Maximizing UAV Target Coverage under Flight Range and Target Service Time Constraints

E. Sevinç and M. Karakaya

***Abstract*—Using Unmanned Aerial Vehicles (UAVs) for reconnaissance purposes requires considering many different criteria such as limited UAV flight range, specified target service time, etc. Furthermore, it is desired that UAV should service more targets as many as possible. Thus, route planning is required to be optimal to cover maximum number of the targets while respecting all the given constraints. This article proposes a genetic algorithm (GA) to creating an optimized route for visiting maximum number of targets under the flight range and target service time constraints. In order to evaluate the success of the proposed GA method, we also developed an alternative approach, based on the Nearest Neighbor (NN) heuristic. To compare the success of these two methods we executed extensive simulation tests. The results indicate the success of the proposed GA method by increasing the number of covered targets compared to the solution based on the NN heuristic.**

***Index Terms*—Unmanned aerial vehicles (UAVs), routing, target coverage, genetic algorithm, optimization.**

I. Introduction

The popularity of employing Unmanned Aerial Vehicles (UAVs) in military and civil business is growing [1], [2]. One of the important challenges stemmed from the requirement of efficient usage of UAVs is scheduling the flight routes such that UAV can observe all or the maximum number of the specified targets. This problem is related with the Travelling Salesman Problem (TSP) [3] and the Vehicle Routing Problem (VRP) [4]. There are several other versions of these base problems including different constraints (service time, number of depots, vehicle range, vehicle capacity, etc.). Service time constraint defines when UAV can visit a target by setting the earliest (ready) and latest (due) visit times. Mobile element (in our case UAV) must be over the targets during the specified service time. Vehicle range can be the duration or the total length of a complete tour that the Mobile element (in our case UAV) can travel (fly) at most.

In the context of well-studied TSP and VRP problems, a typical requirement is that mobile element (travelling salesman or vehicle) should visit all the targets with an optimized (a minimum-distant) route. Moreover, in general, it is mostly assumed that there exists enough number of mobile elements (travelling salesmen or vehicles) to cover all the given locations. Due to the UAV flight range and the target service time, in real-life scenarios, there might not be feasible route to cover all the targets, especially if the number of UAVs is limited. Consequently, increasing the number of targets covered by the UAVs under the given constraints characterizes a new problem. For the problem considering only the flight range constraint, we have proposed an optimization method based on the Ant Colony Optimization [5]. However, adding the target service time constraint increases the complexity of the problem. Thus, this article describes a novel solution for this realistic optimization problem by adapting the Genetic Algorithm (GA) accordingly.

In the context of proposed GA method, each possible route is defined as an individual for the given UAV respecting the constraints. To create new generation, we apply crossover and mutation operations on the initial population. After each generation, the solution which covers more targets with less route distance is selected as the generation-best solution. According to the termination condition, the algorithm stops and outputs the best route found so far as the result. To evaluate the success of the proposed method, another approach, based on the Nearest Neighbor (NN) heuristic, is designed as well. In this solution, an UAV selects the nearest target to move on until its remaining flight range urges the UAV to return the base.

We implemented both solutions using Java and compared them in numerous experiment tests with different parameter settings and various VRPTW benchmark problem data files [6]. The success of the proposed GA method is observed in the results by increasing the number of covered targets up to 25% compared to the solution based on the NN heuristic.

II. Problem Definition

It is assumed that the coordinates of all targets, the service time of each target, and flight range of the UAV are given. The service time is defined as the time window in which the UAV must visit a target. That is, the UAV must service to the target after the ready time and before the due time. Furthermore, we assume that the UAV takes off and lands on the same base which is stationary. Additionally, the UAV flight speed is fixed. The problem is to generate route for the given UAV such that any target is visited in the requested service time, the UAV's total route distance has to be equal or less than the specified flight range, and the number of total targets planned to be visited by the all UAVs is maximized. Thus the target function is to maximize the number of targets to be visited by the UAV respecting the given service time and flight range constraints. In this work we call this problem as Maximum Target Coverage Problem (MTCP).

Manuscript received July 15, 2014; revised September 18, 2014. This work was supported in part by the Atilim University.

Ender Sevinçis with the Department of Computer Engineering of Middle East Technical University, Ankara, Turkey (e-mail: ender@ceng.metu.edu.tr).

Murat Karakaya is with the Department of Computer Engineering of Atilim University, Ankara, Turkey (e-mail: murat.karakaya@atilim.edu.tr).

In order to monitor the targets within the given service times, the UAV may have to wait for some time before visiting the next target to respect the ready time. This time period is termed as lapse time. The UAV can spend the lapse time flying over an extended route to the target or it can return and land on the base. Returning base for passing the lapse time might help to save fuel and flight range. Therefore, any route planning algorithm should take care of how to handle the lapse time effectively.

III. Genetic Algorithm

Genetic Algorithms (GAs) are defined as search procedures based on the mechanics of natural selection and genetics – at least in some qualitative sort of way [7]. Thus, GA is an Evolutionary Algorithm (EA) which is an iterative and stochastic method that works on a set of individuals (population) [8]. The objective of Genetic Algorithm is to develop method and theory to allow the design of GAs that solves hard problems quickly, reliably and accurately [7]-[9].

In order to apply a genetic algorithm to a problem first every potential solution of the given problem is represented by an individual. That is, the solution is encoded as an artificial chromosome or chromosomes. The basic idea is to maintain a population of chromosomes, which represent candidate solutions to the target problem that evolve over time through a process of mating to merge two solution chromosomes to produce a new solution. This solution is attained via an encoding/decoding system. The initial population (a set of potential solutions) is arbitrarily generated or by using a building heuristic. A fitness function is employed to measure the goodness of each individual in the population with regard to the problem at hand. Thus, the GA uses quantitative information for guiding the search process.

GA makes use of selection, crossover and mutation operators. Each chromosome in the population is calculated an associated fitness value to choose competitive chromosomes that will form the next generation. After applying crossover operator on the individuals, GA produces a new population. Mutation operation is employed to ensure that a better (possibly optimal) solution not existing in the chromosome pool can also be randomly generated. Thus, GA carries on by creating successive generations of better and better individuals by applying these simple operations. Thus, finding an optimal solution will be guaranteed if the GA algorithm is run for a very long time to create new generations [9].

IV. Adapting Genetic Algorithm to Maximum Target Coverage Problem

Below, we explain how the GA is modified to generate a solution to the maximum target cover problem. Since we improved GA Algorithm for the UAV routing problem having a time windows (VRPTW), we call it as *Improved GA (IGA)* and the algorithm will be used with the name of *IGA* throughout this work.

IGA is believed to achieve two goals at the same time. First one is to maximize the number of targets in the route and the second one is to minimize the route travelled. The important GA parameter values are given at Table I.

TABLE I: Parameter Values for Genetic Algorithm

Parameter	Value
Generation Number	200
Initial Pool Size	100
Mating Population	50
Crossover type	Truncate, 1-point
Crossover Ratio	0.6 (60%)
Mutation ratio	0.01 (1%)

A. Fitness Value

Fitness value of IGA is the number of targets visited by the given chromosome. If two or more chromosomes have the same target number, we choose the one with the smallest route length. The chromosome with the Highest Fitness Value (HFV) is recorded throughout the generations. After the pre-determined number of generations is processed, the algorithm terminates by outputting the solution (chromosome) with the HFV.

B. Validation

Validation is used to check if the given chromosome is valid according to the given constraints. First of all, targets should be listed in the chromosome only once. If there is any target duplication, we keep the first appearance of this target in the chromosome but remove all the others. Secondly, we check if the route length is less than or equal to the given flight range. If it is longer, then we split chromosome and remove the extra targets such that the new chromosome respects the flight range. Lastly, we check if the visit time of each target is valid according to the given service time. If not, we remove that target from the chromosome. Lastly, we update all the visit times of the targets in the validated chromosome accordingly.

C. Creating Initial Population

Initial population is created randomly selecting the targets by obeying the constraints. Then, each chromosome is validated as explained above and sorted due to fitness value. Our chromosome structure is shown in Fig. 1.

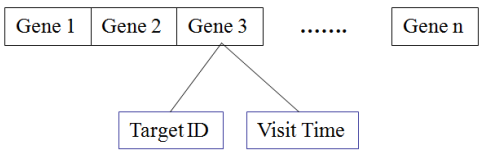

Fig. 1. The proposed chromosome structure.

Any gene has a visited Target ID and the visit time of that target. The visit time to the target also includes the delay times in order to control and obey the specified ready and due times.

D. Crossover

The most important point for the proposed Crossover operator is that we might possibly cross two different sized chromosomes. The main goal of Crossover operation which is described in Fig. 2 is to reach more nodes in a possible shorter path.

Assume that Parent 1 (P1) and Parent 2 (P2) are the two chromosomes (possible solutions). P1 contains 6 genes while

P2 has 7 genes as shown Fig. 2. Each gene has the same structure described in Fig. 1. For example, 1st gene of P1 is "T23" is standing for Target-23 and "1" is standing for Visit Time is minute "1" in time. We use maximum travelling time 3 hours which means 180 min. in the experiments. Because of that reason last gene of any chromosome must be smaller than 180 in this example.

Due to service time and visiting the targets only once constraints, we opt to implement 1-point crossover. In order to achieve the crossover operation, we first decide and find the cut point in the chromosome due to crossover ratio in P1. After deciding the cut point and specify the target at that gene, then, we check whether that target exists in P2 as well. If it is found, crossover is executed as described in Fig. 1. However, if not, we have to pick the adjacent nodes in P1 and check whether they exist in P2. If we cannot find a common target node in P1 and P2, then crossover cannot be executed for this pair of chromosomes and we select another pair of chromosomes.

After creating Offspring1 as shown in Fig. 2, it is re-evaluated again. Thus we look for a better solution and this solution might possibly have either more nodes in the path and/or a shorter path. This search is totally made due to the nature of GA. Here, IGA is believed to show its power, which means that while we are increasing the number of nodes to a possible extent; we are also trying to decrease the possible delays in the last 3-gene block of P2 at the same time. The resulting chromosome (Offspring1) *most probably* seems to have a shorter the path with decreased visiting times to the nodes come from P2.

It has also been mentioned that crossover operation in IGA has possibility to have offspring with more genes/targets than P1 or P2 as seen in Fig. 2. Besides crossover operation, looking for a path having more nodes is a lot more concern for mutation as well.

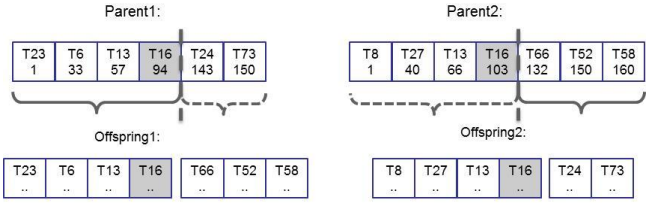

Fig. 2. The proposed crossover operator.

Offspring2 is created in the same way as Offspring1. We produce it by exchanging the rest of P1 and P2 with each other after creating the Offspring1.

E. Mutation

Mutation operator in IGA has two main functionalities mainly. One of them is to work similarly due to mutation ratio as in a common GA [7], [8]. Briefly, two genes are picked randomly in the chromosome, then exchanged and swapped to each other's place in order to look for whole search space.

The other functionality of the IGA's mutation operator is believed to be more important. Here we look for a new node and try to add one more node/gene to chromosome due to ready/due time limitations of the nodes. If possible, then the selected new node is put into a place among the other genes. This node is placed in the sequence where ready/due times of any node are *not* violated. Finally, if IGA decides to make a mutation, it will either make a swap of two genes or add a new node into the chromosome. The more genes with a shorter path are the better solution is. After any mutation operations, the new chromosome is validated.

V. Nearest Neighbor Heuristic to Maximum Target Coverage Problem

To observe the success and effectiveness of the proposed GA, we have executed various simulation tests. First of all, we have implemented an advisory solution based on *Nearest Neighborhood (NN)* heuristic.

In the NN method, route planning contains new target selection process which begins from the base time as the current location. The NN method calculates the UAV travel time from the current location to all non-visited targets. Then, the NN method eliminates the targets whose ready and due times do not fit to the UAV's arrival time to that node. As the final step, the NN method selects the nearest target to the current location, provided that the UAV can have enough remaining flight range to return to the base.

If any new node cannot be selected, then target selection process terminates and the route plan is outputted. The aim is to reach and service the nearest node within the appropriate ready/due times.

But during our experiments, it has been observed that the NN algorithm might produce inferior results. For example, if the nearest node selected has a rather late ready time than the current time, then you have to give delays inevitably and these wasted times in air degrades the performance badly especially when special data sets are considered. In order to get rid of this drawback, we improve the NN method and proposed *Smart NN (SNN)* Algorithm which tries to get rid of this drawback.

In *SNN*, rather than selecting the single nearest one, a set of nearest nodes (in our experiments 3 nodes) to the current one is selected. Then, the node causing the least delay is picked up within that set. The critical point is that selected node might not be the nearest, but very close to it.

In fact, this is a trade-off between time and distance and it is believed to be a good sample of greedy approach. We may not select the nearest node for the sake of having less delay in the air. We may select the second or third closest node at most, but it has been observed in our experiments that we are able to visit more nodes by giving less delay time which causes a precise performance increase.

VI. Simulation Tests and Results

In experimental results, IGA and SNN algorithms are compared due to different various VRPTW benchmark problem data files [6] with changing UAV ranges and parameters.

A. Simulation Environment and Parameters

All the test results given in the following tables and figures are obtained by taking the mean of the results of 20 independent runs. We have used both R and C data sets

described in [6] in order to reach more accurate results. In C data set, all of the nodes are specifically placed on the map which is an advantage for SNN algorithm. But it is better to show that our GA is almost better than SNN algorithm. In our tests, R101 thru R104 among R data sets are used, similarly C101 thru C104 are used among the C data sets. The parameters are shown in Table II. All of the (100%) nodes have been considered and contained in our tests.

For the UAV parameter values, we consider the MQ-1B Predator UAV [10]. The brief technical specs are 130 and 160 km/h are average speeds and we assume that the vehicle will fly for 3 and 6 hours respectively. The data sets, i.e. R and C data sets have been experimented due to two different speed/range occasions which explained below;

1) *1st (Low) speed/range*: 130 km/h × 3 h= 390 km.
2) *2nd (High) speed/range*: 165 km/h × 6 h= 990 km.

TABLE II: SIMULATION PARAMETER SETTINGS

Parameter	Default Value	Range	Notes
Data Set	R101	R101-R104, C101-C104	8 Different VRPTW benchmark problems
Number of Targets	101	101	1st node is selected as the *Base*, the rest 100 nodes are assumed to be targets.
UAV Range	390 km.	390 -990 km.	MQ-1B Predator UAV specs

B. Experiments and Results

In the first set of experiments, we observe the success of two methods on the R data sets given in the VRPWT benchmark problems [6]. The R data sets are good approximations for real life applications and first four data sets are used since they are good samples for the rest, if observed in detail. As seen in the Fig. 3 and Fig. 4, the IGA method produces better results than those of SNN for both speed/range alternatives.

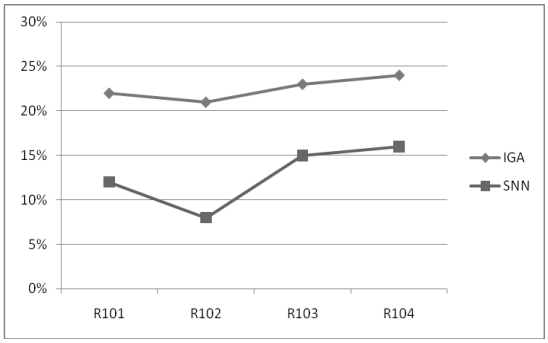

Fig. 3. Comparison of R data sets for 1st speed/range alternative.

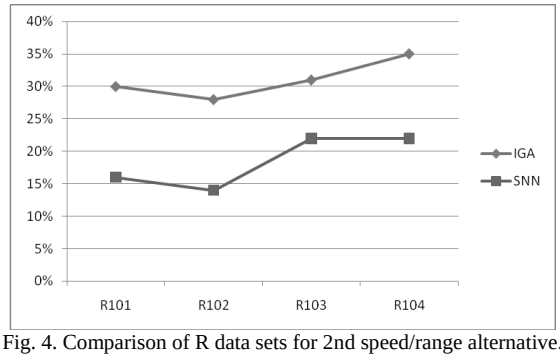

Fig. 4. Comparison of R data sets for 2nd speed/range alternative.

In the second set of experiments, we observe the results of two methods on the C data sets. The C data sets are different than the R data sets. Although there are 101 nodes, all the nodes are observed to be clustered in 5-10 nodes. As the SNN method tends to traverse first through all possible nodes in a cluster and then pass to next cluster, the clustered node topology seems to provide advantage for the SNN method. However, considering the IGA method, since initial population (solutions) is randomly produced at the beginning and these solutions might contain nodes form different clusters, the convergence of the IGA method might take a while. On the other hand, as these clusters are specifically designed for creating the C data sets, one can argue that this in fact might be contrary to real life cases.

Even though this characteristic of the C data sets, as Fig. 5 and Fig. 6 indicate, IGA produces better results when compared to SNN from 5% to 50%. IGA tries to maximize the node number while trying to minimize the path.

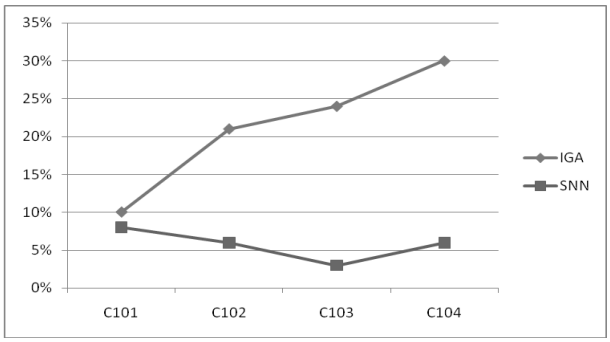

Fig. 5. Comparison of C data sets for1st speed/range alternative.

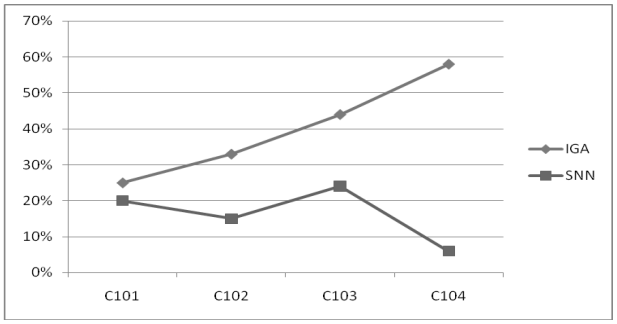

Fig. 6. Comparison of C data sets for 2nd speed/range alternative.

VII. CONCLUSION

The proposed IGA method is proved to be a useful routing algorithm in a complex environment with flight range and service time constraints. We observed via simulation tests that the IGA method is able to increase the number of visited targets successfully compared to the SNN method. Thus, we believe that the crossover and mutation operators are adapted and integrated to the proposed method successfully to produce a good solution to the MTCP.

We would like to develop the IGA method such that it can plan routes for the multiple UAVs as a future work. We also aim to improve the SNN method for small and clustered target layouts.

REFERENCES

[1] J. Everaerts, "The use of unmanned aerial vehicles (UAVs) for remote sensing and mapping," *The International Archives of the*

Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 37, pp. 1187-1192, 2008.
[2] D. Glade, *Unmanned Aerial Vehicles: Implications for Military Operations*, Air Univ Press Maxwell Afb Al., 2000.
[3] K. L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling salesman problem," *Encyclopedia of Operations Research and Management Science*, pp. 1573-1578, 2013.
[4] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345-358, 1992.
[5] M. Karakaya, "UAV route planning for maximum target coverage," *Computer Science & Engineering: An International Journal (CSEIJ)*, vol. 4, no. 1, February 2014.
[6] Msolomon. (2005). VRPTW Benchmark Problems. [Online]. Available: http://w.cba.neu.edu/~msolomon/problems.htm
[7] D. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithm*, Norwell, MA: Kluwer Academic Publishers, ch. 1, 2002.
[8] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Berlin Heidelberg: Springer-Verlag, 2008.
[9] E. Sevinç and A. Coşar, "An evolutionary genetic algorithm for optimization of distributed database," *The Computer Journal*, vol. 54, no. 5, 2011.
[10] MQ-1-Predator-MQ-9-Reaper. (January 2014). UAV Specifications. [Online]. Available: http://www.bga-aeroweb.com/Defense/MQ-1-Predator-MQ-9-Reaper.html#specs

Ender Sevinç was born in İstanbul, Turkey on Sept. 25, 1969. Sevinç received his M.S. and Ph.D. degrees in computer engineering from Middle East Technical University, Ankara/Turkey in 2002 and 2009 consecutively.

Since 1996, he has worked for different positions at different units in the Turkish Land Forces (KKK). In 2010, he worked as the head of the IT Operation Section and was responsible for control, security and effective operation of the IT systems. His team managed various networks, such as Internet, NATO, several LANs and WAN. From 2011 to 2013, he worked as the head of Software Maintenance Section. Mainly acting as the project manager, he managed the software projects due to accepted approaches and methodologies and mostly took place in analyze, design and testing phases. Having been promoted as the head of the branch since February 2013, he has been preparing and managing small/medium-scaled software projects for General Staff Headquarters.

Dr. Sevinç gave C programming lectures during his duty in Middle East Technical University. His major field of study is DDBs, optimization of queries, network and route design by using genetic algorithms for decision support systems.

Murat Karakaya received the B.S.E.E. degree in 1991 from the Turkish Military Academy (KHO), Ankara, Turkey, and the M.S. and Ph. D. degrees in computer engineering from the Bilkent University, Ankara, Turkey in 2000 and 2008, respectively. From 1992 to 2000, he worked as an engineer at different units in the Turkish Land Forces (KKK), Ankara, Turkey. From 2000 to 2005, he worked as an instructor and software engineer at the Turkish Military Academy (KHO), Ankara, Turkey. Then, during 2005-2008 he worked as an IT project manager in the North Atlantic Treaty Organization (NATO) Brussels, Belgium. From 2008 to 2012, he worked as an instructor and software engineer at the Turkish Military School of Electronics, Communications and Information Systems (MEBS) and Turkish Military Academy (KHO), Ankara, Turkey. He joined the Faculty of Atılım University in 2012 and he is currently an asst. professor in the department of Computer Engineering, Ankara, Turkey. His research interests are natural computing, sensor networks, peer-to-peer networks, route optimization, and communications protocol design.