UAV Cooperative Multiple Task Assignments using Genetic Algorithms

Tal Shima, Steven J. Rasmussen, and Andrew G. Sparks

Abstract— **A multiple task assignment problem for cooperating uninhabited aerial vehicles is posed as a combinatorial optimization problem. A genetic algorithm for assigning the multiple agents to perform multiple tasks on multiple targets is proposed. The algorithm allows efficiently solving this NP-hard problem that has prohibitive computational complexity for classical combinatorial optimization methods. It also allows taking into account the unique requirements of the scenario such as task precedence and coordination, timing constraints, and flyable trajectories. 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, providing good feasible solutions quickly. Moreover, it converges near to the optimal solution considerably faster than the other methods for some test cases. This makes real-time implementation for high dimensional problems feasible.**

I. 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 harms way, the lack of a human pilot enables significant weight savings, lower costs, and gives an opportunity for new operational paradigms. To realize these advantages, the UAVs must 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 control algorithms. Scenarios of particular interest are the wide area search and destroy (WASD) [1] 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.

The authors are with the Control and Optimization Branch, Air Vehicles Directorate, Air Force Research Laboratory, Room 305, Building 146, Wright-Patterson AFB, OH, 45433. This research was performed while the first author held a National Research Council Research Associateship Award.

Tal Shima is a Visiting Scientist. Senior member AIAA. shima_tal@yahoo.com

Steven J. Rasmussen is an on-site contractor with General Dynamics Advanced Information Engineering Services. Member AIAA. steven.rasmussen@wpafb.af.mil

Andrew G. Sparks is a Senior Aerospace Engineer. Associate fellow AIAA. andrew.sparks@wpafb.af.mil

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 control problems is complexity. The size of the problem (e.g. number of vehicles, targets, and threats) impacts complexity. However, in scenarios such as WASD and combat ISR, coupling between the performance of the four different tasks (search, classification, attack, and verification) has the most significant impact on complexity [2].

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) [3], [4], the capacitated transhipment problem [1], and the iterative capacitated transhipment problem [5]. Due to the special characteristics of the problem and the requirement for a tractable solution, all of the proposed algorithms are suboptimal in some sense. E.g. the MILP algorithm of [3] uses Euclidean distances while that of [4] uses piecewise UAV trajectories between targets; and hence both do not take into account optimally the need for flyable trajectories. The single task assignment algorithm [1] 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 iterations on the single task assignment algorithm (that utilize in essence a greedy solver) [5] provide a solution to the multiple task assignment requirement, it is heuristic in nature and therefore optimality can not be asserted. Note also that for most significant problems some of this algorithms, such as the MILP ones, take a long time to set up and to execute.

In a recent paper [6] 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 of 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 [7]. This deterministic greedy search method has desirable qualities such as providing immediately a feasible solution, that monotonically improves and, eventually, converges to the optimal solution.

Stochastic and non-gradient search methods 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. Assuming that the search space is not extremely rugged it will quickly yield good feasible solutions and will not converge to a local minima; 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 task assignment problem. The remainder of this manuscript is organized as follows: In the next section, the GA optimization method and its previous use for solving classical task assignments problems is discussed. Then, the UAV cooperative task assignment problem is posed. This is followed by the synthesis of the GA for the studied problem. A performance analysis is then presented and concluding remarks are offered in the last section.

II. Methodology

In this section the GA methodology and its previous use in solving classical assignment problems are reviewed.

A. GA - Brief Review

The GA is a stochastic and non-gradient search method described in many papers and textbooks including [8], and will be only briefly reviewed in this section. It enables searching efficiently a decision state space of possible solutions, as long as the search space is not extremely rugged. The method comprises of 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 taken 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 is of being selected. Crossover is performed in single or multi points across the chromosome, selected randomly. *E.g.*, 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 exchanging randomly one of the genes in the child chromosome. This operator prevents the convergence of the algorithm to a local minimum. The Elitism operator is used to save the best solutions of the generation.

B. GA in Classical Combinatorial Optimization Problems

Classical combinatorial optimization problems have been solved using GAs. Such problems include the travelling salesman problem (TSP), the generalized assignment problem (GAP), and 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 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. Much work in applying GAs to the TSP [8] is concerned with the encoding of the chromosomes and the use of special crossover operators that preserve its validity. In Ref. [9] 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 [10] using a pure GA and a hybrid of the GA with neighboring 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.

III. UAV Task Assignment Problem

Based on [1] a generic UAV task assignment problem is defined. In such a problem the UAVs are required to perform three tasks (classify, attack, and verify) on each of the targets. The requirement of flyable trajectories dictates a lower bound on the turn radius and speed of the aerial vehicles. Thus, the problem is denoted bounded speed task assignment problem (BSTAP) [7].

A. Tasks

It is assumed that the terrain has already been searched (by other UAVs or means) and a few targets have been found. Let $T = \{1, 2, ..., N_t\}$ be the set of targets found and let $V = \{1, 2, ..., N_v\}$ 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 denote N_m as 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 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.

B. Assignment Requirements

The requirements from the assignment algorithm for a feasible efficient 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 efficiency 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 guaranties that an assignment can be performed by the UAV team members. Otherwise, assigned tasks may not be executed and the team coordination could collapse.

C. Tree Representation

In [6] it was demonstrated that the BSTAP can be represented by a tree. This tree not only spans the decision space of the BSTAP, 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 BSTAP. Nodes are constructed until all of the combinations of vehicles, targets, and tasks have been taken into account.

In the case of a BSTAP, 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 the number of tasks N_m is three. Traversing the tree from a root node to a leaf node produces a feasible set of assignments for the UAV group. This makes it possible to find feasible assignments in a known time, *i.e.* node processing rate times N_c.

D. Performance Requirements

The performance metric for the BSTAP analyzed in this paper is defined as the cumulative distance travelled by the vehicles to perform all of the required tasks

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

where R_i is the distance travelled by UAV $i \in V$ until finishing his part in the group task plan. After that instance the UAV has no more group tasks to fulfill and can resume a default task, *e.g.* searching for new targets. The group objective is to minimize Eq. 2 subject to the assignment requirements of subsection III-B. Enforcing these requirements leads to a very large combinatorial problem, discussed next.

E. Combinatorial Optimization Problem

The problem is of minimizing the cost function of Eq. 2 by optimizing the N_c assignments of vehicles to targets. Let $S = \{1, 2, .., N_c\}$ be the set of stages in which the assignment is made, where each stage $l \in S$ corresponds to a row in the tree representation. 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 $X_l = \{x_{1,i,j}, x_{2,i,j}, .., x_{l,i,j}\}$ be the set of assignments up to, and including, stage l. Let $c_{l,i,j}^{X_{l-1}}$ be the cost (*e.g.* distance travelled) of assigning a vehicle $i \in V$ to perform a task on target $j \in T$ at a stage $l \in S$, given the prior assignment history X_{l-1}; $r_{l,i,j}^{X_{l-1}}$ be the resource (*e.g.* fuel) required to perform the task; and b_i as the resource availability (*e.g.* total fuel capacity) of vehicle $i \in V$.

The mathematical formulation of the BSTAP is

$$MinJ = \sum_{l=1}^{N_c} \sum_{i=1}^{N_v} \sum_{j=1}^{N_t} c_{l,i,j}^{X_{l-1}} x_{l,i,j} \tag{3}$$

subject to

$$\sum_{i=1}^{N_v} \sum_{j=1}^{N_t} x_{l,i,j} = 1 \quad \forall \quad l \in S \tag{4}$$

$$\sum_{l=1}^{N_c} \sum_{i=1}^{N_v} x_{l,i,j} = N_m \quad \forall \quad j \in T \tag{5}$$

$$\sum_{l=1}^{N_c} \sum_{j=1}^{N_t} r_{l,i,j}^{X_{l-1}} x_{l,i,j} \leq b_i \quad \forall \quad i \in V \tag{6}$$

Eq. 4 ensures that at each stage exactly one task on a target $j \in T$ is assigned to exactly one vehicle $i \in V$; Eq. 5 ensures that on each target $j \in T$ exactly N_m tasks are performed; and Eq. 6 ensures that the total resource requirement of the tasks from each vehicle $i \in V$ does not exceed its capacity.

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

Fig. 1. Sample chromosome representation.

IV. Genetic Algorithm Synthesis

The BSTAP discussed in the previous section is a very large combinatorial 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 solutions over time, and convergence near to the optimal solution. As discussed in section II it has been shown that GAs can be used to solve assignment problems such as the TSP, GAP, and VRP. In this section we derive a GA, having the desirable attributes sought, proposed for solving the cooperating UAVs BSTAP with is special characteristics.

A. Encoding

The encoding of the GA chromosomes is based on the tree described previously. Each chromosome is composed of two rows and N_c columns called genes. We define the set of genes as $G = \{1, 2, ..., N_c\}$. The first row presents the assignment of vehicle $i \in V$ to perform a task, on target $j \in T$ appearing on 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 significantly the chromosome encoding of the assignments.

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. 1. 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 meanwhile vehicle 1 verifies the kill of target 3.

B. Feasible Chromosomes

We will first define the meaning of the term *feasible chromosome* and then derive the number of such chromosomes in our problem.

Definition 1: Feasible chromosome - Defines an assignment for the $v \subseteq V$ cooperating vehicles performing exactly N_m missions on each of the N_t targets.

Theorem 1: The number of different feasible chromosomes N_f is given by

$$N_f = \frac{(N_c)!}{(N_m!)^{N_t}} {N_v}^{N_c} \tag{7}$$

Proof: *Assigning a vehicle $i_1 \in V$ to perform a task on a given target $j_1 \in T$ on column $g_1 \in G$ is independent from assigning a vehicle $i_2 \in V$ to perform a task on a given target $j_2 \in T$ on 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 $i_1 \in V$ on 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 [11], the number of possible target assignments to a given vehicle sequence (given top row) is $(N_c)!/(N_m!)^{N_t}$.

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

Theorem 2: An upper bound on the number of different feasible assignments is N_f (given in Eq. 7)

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 define the same assignments and the Theorem is proved.* □

C. Fitness

The fitness f of each of the solutions coded in the chromosomes will be based on computing the value of Eq. 2 where

$$f = 1/J \tag{8}$$

This computation is accomplished by using the assignment function of the MultiUAV2 simulation [12] that calculates the relevant $c_{l,i,j}^{X_{l-1}}$ from Eq. 3. The calculation is performed using the Dubin's car model [13] 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 $l \geq 2$ is dependent on the tasks performed by vehicle i prior to step l.

Remark 1: Decoupling the optimization of the path planning from the assignment problem results in a suboptimal solution. However, it greatly simplifies the computational complexity of the problem.

D. The Reproductive Process

The initial population is obtained by employing a random search method of 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. The size of the population is denoted N_s and is selected based on heuristic arguments and is commonly $N_s \in [50, 200]$.

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

1) Selection: First we select randomly two parents based on their fitness. The Roulette wheel selection method is used in this paper.

2) Crossover: The single point crossover method has been chosen, applied with a probability p_c. We also enforce that the children chromosome solutions will be feasible, *i.e.* each target $j \in T$ appears exactly N_m times in the second row. The application of this operator allows utilizing implicitly the 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.

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 prevents it from converging to a local minima.

Note that, generally speaking, crossover or mutation at one of the first genes corresponds to a larger perturbation than in one of the last genes since it has a higher chance of effecting the rest of the group plan.

E. Generations Propagation

We propagate the 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 generating new generations can be repeated until some stopping criteria is met. For an online implementation the algorithm can be run for a specific allotted time. In this study, concerned with the BSTAP of cooperating UAVs the generations have been propagated for a given number of times, denoted N_g.

V. Performance Analysis

The performance of the proposed GA described above is analyzed in this section using simulation. The simulations were conducted using the 6 degrees of freedom MultiUAV2 simulation [12] with the vehicles' autopilot constructed to maintain constant altitude and speed. We assume in these simulations that the vehicles' resource limit is not reached, *i.e.* they each have enough fuel to accomplish any assignment (corresponding, from Eq. 6, to $b_i \rightarrow \infty \ \forall \ j$). The parameters used for the simulation and the GA are summarized in Table I.

GA	Scenario
$N_s = 200$	$N_t \in \{1, 2, 3\}$
$N_e = 6$	$N_v \in \{1, 2, 3, 4\}$
$p_m = 0.01$	$N_m = 3$
$p_c = 0.94$	$Speed = 100m/s$
$N_g = 100$	$Area = 50Km^2$

TABLE I
SIMULATION PARAMETERS.

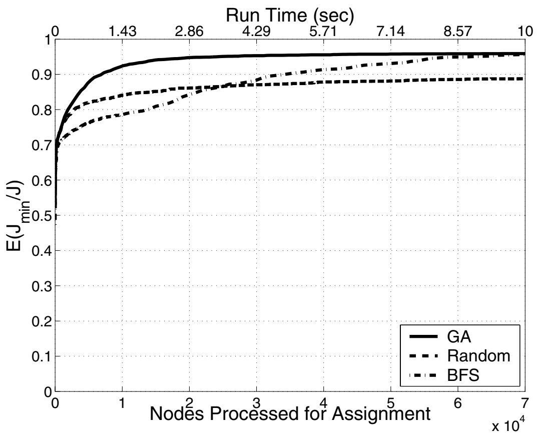

Fig. 2. Monte Carlo results: mean costs for 4x3x3 scenarios.

A. 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 other optimization methods. The random variables are the initial location of the targets and the location and heading of the members of the UAV team. Due to the need to search the entire solution space using the BFS algorithm, and computational limits, the comparison is performed for the medium dimensional scenario of $N_v = 4, N_t = 3$.

In Fig. 2 the average of the cost, normalized by J_{min} (cost of the, *a posteriori* optimal assignment), as a function of the number of nodes computed is plotted for the different algorithms. It is apparent that the mean of the GA solution converges near to the mean of the optimal solution much faster than the BFS and random search algorithms. Note that the standard deviation of the solution using the GA was of the same order as the other methods.

Remark 2: Performing the same comparison for a low dimension problem ($N_v = 2, N_t = 3$) yielded superior performance of the BFS algorithm, since all the decision state space was immediately exhaustively searched. However, trying to perform the comparison for a problem of higher dimension ($N_v = 8, N_t = 5$) proved computationally infeasible on a personal computer (Pentium IV - 2400MHz); since one run time of the BFS algorithm took more than three weeks. For comparison note that the GA provided a solution for the same problem within 1% of the optimal one after 25 sec of run time.

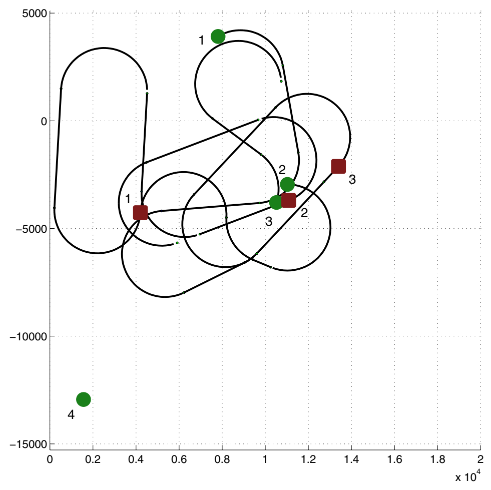

Fig. 3. Example trajectories, for a 4x3x3 scenario, produced by the first feasible assignment from the BFS algorithm.

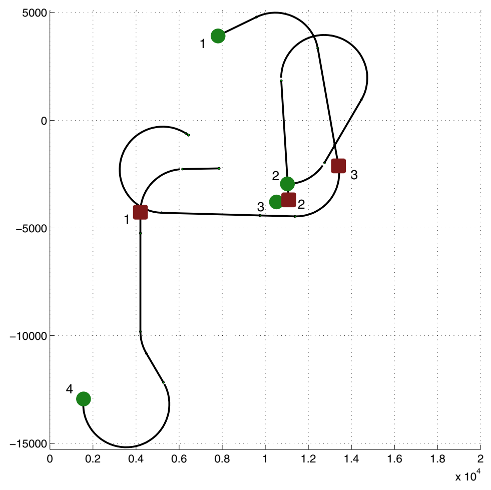

Fig. 4. Example trajectories, for a 4x3x3 scenario, produced by the genetic algorithm.

B. Sample Runs Comparison

The performance of the proposed GA is analyzed in this subsection using a representative sample run. The selected scenario is of medium dimension ($N_v = 4, N_t = 3$), used in the previous section. Corresponding routes for an initial solution and final GA solution with costs of 143,306m and 76,623m are plotted in Figs. 3 and 4, respectively; where a circle represents a vehicle and a square represents a target. The benefit of performing the optimization is immediately apparent by the simplified and shorter trajectories in Fig. 4

For clarification, the optimal cooperative assignment of Fig. 4 is as follows: vehicle 1 classifies and attacks targets 3; next, it verifies the kill of target 1 after it has been classified and attacked by vehicle 4, which then verifies the kill of target 3. Vehicle 2 classifies and attacks target 2 and then vehicle 1 verifies the kill of target 2. Vehicles 3 is not involved in the assignment and thus is conducting the default task of searching for new targets.

VI. Conclusion

The uninhabited aerial vehicles cooperative task assignment problem has been solved using a genetic algorithm. The proposed algorithm allows taking into account the unique requirements of the scenario such as multiple tasks, task precedence and coordination, timing constraints, and flyable trajectories. This stochastic non-gradient search method allows solving efficiently the assignment problem, without resolving to the computational complexity of classic combinatorial optimization methods. The genetic operator of crossover utilizes implicitly the gradients in the problem by performing perturbation around candidate solutions. The mutation genetic operator prevents the algorithm from converging to a local minima.

Using simulations the performance of the algorithm was compared to a stochastic random search and a deterministic branch and bound search methods. The main advantage of using the genetic algorithm has been established as providing near optimal solutions considerably faster than the other search methods; thus, enabling the real-time implementation in high dimensional problems.

References

[1] Schumacher, C. J., Chandler, P. R., and Rasmussen, S. J., "Task Allocation for Wide Area Search Munitions," *Proceedings of the American Control Conference*, Anchorage, Alaska, 2002.

[2] Chandler, P. R., Pachter, M., Swaroop, D., Fowler, J. M., Howlet, J. K., Rasmussen, S., Schumacher, C., and Nygard, K., "Complexity in UAV Cooperative Control," *Proceedings of the American Control Conference*, Anchorage, Alaska, 2002.

[3] Richards, A., Bellingham, J., Tillerson, M., and How, J. P., "Coordination and Control of Multiple UAVs," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, 2002.

[4] Schumacher, C. J., Chandler, P. R., Pachter, M., and Pachter, L., "Constrained Optimization for UAV Task Assignment," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Providence, RI, 2004.

[5] Schumacher, C. J., Chandler, P. R., and Rasmussen, S. J., "Task Allocation for Wide Area Search Munitions Via Iterative Network Flow Optimization," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.

[6] Rasmussen, S. J., Chandler, P. R., Mitchell, J. W., Schumacher, C. J., and Sparks, A. G., "Optimal vs. Heuristic Assignment of Cooperative Autonomous Unmanned Air Vehicles," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Austin, Texas, 2003.

[7] Rasmussen, S. J., Shima, T., Mitchell, J. W., Sparks, A., and Chandler, P. R., "State-Space Search for Improved Autonomous UAVs Assignment Algorithm," *Proceedings of the IEEE Conference on Decision and Control*, 2004.

[8] Goldberg, D. E., *Genetic Algorithms in search, optimization, and machine learning*, Addison-Wesley, Reading, Massachusetts, 1989.

[9] Chu, P. C. and Beasley, J. E., "A Genetic Algorithm for the Generalised Assignment Problem," *Computers and Operations Research*, Vol. 24, 1997, pp. 17–23.

[10] Baker, Barrie, M. and Ayechew, M. A., "A Genetic Algorithm for the Vehicle Routing Problem," *Computers and Operations Research*, Vol. 30, 2003, pp. 787–800.

[11] Bose, R. C. and Menvel, B., *Introduction to Combenatorial Theory*, McGraw-Hill Book Co., New York, 3rd ed., 2000, McGraw-Hill Series in Industrial Engineering and Management Science.

[12] Rasmussen, S. J., Mitchell, J. W., Schulz, C., Schumacher, C. J., and Chandler, P. R., "A Multiple UAV Simulation for Researchers," *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, Austin, TX, 2003.

[13] Dubins, L., "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal position," *American Journal of Math*, Vol. 79, 1957, pp. 497–516.