscape. Physical Review E 1998;57(5):3853–60.
[14] Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley; 1989.
[15] Cantu-Paz E. Implementing fast and flexible parallel genetic algorithms, vol. 3, Genetic Algorithms: Complex Coding Systems, 1st ed. Boca Raton, FL: CRC Press; 1999.
[16] Kuhn HW. The Hungarian method for the assignment problem. Naval Research Logistic Quaterly 1955;2:83–97.
[17] Chu PC, Beasley JE. A genetic algorithm for the generalised assignment problem. Computers and Operations Research 1997;24:17–23.
[18] Baker BM, Ayechew MA. A genetic algorithm for the vehicle routing problem. Computers and Operations Research 2003;30:787–800.
[19] Rasmussen SJ, Mitchell JW, Schulz C, Schumacher CJ, Chandler PR. A multiple UAV simulation for researchers. In: Proceedings of the AIAA modeling and simulation technologies conference, Austin, TX. 2003.
[20] Dubins L. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal position. American Journal of Mathematics 1957;79:497–516.
[21] Bose RC, Menvel B. Introduction to combinatorial theory, 3rd ed. McGraw-Hill Series in Industrial Engineering and Management Science. New York: McGraw-Hill Book Co.; 2000.