GENETIC ALGORITHM FOR DYNAMIC TASK ALLOCATION OF MULTI AUTONOMOUS UNMANNED AIR VEHICLES

Jamal A. F. Azzam
M.Sc, Electrical Engineering, Port Said Shipyard, Suez Canal Authority.

Prof. Dr. Hassan El Deip
Electrical Engineering Dept., Faculty of Engineering, Suez Canal University.

Prof. Dr. Sohiman M. Sharaf
Power And Machines Engineering Dept., Faculty of Engineering, Helwan, Helwan University

To use the mechanisms of the UAVs, we are making a new approach to overcome problems through computing methods and strategies. The primary focus of this study is on the development of a novel genetic algorithm that can be used to allocate tasks effectively and efficiently. The proposed algorithm is tested on a set of benchmark problems, and the results show that it can solve complex assignment problems more effectively than existing methods. The algorithm also demonstrates its capability to handle dynamic environments, where the task requirements change over time. The developed approach offers a promising solution for task allocation problems in the field of unmanned aerial vehicles. This study opens new avenues for further research in the area of autonomous systems, especially in the context of military applications.

ABSTRACT:
Unmanned aerial vehicles (UAVs) have useful military applications, including reconnaissance, search and destroy, search and rescue missions in hazardous environments such as battlefields or disaster areas. Recently, there has been considerable interest in the possibility of using large teams of UAVs functioning cooperatively to accomplish a large number of tasks e.g., attacking targets. However, this requires the assignment of multiple spatially distributed tasks to each UAV along with a feasible path that minimizes effort and avoids threats.

Task Allocation (TA) is one of the core steps to effectively exploit the capabilities of cooperative control of multiple UAV teams. It is an NP-complete problem “non-deterministic polynomial time”. So the computation can’t be implemented in real time, no chance for cooperation among the team members, and no autonomy for these vehicles. The reported papers in this field consider the problem in static condition using different techniques (e.g., auction based, scheduling, linear programming).

In this paper, a new dynamic task allocation algorithm is presented that is based on the principles of genetic algorithm (GA). It discusses the adaptation and implementation of the GA search strategy to the task allocation problem in the cooperative control of multiple UAVs. Simulation results indicate that the GA strategy is a feasible approach for the task allocation problem, and the resulted task assignment is near optimal. This means that the total cost of the team is minimized. A major advantage is its low computation cost.

Accepted April 10, 2007
I. Introduction:
Recently military conflicts have demonstrated the strategic value of UAVs. The roles of UAVs are evolving from reconnaissance purpose to offensive mission as missile launching platform. The capabilities of UAVs will be further improved if multiple UAVs are cooperative [1]. Achievement of cooperation among UAVs requires a method of assigning tasks. TA is an important problem to minimize the overall team cost. Besides, the assigning algorithm must ensure that all targets are approached in an optimal manner.

There are two types of task allocation problems: static and dynamic. Static task allocation means that the assignment may be made at time t such that all of the UAVs are committed, while dynamic task allocation is made at any of several discrete points of time [2]. Determining which of U agents are assigned to which of T targets is a problem of order T^U in complexity, so task allocation is an NP complete optimization problem [3].

There are a wide variety of approaches that have been reported for solving the task allocation problem in various applications. They can be classified into the following categories: network flow optimization [4], market based approach [5], integer linear programming, [6] fuzzy approach [7], and genetic algorithms [8].

Because of the intractable nature of the TA problem and its importance in cooperative control, it is desirable to explore other avenues for developing good heuristic algorithm for the problem. The genetic algorithm (GA) is an intelligent probabilistic search algorithm that models the process of nature selection and genetics [9]. It is an iterative algorithm that maintains a pool of feasible solutions for each iteration. The GA starts with a set of randomly selected chromosomes as the initial population that encodes a set of possible solutions. Variables of a problem are represented as genes in a chromosome, and chromosomes are evaluated according to their fitness values, which are obtained by evaluating the considered fitness.

Recombination typically involves two operators: (1) crossover and (2) mutation. Genetic operators alter the composition of genes to create new chromosomes referred to as offspring. The selection operator is an artificial version of nature selection, a Darwinian survival of the fittest among populations, to create populations from generation to generation. Chromosomes with better fitness have higher probabilities of being selected in the next generation. After several generations, GA can converge to the best solution. GA has many advantages over other heuristic techniques. For example, it can be implemented in a few lines of computer code, it requires only primitive mathematical operators, and it has high probability to escape local minima.

In this paper, an explanation of a genetic algorithm in dynamic case is introduced. For a UAV in a fleet to be autonomous, it has to compute its trajectory and specify its target in real time. If a pop-up threat is arisen or if a member of the fleet is lost re-planning for the trajectory and reallocation for targets has to be done in real time, to optimize the overall mission cost i.e. to minimize the UAVs trajectories threats and length. Consequently, the fuel consumption and the Vehicles flying time are minimized. The proposed algorithm is implemented in dynamic situations. It gives the near optimal solutions in a few seconds which is suitable for fast reaction of the vehicles to the new situations. The allocations in this case may not be the same allocations produced when the mission started. The rest of this paper is organized as follows:

Section 2 description of the GA algorithm, section 3 introducing GA algorithm for task allocations section 4 implementing GA for Dynamic allocations, section 5 the simulation results.
2. Description of the Genetic Algorithm:

The genetic algorithm is a stochastic optimization algorithm that was originally motivated by the mechanisms of natural selection and evolutionary genetics. Over the last decade, GA has been extensively used as search and optimization tools in various problem domains, including science, commerce and engineering. The primary reasons for their success are their broad applicability, ease of use and global perspective. There are some differences between the GA and traditional searching algorithms. They can be summarized as follows [11]:

- The algorithm works with a population of strings searching many peaks in parallel, as opposed to a single point.
- The GA works directly with strings of characters representing the parameter sets, not the parameters themselves.
- The GA uses probabilistic rules instead of deterministic rules.
- The GA uses objective function information instead of derivatives or other auxiliary knowledge.

GA is inherently parallel, because it simultaneously evaluates many points in the parameter space (search space). So, the GA has a reduced chance of converging to local optimum and would be more likely to converge to global optimum. It requires only information concerning the quality of the solution produced by each parameter set (objective function values). This differs from many optimization methods which require derivative information or, worse yet, a complete knowledge of the problem structure and parameters. Since the GA does not require such problem specific information, it is more flexible than that most search methods [2]. Typically, the GA is characterized by the following components:

- A population of encoded solution.
- A fitness function that evaluates the optimality of each solution.
- Genetic operators that generate a new population from the existing population.
- Control parameters.

The basic flow chart of the GA is illustrated in Fig. 1 where \( \epsilon > 0 \) a small number to check convergence.

3. The Proposed Technique:

3.1 Task Allocation (static situation):

For task allocation problems, applying the normal mutation and cross over procedures on a binary representation will lead to illegitimate solutions i.e. assigning a non-existing targets to a UAV (it will produce targets number more than the specified number of targets \( T \)). A number of researchers [9],[11],[12] have created operators that overcome this problem by implementing an operator called ordered crossover to be used. For targets allocation problems the same operator can be used, and the coding can be alphabetic or numeric. This operator builds offspring by choosing a subsequence of UAVs within a list of one parent. It also preserves the relative ordering of UAVs from the other parent.
Assuming the number of UAVs equals the number of targets i.e. \( U = T \), for simplicity numeric codes for targets are used i.e. 1 2 3 4 5 6 7 8 9, and letters are used for UAVs: A B C D E F G H I. As an initia, population the first parent P1 and the second parent P2 can be randomly generated as:

\[
\begin{align*}
P1 & : B C H G E A F I \\
P2 & : F H C D G A I B E
\end{align*}
\]

(1)

This means that for P1 UAV B is assigned to target #1, C is assigned to target #2, H is assigned to target #3 and so on. Two children C1 and C2 are produced in the following way. First, two cut lines are chosen for C1 and C2 for the high fitness chromosomes. The segments between cut points are copied into the following offspring:

\[
\begin{align*}
P1 & : 1 2 3 4 5 6 7 8 9 \\
P2 & : X X X X G E A D X X
\end{align*}
\]

(2)

Next, starting from the second cut point of one parent, the UAVs of the other parent are copied in the same order, omitting UAVs already present between the two cut lines.

\[
\begin{align*}
P1 & : 1 2 3 4 5 6 7 8 9 \\
P2 & : B F H G E A D I B
\end{align*}
\]

(3)

The position of the segments in the two children C1, C2 is the same and depends on the heights fitness. For one parent C1 a mutation operator can be done by exchanging places of randomly selected UAVs in the children chromosomes. Evaluation can be done by using a suitable fitness functions \( f \). The fitness function for any chromosome can be evaluated as \( f_i = 1 / \cos \theta_i \) or \( f_i = Q / \cos \theta_i \), where Q is a suitable weighting factor. As the cost of the chromosome is reduced its fitness is increased. Consequently, its probability to be repeated in the next offspring is increased. It is called the repetition rate (RR) in the next offspring. This RR is evaluated as the fitness of the individual chromosome over the average fitness i.e.

\[
RR_i = f_i / (\sum (f_i) / n)
\]

(5)

The chromosomes that have values \( \geq 1 \) are optimal and kept in the next offspring. So, the solution is approaching optima by increasing all RRi to be \( \geq 1 \) \((i = 1, ..., n)\) with iterations. It must be noted that if the RR increased than 1 it is considered to be only 1 since each UAV is assigned to one target only. The optimal solution for the team is reached when RRi \( \geq 1 \) for \( i = 1, ..., n \) i.e.

\[
\text{average RR} = \frac{\sum (RR_i)}{n} \geq 1
\]

(6)

Given the cost functions \( J \) for \( n \) UAVs to \( m \) targets, the algorithm can be explained as follows:

**Step 1** initiate two random population P1, P2 as shown in Eqn.1

**Step 2** evaluate the cost \( J_i \) for every chromosome in the population. Then compute the fitness function \( f_i \) by Eqn.4.

**Step 3** compute the repetition rate RR for every chromosome by Eqn. 5 for P1, P2.

**Step 4** for any chromosome in P1 where RR \( \geq 1 \) determine two cut lines around it. The same two cut lines for P1 are determined for P2. If the chromosome that have RR \( \geq 1 \) are not adjacent then more than two lines must be determined.

**Step 5** in P1 keep all chromosome in between the two lines in the next offspring C1 as shown in Eqn.2. Execute cross over starting from the first cut line in P2. Omit any existent chromosome (already kept in C1) as shown in Eqn.2.

**Step 6** execute the mutation process, simply by exchanging the position of two chromosomes in C1.

**Step 7** repeat the same steps 4, 5, 6 for P2. till fulfilling the exit condition.

**Step 8** repeat step 3 to step 8 till optimal or near optimal solution is reached. i.e. the converge \( < \epsilon \)
The solution can be detected from the solution average fitness \( \sum (f_i) / n \).  

3.2. Dynamic Reallocation:
The autonomous UAVs plan their trajectories, and the GA algorithm assigns them to targets.
These processes are in real time, while the vehicles are executing the planned trajectories. The situation in the battle field can be changed due to any unexpected reasons, e.g., new threats are detected, new targets are explored, a team member is lost, etc. In this case, the team re-plans trajectories according to the new situation. The GA is used to reallocate the team members.

The initial offspring is randomly chosen, but the last allocation before the new situation arises is the beginning population. So, if at time (t) a new situation arises and the allocation is C(t), C2(t), starting from the current positions of the (n) vehicles the real time trajectory planning algorithm computes the new feasible trajectories to the m targets. The (m x n) minimum cost functions J of these trajectories are produced and passed to the TA algorithm. By its turn the TA reallocates the vehicles from their current positions to the targets. The same algorithm explained in 3.1 is used considering the initial population as: P1 = C1(t), and P2 = C2(t)

4. Simulation Results:

In a real mission, the number of vehicles (m) may be more or less than the number of specified targets (n). So, some targets may be assigned to more than one vehicle (in the first case where m > n). In the other case (m < n) a vehicle may be assigned to more than one target one after another. In this paper, the following assumptions are considered:

- The number of vehicles and targets are equal i.e. m = n, and each target is assigned to one vehicle.
- The vehicles are equipped with the necessary systems such as: sharing information transceivers, anti-jammers, anti-decoys, anti-collision, and different flying phases algorithms (e.g. taking off, landing, attacking) etc.

4.1 Static Allocation:
The algorithm is applied to allocate 6 UAVs (A to F) to 6 targets (1 to 6), the costs of trajectories are shown in the Table 1.

These costs are produced from a trajectory planning algorithm explained in Fig. 2. The cost function J for each trajectory is composed of two elements.

\[ J = J_t + J_h \]  

Where \( J_t \) is the length cost and \( J_h \) is the threat cost.

<p>| Table 1: The Cost Functions For 6 UAVs Trajectories To 6 Targets |
|------------------|---|---|---|---|---|---|---|</p>
<table>
<thead>
<tr>
<th>T</th>
<th>U</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1215</td>
<td>976</td>
<td>1170</td>
<td>2190</td>
<td>2054</td>
<td>2223</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1214</td>
<td>1023</td>
<td>1237</td>
<td>1420</td>
<td>1972</td>
<td>1672</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1557</td>
<td>1397</td>
<td>1617</td>
<td>1498</td>
<td>2526</td>
<td>1280</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1456</td>
<td>1241</td>
<td>1458</td>
<td>1553</td>
<td>1684</td>
<td>1418</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1329</td>
<td>1225</td>
<td>1424</td>
<td>1759</td>
<td>1860</td>
<td>1597</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1916</td>
<td>2354</td>
<td>1498</td>
<td>2936</td>
<td>1328</td>
<td>1291</td>
<td></td>
</tr>
</tbody>
</table>

The task assignment is obtained in 5 iterations as shown in Table 2, with the maximum total fitness 5,661, and average RR ratio = (5,661/6) = 0.94 which is sufficient for a near optimal solution (optimal solution is achieved by verifying Eqn. 6.) (if RR > 1 it is considered only 1, since every UAV is assigned only to one target). Fig. 3 explains these trajectories for vehicles over a hostile territories. The trajectories are planned to avoid the surface to air missiles (SAMs) sites and to minimize the path length and probabilities of detection by the radar sites. The GA is used to allocate each vehicle to a target. The main objective is to optimize the overall cost for the team as a whole.

4.2 Dynamic Allocation:
When the team members are executing their trajectories to the assigned targets a sudden threat is detected (shown as the shaded area in Fig. 3). The threat is detected when the vehicles are on the positions shown in Fig. 3 assuming the team has a rendezvous arrival time.

The trajectory planning algorithm re-plans a new group of the minimum cost feasible trajectories and computes their costs. The costs are listed in Table 3. The GA is used on line to reallocate the targets as shown in Table 4.
Table 2: The Fitness Functions For Each Generation

<table>
<thead>
<tr>
<th></th>
<th>Generation 1</th>
<th>Generation 2</th>
<th>Generation 3</th>
<th>Generation 4</th>
<th>Generation 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P_i F_i(Σf_i / n)</td>
<td>P_i F_i(Σf_i / n)</td>
<td>P_i F_i(Σf_i / n)</td>
<td>P_i F_i(Σf_i / n)</td>
<td>P_i F_i(Σf_i / n)</td>
</tr>
<tr>
<td>1</td>
<td>B 1.44</td>
<td>B 1.34</td>
<td>B 1.43</td>
<td>B 1.34</td>
<td>B 1.29</td>
</tr>
<tr>
<td>2</td>
<td>C 1.137</td>
<td>C 1.06</td>
<td>C 1.132</td>
<td>C 1.06</td>
<td>C 1.03</td>
</tr>
<tr>
<td>3</td>
<td>D 0.939</td>
<td>D 0.877</td>
<td>D 0.917</td>
<td>D 0.877</td>
<td>D 0.98</td>
</tr>
<tr>
<td>4</td>
<td>E 0.799</td>
<td>E 0.822</td>
<td>E 0.73</td>
<td>E 0.989</td>
<td>E 0.895</td>
</tr>
<tr>
<td>5</td>
<td>F 0.756</td>
<td>F 0.822</td>
<td>F 0.73</td>
<td>F 0.822</td>
<td>F 0.986</td>
</tr>
<tr>
<td>6</td>
<td>G 0.73</td>
<td>G 0.73</td>
<td>G 0.73</td>
<td>G 0.73</td>
<td>G 0.73</td>
</tr>
<tr>
<td>Σfitness</td>
<td>5.415</td>
<td>5.59</td>
<td>5.43</td>
<td>5.59</td>
<td>5.661</td>
</tr>
<tr>
<td>Σcost</td>
<td>8905</td>
<td>8092</td>
<td>8832</td>
<td>8092</td>
<td>7753</td>
</tr>
<tr>
<td>Computation Time*</td>
<td>0.49 second</td>
<td>0.49 second</td>
<td>0.49 second</td>
<td>0.49 second</td>
<td>0.49 second</td>
</tr>
</tbody>
</table>

Fig. 2 shows the new trajectories after the new allocation, the population before the popup threat arrived (generation 5) is used as an initial population. The algorithm reached a near optimal solution (average \( RR = \frac{5.976}{6} \) = 0.996, in 0.36 seconds.

Fig. 2 Targets Allocations For 6 Vehicles

Table 3: The Costs Of New Trajectories After a Pop-up Threat Is Detected

<table>
<thead>
<tr>
<th>T</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>583</td>
<td>468</td>
<td>562</td>
<td>1051</td>
<td>986</td>
<td>1067</td>
</tr>
<tr>
<td>2</td>
<td>583</td>
<td>491</td>
<td>594</td>
<td>682</td>
<td>947</td>
<td>803</td>
</tr>
<tr>
<td>3</td>
<td>747</td>
<td>671</td>
<td>776</td>
<td>712</td>
<td>732</td>
<td>514</td>
</tr>
<tr>
<td>4</td>
<td>699</td>
<td>596</td>
<td>700</td>
<td>1603</td>
<td>808</td>
<td>581</td>
</tr>
<tr>
<td>5</td>
<td>662</td>
<td>588</td>
<td>684</td>
<td>844</td>
<td>893</td>
<td>767</td>
</tr>
<tr>
<td>6</td>
<td>920</td>
<td>1130</td>
<td>719</td>
<td>977</td>
<td>637</td>
<td>520</td>
</tr>
</tbody>
</table>

Table 4: Reallocation Of Targets

<table>
<thead>
<tr>
<th>T</th>
<th>P</th>
<th>( F_i(\Sigma f_i / n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B</td>
<td>1.30</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>0.90</td>
</tr>
<tr>
<td>3</td>
<td>I</td>
<td>1.00</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>0.88</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>0.93</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>0.966</td>
</tr>
<tr>
<td>Σf_i</td>
<td>5.976</td>
<td></td>
</tr>
<tr>
<td>Σcost</td>
<td>1802</td>
<td></td>
</tr>
<tr>
<td>Computation Time</td>
<td>0.36 second</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3 The New Trajectories For Newly Assigned Targets

* On Intel 3 GHz, 256 MB Memory, 300 MHz
Major advantage of GA algorithm is that it searches for optimization (as stated in Eqn. 6) in a parallel manner. Consequently, it produces the near optima solution in a few seconds. This is a vital element for the vehicle autonomy i.e. to plan its trajectories to its assigned targets in real time. It is worth noting that, the trajectory of the vehicle D to its pre-assigned target (target: 4) is no longer feasible because of the new threat (Fig. 3). Hence, its new trajectory is forced to turn around the new threat along with two additional SAM sites namely number 2 and 3. The cost of that trajectory is increased as explained in table 3. From table 4 and Fig. 3 it is clear that the UAVs C and D switched their targets to optimize the total cost of the fleet. The trajectories and allocations of the other vehicles A, B, E, and F are the same as before the replanning. The average fitness of the group is 936.

The main results of the algorithm are:
1) the search for optimal allocation is inherently parallel and very fast (the computation time is stated in table 2 and table 4).
2) the resultant allocation can be optimal or near optimal (average RRI =1)
3) the algorithm can fall on local minima so the algorithm has to observe if the solution failed in a local minimum or not. However escaping from it is easy and can be done by exchanging two vehicle symbols positions.
4) the result for different initial population can be different, especially if the number of UAVs is limited.
5) GA algorithm enables the autonomy of the UAVs

Conclusions:
This paper introduces a genetic algorithm for assignment of autonomous multi unmanned air vehicles. The assignments are produced in both static and dynamic environments. The application This a crucial issue for vehicles safety, fuel consumption and flying time.

References
shows a lot of advantages of the algorithm in static and dynamic targets allocation such as:
1) The produced solution is optimal or near optimal i.e. minimizing overall the fleet trajectories cost.


