MISSION PLANNING FOR UNMANNED AERIAL VEHICLE (UAV) TEAMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERDAR YILGIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2014

Approval of the thesis:

MISSION PLANNING FOR UNMANNED AERIAL VEHICLE (UAV) TEAMS

submitted by **SERDAR YILGIN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Prof. Dr. Faruk Polat
Supervisor, **Computer Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Ahmet Coşar
Computer Engineering Dept., METU _____

Prof. Dr. Faruk Polat
Computer Engineering Dept., METU _____

Prof. Dr. Veysi İşler
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Halit Oğuztüzün
Computer Engineering Dept., METU _____

Assist. Prof. Dr. Mehmet Tan
Computer Engineering Dept., TOBB ETU _____

Date: 04.09.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: SERDAR YILGIN

Signature :

ABSTRACT

MISSION PLANNING FOR UNMANNED AERIAL VEHICLE (UAV) TEAMS

Yılgın, Serdar

M.S., Department of Computer Engineering

Supervisor: Prof. Dr. Faruk Polat

September 2014, 71 pages

In recent years, use of Unmanned Aerial Vehicle (UAV) especially for reconnaissance and combat missions has become very popular in worldwide. There is no onboard human operator exists for UAVs and they are generally controlled by remote human operators. Depending on the operational environment; sometimes it becomes nearly impossible to provide optimal or an acceptable UAV – target assignment and scheduling, satisfying the constraints required to accomplish the mission, for the operators in control center. In this scheme, computer support became inevitable to be able to acquire more suitable scheduling and assignments for the missions, involving more than a few UAVs and targets, in shorter durations. In this thesis, we study different approaches for UAV mission planning problem and analyze their performances. We designed a genetic algorithm instance with customized encoding, crossover and fitness calculation that all these algorithm components are somehow related to problem domain. A brute – force and a greedy algorithm are also developed for this problem with the aim of comparison. As the result, by utilizing developed algorithms, it has become possible to evaluate effectivity and efficiency of the proposed genetic algorithm.

Keywords: Unmanned Aerial Vehicle, Genetic Algorithm, Mission Planning, UAV – Target Assignment and Scheduling, Operational Environment

ÖZ

İNSANSIZ HAVA ARACI (İHA) TAKIMLARI İÇİN GÖREV PLANLAMA

Yılgın, Serdar
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Faruk Polat

Eylül 2014, 71 sayfa

Son yıllarda, özellikle keşif ve çarpışma görevlerinde İnsansız Hava Aracı (İHA) kullanımı dünya çapında fazlaca tercih edilir bir hale gelmiştir. İHA'lar için hava aracı üzerinde bir insan operatör bulunmaz ve bu araçlar genellikle uzak kontrol merkezlerindeki insan operatörler tarafından kontrol edilirler. Operasyonel ortam şartları doğrultusunda; kontrol merkezindeki operatörlerin, görevin başarıyla sonuçlanmasını sağlayacak gereksinimleri en uygun veya kabul edilebilir uygunlukta yerine getiren bir İHA – hedef eşleşmesi ve zaman planlaması yapabilmeleri bazen neredeyse imkânsız bir hal alır. Bu durumda özellikle birkaç İHA ve birkaç hedeften fazlasını içeren görevlerde, daha kısa bir zaman dilimi içerisinde daha uygun zaman planlamaları ve atamalar sağlayabilmek için bilgisayar desteği almak kaçınılmaz olmuştur. Bu tezde, İHA görev planlaması için farklı yaklaşımlar üzerinde çalıştık ve bu yaklaşımların başarılarını analiz ettik. Özelleştirilmiş çaprazlama, kodlama ve değerlendirme yöntemlerini içeren ve bileşenlerinin alan bilgisi doğrultusunda belirlendiği bir genetik algoritma örneği tasarladık. Bu problem için karşılaştırma amaçlı, kaba kuvvet ve açgözlülük yöntemleri üzerine kurulmuş 2 algoritma daha geliştirildi. Sonuç olarak, geliştirilen bu algoritmalar kullanılarak, önerilen genetik algoritmanın yürürlük ve etkinliğini değerlendirebilmemiz mümkün oldu.

Anahtar Kelimeler: İnsansız Hava Aracı, Genetik Algoritmalar, Görev Planlama, İHA – Hedef Ataması ve Zaman Planlaması, Operasyonel Ortam

To Computer Science

ACKNOWLEDGMENTS

I would like to present my deepest thanks to my thesis supervisor Prof. Dr. Faruk Polat for his valuable guidance, motivation and support throughout this thesis study.

I am very grateful to my family and colleagues in Havelsan A.Ş. for all their patience and tolerance.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are somehow autonomous aircrafts and do not require onboard human controller (pilot) for management. The idea of using UAVs for various military missions receives growing attention day to day, especially in last two decades. Lack of dependency for an onboard controller provides some other desirable features such as having less weight compared to traditional airplanes and accordingly need for lesser fuel amount for flying same distances. Also, some equipment used for safety and surveillance of human controller are not necessary for UAVs. So, for some type of missions, it is possible to lower cost of operation by using UAVs instead of traditional airplanes. As the result, having cost advantage and no need for human intervention encourage the use of UAVs for realization of D-cube (Dull, Dirty, Dangerous) missions [19].

UAV mission planning is a kind of resource allocation problem requiring assignment of scarce resources to relatively big amount of targets by taking time constraints of targets into consideration. Workload variety of targets is also an important issue that number of UAVs needed by any target is directly proportional to size of the area covered by that target and inversely proportional to time interval that the target required to be served for. In this thesis, additional to these mentioned requirements, we also consider fuel consumption of UAVs. Accordingly, fuel levels of UAVs need to be checked to be able make them to return to base location safely. Regarding to this described scenario, it is possible to classify UAV mission planning problem as a combinatorial optimization problem, which requires finding the best or an acceptable solution, with respect to predefined requirements, from a big solution set. So, this

scheduling and planning task is NP – Hard (**N**on – deterministic **P**olynomial – time hard) [21].

It is not always possible for UAV operators to reach acceptable level of optimality in assignment and scheduling, especially for the operational environments involving many UAVs and targets. For some scenarios in which available UAVs are scarce and just a few more than the number required for accomplishing the mission with an optimal assignment and scheduling, probability of failure for the mission is very high without computer support. Even in case of success for those typical scenarios, cost of the operation will probably become more than the cost resulted from computer supported one. Requirement for such an algorithm meeting the requirements and providing optimal or more suitable solutions in shorter durations constitutes the motivation of this study.

In this thesis, we focus on genetic algorithm formulations for UAV mission planning problem. Problem specific encoding, crossover and fitness calculation schemes are proposed for genetic algorithm solution of this assignment and scheduling problem through this thesis. Customized genetic algorithm approaches are implemented and examined through experiments.

Greedy and exhaustive search algorithms are also implemented for the UAV mission planning problem. Since exhaustive search algorithm requires long time to terminate, it is not implemented in a standard manner but including a preprocessing procedure to be able to provide early elimination for the cases showing unfeasibility or lesser suitability obviously. Solutions of greedy and exhaustive algorithms are used for evaluating implemented genetic algorithm customizations.

We also implemented a framework by utilizing Qt and OpenGL technologies with C++ programming language, to be able to create, configure and visualize problem instances. This framework also enables users to export produced scenarios to text files in xml format and they can also import saved scenarios into framework environment by loading already saved text files. Running any of the algorithms for the

current scenario is possible with this framework and evaluation of them becomes very easy. A 2D simulation of any solution instance, resulted from the implemented algorithms, is also possible in this framework to be able ease feasibility detection especially for the scenarios involving interaction of several UAVs and targets in operational environment.

The remainder of the thesis is organized as follows:

Chapter 2 – Background and Related Work provides base information about Unmanned Aerial Vehicles and planning of their cooperative mission in a previously analyzed operational environment. Then, related studies in literature are introduced and analyzed. A detailed comparison, illustrating pros and cons of the approaches, is also provided. Lastly, algorithms for target assignment and scheduling are mentioned briefly.

Chapter 3 – Our Work presents our study in details. Components of the genetic algorithm, designed in the context of this research, and their customizations are described. Brute – force and greedy algorithms, that are also designed to work on the same scenarios on which the genetic algorithm works, are also introduced. Details of the framework, which is implemented to be able to compose various mission scenarios and run the algorithms for these scenarios, are illustrated in this chapter.

Chapter 4 – Evaluation of the Results provides an outline of the test results gathered by running the genetic, brute – force and greedy algorithms on specified scenario sets. Analysis and evaluation of acquired test results are also provided in this chapter.

Chapter 5 – Conclusion and Future Work includes gained acquisition remarks and concluding points. Statements about the future work are also provided in this chapter.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter aims to present an overview of the UAV mission planning problem with all its subcomponents. It also includes current studies in literature and the approaches proposed in these studies. Main algorithms and formulations used to solve assignment and scheduling problems are described briefly in this chapter.

2.1 Problem Formulation

Key components of the UAV mission planning problem are described briefly as follows:

- **Base:** It is the initial location of all UAVs. The UAVs are going to depart that location and also they all are expected to return there after completion of their missions.

- **Fleet:** It is a group of UAV constituting a taskable unit. Size of a fleet varies according to the aim of the UAVs. It is also possible to let several fleets to operate cooperatively for the same mission.

- **Target:** It is the entity that requires some predefined number of UAVs for the mission to be accomplished. In general, there is less number of UAVs than the number required to be able to fulfill all targets' needs. That's why, an efficient assignment and scheduling scheme is needed.

- **Operational Environment:** It stands as the area of the interest together with logical and physical factors playing some roles in this area during the operation. There are also some virtual factors of operational environment which are also important for operations. These factors can be listed as follows:

    a. **Radar Zones and Prohibited Areas:** They are specified fields which are already known to be somehow in the sensor coverage of enemies' defense forces.

    b. **Safe Graph:** It is the graph connecting all targets (nodes) in the operational area to base point of the fleet by taking already known radar zones and prohibited areas into consideration and guaranteeing no intersection with them. This graph also contains distance values between each neighbor target pair available in the operational area.

- **The Goal:** It is the aim of providing availability of desired number of UAVs in desired time interval for each target exists in operational environment.

2.2 Related Work

UAVs do not require an onboard human controller but they need to follow some sort of control logic, and degree of autonomy varies according to provided scheme of that logic. Basically, there are two main approaches in controlling UAVs: First one is centralized approach and the second one is decentralized (distributed) approach. A hybrid application of centralized and decentralized approaches is also possible. In centralized approach, all UAVs involved in the mission are controlled by a central algorithm and mission planning scheme is based on this central logic [1, 2, 6, 8, 11, 12, 13]. Conversely, decentralized approach offers more and sometimes full autonomy. Each UAV is generally controlled by one operator but sometimes it can be necessary to assign more than one operator for each UAV. Because of the situation, it is difficult to afford cooperative missions involving UAV swarms. So, the need for

decentralized approaches, aiming to use fully autonomous agents, aroused [3, 4, 5, 7, 9, 10, 16]. In this logic, each UAV is supposed to have same situational awareness to decide about its next step; and a strong communication among the UAVs is also an important requirement to be able to accomplish cooperative missions. In hybrid (centralized – distributed) approach, not fully but partially autonomous UAV usage is proposed. The idea is to provide some level of abstraction to the central controller; that it is rather preferable to be able to control a UAV team instead of a single UAV for an UAV operator [14].

Mission completion of each observed target requires classification, attack and verification tasks to be accomplished for that target in specified order. This ordering comes from the nature of the mission; and so, the order has to be preserved. As stated, task precedence is also an important issue that needs to be considered in UAV mission planning [4, 13, 15, 16].

UAV mission planning algorithms should take the constraints coming from operational environment and nature of UAVs into consideration and behave accordingly. The algorithms are supposed to handle two core components of the planning procedure, which are path planning and target assignment. Especially the ones, focusing on path planning issue, consider about UAV movement capabilities (e.g. minimum turning radius, maximum speed to be reached) and no – fly zones located in operational environment. Accordingly, the algorithms are designed to handle the problem by also caring about these constraints [7, 11, 12]. The other ones, focusing on target assignment issue, also consider about some constraints; but for instance, coming from designed strategy (e.g. target dependency and target priority) and nature of the problem (e.g. task precedence). So, these kinds of algorithms are designed to include priority, dependency and precedence requirements into problem definition set and behave accordingly [13, 15]. Also there are some full mission planning algorithms aiming to deal with path planning and target assignment issues together [1, 4, 5, 9, 16].

There are also some other constraints, such as heterogeneity of UAVs, fuel capacity of UAVs and heterogeneity of targets forming operational environments, that should be considered by mission planning algorithms. Fuel capacity constraint is considered in many mission planning algorithms both with and without regarding type of the UAVs [4, 7, 9, 13, 14, 16]. Heterogeneity is also an important concept for both of the UAVs and the targets. It is because some features, such as fuel capacity, operational capability and average speed, pose differences according to the type of an UAV. In addition, UAVs are generally specialized and equipped according to the mission type, the UAV is intended to realize, that task assignment is required to be done according to the type of the UAVs [4, 13, 15]. In a similar way, each target does not need same amount of resource to be handled. This constraint is included in [5] to be able to provide higher degrees of reality by designing the assignment and scheduling scheme with taking specific target resource need into account. Heterogeneity of targets has also introduced another important constraint, known as time – window in the literature, and involving for each target to be served in previously specified time interval with an affordable delay [5, 11, 26, 27].

Most of the proposed mission planning algorithms assume the scenarios, going to be realized, to be based on a certain and static environment. In parallel to this intuition, the algorithms are run once and the scenario continues in an offline fashion according to the command set decided in initialization phase [5, 8, 11, 13]. But in real life, changes for operational environments are frequent. So, to be able to provide modeling of the scenario as close as possible to real life, there are also some dynamic algorithms, generally proposed in relatively near history, aiming to provide acceptable and quick responses to the changes, which are happening in the current scenario and requiring immediate update of situational awareness [1, 4, 6, 7, 9, 14, 16].

It is possible to grasp feasible and acceptable solutions for UAV mission planning problem both using exact and heuristic – based approaches. The choice of which approach to use is basically related to problem domain and the specified requirements to be provided. In the case that number of targets is more than a few, exact approaches generally terminate after a substantial amount of time, which is not

between limits of acceptation in general; but the appealing result of these kinds of approaches is to reach the optimal solution. On the other hand, deterministic and / or stochastic heuristic – based algorithms generally terminate in an acceptable time amount and result with the optimal or nearly optimal feasible solutions for the problem. As it can be inferred, for heuristic – based algorithms it is not guaranteed to reach the optimal solution.

Proposed exact algorithms work in a straightforward manner and aim to pick up the best solution, maximizing (multi)objective function, from the pool of all possible solutions generally enumerated in offline time. To be able to incorporate the constraints into the problem environment, this kind of algorithms generally use problem – based specified tree structures with the leaf nodes containing feasible solutions for the processed scenario. Another advantage of the tree structure is the feature of lending itself for efficient and quick search techniques. In spite of structural advantages of this so – called decision trees, necessity of enumerating all feasible solutions generally requires unacceptable CPU time for most of the mission planning scenarios including many agents. In [8] a new exact algorithm not using decision tree but some kind of linear formulations with polynomial number of binary variables is introduced. By using a new set of properties and inequalities, such as symmetric breaking inequalities, boundaries on profits, generalized subtour eliminations and clique cuts from graphs of incompatibilities, the algorithm achieves to be competitive with the best performing literature algorithms in both of effectivity and efficiency aspects. On the other hand, the algorithms introduced in [4] and [15] utilize decision tree structure like many of the exact mission planning algorithms do. By using advantages of this specified tree structure, enumerating only feasible solutions not all possible permutations of UAV and target sets, and utilizing best – first and depth – first search procedures on final decision tree; these algorithms also have potential of planning mission scenarios including higher number of instances in acceptable time limits.

There are so many heuristic – based algorithms proposed for UAV mission planning problem compared to the exact ones. Simulated Annealing (SA), Genetic Algorithms (GA), Tabu Search (TS) and Neighborhood Search (NS) heuristics and somehow

varied versions of them are mostly used heuristics in UAV mission planning problem for both or individually handling of path planning and target assignment issues [5, 7, 12, 13, 14]. In some heuristic – based algorithms, constraints and requirements are represented by inequalities where all these inequalities together with the objective function constitute a Mixed Integer Linear Programming (MILP) instance, which is known as NP – hard. After construction of this MILP instance, according to the operational environment and the specifications of the problem, the algorithms start to enumerate feasible solution sets by changing order and match of scenario elements according to the rules implied by predefined heuristic(s) aimed to be utilized in algorithm design phase. Typically, heuristic – based algorithms either incorporate a general mechanism for restoring feasibility of the system after each move or use the problem structure to guide the search effectively by eliminating constraints [10]. The most important issue for this kind of algorithms is avoidance of sticking on local optimality causing the algorithm to terminate without covering entire solution space**.** Main advantage of heuristic – based approaches is that they are generally capable of resulting with a good feasible solution in an acceptable time period, which enables the algorithms to be up to date with respect to changing operational environment situations and behave accordingly.

In parallel to that growing demand for the use of UAVs, essentiality of robust algorithms to solve the target assignment problem for a UAV team in an effective way becomes inevitable. It is because; the real environment and the equipments used to raise the level of autonomy are quite noisy. However, uncertainty of the operational time environment should also be taken into consideration and normalized to be able to need minimum level of control, required to manage the UAVs, for providing completion of a mission with an acceptable rate of success. Based on this reality, in [2, 3, 6, 7] uncertain and noisy conditions are tried to be handled by introducing alternative strategies and delta values enabling compensation of the aimed strategy by keeping the key attributes of the strategy in a predefined range. Specifically in [2] , some delta values are injected to the formulations used in realization of mission planning algorithm. Also, by enabling span of these delta values according to predefined factor, the algorithm allows the controller to tune sensitivity according to uncertainty level

of operational environment. As another approach, [3] offers use of decentralized autonomous agents with the same central algorithm deployed to be able to need minimum level of communication while realizing cooperative missions. It also introduces availability of a backup (second) algorithm for all UAVs, enabling them to operate in a sensible manner in case of loss of communication. In [7], replanning of the mission is considered when a change is detected in situational awareness; and it is required to be handled in current scenario. Lastly, [6] serves as a combination of [2] and [7] with the aim of reducing imperfectness of individual algorithms by normalizing negative effects of the situations with utilization of the relatively powerful strategies.

2.3 Resource – Target Assignment and Scheduling Methodologies

Main algorithms and formulations used for resource – target assignment and scheduling problem are described briefly in this part. The algorithms with possible modifications are listed as follows:

2.3.1 Exhaustive (Brute – Force) Search Algorithm

This kind of algorithms work by enumerating all possible solutions and checking which of them yielding the best rank. Because of the need for covering all workspace, time consumption of the algorithms is generally too high to accept for most of the problems especially for the ones required to be solved in online fashion. So, some kind of elimination mechanisms and heuristics are introduced to be able to lower time consumption of exhaustive search algorithms. Main variations of exhaustive search algorithms are:

- **Branch and Bound:** Main idea is to branch the problem space to adequate number of levels and provide upper and lower bounds for every sub branches in each level. Deciding about the number of levels to branch the problem structure is an important issue for this kind of algorithms. Because, over

depth may require extra time to terminate searching while less depth may cause the solutions to diverge from optimality. After an acceptable branching is provided, the idea is to decide to continue to process or stop processing for a solution candidate, by forecasting its potential to perform better or worse than the current best solution, using lower and upper bounds.

- **Greedy Bound:** Unaffordable operations are detected greedily and they are not performed for solution candidates. In this scheme, the candidates are processed with the expectation of reaching better result quality at the end.

2.3.2 Greedy Algorithm

These kinds of algorithms work by utilizing locally optimum solutions to be able to reach globally optimum solution. The algorithms try to increase total profit for each step by proceeding with possibly maximum profit of the scenario for that step. So, greedy algorithms generally suffer from being stuck in local optimality. Accordingly, it is not possible to grasp best solution using greedy algorithms in general but it is common to reach an acceptable solution in relatively short time compared to traditional exhaustive search algorithms. It is also possible for greedy algorithms to use some heuristics and bounding strategies, like the ones mentioned in exhaustive search algorithms, to be able to boost effectivity and shorten execution time.

2.3.3 Integer Linear Programming

Linear programming is a method of solving problems by using the mathematical model which is inferred from problem domain. All of the constraints are represented as mathematical equations in this model. Specifically, integer linear programming is a linear programming instance in which all or some of the problem domain variables are restricted to be integers. The method is generally used to solve optimization problems.

A standard linear programming instance consists of a linear objective function containing the values to be maximized, a set of inequalities keeping consistency of problem domain constraints for candidate solutions and non – negative decision variables, which both linear objective function and constraints are based on.

Three possible outcomes are possible for a standard linear programming instance. It may be impossible to find proper values for decision variables satisfying all of the constraints. Inversely, as another result it may be possible to find optimal values for decision variables satisfying all of the constraints. As the last result, the instance may be unbounded that given constraints are not enough for bounding decision variables' value range in solution space of the problem.

2.3.4 Genetic Algorithm

The elements of a standard genetic algorithm are listed and described briefly as follows:

- **Chromosome:** It is the data structure in which an individual solution is encoded.

- **Population:** It is the set of solutions (chromosomes) which is also serving as the pool of chromosomes to be selected for crossover procedure in genetic algorithm context.

- **Fitness:** It is the rank value associated to each chromosome. This value indicates in which degree the chromosome fits to the predefined criterions.

- **Selection:** It is the process of selecting (mating) two parent chromosomes for crossover operation.

- **Crossover:** It is the process of generating a new chromosome by applying some kind of split – and – merge procedure for genetic information of two selected parent chromosomes.

- **Mutation:** It is the operation of replacing some randomly chosen bits of a chromosome with the ones already existing in problem specification.

The following procedures have substantial importance in the design of a genetic algorithm; and so, they should be organized by taking problem specific details into consideration:

- **Chromosome Encoding:** Main aim is to encode any possible solution of the problem into a data structure called chromosome. It is one of the most important procedures for a genetic algorithm. Because chromosome structure is the core component of a genetic algorithm that the following procedures are all subject to variance according to determined chromosome structure. Widely used encoding schemes are binary, real number, permutation and data structure encodings. However, use of binary and real number encodings are more frequent.

- **Population Initialization:** It is another important procedure in genetic algorithm aiming to provide initial population for the algorithm. All new chromosomes are generated by realizing crossover and mutation operators on current population members. So, it is obvious that fitness of a child directly related with fitness of its parents. In this direction, it is common to run another algorithm, terminating fast like greedy ones, to provide an initial population having higher level of fertility.

- **Fitness Calculation:** It is the procedure of evaluation for each solution by measuring how much rank the solution provides regarding to the predefined objectives of the mission. The issue here is to find the objective set aimed to optimize in final solution. If there is only one objective it would be easier and

quicker to reach an optimal or acceptable solution. But in the case of availability of multiobjective scheme, it wouldn't be easier and quicker to provide the acceptable solution; even in some problems, depending on the objectives' nature, it wouldn't be possible. This is because in a multiobjective design, objectives are often contradictory to each other and to be able to find a solution satisfying all the objectives is really a difficult procedure.

- **Selection Operator Determination:** In literature, mostly used selection techniques are given below:

1. **Roulette Wheel Selection:** In this selection method, all chromosome fitness values are summed up and total fitness value (TF) is obtained. Then, a random number (R) in range [1 - TF] is generated. After that, while going through all the population in a loop, fitness values of all chromosomes are summed up as partial fitness value (PF) and current chromosome is selected as a parent when PF is greater than R. In this scheme, the scenario depicted in Figure 2.1 has Chromosome5 to be selected as one parent with about 61% probability.

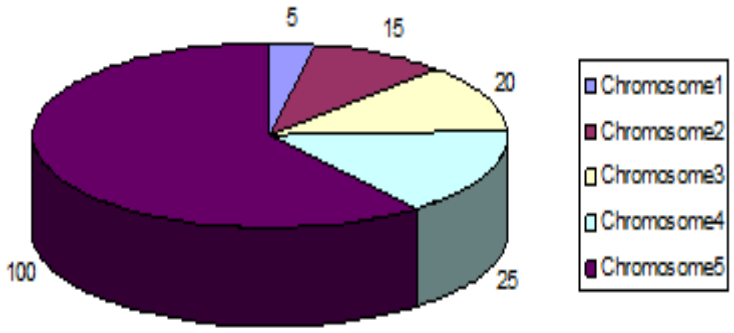

Figure 2.1: Roulette Wheel Selection Probability Distribution

2. **Rank Selection:** It is somehow a normalization of Roulette Wheel Selection technique. At first, according to fitness values they have, some kind of ranking procedure is applied to all chromosomes in the population without violating their respective fitness position. Then, same selection

procedure, utilized for Roulette Wheel Selection method, is utilized for this method but this time not based on fitness values but based on rank values of the chromosomes. This method is proposed with the aim of providing variety by hindering excessive dominance of a few individuals to the rest in current population. Compared to Roulette Wheel Selection method, with the same scenario illustrated above, for this scheme Chromosome5 has about 33% selection probability as shown in Figure 2.2.

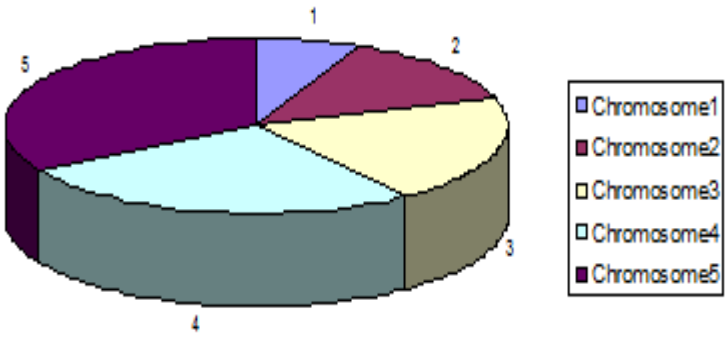

Figure 2.2: Rank Selection Probability Distribution

3. **Tournament Selection:** This selection type involves realizing some tournaments on some randomly chosen chromosomes. After all tournaments held, the one achieving the highest score is selected. Similar to the relation between Roulette Wheel Selection and Rank Selection techniques, increasing number of tournaments decreases variety by allowing dominant chromosomes to reach stability for this method.

4. **Steady State Selection:** Main aim of this selection method is to transfer big portion of the population to the next generation by only allowing a few least fit chromosomes of the population to be replaced by new off springs. However, this method only allows a few best fit chromosomes to be selected as parents for crossover procedure.

5. **Elitism:** In this method some number of best fit chromosomes is directly copied to next generation to be able to provide availability of fertile members for next generations.

- **Crossover Scheme Determination:** Crossover scheme design is very important for a genetic algorithm; that the scheme can influence performance of the algorithm in either negative or positive directions. In general, there exist two schemes mostly used in literature which are single and multi point crossover schemes. Although the schemes can be grouped in two main titles, they generally vary based on the subcomponent structure decided in chromosome encoding phase of the algorithm.

1. **Single Point Crossover Scheme:** In this crossover scheme, crossing two parent chromosomes at one specific point is realized. After crossing operation of parents' genetic information, two off springs are generated by exchanging grasped trailing parts between parents and merging them with heading parts. Procedure for single point crossover operation is depicted in Figure 2.3.

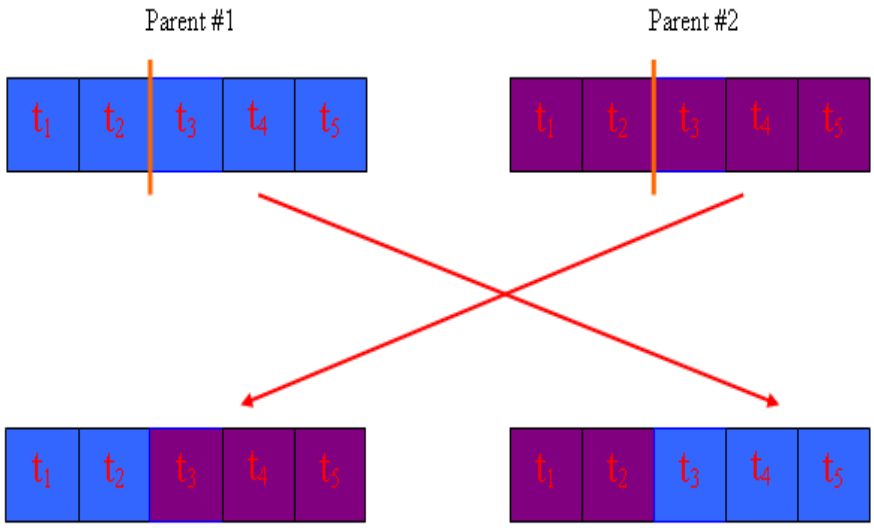

Figure 2.3: Single Point Crossover Procedure

2. **Multi Point Crossover Scheme:** In this crossover scheme, crossing two parent chromosomes at more than one specific point is realized. After crossing operation of parents' genetic information, two off springs are generated by exchanging grasped even – numbered parts between parents and merging them with odd – numbered parts. Procedure for multi point, two point for instance, crossover operation is depicted in Figure 2.4.

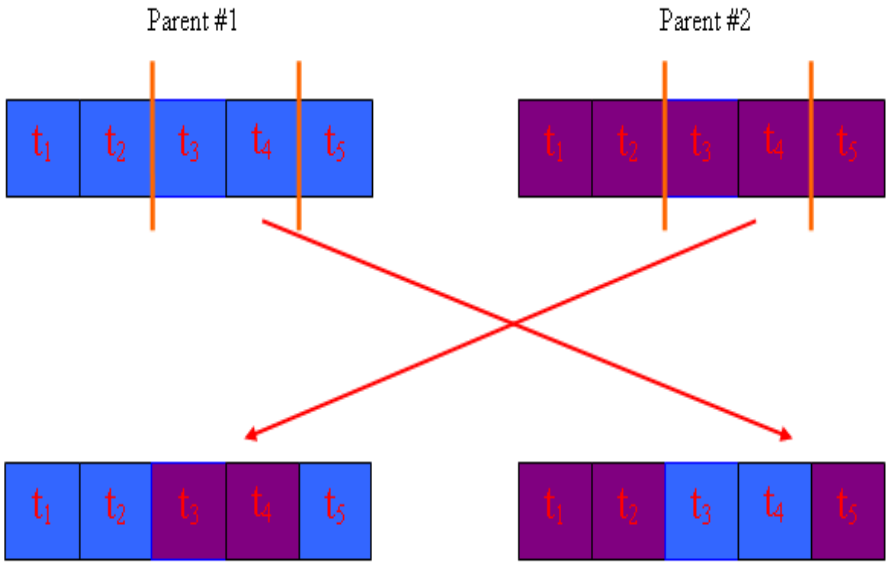

Figure 2.4: Two Point Crossover Procedure

- **Mutation Scheme Determination:** Mutation scheme is also an important issue in a genetic algorithm design. Although mutation is not a must operation in a genetic algorithm, availability of a useful mutation enriches diversity in current population and let the solution set not to converge to local optimality and stuck around that point.

  In general one or more randomly chosen atomic component change is used as the mutation scheme. In this changing procedure, inclusion of logic to hinder violation of solution's feasibility is also a common practice [23, 24, 25]. Another remarkable point about mutation scheme is the mutation rate specified

to determine whether mutation should be applied or not for current chromosome. This rate is important, because introducing very high possibility for mutation may cause the algorithm to converge late whereas introducing very low possibility may cause the algorithm stuck in local optimality and produce results further from global optimality. Also, this mutation rate need not stay same that it is possible to use changing mutation rate according to intermediate results of the algorithm [22]. Best values for mutation rate are reported to be between %0.5 and 1% [22]. Procedures for one – bit and multi – bit mutation operations are depicted in Figure 2.5.a and Figure 2.5.b respectively.

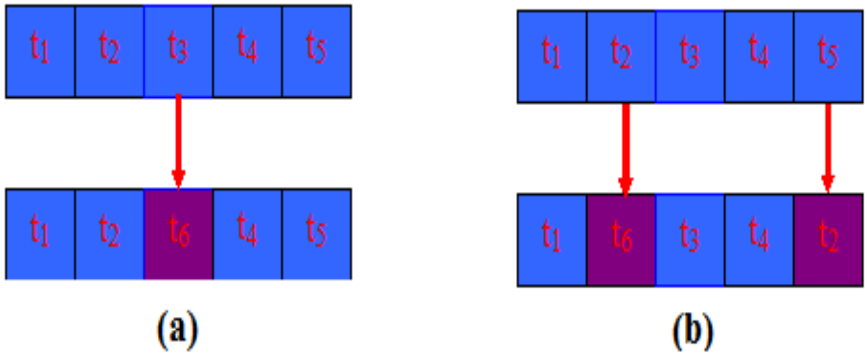

Figure 2.5: (a) One-bit Mutation Procedure, (b) Multi-bit Mutation Procedure

As a combination of the components and procedures introduced in this section, a standard genetic algorithm consists of the following steps:

1. Encoding of the chromosome structure

2. Initialization of the first population

3. Fitness calculation of the current population

4. Test of the current population to find out whether any chromosome satisfying end condition or not. If such a solution exists, the algorithm is terminated.

5. Generating new candidate solutions using crossover and mutation operators.

6. Replacement of old chromosomes by the new ones.

7. Go back to step 3. (Loop until fail criterion is not met)

CHAPTER 3

OUR WORK

This section presents details of our study on UAV – target assignment and scheduling procedures in UAV mission planning problem. First, problem formulation with requirements and operational environment assumptions are defined. Then, input set generation procedure and generated input sets' features are explained. Finally, the architecture of our framework, details of implementations and customizations of developed algorithms are described.

3.1 Problem Formulation and Constraints

In this study, UAV mission planning problem is solved by specifying which UAVs to operate on which routes and serve for which targets at which time periods of the mission. However, the key point here is to provide a feasible solution with respect to all requirements and constraints coming from problem domain. The requirements and constraints specified for the problem can be listed as follows:

1. **UAV Count Requirement:** Number of UAVs needed to handle a target may vary according to features of that target. This number is specified for each target by the user in scenario development phase and can be updated before each execution of that scenario.

2. **Time Window Requirement:** For a target, beginning and ending of service time may vary according to current internal and external factors the target is

subject to. So, it is required to serve that target during this time interval. This time interval is specified by the user in scenario development phase and can be updated before each execution of that scenario.

3. **Radar Zone Avoidance Requirement:** Radar zones and prohibited areas, which are known to exist in some specific locations in the operational environment, are assumed to be cared by the user during scenario development phase and linkages of targets are provided accordingly. However, existence of such a linkage procedure entails indirect transmissions among targets; and so, shortest path usage is required to be able to save resources by letting UAVs to go through the shortest paths.

4. **Fuel Level Consistency Requirement:** Fuel capacity of UAVs may also vary; and so, planning of UAVs' missions is required to happen accordingly. Since safety of resources has the highest priority, mission plans have to check fuel level of each UAV before deciding to direct it to serve for a target. In this work, UAVs are assumed to be generic; accordingly, fuel capacity is only allowed to be specified for all UAVs not individually. This fuel capacity is specified by the user in scenario development phase and can be updated before each execution of that scenario.

5. **Base Station Return Requirement:** As stated before, safety is one of the most important issues for this kind of missions. So, it is required to consider about return of UAVs to base station and compose mission plans accordingly.

Although main issue is to provide availability of desired number of UAVs in desired time interval for each of the targets exists in operational environment, providing the solution as optimal as possible regarding to some predefined objectives is also an important issue. There are mainly three objectives; which are minimizing total distance covered by UAVs, maximizing number of targets to be served during the mission and minimizing number of UAVs required to be used for the mission by maximizing UAV reuse. Approaches to the objectives are described in detail in Section 3.5.

An operational environment instance with target configurations, placed near of each target, is shown in Figure 3.1. Common and algorithm specific parameters and their specified values for the scenario can be seen in Figure 3.2.

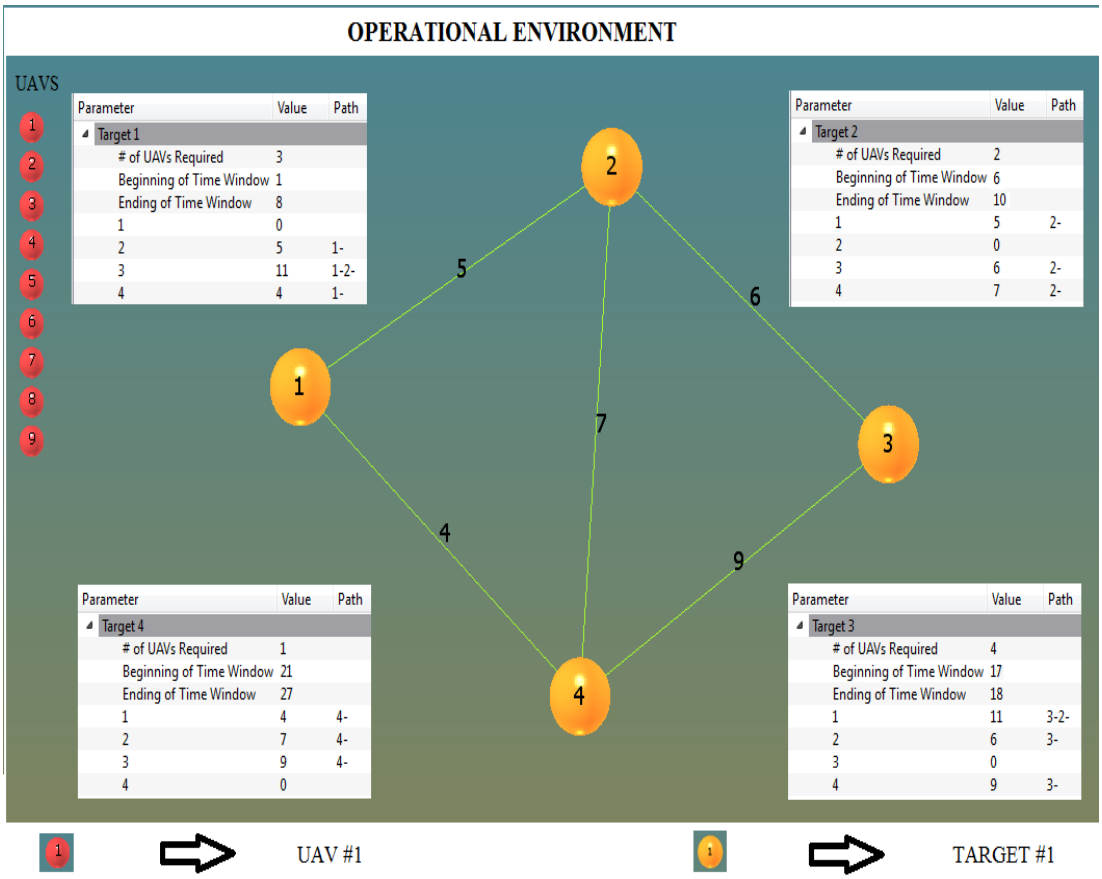

Figure 3.1: An Operational Environment with Target Configuration

Parameter	Value
Algortihm	
# of UAVs	9
Fuel Capacity	400
# of Candidate Solutions	30
Crossover Rate	100
Mutation Rate	5
Mutation Scheme	Multi-Point
Crossover Type	Usual
Crossover Scheme	Two-Point
Selection Method	Rank

Figure 3.2: Parameter Configuration for the Algorithms

Optimal solution of the problem instance given in Figure 3.1 and Figure 3.2 is shown in Figure 3.3. As can be seen from the figure, red cross marks are available to indicate which UAVs served for which targets during the mission. For the UAVs serving more than one targets, it is obvious that scheduling (serving order) is the order of beginning points of time windows of the targets in increasing order.

UAV / Target	Target # 1	Target # 2	Target # 3	Target # 4
UAV # 1			X	
UAV # 2	X			
UAV # 3	X			
UAV # 4	X			X
UAV # 5		X	X	
UAV # 6			X	
UAV # 7				
UAV # 8		X	X	
UAV # 9				

Figure 3.3: An Optimal Solution Instance of the Problem

3.2 Operational Environment Assumptions

It is an obligation to introduce some limitations and make some assumptions about some parameters of the problem to be able to acquire a platform enabling comparison of developed algorithms' performances fairly and easily. So, in the context of this study about UAV mission planning, some aspects related to elements of operational environments are exposed to some assumptions to be able to deal with a limited and well – defined problem domain. Assumptions concerning operational environment can be listed as follows:

1. Radar zones and prohibited areas are assumed to be taken into consideration by the user and scenario elements are placed and linked accordingly.

2. UAVs are assumed to be homogenous in the aspects of maximum speed, fuel level they can carry at most and fuel consumption / distance rate.

3. All UAVs are assumed to be stabilized to different altitudes and accordingly it is assumed that no crash is going to happen during missions.

4. Maximum / minimum turning radius limits of UAVs are not taken into consideration and they are assumed to turn any radius on any path defined by the user.

5. Targets are assumed to be handled with all – or – nothing principle regarding to requirement of providing needed number of UAVs to targets. So, it is not possible to handle any target partially by assigning less number of UAVs to that target.

6. It is assumed that no real – time change is going to happen for any of the user designed scenarios.

7. Weather condition related aspects, like visibility, temperature, contamination, cloud density and cloud altitude, which are affecting aerodynamics and / or flight capability of UAVs, are assumed to be stable regardless of the time scenario created.

8. Altitudes of landforms and buildings are not taken into consideration and all UAVs are always assumed to be flying at higher altitudes. So, no crash among landforms, buildings and UAVs is going to happen during missions.

3.3 Framework Architecture

A graphical framework, letting users to solve and manage problem instances easily, is implemented by utilizing Qt and OpenGL technologies with C++ programming language in MS Visual Studio 2008 platform. Main features provided by this framework can be listed as follows:

1. It enables users to create, update, delete and visualize scenarios involving mission planning of UAVs. Two sample scenarios (problem instances), designed using this framework, can be seen in Figure 3.4 and Figure 3.5.

2. It is possible to save any designed problem instance in XML format and users can load any available scenario in anytime required to resolve the problem instance available in that scenario. An XML – formatted scenario file of the problem instance shown in Figure 3.5 can be seen in Figure 3.6.

3. All algorithms, developed in the context of this thesis study, are designed to work on the same input format. So, this graphical framework eases performance test procedures of developed algorithms by enabling execution of any developed algorithm on the same scenario without need of any extra configuration change.

4. Once a feasible solution is grasped, by using any of the developed algorithms, for any predesigned problem instance; the framework enables users to simulate motions of the UAVs, according to timeline and coordinates provided by the solution, in 2D virtual operational environment.

5. A user – friendly graphical interface, enabling users to work with easily, is available for the framework.

6. It allows users to change configuration of any developed algorithm before execution. Algorithm specific parameters are arranged to be configurable as much as possible through this framework.

7. The framework enables users to collect test results of implemented algorithms in a file, which is outputted in text format.

8. It also enables generation of randomly configured problem instances with respect to limit values of configurable parameters where these parameters can

be seen in the graphical interface depicted in Figure 3.7. Users are allowed to configure the parameters via this graphical interface.

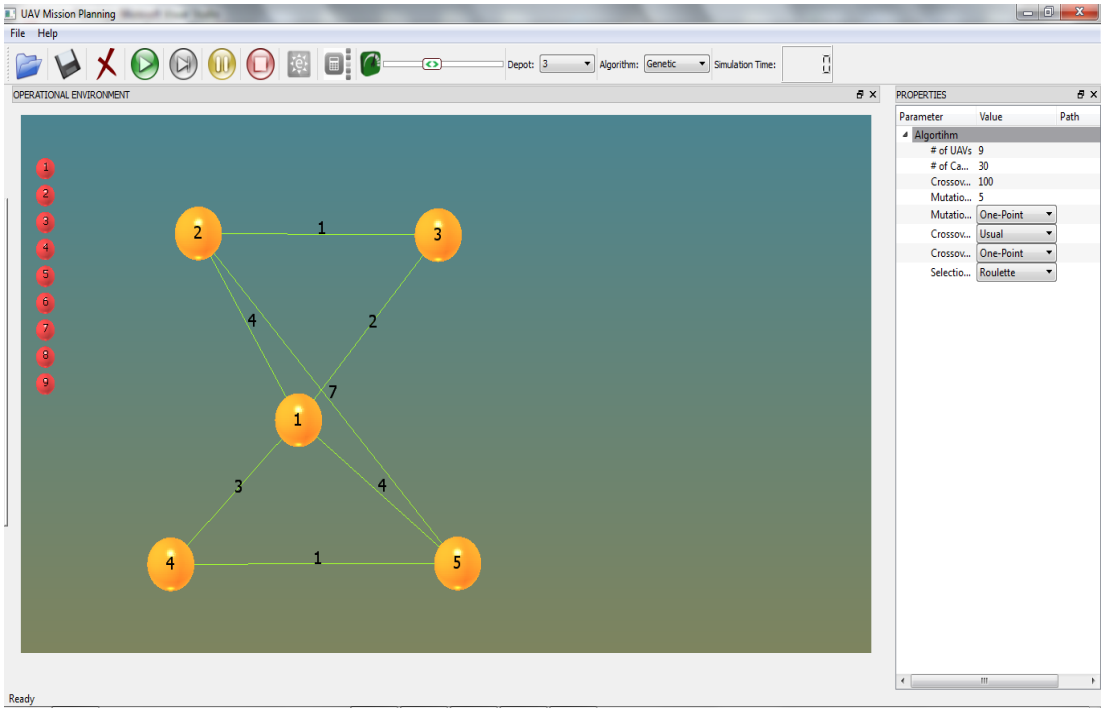

Figure 3.4: UAV Mission Planning Scenario Sample # 1

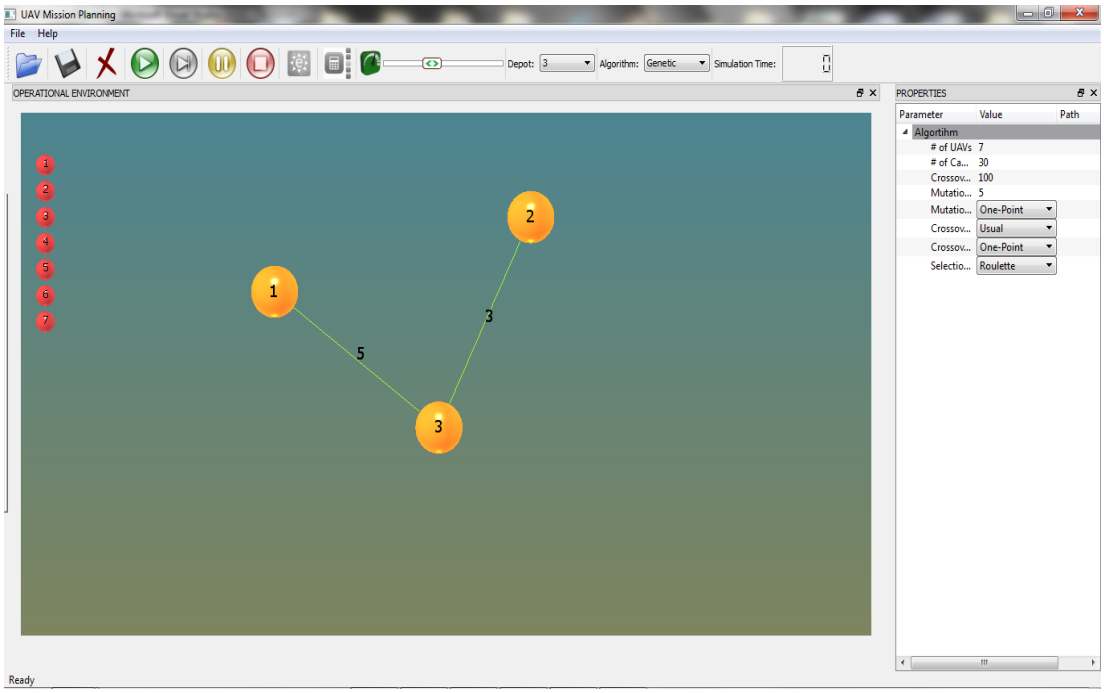

Figure 3.5: UAV Mission Planning Scenario Sample # 2

```
<Environment>
    <GeneticAlgorithm>
        <SolutionCount>30</SolutionCount>
        <CrossoverRate>100</CrossoverRate>
        <MutationRate>5</MutationRate>
        <MutationScheme>0</MutationScheme>
        <SelectionMethod>0</SelectionMethod>
        <CrossoverType>0</CrossoverType>
        <CrossoverScheme>0</CrossoverScheme>
    </GeneticAlgorithm>
    <Targets>
        <Target>
            <Name>1</Name>
            <Offset>-3.63708,1.57686</Offset>
            <UAVsRequired>3</UAVsRequired>
            <TimeWindow>15,21</TimeWindow>
            <Links>
                <Link>1,5,3</Link>
            </Links>
        </Target>
        <Target>
            <Name>2</Name>
            <Offset>1.79873,3.00317</Offset>
            <UAVsRequired>6</UAVsRequired>
            <TimeWindow>31,33</TimeWindow>
            <Links>
                <Link>3,3,2</Link>
            </Links>
        </Target>
        <Target>
            <Name>3</Name>
            <Offset>-0.150555,-1.02219</Offset>
            <UAVsRequired>4</UAVsRequired>
            <TimeWindow>1,11</TimeWindow>
            <Links>
                <Link>3,3,2</Link>
                <Link>1,5,3</Link>
            </Links>
        </Target>
    </Targets>
    <UAVs>
        <UAVCount>9</UAVCount>
        <UAV>
            <Name>1</Name>
            <Fuel>400</Fuel>
        </UAV>
        <UAV>
            <Name>2</Name>
            <Fuel>400</Fuel>
        </UAV>
        <UAV>
            <Name>3</Name>
            <Fuel>400</Fuel>
        </UAV>
        <UAV>
            <Name>4</Name>
            <Fuel>400</Fuel>
        </UAV>
        <UAV>
            <Name>5</Name>
            <Fuel>400</Fuel>
        </UAV>
        <UAV>
            <Name>6</Name>
            <Fuel>400</Fuel>
        </UAV>
        <UAV>
            <Name>7</Name>
            <Fuel>400</Fuel>
        </UAV>
    </UAVs>
</Environment>
```

Figure 3.6: XML – formatted Scenario File Sample

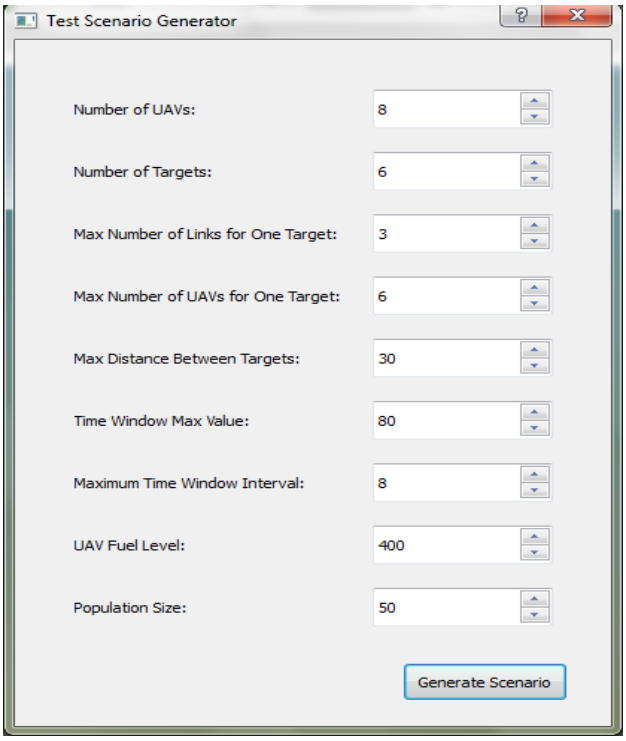

Figure 3.7: GUI of Automatic Test Scenario Generation Tool

3.4 Input Set Generation

Generation of input sets is realized by using the framework implemented in the context of this study. It is so easy to work with the architecture; that various problem instances can be created on this framework both manually and automatically. Once the problem instances are created and saved, it is possible to use them again and again.

As the framework is designed generically to serve for each implemented algorithm, so is the input whose format is illustrated before in Figure 3.6. Hence, it becomes easy to compare implemented algorithms' performances for same scenarios.

For input set generation procedure, one of the key concepts is variety of problem instances regarding to problem domain, configuration parameters and requirements specified for the problem. For this problem, scenarios having target number variety,

UAV number variety with respect to target number, time window variety of targets, fuel level variety of UAVs and UAV need variety of targets are provided for all of the implemented algorithms by using the framework.

As the result, by changing user configurable parameters, mentioned above, two input sets containing high degree of variation are obtained for the test procedure automation of this study. Details and features of these input sets are given in Chapter 4.

3.5 Algorithm Implementations and Customizations

This part explains the algorithms and customizations made for them in the context of this thesis study.

As a preprocessing procedure shortest paths among targets are computed according to graphs of the operational environments given in scenarios. Dijkstra's shortest path algorithm is used for computing shortest paths among targets which are assumed to be linked according to radar zone and prohibited area locations so that UAVs would not be noticed by enemies.

3.5.1 Exhaustive (Brute – Force) Search Algorithm

Conventional exhaustive search algorithm is implemented with an additional early elimination procedure. This elimination procedure is applied for all solution candidates, enumerated by the conventional exhaustive search algorithm, to be able shorten execution time by eliminating obviously non – feasible or lesser suitable ones without proceeding with any other operations for them. For this problem, the elimination procedure is to check availability of solution candidates having more infeasible targets compared to the one marked as the current best solution. After detection of such a situation, obviously there is no need to continue with the other steps for that solution candidate.

We implemented this algorithm with the aim of acquiring optimum solutions for small sized input sets, and use them as reference solutions in comparison with outputs of the other developed algorithms. Main steps of this algorithm can be given as follows:

1. Try to enumerate one different solution candidate, if there is no different solution candidate, terminate and return with best solution if one exists otherwise return null.

2. Apply early elimination procedure to the current solution candidate, if the procedure results with failure for this candidate then discard it and go back to step 1.

3. Check for feasibility of each target for current solution candidate; if it is fully infeasible, then discard the candidate and go back to step 1.

4. Compute fitness value of solution instance.

5. Mark the current solution instance as the best solution if the fitness value is higher than the fitness value of current best solution. Go back to step 1.

3.5.2 Greedy Search Algorithm

A greedy algorithm based solution is provided for this UAV – target assignment and scheduling problem to be able compare performances of this algorithm with ones of genetic algorithm solution. Although exhaustive search algorithm is implemented with the same aim, for the scenarios including more than a few UAVs and targets it is totally impractical to use exhaustive algorithm because of its huge running time. So, greedy algorithm is intended to be used in performance comparison of genetic algorithm especially for large sized input set.

It is generally not possible to reach optimal solutions by using greedy algorithms because of their working principles, which impose proceeding with locally optimum solutions instead of looking the whole picture and behaving accordingly. But these principles provide the feature of terminating in shorter time durations compared to almost all other algorithms in literature. For this study, the same conditions are also valid that our greedy algorithm provides nearly optimal solutions in shortest time durations compared to exhaustive search and genetic algorithms. Main steps of this algorithm can be given as follows:

1. Sort the targets according to their beginning points of time windows in increasing order and insert them into a list in that order.

2. Get next target from sorted target list. If there is no target available in the list, return the existing solution.

3. For the current target, look for available and nearest UAVs in operational environment greedily and try to reserve required number of UAVs to meet the need of current target. If there is not enough number of UAVs available for current target mark it as infeasible.

4. For current target, compute local fitness by taking feasibility into consideration and sum up this value to global fitness. Go back to step 2.

With the aim of increasing UAV reuse instead of using an unused UAV from base, a penalty distance is included additional to current distance, specified in the scenario, in the case of a new UAV is intended to be used.

3.5.3 Genetic Algorithm

A genetic algorithm based solution is also provided for UAV – target assignment and scheduling procedures in UAV mission planning problem. Main aim of this study is

to implement this genetic algorithm to be able to reach an acceptable level of optimality for the problem in shorter times compared to manual and other possible computer assisted solutions.

Genetic algorithm design is substantially related to the problem domain and it requires modifying the components of a genetic algorithm according to the domain requirements. Accordingly, for this problem instance; we have approached critical components, like encoding, crossover and fitness computation, of genetic algorithm with a customized fashion. Main components of the genetic algorithm with customizations are explained in detail as follows:

3.5.3.1 Encoding

Use of real number encoding scheme is chosen regarding to this optimization problem's nature. As it is possible to assign more than one UAV to one specific target, some kind of hierarchical encoding, providing abstraction where necessary, is planned to use. In this encoding structure, targets are located into first hierarchy level and the UAVs are located below them as the second hierarchy level. Visualization of the encoding is provided in Figure 3.8.

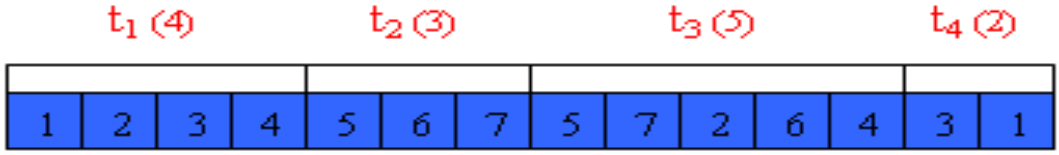

Figure 3.8: Chromosome Structure with Encoding Format

The encoding illustrated in Figure 3.8 represents a solution instance for a scenario in which the operational environment is designed to contain 4 targets, which are t_1, t_2, t_3 and t_4, and number of UAVs needed to be able to cover the targets are 4, 3, 5 and 2 respectively. When looking at the second level of hierarchy, there are 7 different UAVs which are indexed from 1 to 7; and UAV – target assignment has taken place

as depicted below. Accordingly, the encoding implies the UAVs, which are assigned to any target (i.e. located below that target in the figure) to be serving for that target in specified time window of the target.

$$t_1 = \{u_1, u_2, u_3, u_4\}$$
$$t_2 = \{u_5, u_6, u_7\}$$
$$t_3 = \{u_5, u_7, u_2, u_6, u_4\}$$
$$t_4 = \{u_3, u_1\}$$

As the problem specification does not allow real time change in operational environment of the scenario, fixed length of chromosome usage is encouraged. However, fixed length chromosome encoding has provided easiness for application of genetic algorithm operators. In addition, two – level hierarchic encoding scheme has also made customizations of the other operators easy.

3.5.3.2 Population Initialization

Population initialization is an important issue for a genetic algorithm design. Although initialization procedure is not one of the key operations, an efficient initialization directly affects running time required to find optimal solutions. It is because starting execution with a population consisting of highly fertile members (i.e. members having higher fitness values) is expected to enhance overall fertility during algorithm running process. So, in this genetic algorithm design procedure, population initialization procedure is realized by caring about following conditions and removing negative effects of them if possible.

- **Infeasible Operational Environment Checking:** An early checking of designed operational environment, whether specified number of UAVs is enough to meet minimum needs of targets or not. If it is not, an early termination of the algorithm resulting with failure occurs.

- **Time Window Feasible Assignment Providing:** Main aim here is to provide UAV – target assignment feasibility with respect to time window requirements of the targets. In this context, without caring about paths and distances among targets, just infeasible scheduling (i.e. scheduling requiring any UAV to be serving for at least two targets at the same time) is avoided as much as possible for each member of initial population.

- **Keeping Used UAV Set Minimal:** Reuse of UAVs is tried to be maximized while generating initial population. In this context, without losing local and global feasibility, number of UAVs to be used is kept as minimum as possible.

3.5.3.3 Crossover

Crossover is the key operation for a genetic algorithm since solution space variation and diversity are provided by this operation. In the context of this study, based on the hierarchy provided in chromosome encoding phase, two crossover techniques are used together. First of them, realizing crossover operation for the first hierarchy level (i.e. target level), is Target Level Crossover technique and the second one is UAV Level Crossover technique which is realizing crossover operation for the second hierarchy level (i.e. UAV level). Both of these crossover techniques are explained in detail below:

1. **Target Level Crossover:** In this crossover technique, abstraction provided by the chromosome encoding is utilized. Hence, it has become possible to think the chromosome as if just including one level (first hierarchy level) containing targets as atomic elements. Accordingly, with this new structure, crossover is applied using both one – point and two – point schemes which are shown in Figure 3.9 and Figure 3.10 respectively.

Decision of which scheme to use is left to user by allowing choice of scheme before execution.

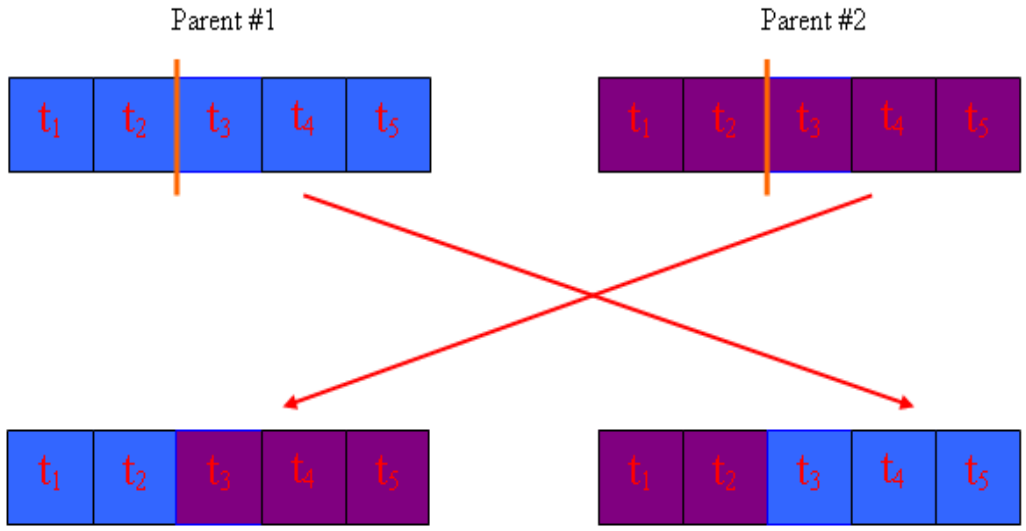

Figure 3.9: Target Level Single Point Crossover Scheme

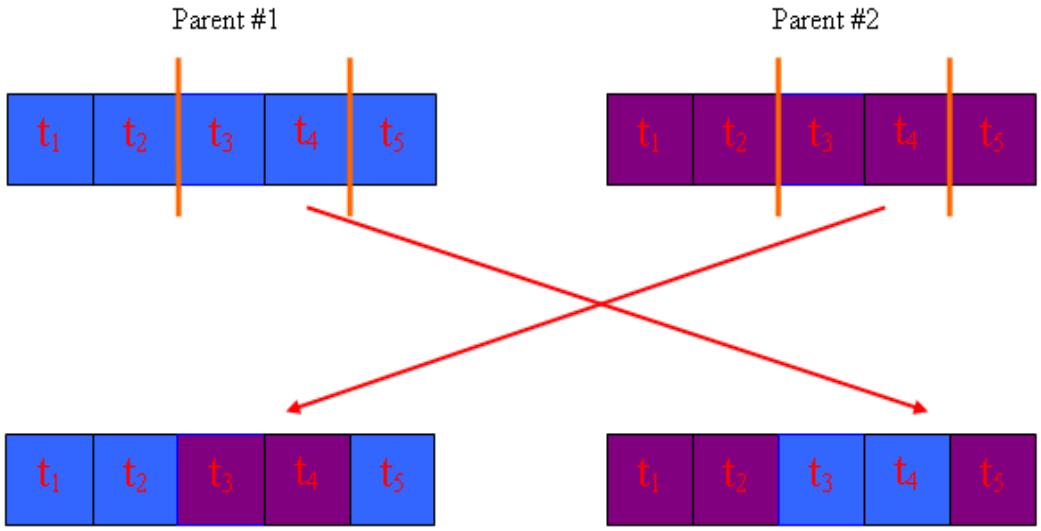

Figure 3.10: Target Level Two Point Crossover Scheme

2. **UAV Level Crossover:** In this crossover technique, abstraction provided by the chromosome encoding is utilized. Hence, it has become possible to think the chromosome as if just including one level (second hierarchy level) containing UAVs as atomic elements. Accordingly, with this new structure, crossover operation is applied using both one – point and two – point schemes which can be seen in Figure 3.11 and Figure 3.12 respectively. Decision of which scheme to use is left to user by allowing choice of scheme before execution.

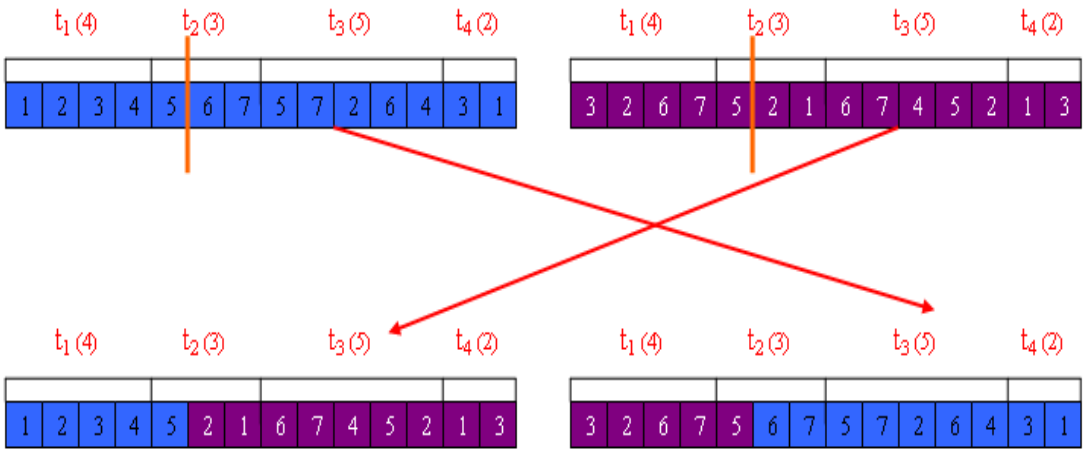

Figure 3.11: UAV Level Single Point Crossover Scheme

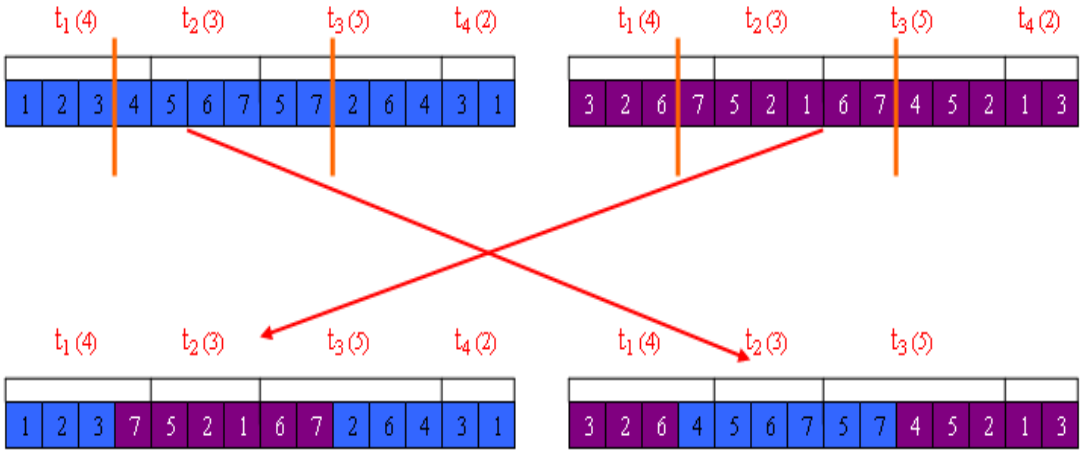

Figure 3.12: UAV Level Two Point Crossover Scheme

In this algorithm, both of the UAV Level Crossover and Target Level Crossover techniques are used during the same session (execution). For each iteration, decision of which technique to be utilized is made randomly but with equal probability. By using this customized crossover technique, both the two levels of the encoding are exploited for the sake of generating strong members having higher fertility.

Specifically, utilization of Target Level Crossover provides generating members without losing subchromosome (i.e. target) fitness and enabling integration of highly fertile subchromosomes into one specific chromosome. However, utilization of UAV Level Crossover provides generating subchromosomes having higher level of fertility by just dealing with UAVs in subchromosomes. So, it becomes best idea to utilize both of the techniques to acquire more fertility for subchromosomes and accordingly, acquire highly fertile chromosomes by gathering these subchromosomes together.

A more standard version of this genetic algorithm utilizing only UAV Level Crossover technique is also provided with the aim of comparison. Hence, it becomes possible to evaluate efficiency and effectivity of two – level crossover operation in a comparative manner.

3.5.3.4 Mutation

Single and multibit mutation is used in the implementation of the genetic algorithm designed in the context of this study. The mutation scheme to be used is left to the user and can be specified before each execution of the algorithm. In application of multibit mutation, number of points to be mutated is restricted by a specific number, which is randomly chosen between one and half of the target count available in the scenario.

Both single and multibit mutation procedures are only applied for UAV level of the chromosome according to the rate specified by the user. For both of the mutation schemes, some kind of logic, trying to avoid from mutation of any chosen bit with an

already existing bit for that target, is also inserted into implementation procedure of the mutation operator.

3.5.3.5 Selection

Rank and Roulette Wheel selection techniques are both used in the implementation of the genetic algorithm and decision of which selection technique to be used is left to user.

3.5.3.6 Fitness Calculation

Multiobjective fitness function, mainly aiming to asses three objectives, is used in the implementation of the genetic algorithm. As it is easier and quicker to find optimal solution using singleobjective fitness functions, multiobjectivity is an unavoidable real life practice. Accordingly, we have tried to include as many problem related objectives as possible into our solution model. In the context of this study we care about following objectives:

1. Minimizing total distance covered by all involved UAVs.

2. Minimizing number of UAVs to be used for the mission.

3. Maximizing number of targets to be served during the mission.

4. Fuel levels of UAVs are also taken into consideration and planning of their missions is realized accordingly.

The multiobjective fitness function, used for assessing fitness of chromosomes (i.e. solutions) generated by all the algorithms, can be mathematically formulized as follows:

$$Fitness = (\sum_{i=1}^{t} dm_i - \sum_{j=1}^{n} dt_j) + (t-n)\times 100 + (2s-k)\times 500$$

where k is the total number of targets available in operational environment, s is the number of targets served during the mission, t is the total number of UAVs available in operational environment, n is the number of UAVs used for the mission, dm_i is the maximum distance i_{th} UAV can travel using available fuel and dt_j is the distance travelled by j_{th} used UAV during the mission.

Here in this fitness function:

- First part, which is the difference of total distance to be covered using all UAVs and the distance covered during the mission, is available to keep first objective.

- Second part, which is the difference of total number of UAVs available and the number of UAVs used for the mission, is available to keep second objective.

- Third part, which is the difference of twice of the number of targets served during the mission and total number of targets available, is available to keep third objective.

However, fourth objective is kept logically by directing UAVs according to their current fuel level and providing their safe return to base station. Also note that in fitness function, the second and the third parts are multiplied by 100 and 500 respectively to be able to boost their relative effectivity in total fitness. These values are determined according to relative importance of the objectives in the mission and configuration of the input scenarios. In addition, first variable of third part is multiplied by 2 with the aim of providing positive fitness for each served target.

CHAPTER 4

EVALUATION OF THE RESULTS

This chapter provides experimental results obtained by realizing test procedures for all the implemented algorithms. Graphical visualizations of these results are also given in a comparative manner.

Mainly there are two input sets, which are small sized and large sized sets, and both of their two subsets. These subsets are produced based on randomness of operational environment composition; that one randomly (automatically) and one consciously (manually) composed input subsets are provided for testing phase. Main aim of consciously composing is to provide some fully feasible missions which are hard to get by randomly composing, especially for large sizes.

Evaluations and comparisons of the algorithms are performed according to objective function output values acquired for each of 4 input subsets. Objective based test results are also given with the aim of enabling evaluation and comparison of the algorithms' performances based on a single objective or combination of some of them.

All the tests are run on a laptop computer having 4 GB RAM, 2.53 GHz Intel(R) Core(TM) i5 CPU and 64 bit Windows 7 Home Premium operating system. However, testing framework is compiled to work as a 32 bit application.

Although, configuration change is allowed for each different execution of the algorithms, some default values, which are determined according to similar studies in the literature, are also provided for some of them.

4.1 Testing Procedures and Configuration of the Algorithms

Testing configurations and evaluation procedures of the implemented algorithms are given below:

4.1.1 Exhaustive Search Algorithm Implementation

This algorithm is used to find optimum solutions for small sized problem instances. These optimum solutions are used as reference to assess quality of solutions generated by the proposed methods. However, elapsed time values are very high for this algorithm. So, it is impractical to compare efficiency of the algorithm and use it for solution of this problem in real life.

No bounding logic is available for this algorithm but some logic providing early elimination of useless solution candidates is included to the implementation of the algorithm. In addition, maximum fitness based sorting method is applied to all feasible solution candidates to be able to find best performing solution(s).

4.1.2 Greedy Search Algorithm Implementation

Greedy search algorithm is applied to all problem instances. The aim is to provide solutions to be able to evaluate effectivity and efficiency of the genetic algorithm, by comparing the solutions with ones resulted from the genetic algorithm, especially for the large sized input set.

As this algorithm tries to assign UAVs to targets greedily, elapsed time values are generally shorter compared to other algorithms. Elapsed time values of the algorithm given in Section 4.2 show that this algorithm terminates in acceptable time for the problem instances of this study.

No bounding logic is provided for this algorithm. However, maximum fitness based sorting method is applied to all feasible target assignment candidates to be able to find best performing ones with the aim of reaching best performing solution by merging them.

4.1.3 Genetic Algorithm Implementation

This is the main algorithm developed for UAV mission planning problem with the aim of acquiring best or acceptable results in reasonable time. All problem instances available in both of the input sets are solved by using variations of this algorithm.

Variations of the algorithm, evaluated in this study, can be listed in Table 4.1 as follows:

Table 4.1: Variations of Genetic Algorithm

GA Algorithm	**Specification**
GA_Single_Rank (GA_1)	Genetic algorithm with two – level single random point crossover and rank selection
GA_Multi_Rank (GA_2)	Genetic algorithm with two – level two random point crossover and rank selection
GA_Single_Roulette (GA_3)	Genetic algorithm with two – level single random point crossover and roulette selection
GA_Multi_Roulette (GA_4)	Genetic algorithm with two – level two random point crossover and roulette selection
GA_Simple_Multi_Rank(GA_5)	Genetic algorithm with one – level (UAV level) two random point crossover and rank selection

It is possible to say that genetic algorithm is the most preferable algorithm for this problem when looking at the overall results reflecting algorithms' performances in both of the effectivity and efficiency aspects.

Main configuration parameters of the genetic algorithm and the values specified for them, according to similar studies in literature, are given below:

- Crossover probability = 0.85
- Mutation probability = 0.05
- Population count = 1
- Population size = 50
- Number of generations = 200

4.2 Results of Experiments

One of the two most important differences of the algorithms is the elapsed time change according to input size and configuration. This change is directly related with time complexities of the algorithms. The other most important difference is the effectivity, which is directly related with design of the algorithms. All of the algorithms are compared in both of these two differing aspects using both of the input sets.

Mainly there exist two input sets for the evaluation, which are small and large sized input sets, regarding to requirement of providing requested number of UAVs to the targets of the operational environment in specified time intervals. Also, for each of the input sets, there exist two subsets, which are randomly and consciously composed subsets, regarding to need for evaluating the effect of randomness factor, which highly exists in real life.

Experiments are realized using subsets of both of the input sets whose detailed specifications are given below:

4.2.1 Results of Small Sized Input Set

Main issue here is to avoid time complexity of exhaustive search algorithm which is highly impractical for the scenarios providing huge number of possible assignment schemes. So, some specific scenarios, not having potential of generating more than 65 million assignment schemes, are generated in order to be able to compare performances of the implemented algorithms. For all of the scenarios of this set, number of UAVs to be available in operational environment is decided to be same and equals to 8. Number of targets to be available in operational environment is also decided to be same and equals to 5. Variation of the scenarios is provided by varying number of possible assignment schemes. Additionally, fuel level of UAVs is set to 300 distance units for all members of this input set.

This input set is divided into two subsets, which are randomly composed subset (Input subset 1) and consciously composed subset (Input subset 2) and each of which containing 12 different scenarios.

Scenario		Exhaust.		GA_1		GA_2		GA_3		GA_4		Greedy	
Assignment	Targets	Gain	Time	Gain	Time	Gain	Time	Gain	Time	Gain	Time	Gain	Time
802816	5	4169	70	4167	.078	4169	.094	4167	.094	4169	.109	4167	.001
1404928	5	4828	73	4828	.062	4828	.063	4828	.062	4828	.079	4828	.001
2508800	5	3496	113	3496	.109	3496	.122	3489	.110	3496	.109	2864	.001
5619712	5	2822	380	2822	.093	2822	.093	2822	.082	2822	.093	2822	.001
5619712	5	3916	104	3916	.064	3916	.081	3916	.094	3916	.103	3916	.001
5619712	5	4138	35	3938	.078	4138	.124	4138	.156	4138	.171	3938	.001
9834496	5	4564	284	4362	.078	4563	.106	4358	.094	4362	.109	3950	.001
9834496	5	3114	198	3114	.109	3114	.116	2816	.106	3114	.156	2816	.001
9834496	5	3910	60	3910	.081	3910	.092	3910	.111	3910	.127	3910	.001
19668992	5	3504	1402	3504	.093	3504	.094	3504	.092	3504	.112	3504	.001
34420736	5	4172	1102	4172	.094	4172	.096	4172	.109	4172	.110	3972	.001
49172480	5	3282	2811	3266	.124	3282	.167	3082	.140	3266	.164	2850	.001

Figure 4.1: Test Results of Input Subset 1

4.2.1.1 Results of Randomly Composed Input Set

This input subset includes 12 small sized and randomly composed scenarios. In this subset number of possible assignment schemes varies from 802816 to 49172480. Scenarios of this subset are generated automatically by the framework.

Both of elapsed time and fitness values measured for each algorithm, using this input subset, can be seen in Figure 4.1.

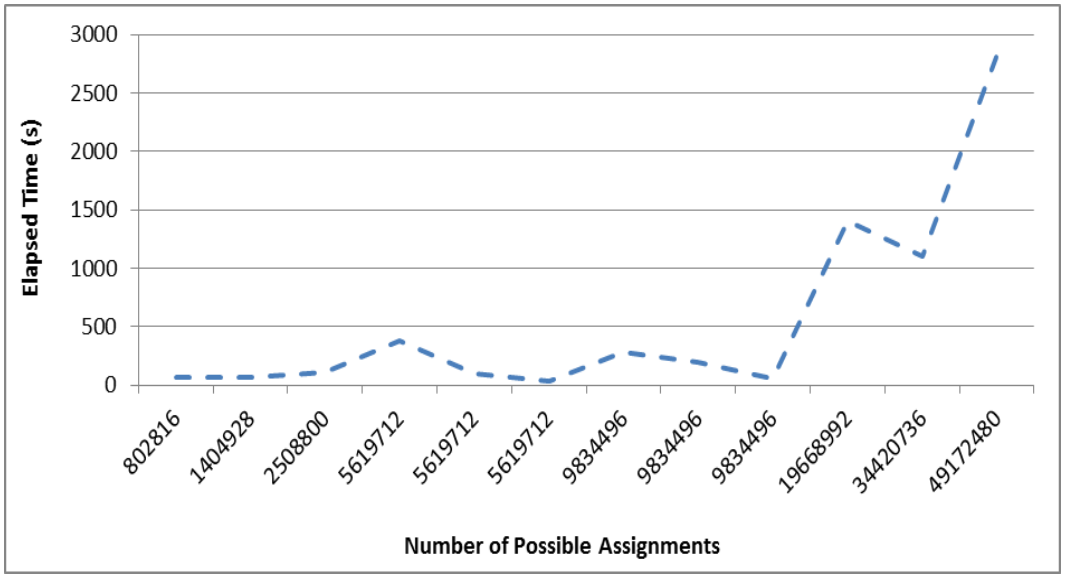

Figure 4.2: Elapsed Time of Brute Force Algorithm for Input Subset 1

For the traditional application of exhaustive search algorithm, time complexity of the algorithm is directly proportional to number of possible assignments. However, with early elimination, it is not the only factor affecting time complexity; occurrence and count of early eliminations are also important. Main issue here is the time, the algorithm reached to best solution while enumerating all possibilities. It is possible to observe mentioned situation samples in Figure 4.2 that time complexity tends to increase proportionally but exceptional scenarios are available because of the situation.

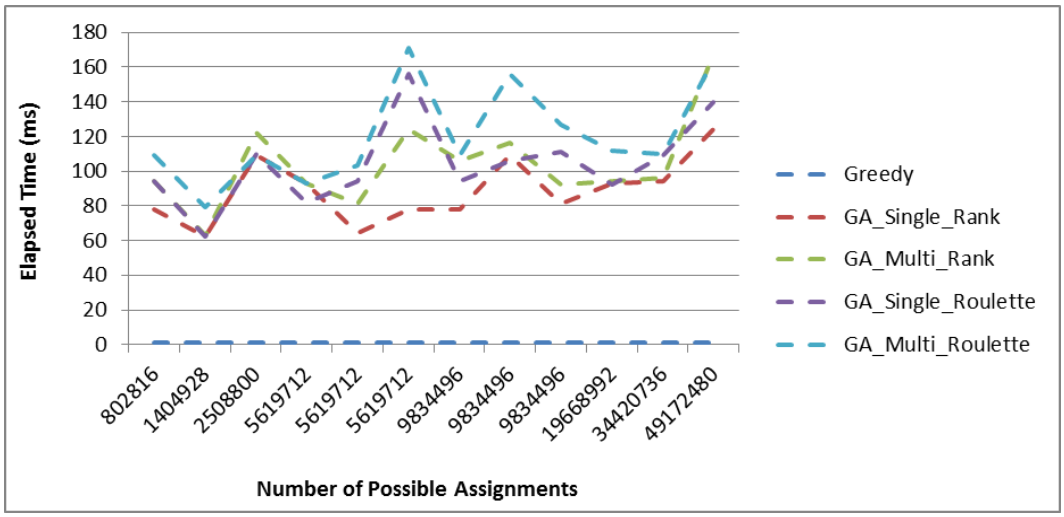

Figure 4.3: Elapsed Time of the Algorithms for Input Subset 1

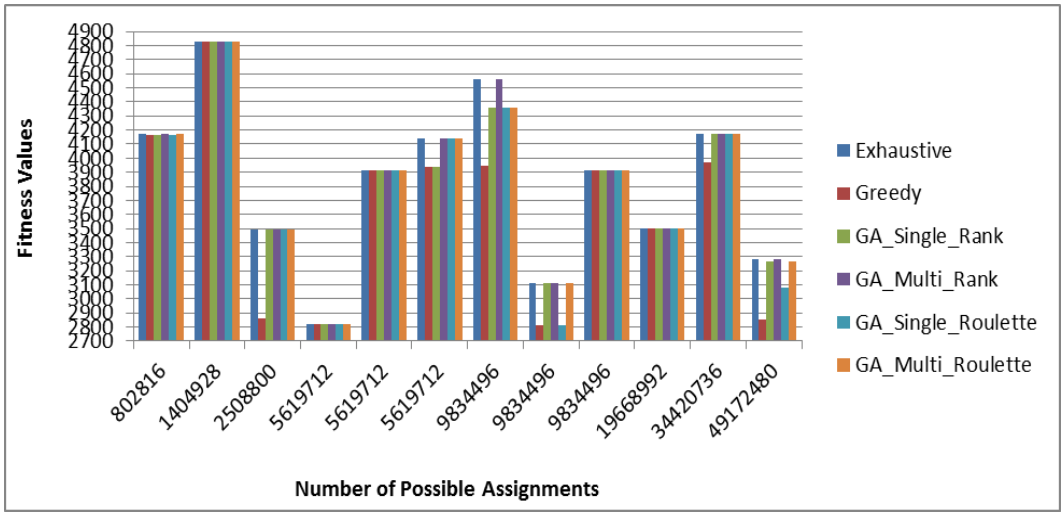

Figure 4.4: Fitness Values of the Algorithms for Input Subset 1

As it can be seen in Figure 4.3, for this randomly composed subset, rank selection yields better time complexity compared to roulette selection for genetic algorithm variations on the average. However, multi (two) point crossover operation takes more time compared to single point one. Besides, we can see that efficiency of greedy algorithm is the best as expected.

Figure 4.3 also shows that elapsed time change of greedy algorithm and genetic algorithm variations is not directly related with possible number of assignments. Really, this is another expected result because of the fact that main factor affecting these algorithms' efficiency is the size of the chromosome data structure, which is determined according to number of targets available in operational environment and number of UAVs needed to serve for those targets.

As expected, exhaustive search algorithm is the most effective algorithm for this randomly composed input subset according to fitness values given in Figure 4.4. On the contrary, greedy algorithm yields the worst of all the algorithms on the average; but it is not so far from the reference. However, genetic algorithm variations provide close fitness values to ones provided by the reference algorithm. It is possible to see that variations of genetic algorithm applying multi point crossover operation are slightly better than the ones applying single point. Additionally, rank selection outperforms roulette selection for a few scenarios. So, GA_Multi_Rank algorithm applying multi point crossover operation for two levels and using rank selection reaches to top effectivity, which is almost same with the reference, among all the genetic algorithm variations.

Scenario		Exhaust.		GA_1		GA_2		GA_3		GA_4		Greedy	
Assignment	Targets	Gain	Time	Gain	Time	Gain	Time	Gain	Time	Gain	Time	Gain	Time
7024640	5	4166	311	4166	.152	4166	.156	4166	.156	4166	.187	3966	.001
7024640	5	4608	113	4608	.093	4608	.094	4608	.092	4608	.109	4608	.001
7024640	5	4200	112	4000	.118	4196	.126	4000	.134	4000	.140	4000	.001
11239424	5	4398	766	4392	.114	4398	.124	4392	.127	4398	.141	4382	.001
12293120	5	4434	87	4434	.092	4434	.096	4434	.102	4434	.114	4434	.001
12293120	5	3440	287	3440	.148	3440	.161	3240	.146	3440	.125	3040	.001
19668992	5	4646	276	4646	.093	4646	.094	4646	.091	4646	.101	4646	.001
24586240	5	4816	1978	4816	.107	4816	.109	4816	.108	4816	.117	4816	.001
24586240	5	3966	156	3966	.108	3966	.109	3966	.124	3966	.127	3966	.001
49172480	5	4767	2771	4767	.107	4767	.118	4767	.122	4767	.140	4762	.001
49172480	5	4832	2482	4832	.097	4832	.107	4832	.098	4832	.112	4832	.001
61465600	5	4236	1983	4234	.138	4236	.148	4233	.140	4236	.156	4035	.001

Figure 4.5: Test Results of Input Subset 2

4.2.1.2 Results of Consciously Composed Input Set

This input subset includes 12 small sized and consciously composed scenarios. In this subset number of possible assignment schemes varies from 7024640 to 61465600. Scenarios of this subset are generated manually by us.

Both of elapsed time and fitness values measured for each algorithm, using this input subset, can be seen in Figure 4.5.

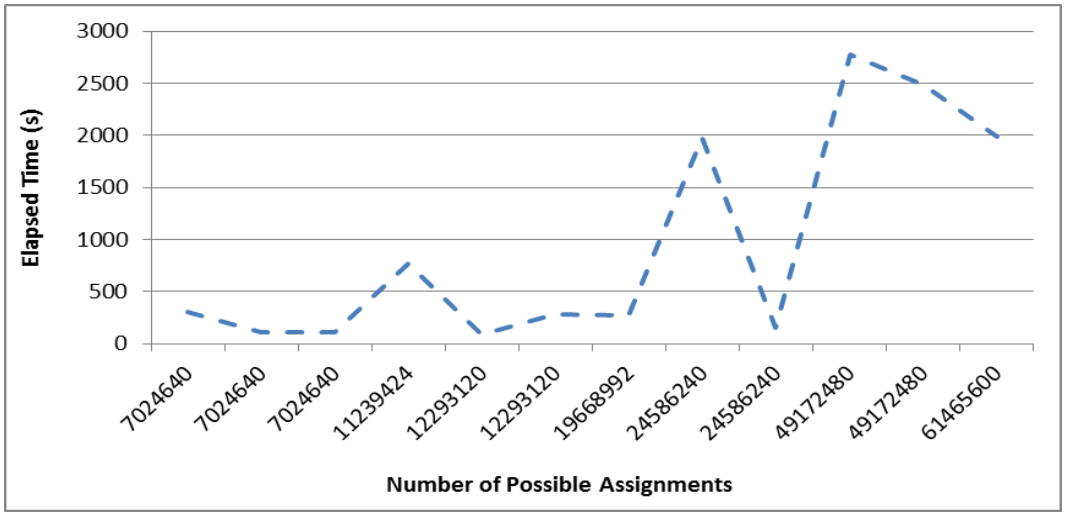

Figure 4.6: Elapsed Time of Brute Force Algorithm for Input Subset 2

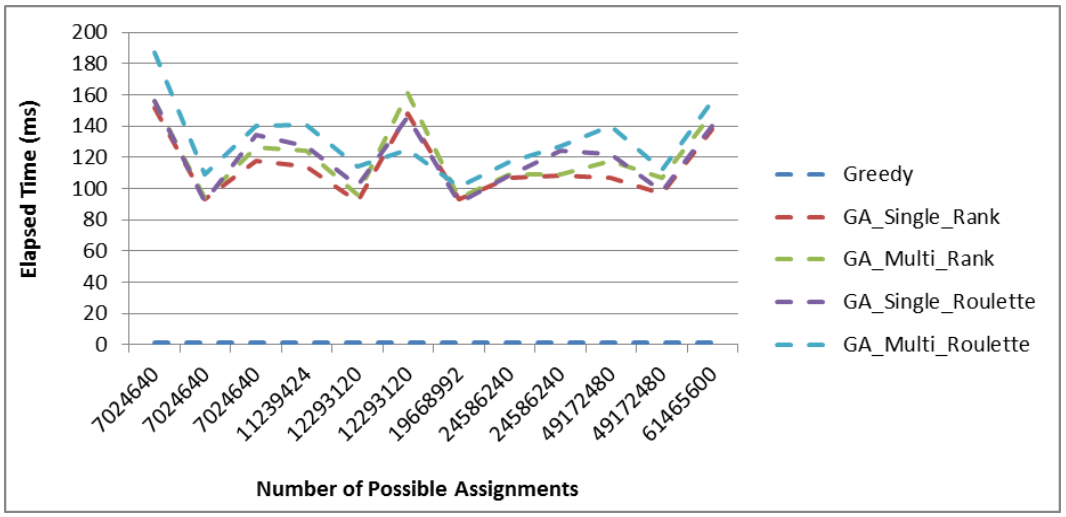

Figure 4.7: Elapsed Time of the Algorithms for Input Subset 2

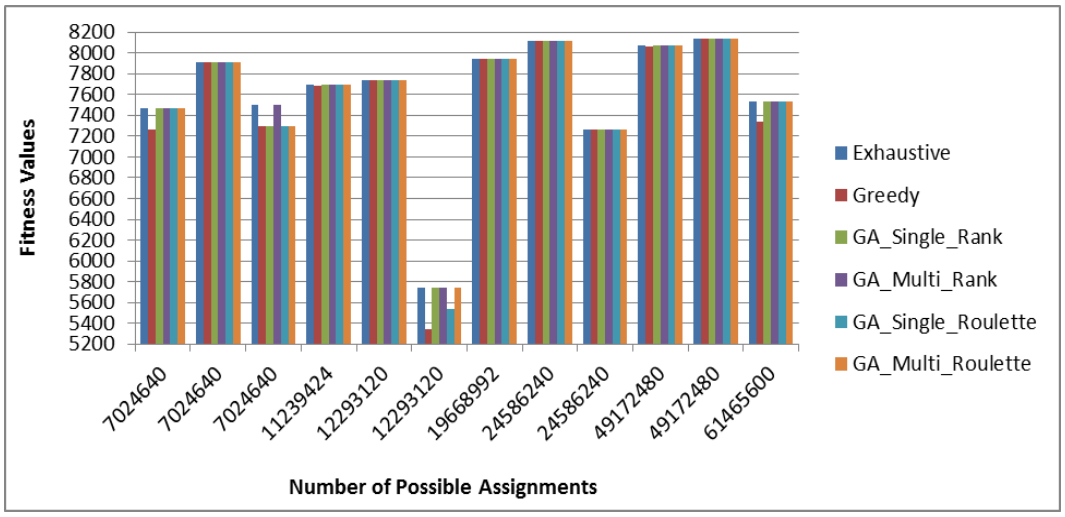

Figure 4.8: Fitness Values of the Algorithms for Input Subset 2

For this consciously composed subset, we have 11 fully feasible scenarios which are 7 for randomly composed one. Although higher fitness values are observed because of this difference, this situation does not make a substantial change in relative efficiency and effectivity of the algorithms. So, regarding to test results depicted in Figure 4.6, Figure 4.7 and Figure 4.8, efficiency and effectivity evaluation of the algorithms occurs similar to the previous one occurred for randomly composed input set.

4.2.2 Results of Large Sized Input Set

This input set consists of some specific scenarios having more than 65 million different assignment schemes. Because of high time complexity of exhaustive search algorithm and availability of large sized inputs, evaluation of this input set is realized without this algorithm. Main aim for this evaluation is to analyze quality of outputs for the genetic algorithm by comparing fitness and elapsed time values of this algorithm with the ones resulted from the greedy algorithm. For all of the scenarios of this set, number of UAVs to be available in operational environment is decided to be same and equals to 20. But this time, number of targets is increased by one from scenario to scenario. Additionally, fuel level of UAVs is set to 400 distance units for all members of this input set.

This input set is also divided into two subsets, which are randomly composed subset (Input subset 3) and consciously composed subset (Input subset 4) and each of which containing 12 different scenarios.

Scenario		GA_1		GA_2		GA_3		GA_4		Greedy	
UAVs	Targets	Gain	Time	Gain	Time	Gain	Time	Gain	Time	Gain	Time
20	6	10261	.234	10461	.280	10259	.264	10461	.491	9957	.016
20	7	10116	.265	10116	.281	10116	.265	10116	.281	9810	.017
20	8	9662	.249	9871	.265	9653	.267	9799	.267	9244	.019
20	9	10245	.737	10450	.972	10215	.749	10250	.827	9935	.021
20	10	9832	.499	9848	.671	9832	.501	9848	.686	9832	.022
20	11	12143	.567	12238	.599	12138	.571	12229	.608	12029	.024
20	12	9796	.873	9827	.951	9813	.902	9830	1.295	9430	.027
20	13	10035	1.108	10267	1.488	10013	1.201	10075	1.328	9843	.032
20	14	11628	.637	11819	.662	11614	.688	11631	.714	11304	.034
20	15	9849	.780	10077	.812	9849	.795	9998	.842	9466	.034
20	16	8682	.1186	8682	.1216	8682	.1199	8682	.1277	8270	.036
20	17	9873	.694	9956	1.748	9873	.697	9956	1.860	9651	.037

Figure 4.9: Test Results of Input Subset 3

4.2.2.1 Results of Randomly Composed Input Set

This input subset includes 12 large sized and randomly composed scenarios. In this subset number of targets varies from 6 to 17. Scenarios of this subset are generated automatically by the framework.

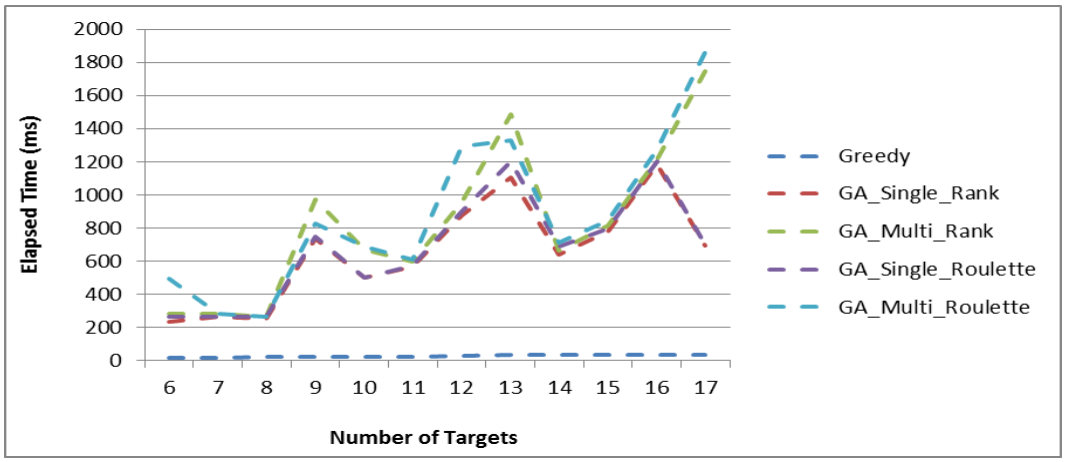

Figure 4.10: Elapsed Time of the Algorithms for Input Subset 3

Both of elapsed time and fitness values measured for each algorithm, using this input subset, can be seen in Figure 4.9.

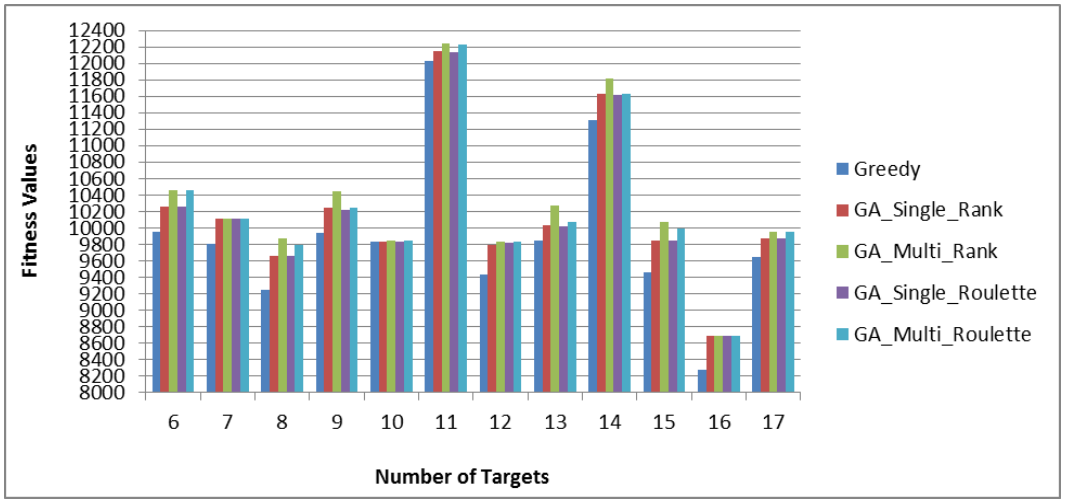

Figure 4.11: Fitness Values of the Algorithms for Input Subset 3

As stated before, there is no reference algorithm available for this input subset. So, comparison of just greedy and variations of genetic algorithms is provided in this section. Because of varying number of targets, size of the chromosome data structure grows and shrinks accordingly. Mainly, both of greedy and genetic algorithms' elapsed time values are directly related to that size as shown in Figure 4.10. However, there are also some other factors, such as relative locations of targets, relative conditions of target time windows and number of distinct UAVs to be used, that are also affecting time complexity. Accordingly, it is possible to see in Figure 4.10 that time complexity of all algorithms tends to increase as the number of targets increase; but for genetic algorithm, there are also some scenarios affected from these mentioned factors and caused some extra points to emerge in the graph.

Relative performances of the algorithms stay similar to the one happened for small sized scenarios that greedy algorithm is the most efficient and the least effective algorithm again. According to the results depicted in Figure 4.10 and 4.11, genetic algorithms applying multi point crossover operation are more effective and less efficient in average when compared to ones applying single point. Also, rank selection is again performing slightly better than roulette selection, for both aspects, in average.

4.2.2.2 Results of Consciously Composed Input Set

This input subset includes 12 large sized and consciously composed scenarios. In this subset number of targets varies from 6 to 17. The scenarios available in this subset are generated manually by us.

Both of elapsed time and fitness values measured for each algorithm, using this input subset, can be seen in Figure 4.12.

Scenario		GA_1		GA_2		GA_3		GA_4		Greedy	
UAVs	Targets	Gain	Time	Gain	Time	Gain	Time	Gain	Time	Gain	Time
20	6	11336	.099	11336	.113	11336	.104	11336	.121	11336	.016
20	7	11329	.234	11329	.265	11329	.343	11329	.406	10931	.017
20	8	11389	.711	11589	.742	11387	.717	11585	.787	10366	.018
20	9	12090	.436	12290	.733	12082	.530	12282	.792	11690	.020
20	10	12096	.484	12099	.577	12096	.492	12096	.543	11896	.022
20	11	12359	.327	12558	.342	12359	.327	12359	.406	12354	.024
20	12	12750	.343	12854	.674	12750	.436	12754	.468	12650	.026
20	13	13616	1.030	13617	1.112	13427	.827	13427	.967	12806	.028
20	14	12242	.921	12444	.942	12047	.967	12244	1.404	11644	.031
20	15	12092	.936	12242	1.397	12238	1.498	12238	1.575	12038	.033
20	16	14912	1.249	15112	1.260	14889	1.311	15101	1.370	14291	.036
20	17	12796	1.716	12796	1.732	12796	1.606	12797	2.171	11825	.041

Figure 4.12: Test Results of Input Subset 4

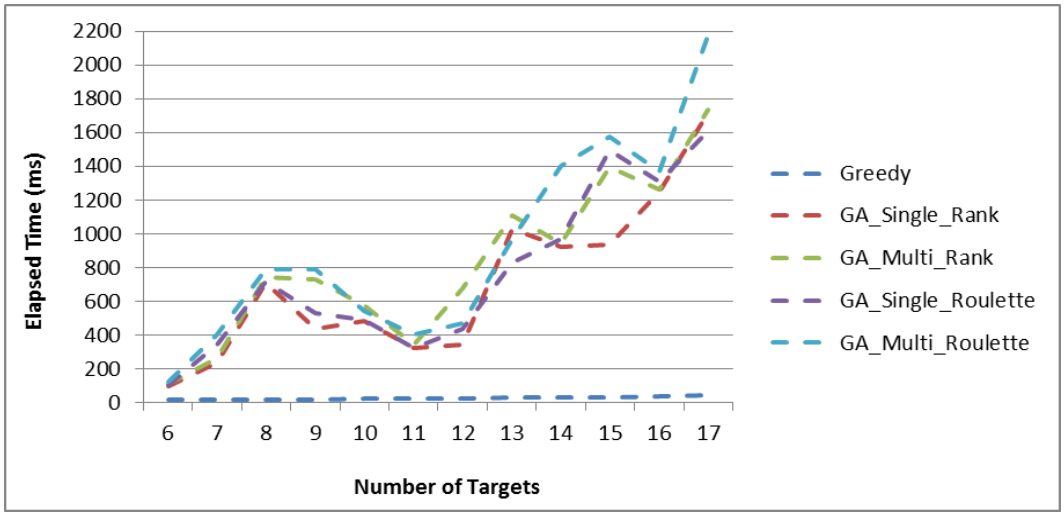

Figure 4.13: Elapsed Time of the Algorithms for Input Subset 4

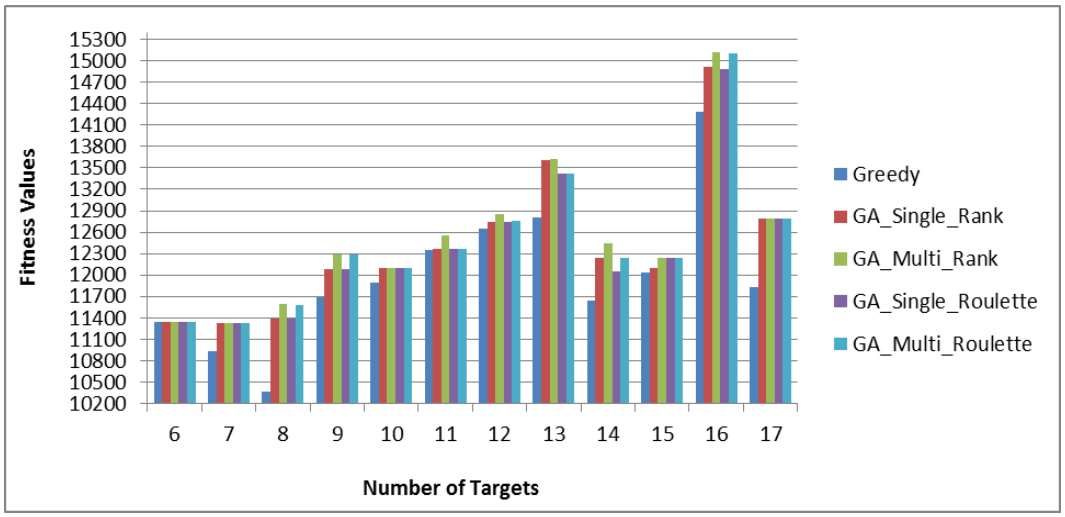

Figure 4.14: Fitness Values of the Algorithms for Input Subset 4

For this consciously composed subset, we have 9 fully feasible scenarios which are 3 for randomly composed one. Although, higher fitness values are observed because of this difference, this situation does not make a substantial change in relative efficiency and effectivity of the algorithms. So, regarding to test results depicted in Figure 4.13 and Figure 4.14, efficiency and effectivity evaluation of the algorithms occurs similar to the previous one occurred for randomly composed input subset of this set.

According to all of the test results gathered for 4 input subsets, remarkable inferences are listed as follows:

- Although evaluated only for small sized set, exhaustive search algorithm with early elimination provides the best effectivity and the worst efficiency.

- Greedy algorithm performs the worst effectivity and the best efficiency.

- For genetic algorithms implemented for this problem, multi point crossover reaches more effective fitness values compared to single point ones that the situation is also emphasized in [28], which also deals with a similar problem. However, single point crossover is more efficient than the multi point one.

- For genetic algorithms implemented for this problem, rank selection is more efficient than roulette selection in general. Moreover, it achieves better effectivity for some specific scenarios.

- As a consequence of third and fourth conclusions, GA_Multi_Rank algorithm provides highly appealing results when evaluated for both of effectivity and efficiency aspects together. Although time complexity of the algorithm is not even best of genetic algorithm variations, it presents nearly optimal solutions in terms of effectivity. Moreover, the average time complexity of the algorithm is in acceptable range to be utilized for this problem.

4.2.3 Objective Based Evaluation of Test Results

Availability of three different objectives in fitness function lets us to present objective based test results in addition to fitness based test results with the aim of analyzing objective based performances of the algorithms. However, these objective based test results are only given for large sized input set because of the reality that performances of the algorithms for small sized input set do not pose substantial differences.

Scenario		GA_1			GA_2			GA_3			GA_4			Greedy		
UAV	Targets	Dis.	U	Tar.	Dis.	U	Tar.	Dis.	U	Tar.	Dis.	U	Tar.	Dis.	U	Tar.
20	6	1439	13	6	1339	12	6	1441	13	6	1339	12	6	1643	14	6
20	7	1584	8	6	1584	8	6	1584	8	6	1584	8	6	1690	10	6
20	8	2338	10	7	2229	9	7	2347	10	7	2301	9	7	2556	12	7
20	9	2555	17	9	2450	16	9	2585	17	9	2550	17	9	2665	19	9
20	10	2368	18	9	2352	18	9	2368	18	9	2352	18	9	2368	18	9
20	11	1957	14	11	1962	13	11	1962	14	11	1971	13	11	1971	15	11
20	12	2404	18	10	2373	18	10	2387	18	10	2370	18	10	2570	20	10
20	13	2565	19	11	2433	18	11	2587	19	11	2525	19	11	2657	20	11
20	14	2572	18	13	2481	17	13	2586	18	13	2569	18	13	2696	20	13
20	15	2051	16	11	1923	15	11	2051	16	11	1902	16	11	2234	18	11
20	16	2618	17	11	2618	17	11	2618	17	11	2618	17	11	2730	20	11
20	17	2627	20	13	2644	19	13	2627	20	13	2644	19	13	2849	20	13
Average:		2257	15.7	9.8	2199	15.0	9.8	2262	15.7	9.8	2227	15.3	9.8	2386	17.2	9.8

Figure 4.15: Objective Based Test Results of the Algorithms for Input Subset 3

Objective based performances of the algorithms for large sized randomly and consciously composed subsets can be seen in Figure 4.15 and Figure 4.16 respectively. With the aim of providing objective based comparison, total distance covered by all UAVs (Dis.), total number of UAVs used for the mission (Uav) and total number of targets served during the mission (Tar.) are given regarding to first, second and third objectives respectively.

As can be seen in Figure 4.15, for randomly composed subset, there is no difference for performances of the algorithms in target coverage aspect. But, for distance coverage issue, effectivity of genetic algorithms is superior to effectivity of greedy one; that the difference varies from %5.2 to %7.8 on the average and GA_Multi_Rank algorithm provides the highest effectivity. In addition, it is possible to lower total covered distance up to %18.5 and save corresponding level of fuel for some specific scenarios. For UAV usage issue, again genetic algorithms outperform greedy algorithm from %8.7 to %12.8 on the average and GA_Multi_Rank algorithm reaches to top effectivity again. Besides, for some specific scenarios, it is possible to save UAVs up to %25 using genetic algorithm variations. As the result, it can be concluded for this input subset that the genetic algorithms yield more effective results compared to ones of the greedy algorithm on the average for two of the three objectives.

Scenario		GA_1			GA_2			GA_3			GA_4			Greedy		
UAV	Targets	Dis.	Uav	Tar.	Dis.	Uav	Tar.	Dis.	Uav	Tar.	Dis.	Uav	Tar.	Dis.	Uav	Tar.
20	6	864	8	6	864	8	6	864	8	6	864	8	6	864	8	6
20	7	1171	10	7	1171	10	7	1171	10	7	1171	10	7	1369	12	7
20	8	1511	11	8	1311	11	8	1513	11	8	1315	11	8	1934	17	8
20	9	1310	11	9	1110	11	9	1318	11	9	1118	11	9	1510	13	9
20	10	1704	12	10	1701	12	10	1704	12	10	1704	12	10	1704	14	10
20	11	1741	14	11	1642	13	11	1741	14	11	1741	14	11	1746	14	11
20	12	1850	14	12	1746	14	12	1850	14	12	1846	14	12	1850	15	12
20	13	1584	13	13	1683	12	13	1573	15	13	1673	14	13	1994	17	13
20	14	2158	16	13	2056	15	13	2253	17	13	2256	15	13	2356	20	13
20	15	2408	20	14	2358	19	14	2362	19	14	2362	19	14	2462	20	14
20	16	1688	14	16	1688	12	16	1711	14	16	1699	12	16	2009	17	16
20	17	2704	20	16	2704	20	16	2704	20	16	2703	20	16	2675	20	15
Average:		1724	13.6	11.2	1670	13.1	11.2	1730	13.7	11.2	1704	13.3	11.2	1873	15.6	11.1

Figure 4.16: Objective Based Test Results of the Algorithms for Input Subset 4

As can be seen in Figure 4.16, for consciously composed subset, there is only one scenario posing difference in target coverage aspect; that genetic algorithms cover one more target compared to greedy algorithm in that scenario. For distance coverage issue, effectivity of genetic algorithms is superior to effectivity of greedy one; that the difference varies from %7.6 to %10.8 on the average and GA_Multi_Rank algorithm provides the highest effectivity. In addition, it is possible to lower total covered distance up to %32.2 and save corresponding level of fuel for some specific scenarios. For UAV usage issue, again genetic algorithms outperform greedy algorithm from %12.2 to %16 on the average and GA_Multi_Rank algorithm reaches to top effectivity again. Besides, for some specific scenarios, it is possible to save UAVs up to %35.3 using genetic algorithm variations. As the result, it can be concluded for this input subset that the genetic algorithms yield more effective results compared to ones of the greedy algorithm on the average for all of the three objectives.

For large sized and randomly composed subset, although fitness based test results show superiority of genetic algorithms' average effectivity from %2.8 to %4.1 compared to average effectivity of greedy algorithm, objective based evaluation shows that the reason of the small difference is relative effectivity of the third objective in fitness; that the objective is achieved equally well by all of the algorithms. However, there are substantial performance differences in terms of first and second objectives.

For large sized and consciously composed subset, although fitness based test results show superiority of genetic algorithms' average effectivity from %3.4 to %4.5 compared to average effectivity of greedy algorithm, objective based evaluation shows that the reason of the small difference is relative effectivity of the third objective in fitness; that the objective is achieved almost equally well by all of the algorithms. However, there are substantial performance differences in terms of first and second objectives.

As the general result, it is possible to conclude for two subsets of large sized input set that the genetic algorithms yield reasonably more effective results compared to ones of the greedy algorithm on the average especially for the first and second objectives.

4.2.4 Comparison of Genetic Algorithms' Performances

As stated before; another genetic algorithm, applying crossover operation only for UAV level of the chromosome data structure, is also implemented with the aim of making efficiency and effectivity comparison of two – level crossover operation possible. Hence, two genetic algorithms, applying specified crossover operation to one level and two levels of the chromosome structure, are obtained for comparison process.

Both of the algorithms are arranged to use multi point crossover scheme and rank selection technique, which constitute the best performing genetic algorithm configuration of this problem, together.

Performances of the GA_Multi_Rank and GA_Simple_Multi_Rank algorithms are compared with respect to both of the fitness and objective based test results. However, the objective based comparison is only given for large sized input set because of the reality that performances of the algorithms for small sized input set do not pose substantial differences.

4.2.4.1 Fitness Based Comparison

Fitness based comparison of the genetic algorithms is provided in this section. The algorithms, GA_Multi_Rank and GA_Simple_Multi_Rank, are both tested with already composed small sized and large sized input sets and both of their randomly and consciously composed subsets. For this time, gathered fitness and elapsed time results are not tabulated but graphically visualized in a comparative manner.

With respect to efficiency and effectivity aspects, evaluations of the algorithms, for all input subsets, are given below:

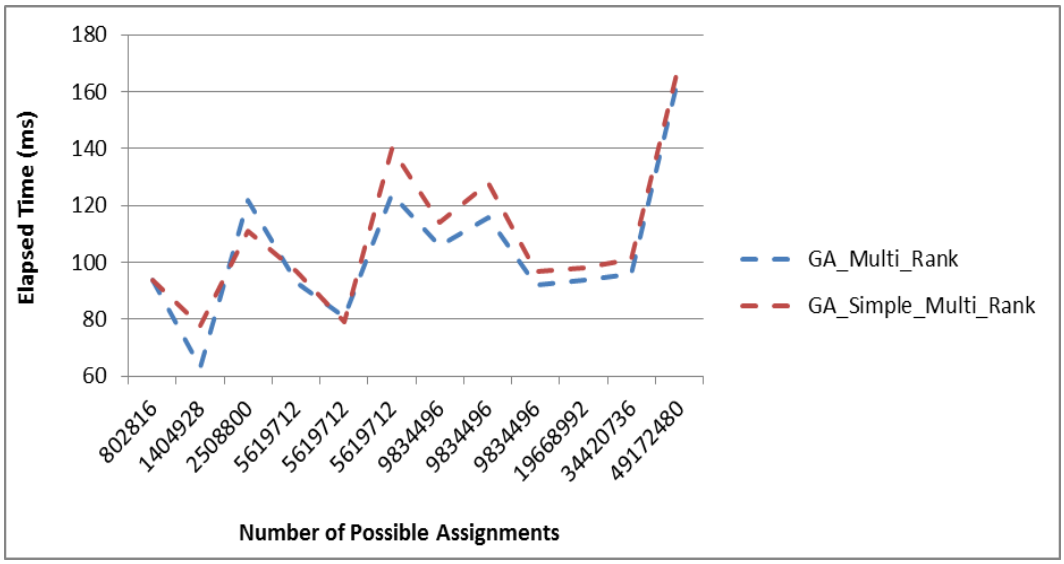

Figure 4.17: Elapsed Time of Genetic Algorithms for Input Subset 1

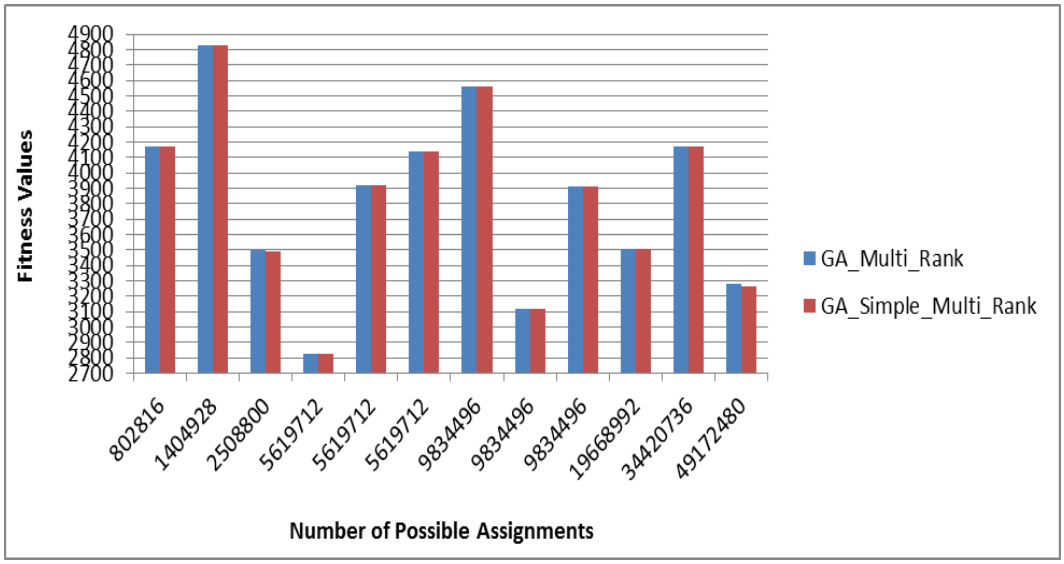

Figure 4.18: Fitness Values of Genetic Algorithms for Input Subset 1

As it can be seen in Figure 4.17 and Figure 4.18, for the small sized and randomly composed input subset, both of the algorithms perform very close to each other with respect to both of the evaluation criterions. Accordingly, it is possible to say that two – level crossover scheme does not make any substantial difference for this input subset.

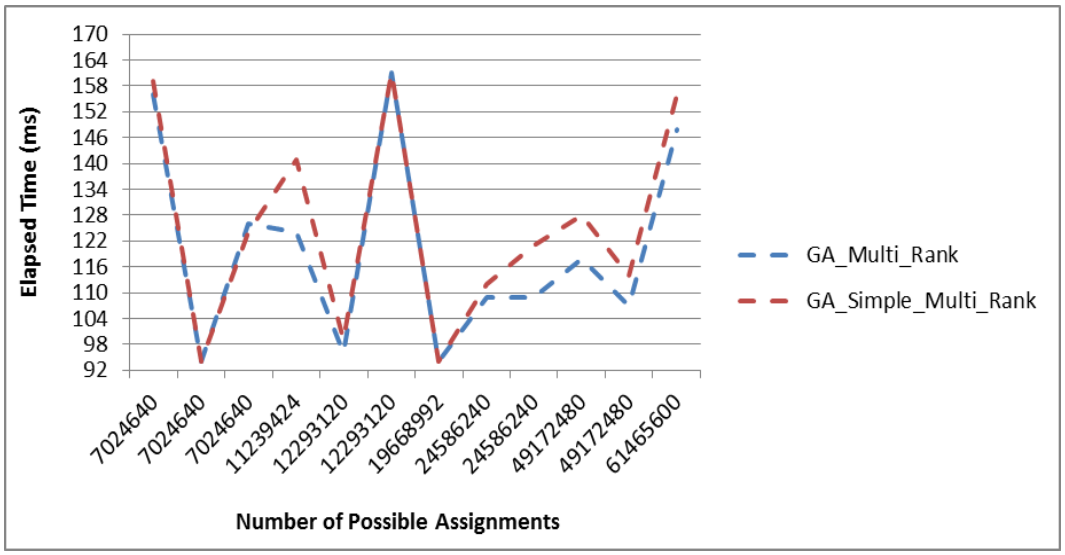

Figure 4.19: Elapsed Time of Genetic Algorithms for Input Subset 2

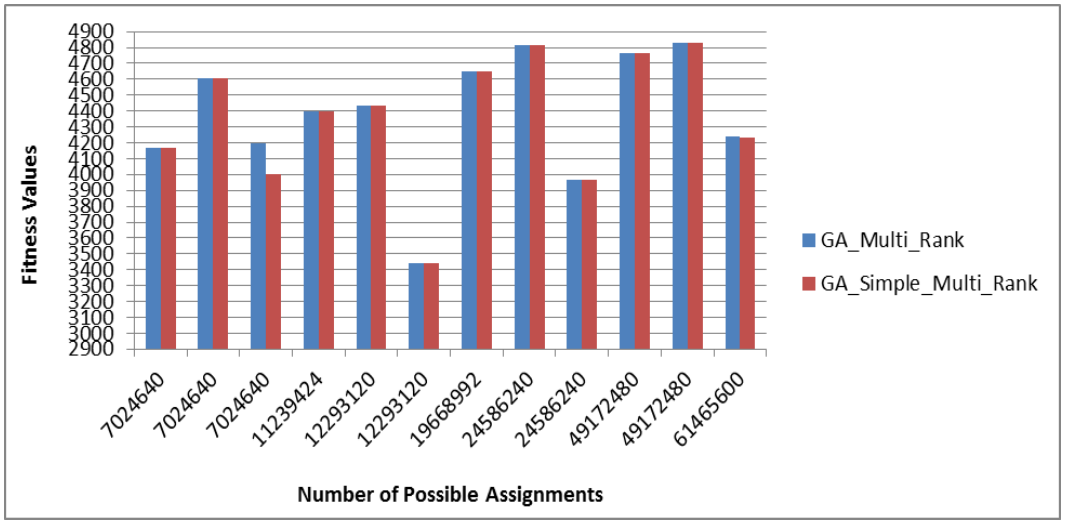

Figure 4.20: Fitness Values of Genetic Algorithms for Input Subset 2

Figure 4.19 and Figure 4.20 show that, for the small sized and consciously composed subset, GA_Multi_Rank algorithm achieves better effectivity for one specific scenario and also results with slightly better average elapsed time compared to GA_Simple_Multi_Rank algorithm. However, it is not realistic enough to declare GA_Multi_Rank algorithm as the better one for small sized input set, because there is no obvious outperforming situation available especially for effectivity aspect.

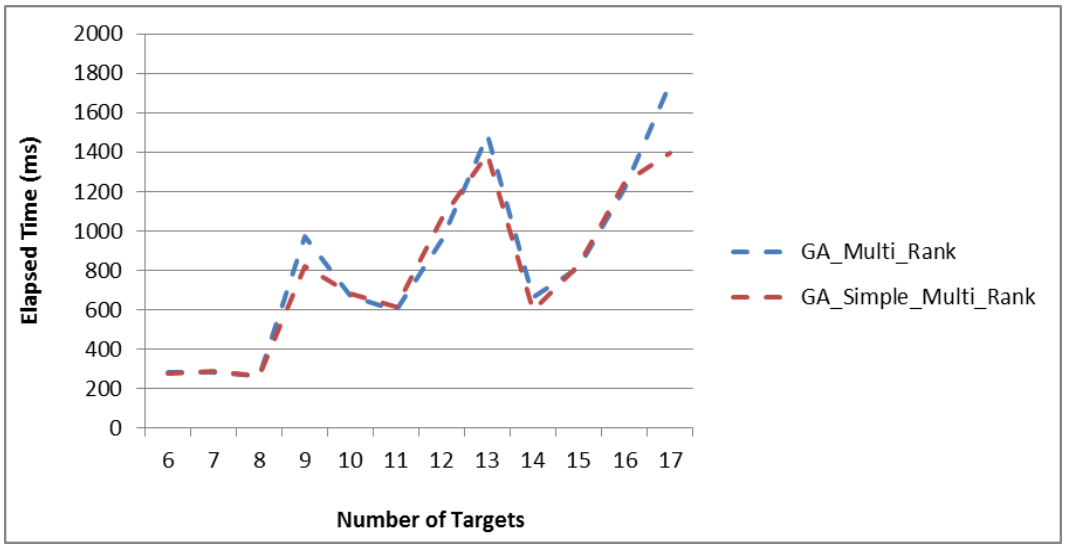

Figure 4.21: Elapsed Time of Genetic Algorithms for Input Subset 3

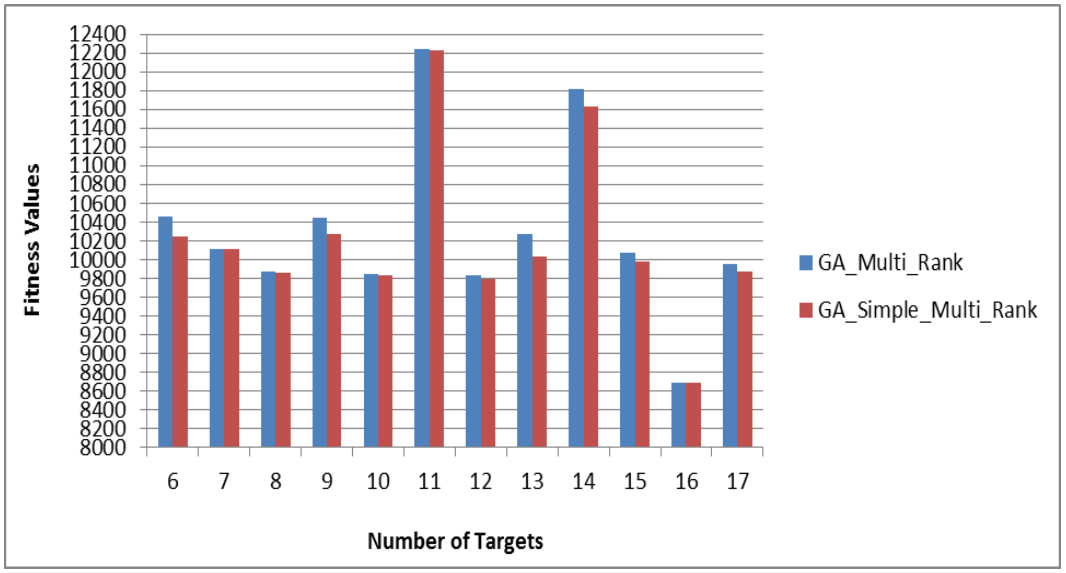

Figure 4.22: Fitness Values of Genetic Algorithms for Input Subset 3

According to graphical representations of test results given in Figure 4.21 and Figure 4.22, for input subset 3, average elapsed time values of both genetic algorithms are very close to each other again. However, GA_Multi_Rank algorithm obviously provides better fitness values for 4 of 12 scenarios for this time. Also, there exist extra 6 scenarios in which performance difference happen so slightly.

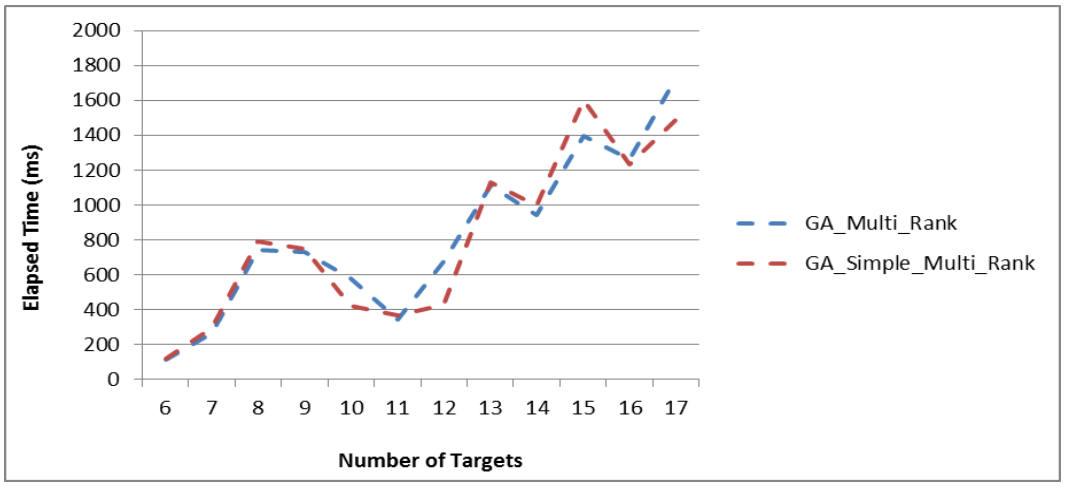

Figure 4.23: Elapsed Time of Genetic Algorithms for Input Subset 4

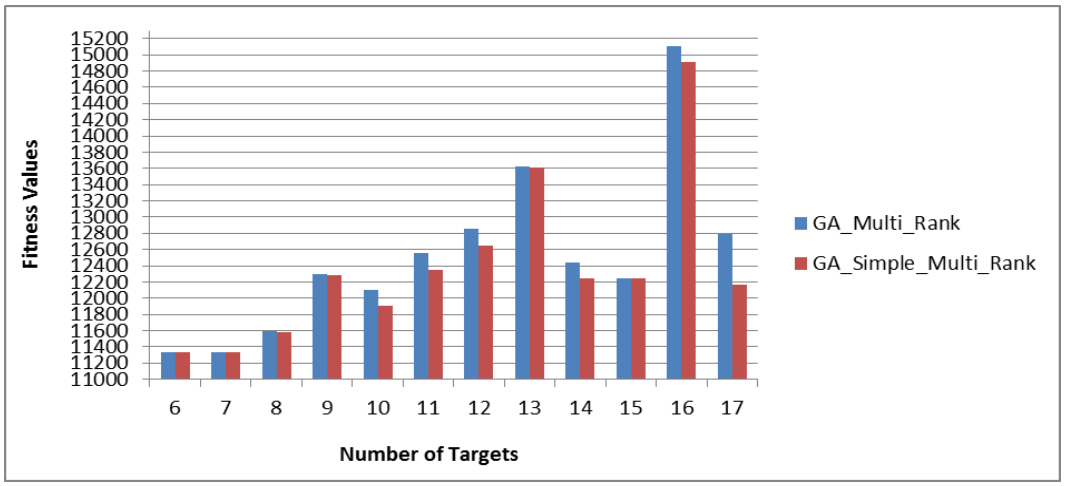

Figure 4.24: Fitness Values of Genetic Algorithms for Input Subset 4

For input subset 4, comparison of the algorithms occurs similar to the previous one, occurred for large sized and randomly composed subset, as can be seen in Figure 4.23 and Figure 4.24. Again very close performances are available for efficiency aspect, but GA_Multi_Rank algorithm outperforms for some specific scenarios of this input subset in terms of effectivity.

According to all fitness based test results for 4 input subsets, it may be concluded that the two – level crossover operation is more effective than the one – level one for this problem, especially for large sized input subsets regardless of composition way.

4.2.4.2 Objective Based Comparison

Objective based comparison of the genetic algorithms is provided in this section. The algorithms, GA_Multi_Rank and GA_Simple_Multi_Rank, are both tested with already composed large sized input set and its randomly and consciously composed subsets. For this time, gathered fitness results are not graphically visualized in a comparative manner but presented in tabular form.

Objective based performances of the algorithms for large sized randomly and consciously composed subsets can be seen in Figure 4.25 and Figure 4.26 respectively. With the aim of providing objective based comparison, total distance covered by all UAVs, total number of UAVs used for the mission and total number of targets served during the mission are given regarding to first, second and third objectives respectively.

Scenario		GA_Multi_Rank			GA_Simple_Multi_Rank		
UAVs	Targets	Distance	Used UAVs	Targets	Distance	Used UAVs	Targets
20	6	1339	12	6	1457	13	6
20	7	1584	8	6	1584	8	6
20	8	2229	9	7	2247	9	7
20	9	2450	16	9	2530	17	9
20	10	2352	18	9	2368	18	9
20	11	1962	13	11	1971	13	11
20	12	2373	18	10	2404	18	10
20	13	2433	18	11	2563	19	11
20	14	2481	17	13	2574	18	13
20	15	1923	15	11	1923	16	11
20	16	2618	17	11	2618	17	11
20	17	2644	19	13	2627	20	13
Average:		2199	15	9,75	2239	15.5	9,75

Figure 4.25: Objective Based Test Results of Genetic Algorithms for Subset 3

As can be seen in Figure 4.25, for randomly composed subset, there is no difference for performances of the genetic algorithms in target coverage aspect. But, for distance coverage issue, effectivity of GA_Multi_Rank algorithm is better than effectivity of GA_Simple_Multi_Rank algorithm; that the difference is about %1.8 on the

average and about %8.1 at maximum. For the aspect of UAV usage, again GA_Multi_Rank algorithm outperforms GA_Simple_Multi_Rank algorithm about %3.2 on the average and about %7.7 at maximum. As the result, it can be concluded for this input subset that GA_Multi_Rank algorithm provides more effective results compared to ones of the GA_Simple_Multi_Rank algorithm on the average for two of the three objectives.

Scenario		GA_Multi_Rank			GA_Simple_Multi_Rank		
UAVs	Targets	Distance	Used UAVs	Targets	Distance	Used UAVs	Targets
20	6	864	8	6	864	8	6
20	7	1171	10	7	1171	10	7
20	8	1311	11	8	1316	11	8
20	9	1110	11	9	1114	11	9
20	10	1701	12	10	1700	14	10
20	11	1642	13	11	1746	14	11
20	12	1746	14	12	1850	15	12
20	13	1683	12	13	1584	13	13
20	14	2056	15	13	2156	16	13
20	15	2358	19	14	2358	19	14
20	16	1688	12	16	1788	13	16
20	17	2704	20	16	2534	18	15
Average:		1670	13.1	11,25	1682	13.5	11.17

Figure 4.26: Objective Based Test Results of Genetic Algorithms for Subset 4

As can be seen in Figure 4.26, for consciously composed subset, there is only one scenario posing difference in target coverage aspect; that GA_Multi_Rank algorithm covers one more target compared to GA_Simple_Multi_Rank algorithm in that scenario. However, for distance coverage issue, effectivity of GA_Multi_Rank algorithm is better than effectivity of GA_Simple_Multi_Rank algorithm; that the difference is about %0.7 on the average and about %6 at maximum. For the aspect of UAV usage, again GA_Multi_Rank algorithm outperforms GA_Simple_Multi_Rank algorithm for about %3 on the average and about %14.3 at maximum. As the result, it can be concluded for this input subset that the GA_Multi_Rank algorithm provides more effective results compared to ones of the GA_Simple_Multi_Rank algorithm on the average for all of the three objectives.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusions

In this thesis, we analyzed UAV mission planning problem especially for UAV – target assignment and scheduling aspects. We formulated the problem as a combinatorial optimization problem and served it in a generic way that all available algorithms can be easily designed for.

Genetic, greedy and exhaustive search algorithms are designed and implemented for this problem. Additionally, some customizations are made to these implementations regarding to specific features coming from problem domain. A graphical framework, easing scenario management and allowing a generic testing procedure regardless of which algorithm to be tested, is also developed in the context of this study. Each implemented algorithm is tested in both of effectivity and efficiency aspects and its performance is compared to the rest ones'.

Two main input sets and subsets posing different problem domain related features are composed for evaluation of all implemented algorithms and their possible variations. Finally, resulting comparable values, such as fitness and elapsed time, are tabulated for each of 4 input subsets. Also, for large sized input set, objective based performances of the algorithms are also given for all of the three objectives in tabular form.

Exhaustive search algorithm gives the best solutions for all kind of scenarios in terms of solution quality. However, the algorithm provides the worst efficiency, especially for operational environments including more than a few infeasible targets, when compared to efficiency of the other algorithms. Because of this huge running time of the algorithm, it is not preferable to use especially for real time problem domain. Also, regardless of real time requirement of the problems, it is also not practical for some problems, waiting decisions not in real time but in acceptable time, like this UAV mission planning problem.

Greedy algorithm gives the most efficient solutions requiring the least elapsed time for execution; but effectivity of this algorithm is low compared to the other algorithms. Also, the solutions tend to diverge more from optimality, as complexity of operational environment increases. So this algorithm is not also suitable enough for UAV mission planning problem.

After comparison of all experimental data, we can conclude that genetic algorithm is practically the best algorithm for our problem by keeping both effectivity and efficiency criterions in acceptable interval. Solutions of genetic algorithms are produced nearly as quickly as greedy algorithm (smaller than 2.2 seconds for all input scenarios) for all customizations of the problem. Also fitness values are not much worse than the ones produced by exhaustive search algorithm and they differs less than %1 at maximum. In addition, objective based evaluation, which is realized only for large sized input set, shows us superiority of genetic algorithm variations to greedy algorithm in terms of effectivity, especially for the first and second objectives of the problem.

Finally, we can say that the genetic algorithm GA_Multi_Rank, which is utilizing UAV and Target Level Crossover techniques together with two point scheme and using rank selection operator, stands as the most preferable algorithm for this assignment and scheduling problem. This algorithm finds good quality solutions in preferable and acceptable levels with respect to requirements emerging from nature of the UAV mission planning problem.

5.2 Future Work

In this thesis, three main objectives are cared about and gain related to these objectives are tried to be maximized. It is possible to increase number of objectives related to problem domain such as priority of targets regarding to configurations of operational environments.

It would be more realistic to think operational environment in a dynamic fashion instead of static which is done in this thesis. Because, it is very common to face with unexpected factors for this type of missions. Accordingly, it would be a good practice to care about extra targets popping up in operational time and design algorithm to react operational time changes.

This study has assumed availability of homogenous UAVs for each implemented algorithm but it would be more realistic to be cared about UAV types and managing them according to hard and soft features associated with them. In this context, it can also be possible to divide the mission into three main stages which are classification, attack and verification as stated in [4] and manage accordingly.

It can also be a good practice to modify 2D graphical framework to be able to make it working in 3D fashion. Then, it would become more useful and realistic for the users, especially in simulation phase to see what would happen during scenarios.

REFERENCES

[1] J. S. Bellingham, M. J. Tillerson, A. G. Richards and J. P. How, "Multi-Task Assignment and Path Planning for Cooperating UAVs," Conference on Cooperative Control and Optimization, Nov. 2001.

[2] L. F. Bertuccelli, M. Alighanbari and J. P. How, "Robust Planning for Coupled Cooperative UAV Missions," 43rd IEEE Conference on Decision and Control Dec. 14 – 17, 2004, Vol. 3, pp. 2917 – 2922.

[3] Mehdi Alighanbari and Jonathan P. How, "Decentralized Task assignment for Unmanned Aerial Vehicles," 44th IEEE Conference on Decision and Control, and the European Control Conference Dec. 12 – 15, 2005, pp. 5668 – 5673.

[4] S. J. Rasmussen, T. Shima, J. W. Mitchell, A. G. Sparks and P. Chandler, "State-Space Search for Improved Autonomous UAVs Assignment Algorithm," 43rd IEEE Conference on Decision and Control Dec. 14 – 17, 2004, Vol. 3, pp. 2911 – 2916.

[5] Adam J. Pohl and Gary B. Lamont, "Multi-Objective UAV Mission Planning Using Evolutionary Computation," Proceedings of the 2008 Winter Simulation Conference Dec. 7 – 10, 2008, pp. 1268 – 1279.

[6] Mehdi Alighanbari, Luca F. Bertuccelli and Jonathan P. How, "A Robust Approach to the UAV Task Assignment Problem," Proceedings of the 45th IEEE Conference on Decision and Control Dec. 13 – 15, 2006, pp. 5935 – 5940.

[7] David Rathbun, Sean Kragelund, Anawat Pongpunwattana and Brian Capozzi, "An Evolution Based Path Planning Algorithm for Autonomous Motion of a UAV through Uncertain Environments," Proceedings of the 21st IEEE Digital Avionics System Conference 2006, Vol. 2, 8D2-1 - 8D2-12.

[8] Duc-Cuong Dang, RachaEl-Hajj and Aziz Moukrim, "A Branch-and-Cut Algorithm for Solving the Team Orienteering Problem," C. Gomes and M. Sellmann (Eds.): CPAIOR 2013, LNCS 7874, pp. 332 – 339, 2013.

[9] Domenico Pascarella, Salvatore Venticinque and Rocco Aversa, "Agent Based Design for UAV Mission Planning," 2013 Eighth International Conference on P2P, Paralleled, Grid, Cloud and Internet Computing Oct. 28 – 30, 2013, pp. 76 – 83.

[10] Stephen Leary, Markus Deittert and John Bookless, "Constrained UAV Mission Planning: A Comparison of Approaches," 2011 IEEE International Conference on Computer Vision Workshops Nov. 6 – 13, 2011, pp. 2002 – 2009.

[11] Pieter Vansteenwegen, WouterSouffriau, Greet Vanden Berghe and Dirk Van Oudheusden, "Iterated local search for the team orienteering problem with time windows," Computers and Operations Research Dec. 2009, Vol. 36, Issue 12, pp. 3281 – 3290.

[12] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe and Dirk Van Oudheusden, "A guided local search metaheuristic for the team orienteering problem," European Journal of Operational Research July 2009, Vol. 196, Issue 1, pp. 118 – 127.

[13] Tal Shima, Steven J. Rasmussen and Andrew G. Sparks, "UAV Cooperative Multiple Task Assignments using Genetic Algorithms," 2005 American Control Conference Portland, OR, USA, June 8 – 10, 2005, Vol. 5, pp. 2989 – 2994.

[14] Yi Wei, M. Brian Blake and Gregory R. Madey, "An Operation-time Simulation Framework for UAV Swarm Configuration and Mission Planning," 2013 International Conference on Computational Science, 2013, Vol. 18, pp. 1949 – 1958.

[15] Steven J. Rasmussen and Tal Shima, "Branch and Bound Tree Search for Assigning Cooperating UAVs to Multiple Tasks," Proceedings of the 2006 American Control Conference Minneapolis, Minnesota, USA, June 14 – 16, 2006.

[16] Christoph Rasche, Claudius Stern, Willi Richert, Lisa Kleinjohann and Bernd Kleinjohann, "Combining Autonomous Exploration, Goal-Oriented Coordination and Task Allocation in Multi-UAV Scenarios," 2010 Sixth International Conference on Autonomic and Autonomous Systems, ICAS 2010, Cancun, Mexico, March 7 – 13, 2010.

[17] Phillip R. Chandler, Meir Pachter, Dharba Swaroop and Jeffrey M. Fowler, "Complexity in UAV Cooperative Control," Proceedings of the American Control Conference Anchorage, AK May 8 – 10, 2002, Vol. 3, pp. 1831 – 1836.

[18] Ibrahim H. Osman, "Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches," OR Spektrum 1995, Vol. 17, Issue 4, pp. 211 – 225.

[19] L. A. Ingham, "Considerations for a roadmap for the operations of Unmanned Aerial Vehicles (UAV) in South African airspace," PhD thesis in Stellenbosch University, 2008.

[20] Bahram Alidaee, Haibo Wang and Frank Landram, "On the Flexible Demand Assignment Problems: Case of Unmanned Aerial Vehicles," IEEE Transactions on Automation Science and Engineering Oct. 2011, Vol. 8, Issue 4, pp. 865 – 868.

[21] A. Schrijver, "A course in combinatorial optimization," CWI and University of Amsterdam, 2006.

[22] Matthias Kühn, Thomas Severin, Horst Salzwedel, "Variable Mutation Rate at Genetic Algorithms: Introduction of Chromosome Fitness in Connection with Multi – Chromosome Representation," International Journal of Computer Applications (0975 – 8887) June 2013, Vol. 72, No. 17.

[23] P. Victer Paul, P. Dhavachelvan, R. Baskaran, "A Novel Population Initialization Technique for Genetic Algorithm," 2013 International Conference on Circuits, Power and Computing Technologies March 20 – 21, 2013, pp. 1235 – 1238.

[24] Rakesh Kumar, Sudhir Narula, Rajesh Kumar, "A Population Initialization Method by Memetic Algorithm," International Journal of Advanced Research in Computer Science and Software Engineering April 2013, Vol. 3, Issue 4.

[25] Shahryar Rahnamayan, Hamid R. Tizhoosh, Magdy M. A. Salama, "A Novel Population Initialization Method for Accelerating Evolutionary Algorithms," Computers and Mathematics with Applications May 2007, Vol. 53, Issue 10, pp. 1605 – 1614.

[26] Beatrice Ombuki, Brian J. Ross, Franklin Hanshar, "Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows," Applied Intelligence 2006, Vol. 24, pp. 17 – 30.

[27] Marius M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," Operations Research 1987, Vol. 35, pp. 254 – 265.

[28] Jorge Mendes, "A Comparative Study of Crossover Operators for Genetic Algorithms to Solve the Job Scheduling Problem," WSEAS Transactions on Computers April 2013, Vol. 12, Issue 4, pp. 164 – 173.