Faculty of Engineering and Material Science
Mechatronics Engineering Department
German University in Cairo

A Metaheuristic Approach to Multi-robot Task Allocation

Master Thesis

Author:	Mohamed Yehia Mohamed Saleh Badreldin
Supervisor:	Assoc. Prof. Alaa Khamis
Submission Date:	04 May, 2013

This is to certify that:

(i) the thesis comprises only my original work toward the Masters Degree

(ii) due acknowledgment has been made in the text to all other material used

Mohamed Yehia Mohamed Saleh Badreldin
04 May, 2013

Acknowledgment

First and foremost I offer my sincere gratitude to my supervisor, Professor Alaa Khamis, who has been supportive throughout my thesis with his patience, understanding and knowledge while allowing me enough room to work in my own way. This Master work would not have been possible without his vision and vast skills in many areas of research. I acknowledge that without his encouragement, efforts and constructive guidance this thesis would not have been written or completed. Professor Alaa was not only a supervisor but a mentor and a bigger brother. I will always appreciate what I have learned from him as my lecturer and as my supervisor. I am deeply indebted to the reviewers committee Prof. Dr. Hazem Abbas and Dr. Rabie Ramadan for their time and effort in reviewing this work.

A very special thanks goes out to Professor Imam Morgan the head of the Mechatronics Engineering Department, without his motivation, encouragement and enduring support throughout my studying and working period at the German University in Cairo I would not have the vision or interest in research. He always provided us, his students with direction and support and became more of a father than a professor.

My sincere thanks go to my friends, colleagues and the members of the Robotics and Autonomous Systems (RAS) research group Eng. Ahmed Hussein, Eng. Omar Mahmoud, Eng. Ahmed Wagdy, Eng. Fady Magdy, Eng. Ahmed Adel and Eng. Heba Ali for their support, encouragement and valuable advice. I would also like to thank the German University in Cairo for offering me the opportunity to conduct my master thesis.

Above all, I would like to thank God for everything I have and everything I am granted.

Dedication

To the woman who sacrificed all her life to gift me my life
My Mother
To the man who always dreamed of seeing me better
My Father
To the girl who never hesitated to support me in what I believed in
My Fiance and My Future Wife
To the mentor who taught me how to think, work, act and write professionally
Dr. Sherif Zaidan

Abstract

Multi-robot System (MRS) have recently become a topic of interest in the field of robotics research since they can be utilized in many diverse areas of application. MRS have the potential of replacing current available systems and approaches through providing more safe, more efficient and generally enhanced overall system performance. The fact that MRS applications are complex and therefore must be divided into a set of subtasks necessitates the incorporation of a task allocation approach to the MRS that will be responsible of assigning the robots to the available tasks to be executed.

Task allocation is one of the most important research problems in the field of MRS which has been proven to be complex and of extended difficulty. Over the last decade various task allocation approaches have been developed for multi-agent systems as well as for MRS and these approaches were successful in various application domains of MRS. However some limitations were reported in the previously proposed approaches, such as lacking general applicability, unsuccessful with large scale applications, unable to handle extended constrained problems and finally not suitable for dynamic task allocation applications.

In this thesis, a metaheuristic optimization-based approach was proposed for solving the task allocation problem in MRS. Simulated annealing (SA) and genetic algorithm (GA) were the metaheuristic optimization algorithms implemented. The proposed approach addresses general task allocation problems that are heavily constrained, with large number of heterogeneous robots and heterogeneous time-extended tasks to be allocated. Both algorithms were extensively tested over different test scenarios. Finally a comparative study was carried out between the proposed metaheuristic approaches and a previously implemented market-based approach. The experimental results show that the metaheuristic approach outperforms the market-based approach.

Contents

List of Figures

List of Tables

Acronyms

AEP	ALLIANCE Efficiency Problem
GA	Genetic Algorithm
MRTA	Multi-robot Task Allocation
MRS	Multi-robot System
mTSP	Multiple Traveling Salesman Problem
SA	Simulated Annealing
TSP	Traveling Salesman Problem

Chapter 1

Introduction

1.1 Motivation

In the last few years, the field of research in mobile robotics has encountered a significant shift as the researchers in this field have recently started focusing on MRS rather than single robot systems. This increased interest in the community of mobile robotics research towards MRS comes from the significant advantages and higher potential provided by MRS than single robot systems. MRS can be simply understood to be a group of robots cooperating together for accomplishing a certain task or mission. From this simple definition, one can infer the importance, higher potential and extended capabilities for a robot team working together rather than a single robot working alone. The advantages of a robot team are many, some examples of these advantages include but are not limited to resolving task complexity, increased system reliability, increased system performance and finally easier and simpler design [2]. MRS have a high potential to be used in various applications either replacing single robot systems thus increasing the performance of the whole system or either to be used in applications where one robot will fail in executing the required tasks. Some examples of MRS applications are autonomous mobile surveillance systems, search and rescue operations, humanitarian demining, exploration of abandoned areas or other planets and finally cooperative manipulation or transportation of objects [3].

Recently the complexity of MRS has severely increased. This increased complexity has two main reasons, the increase of problem complexity and the increase of multi robot team complexity. First the complexity of tackled problems of MRS has strongly increased as the hopes and objectives of researchers in MRS have increased and no longer limited to two robots observing a moving target or cooperatively pushing a box together. Nowadays the expectations of MRS include larger number of robots working cooperatively on solving complicated tasks over an extended period of time. Second the complexity of multi robot teams also increased to cope with the requirements of the tackled problems. Now multi robot teams have a larger number heterogeneous robots with different and various capabilities [4].

In conclusion, researchers are now focusing on MRS due to their importance and higher potential. One of the main areas of research in this field is the task allocation problem in MRS, where the mapping of robots to tasks is done in order to increase the overall performance of the system. The task allocation problem is a major issue in MRS, it is a problem concerning utilizing of the available resource. In MRS, the available resources are the robots which are used to solve a problem or to perform a certain task. Thus in order to increase the performance of the system, one must efficiently utilizes the available robots in order to solve the required tasks. Since the decision of which robot will do which task has a significant effect on the performance of the system, therefore the allocation of the tasks to the proper robots strongly affects the performance of the system [5]. The task allocation problem is proved to be one of the toughest problem especially when it comes to complex robot teams that are required to solve and execute complex problems and tasks. Therefore it was of a great importance to search for, develop and propose an efficient approach to solve the MRTA problem.

1.2 Objective

The main objective of this thesis is to propose a generic approach to solve the MRTA problem and to present a framework that is efficiently capable of modeling any instance of the MRTA problem and to provide a solution to this model which is a suitable solution to the given problem. The framework will generate a solution in which the given robots are efficiently allocated to the given tasks in such a way to maximize the overall performance and minimize the total cost of the allocation. The framework will take into consideration the real world constraints of the system, the requirements of the tasks and the capabilities of the robots.

As previously mentioned in 1.1 that the complexity of the MRS has strongly increased. Therefore it was one of the main objectives of this thesis to propose an approach that is capable of handling complex MRTA problems that include a large number of heterogeneous tasks with different requirements. Another objective is to propose a scalable approach that is capable of generating solutions to instances of the MRTA problem that includes a large number of robots and tasks.

1.3 Thesis Structure

The remainder of this thesis is structured as follows:

- **Chapter 2** gives a general overview of MRS, their significance and various range of applications. The MRTA problem is also presented in this chapter highlighting the previously proposed problem formulations and solving techniques.
- **Chapter 3** introduces a general classification of optimization approaches, discussing in detail the metaheuristic optimization-based approach and the motivations of using this approach. In this chapter also a classification of metaheuristic algorithms is classified with a brief information about some of the well-known algorithms.

- **Chapter 4** presents in detail the formulation followed for the MRTA problem as well as the proposed metaheuristic optimization-based approach used to solve the problem.
- **Chapter 5** presents the experiment setup and the experimental results for testing the developed algorithms. Also a comparative study between the proposed approaches and a market-based approach is conduced at the end of this chapter.
- **Chapter 6** summarizes the work carried out in this thesis with conclusion and future work recommendations.

Chapter 2

Multi-robot Task Allocation

In this chapter, the task allocation problem in MRS is introduced. The main aim of this chapter is to give an overview of MRS and the task allocation problem in such systems. Further information about MRS is discussed in 2.1 followed by introducing the task allocation problem in MRS 2.2. Finally a conclusion about this chapter is given in 2.3.

2.1 Multi-robot Systems (MRS)

2.1.1 Definition

Although several definitions have been proposed in the literature for MRS, however no formal definition can be given. This lack of formal definition is mainly caused due to the fact that the previously proposed MRS systems differ in many aspects. Such as the number of robots used, the nature of robots, their level of capabilities and the type of tasks they are performing. On the other hand, the different MRS previously proposed have a main similarity which is that these systems include a number of robots working together to reach a common goal. Therefore, MRS can be defined as a group of physical robots that work together in order to perform some collective behavior through the interaction between the robots and themselves, and the interactions between the robots and the environment [6].

2.1.2 Why MRS?

As previously mentioned, recently there has been an extensive activity in the field of multi-robot research because of the advantages and high potentials of such systems. These advantages are, but not limited to, the following [7] [8]:

1. **Resolving task complexity:** resolving the complexity of the task can be done through breaking down the main task that was to be executed with a single robot into several subtasks, and then assigning each agent of the MRS a certain subtask.

This breaking down of the main task can strongly help in resolving the complexity of the task, as all or some of the agents according to their availability can cooperate on working on a certain subtask, which will decrease the degree of complexity of this task.

2. **Increasing the performance:** MRS have higher potential of enhancing the overall performance of the system. For example MRS have higher ability to cover more area while navigating or exploring in a certain environment which is an important advantage in many situations such as search and rescue situations. Another enhancement is in terms of time required to finish the task, since some agents can work in parallel when executing sub-tasks that are not dependent on each other, which can strongly decrease the amount of time required to finish the whole task.

3. **Enhancing reliability:** in systems at which only one robot is responsible for executing the task, failure of this robot means failure of the whole system or a full stoppage of the task execution till fixation or replacement of the robot. This significant drawback can be avoided through the use of MRS, at which the failure of one of the agents will not cause total failure of the system. Also the role of the agent which failed can be fully or partially compensated by other agents, which will also avoid the system expected delays of task execution due to this failure. All of these capabilities will increase the robustness and the reliability of the system.

4. **Simpler design:** in MRS the main tasks will be subdivided into smaller tasks that will be executed by multi-robots instead of a single robot. The capabilities of these multi-robots will certainly be less than the capability of the single robot and thus the design of these multi-robots will be simpler, smaller and cheaper than the single powerful robot.

2.2 Task Allocation in MRS

Over the last few years, several MRS problems were addressed, and since the MRS have a higher potential of solving complex and sophisticated problems, the complexity of the addressed problem increased with time. Therefore the complexity of the MRS had to increase to accommodate the complexity of the addressed problems. As more research was done on MRS, the researchers have encountered the question "Which robot will execute which task?" In order to answer this question, more focus was directed towards the task allocation problem in MRS.

2.2.1 MRTA Problem Definition

The MRTA problem is a problem that arises in MRS where a number of robots are working together in the aim of achieving a common goal or task which is sub divided into a number of subtasks. The problem can be defined as follows:

1. Given a set of available robots, $R = \{\mathrm{R}_1, \mathrm{R}_2, \ldots\ \mathrm{R}_\mathrm{n}\}$

2. Given a set of available tasks, $T = \{T_1, T_2, \ldots\ T_m\}$
3. Allocation of the available tasks to the robots occurs, $A : T \to R$
4. The output set S is the optimal allocation of the tasks to the available robots

$$S = \{\ (\ R_1\ T_1\ )\ (\ R_2\ T_2\ )\ \ldots\ (\ R_k\ T_k\ )\ \}\ \text{for}\ 1 \leq k \leq m \tag{2.1}$$

5. This allocation S minimizes or maximizes a certain objective function in order to get the best performance of the system

The increased complexity of MRS and the increased complexity of the addressed problems have severely affected the MRTA problem. This increased complexity of the MRTA comes from different sources. One of the main sources of complexity is the increase of the number of used robots and tasks, the heterogeneous nature of used robots and tasks and other difficult constraints and variations that surround the problem. This increased complexity of the targeted systems had made the MRTA a very complex and difficult problem to solve. Therefore, in order to propose an approach that is adequately capable of providing optimal solutions to different instances of the MRTA problem, one must have a deep understanding of the different aspects of this problem. In Figure 2.1 a taxonomy of the MRTA problem is introduced. This taxonomy is a an attempt to give an overview and a better understanding of the different aspects of the MRTA problem where the details of this taxonomy is explained in the following sections.

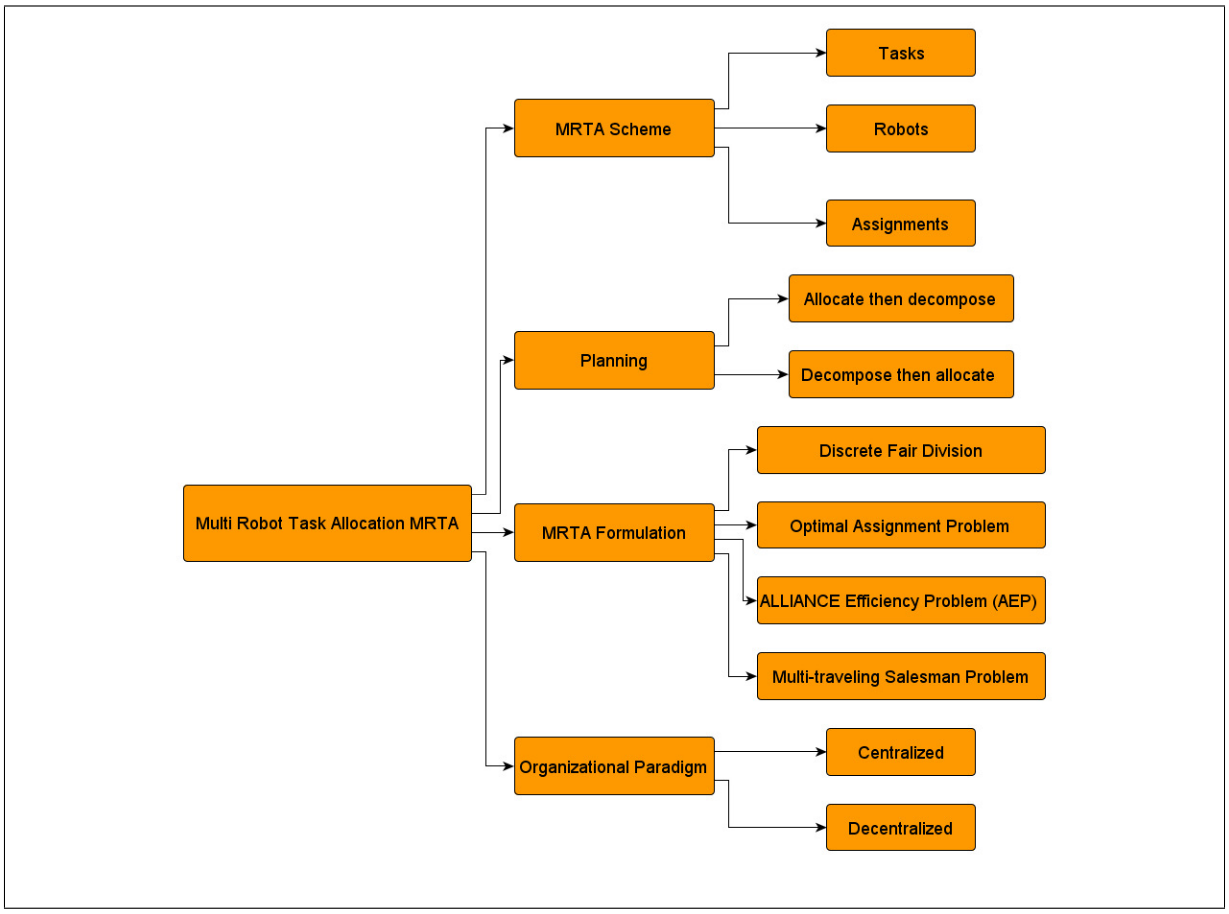


Figure 2.1: MRTA Taxonomy [1]

2.2.2 MRTA Scheme

For the purpose of solving the MRTA problem, the problem must first be classified. In [9] a formal taxonomy of MRTA was introduced, in this taxonomy the MRTA problem can be generally classified according to the following:

- The type of robots
- The type of tasks
- The type of task assignment

First the type of robots used in the systems, in this taxonomy the robots are classified either as Single-Task robots (ST) or as Multi-Task robots (MT). ST robots are the type of robots which are capable of performing a single task at a time. On the other hand MT robots are the type of robots that are capable of executing or doing progress on more than one task at the same time. In the literature, most of the proposed MRS include ST robots due to the lack of robots that are capable of performing more than one task simultaneously.

Second the type of tasks to be allocated, in the proposed taxonomy the tasks are classified either as Single-Robot tasks (SR) or Multi-Robot tasks (MR). SR tasks can be executed by one robot only, while MR tasks are the tasks that require simultaneous cooperation of more than one robot in order to be able to execute such kind of tasks.

Third the type of task assignment, there are two types of task assignment either an Instantaneous Assignment (IA) or Time-extended Assignment (TA). The IA is the type of assignment that tries to optimally allocate the robots and the tasks currently available with no consideration to future allocations. TA assignment aims for the optimal allocation for all of the available tasks taking into consideration the relationship between the tasks and their time requirements and thus it is more suitable for future allocations if at the current allocation no information about all the tasks is available [2].

In this thesis, the work introduced focuses on developing a task allocation approach capable of handling single-task robots, single-robot tasks and both instantaneous and time-extended assignments.

2.2.3 Planning

As previously explained in 2.1.2 that in MRS, the complexity of the task can be resolved through dividing the complex task into smaller subtasks which will be then executed by the available team of robots after the allocation of each task to the most suitable robot. In real world applications such as search and rescue applications, the mission of the team of robots will usually include executing more than one complex task where each complex task will be divided into a set of subtasks. In this step an important question must be answered, which tasks will be allocated to the robots whether the complex tasks or the subtasks (simple tasks)?

Before deciding which strategy will be used to solve the MRTA problem, one must understand the difference between complex task allocation and simple task allocation. Simple and complex task allocation can be defined respectively as follows:

Simple Task Allocation: Given a set of mobile robots R each looking for one task, and a set of tasks T each requires one mobile robot. The simple task allocation can be defined as mapping each task to a mobile robot in order to be executed.

Complex Task Allocation: Given a set of mobile robots R and a set of tasks T. Let $G \in T$ be a group of tasks that is decomposable into other subtasks $K \in G$. The complex task allocation can be defined by as mapping each subtask to a mobile robot to be responsible of completing it [10].

And thus in this phase a decision must be made between one of the two allocation strategies:

- Decompose then allocate
- Allocate then decompose

First decompose then allocate strategy [11], in this strategy all of the main tasks are decomposed into a number of subtasks and thus all the information about the subtasks is considered as a global information, or in other words the subtasks are now accessible to all the members of the robot team. Then the allocation phase starts by searching for the optimal allocation that optimizes the objective function through allocating all of the subtasks to the team of robots. In this case, the optimal allocation might include a robot that is assigned to different subtasks from different main tasks as in Figure 2.2. The main advantage of such a strategy is that it is more likely to find more optimal solution using this strategy than the allocate then decompose strategy, since all tasks information is available globally and thus the allocation process occurs depending on a more detailed information and thus can give better results.

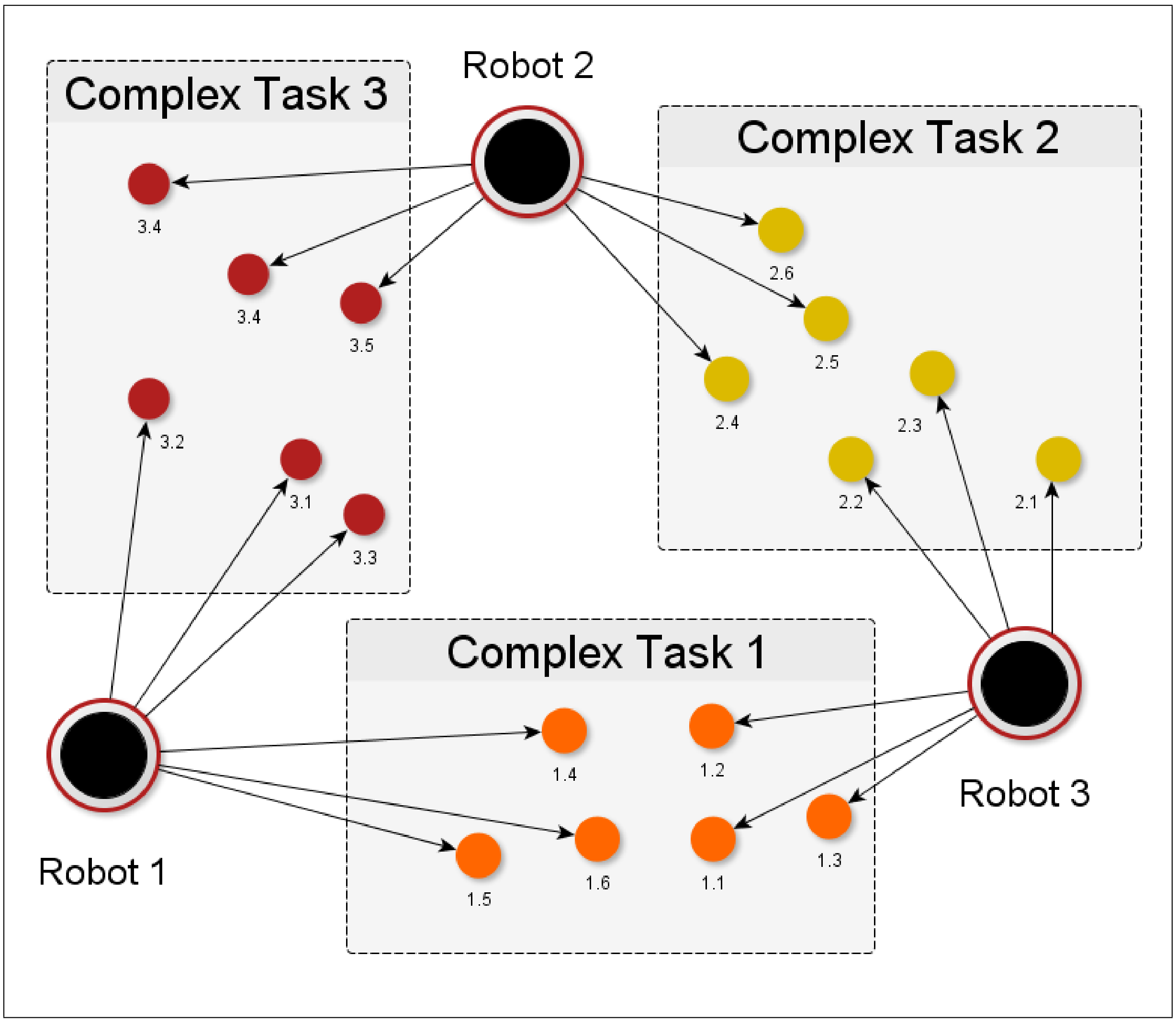


Figure 2.2: Decompose Then Allocate

Second allocate then decompose [12], in this strategy the main tasks are first allocated to the different robots of the robot team depending upon the requirements of the tasks and the capabilities of the robots in order to find the optimal allocation that optimize the system performance as in Figure 2.3. The second step is to decompose the main tasks that have been previously assigned to the robots into a set of subtasks that will be executed by the assigned robots as in figure 2.3. This allocation strategy is considered a higher level strategy as it includes a high level of abstraction since the allocation process depends on the main tasks disregarding the details of the subtasks. This strategy is more suitable to be used with decentralized and hierarchical control paradigms than with centralized controllers [1].

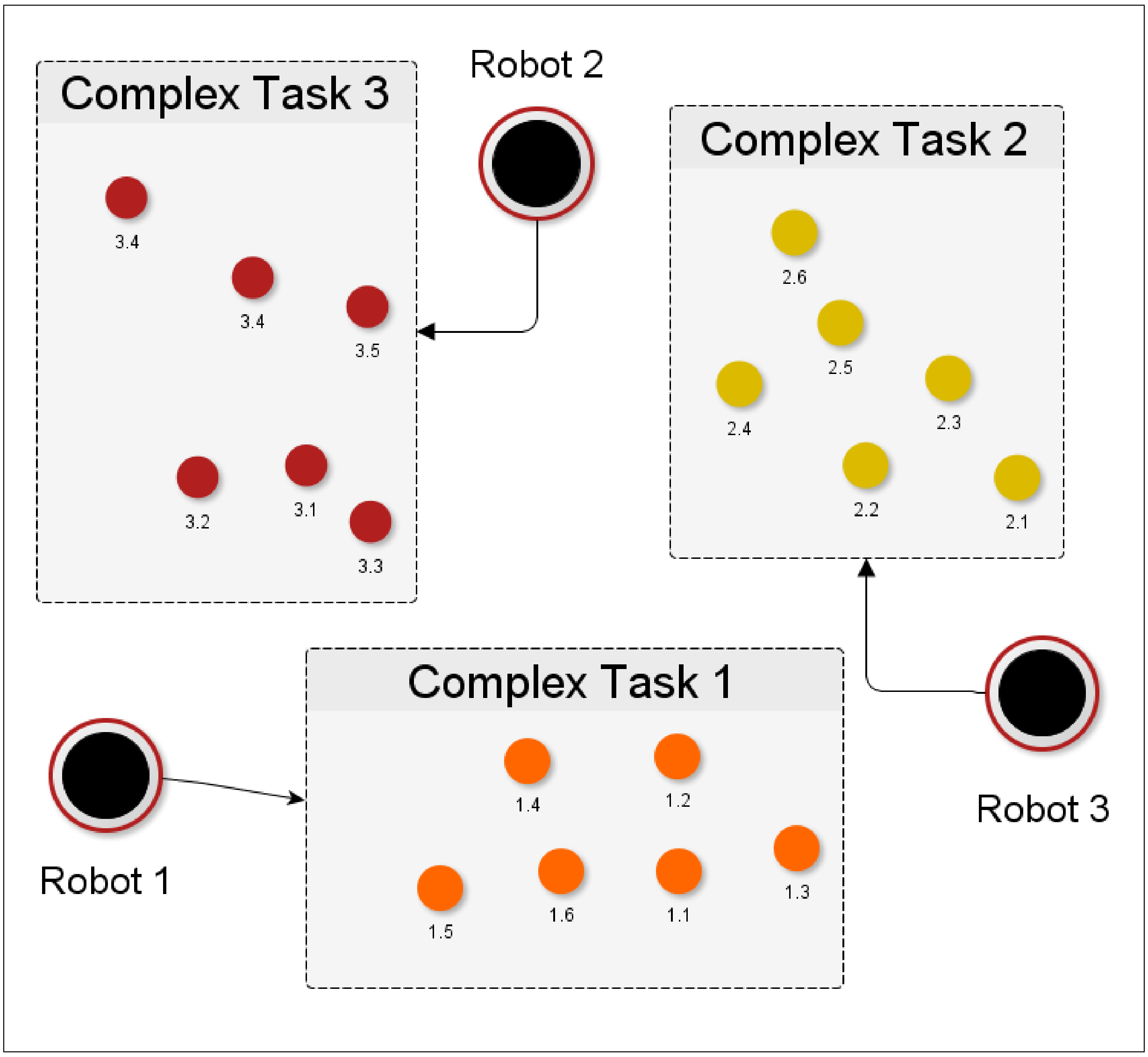


Figure 2.3: Allocate Then Decompose

The work proposed in this thesis focuses on the decompose then allocate strategy. The decomposition role is assumed to be done by a human expert.

2.2.4 MRTA Formulation

2.2.4.1 Discrete Fair Division

Fair division is a mathematical theory based on an idealization of a real life problem. The real life problem is the one of dividing goods or resources among several individuals so that each individual considers the part he or she receives to be a fair portion. It provides explicit criteria for various different types of fairness. Its aim is to provide algorithms to achieve a fair division, or prove their impossibility, and study the properties of such divisions both in theory and in real life. That is, fair division may be considered part of the larger research area of multi-agent resource allocation [13].

There are different types of methods for fair division, such as:

- **Cake-Cutting Problem:** is presented in [14] where agents arrive, take a piece of cake, and immediately depart. The cake cutting setting deals with the allocation of a single, heterogeneous divisible resources. The author suggested numerous desirable properties for cake cutting mechanisms in this setting, and showed that adaptations of classic mechanisms achieve these properties. The same problem is presented in [15], however, the cake cutting setting deals with the allocation of multiple homogenous divisible resources in dynamic environment. Moreover, the trust algorithm in cake cutting problem is proposed in [16] to fairly divide tasks to multiple-agents.

- **Method of Lone Divider:** is based on the concept of dividing independent goods into number of sets and agents choose and list all the acceptable goods and then goods are distributed fairly among all agents. Genraly for *N* agents, a random agent is selected (Divider); the divider divides the goods into *N* sets $\{S_1, S_2, ... S_N\}$, then the other agents (Choosers) independantly list all acceptable goods. The decision of distributing the goods is now based on two cases; either the Choosers select acceptable goods in the same set, then each Chooser is given acceptable goods and the Divider gets the remaining goods, or the Choosers select conflicted goods from different sets then the issue can be solved by distributing one of the conflicting goods to the Divider and recombine the remaining goods to repeat the whole procedures again. The process keeps rotating till only two agents are remaining, in this case the distribution can be made with Divider-Chooser Method [17].

- **Method of Markers:** is another fair division method for multi-agents with discrete goods. Rather than a randomly selected agent to divide the goods, all the agents get their chance to divide the goods into number of fair sets in which they find it acceptable to get any of these sets. For *N* agents, each gets *N-1* markers to be able to divide the goods into *N* sets. To be able to divide the goods easily, the goods are sorted in one single row and the agents add their markers independantly. After the proccess of setting the markers is finished, scan the row of goods from left to right and locate the first marker. The agent owning this marker goes first and gets the first set, in the case of a tie, flip a coin. Next remove all the markers of this agent. Continue scanning from left to right, and locate the first second marker. The agent owning this marker then receives the second set between his markers. Continue this process until everyone has received a a fair set and the last agent will receive the last set. Sometimes a surplus of the goods remains without being taken in any set, in this case the agents get to go in some random order and pick one of the goods at a time until all the surplus goods are given out [17].

- **Method of Sealed Bids:** is very similar to closed bid auctions, however with fair division among the agents. The procedures start with having each agent gives an honest sealed bid for the goods. Then the fair share is computed by dividing the total of the agents' bids by the number of particpating agents. The distribution of the goods starts with each item goes to the highest bidder for that item and then the agent adds or subtracts from the pot the difference between his fair share and the total value of the goods allocated to him. All the remaining values from the bids in the pot are divided evenly among all the agents. This way all agents can be considerd as buyers and sellers. Also, it is a must that each agent has enough purchasing power to enter the bidding process [18].

2.2.4.2 Optimal Assignment Problem

The assignment problem is a fundamental research topic in the field of operations research in mathematics [19]. Optimal Assignment Problem (OAP) can be defined as follows. A number of workers (m) and a number of jobs (n) are given. Each job requires one worker and each worker is searching for a job to do. A positive value is assigned for each worker providing information about the level of performance of this worker related to each one of the given jobs. Information is also provided about the priority of the jobs. The main aim is to assign the workers to the jobs in order to maximize the profit through maximizing the overall performance of the workers taking into consideration the priorities of the tasks [9]. A more elaboration on the history and variation of the assignment problem in the context of linear programming is found in [20] [21].

An obvious analogy can be found between the OAP definition and the MRTA problem definition as follows: Given a number of robots (m) each capable of performing one task at a time, and a number of tasks (n) where each task requires one robot for its fulfillment. The tasks are weighted, assigned priorities and information about the performance of the robots with respect to each task is also provided. Thus the main objective is to assign the available robots to the available tasks in order to maximize the overall performance of the system and therefore maximizing the overall profit [9]. Therefore it could be claimed that the MRTA problem could be modeled as an instance of the OAP and thus makes use of the techniques applied to solve the OAP in order to solve the MRTA problem. In [19] [22] [23] [24] [25] [26] the MRTA problem was formulated as an instance of the OAP and different solving strategies were used to find the optimal assignment of the available robots to the given tasks. Since the MRTA problem is a dynamic problem that require reassigning of the robots to the tasks according to the available information about the status of the robots and the requirements of the tasks, therefore the problem needs to be solved in an iterative way over time in order to adapt to changes in the system and to keep on enhancing the overall performance of the system [9] [27].

2.2.4.3 ALLIANCE Efficiency Problem (AEP)

Another type of problem formulation is the behavior-based mechanism, such as ALLIANCE architecture [28]. It is a software architecture that simplifies the cooperative control of teams of heterogeneous mobile robots performing complex tasks that may have ordering dependencies. This approach builds on the sub-submission architecture [29] by adding behavior sets and motivations for achieving action selection without explicit negotiations between robots.

ALLIANCE architecture gives all robots the ability to decide which task to perform based upon their current situation, with no centralized control utilized. It is a fully distributed architecture that integrates the use of mathematically modeled motivations, such as impatience and acquiescence, within each robot to complete adaptive action selection as illustrated in figure 2.4. Nevertheless, the ALLIANCE architecture offers one solution to multi-robot cooperation through allowing fault tolerant [28].

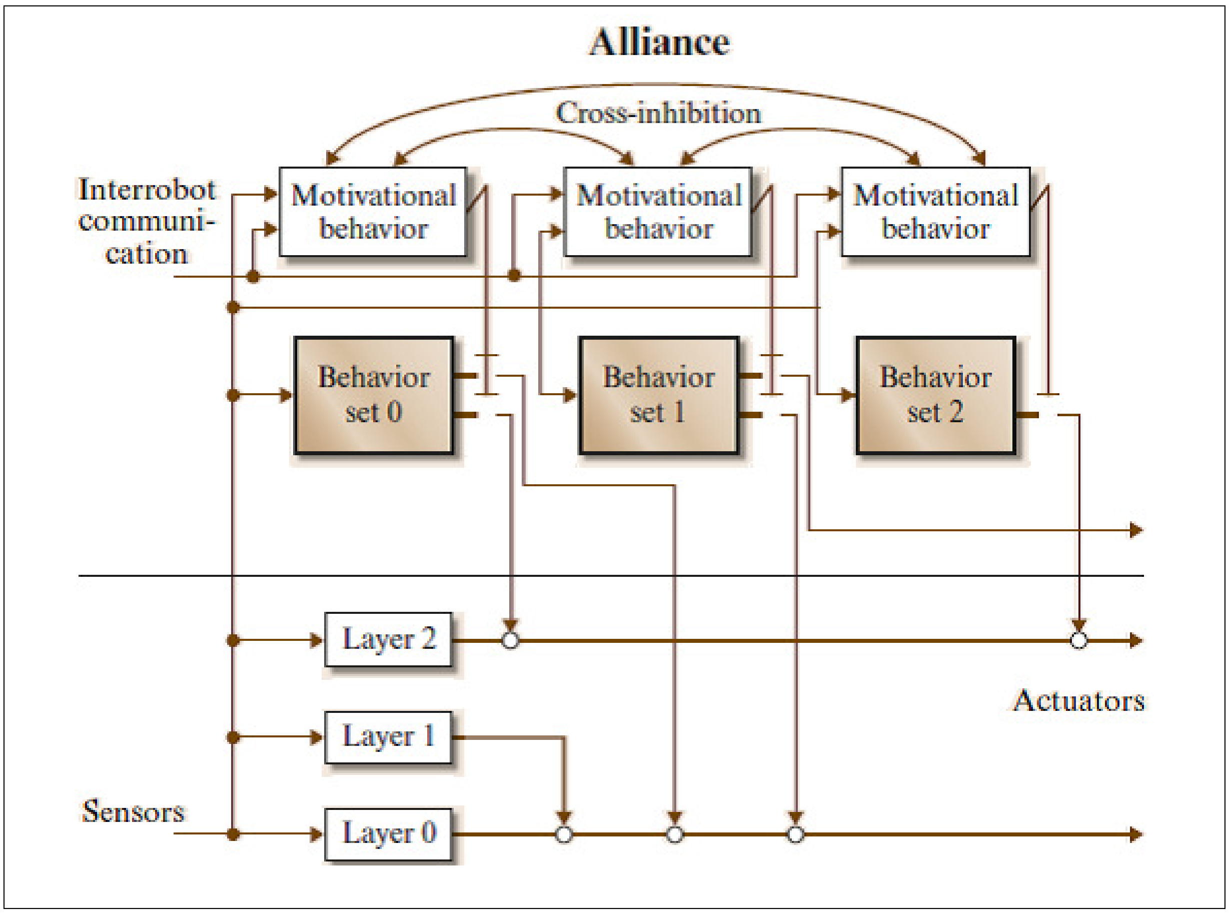


Figure 2.4: ALLIANCE Archtiecture [2]

Although, the cooperative team under ALLIANCE architecture is robust; the main weakness of this architecture is its restriction to independent subtasks. This means that complex task must be divided into subtasks that have ordering dependencies in the preconditions. It is interesting to note that with this restriction, the ALLIANCE architecture is guaranteed to allow the robot team to perform its task for a broad range of applications [28].

The MRTA can be solved as ALLIANCE Efficiency Problem (AEP), which is considered as NP-hard problem [30] [31]. In the AEP, given is a set of complex tasks and the objective is to allocate a subset of these tasks to multi-robots in such a way to minimize the overall time taken by each robot to successively perform its allocated tasks. Thus in order to solve the AEP, one must construct a schedule for the dependencies order of the subtasks.

2.2.4.4 Multiple Traveling Salesman Problem

mTSP can be considered as the generalization of the original Traveling Salesman Problem (TSP) which is used as a platform for the study of general methods that can be applied to a wide range of discrete optimization problems. In TSP n cities are given and a travelling salesman must visit these n cities and return back home, making a loop (roundtrip). He would like to travel in the most efficient way (cheapest way, fastest route or shortest route or some other criterion). Similar to the TSP, in the mTSP a number of nodes are given and the distances between these nodes are also given but the main difference in the mTSP is that instead of a single salesman, a number of salesmen are given. The salesmen are required to cover all the available nodes and return back to their starting node. A number of variations of the original mTSP was introduced in by different researchers to accommodate the mTSP to their problems. These variations included and not restricted to the following [32]:

- **Salesmen starting node:** the salesmen may all start from a single depot node and then all of them must return back to the same node or every salesman can start from a certain node, and thus each salesman must return back to his starting node
- **Number of salesmen:** the number of salesmen used in different applications varied according to the type and requirements of the application itself. In some applications the number of salesmen was dynamic such that after each iteration the number of salesman may or may not change
- **City time frame:** in some applications the task of the salesman was not only to visit the city, but also to stay in the city for a certain time in order to move to the next city
- **Fair division of salesmen:** another variation of the general mTSP is the addition of constraints that specifies the maximum number of cities or the maximum distance that can be traveled by a single salesman. This variation was used in applications that are concerned by the fair division of the available resources (salesmen)

In the literature, various researchers have used the mTSP as a solution model for the MRTA due to the strong analogy between the two problems. In [33] the authors proposed to solve the MRTA for heterogeneous robots simultaneously with the path planning problem using a general mTSP problem formulation model. Simulated annealing metaheuristic approach was proposed as the optimization technique for solving the MRTA problem. The objective function used to evaluate the performance of the system was the MinMax strategy where the role of the applied algorithm was to minimize the worst case allocation for each robot. In the simulation phase two algorithms were compared, the simulated annealing approach and a previously implemented auction based approach. In [34], the task allocation problem in MRS was addressed when the authors proposed a solution to the assignment of multiple UAV's to multiple targets in a military application scenario. The proposed solution uses Ant-colony algorithm to solve an instance of the mTSP problem which is the formulation model used for solving the addressed task allocation problem. The mTSP was also used in [35] as a formulation for the MRTA problem. The author of this thesis, proposed an auction based approach to solve the MRTA problem after formulating the problem as an instance of the mTSP. In [36] a set of heterogeneous UAVs is required to be used in surveillance mission, where a set of targets should be observed by these UAVs. The problem can be considered a MRTA problem, and can be modeled as a heterogeneous multiple depot, multiple UAV routing problem which is an another variation of the mTSP. Another approach is used to solve this problem which is to model this problem as a TSP where each target is modeled as a city to be visited only once by the traveling salesman (one of the UAVs) and then any known or novel approach can be used or developed to solve the TSP model in the aim of minimizing the traveling cost (distance) of the UAVs to cover all targets. The authors in this article proposed a transformation method to transform the mTSP into an instance of the Asymmetric TSP in order to use previously developed solving approaches to solve the Asymmetric TSP problem and thus solve the addressed MRTA problem. Also [37] used the mTSP to propose a framework to solve dynamic task allocation scheme for MRS.

The mTSP formulation is the adopted formulation and thus the MRTA problem in this thesis is solved as an instance of the mTSP.

2.2.5 MRTA Approaches

2.2.5.1 Market-based Approaches

Throughout time, humans have dealt with coordination and allocation problems for thousands of years with sophisticated market economies in which the individual pursuit of profit leads to the redistribution of resources and an efficient production of output. The principles of a market economy can be applied to multi-robot coordination [38]. Market-based multi-robot coordination approaches have received noteworthy attention within the robotics research community in recent years. They have been successfully implemented in a variety of domains ranging from mapping and exploration to robot soccer.

Market-based approaches are focused on the concept of utility functions, which can represent the ability of the agents to measure their interest in specific tasks for trading. In MRTA systems, the utility functions show how the robots skills can match the tasks requirements. A lot of market-based approaches were developed for multi-agent coordination [39] [40] [41]. Moreover, a lot of surveys are completed on the same topic [9] [38].

In general, auctions are, scalable, computationally cheap, and have reduced communication requirements. They can be performed centrally, by an auctioneer, or by the robots themselves, in a distributed way. Therefore, market-based coordination approaches have been studied in countless multi-agent systems [42] [43] [44] [45] [46].

The market-based approach is mainly based on the auctions systems. Auctions have mainly two general categories; simple auctions and combinatorial auctions. MRTA problem are solved using both categories to reach an optimal allocation and results were acceptable [47] [48] [1]. Nowaways, new architectures are being implemented as enhancement to the traditional methods, such as Murdoch [39] and Trader Bots [49] and many others [50] [51].

2.2.5.2 Optimization-based Approaches

Optimization is the branch of applied mathematics focusing on solving a certain problem in the aim of finding the optimum solution for this problem out of a set of available solutions. In other words, optimization techniques are applied in order to maximize the profit (maximization problem) or reduce the damage (minimization problem) of a certain problem. The set of available solutions is restricted by a set of constraints, and the optimum solution is chosen within these constrained solutions according to a certain criteria. These criteria define the objective function of the problem, where the objective function is a mathematical expression combining some variables in order to describe the goal of the system [52]. There is a wide variety of optimization approaches available, and the use of these approaches depends on the nature and the degree of complexity of the problem to be optimized.

By reviewing the literature, it was found that different optimization approaches have been used in order to solve the general task allocation problem and was also used in order to solve the MRTA problem. In [26], a mixed integer linear programming optimization approach was used in order to allocate heterogeneous robots for maximizing the coverage area of the regions of interest. Also in [53] a mixed integer linear programming approach was used for solving the task allocation problem in the context of UAV cooperation. In [54] [55], a simulated annealing approach was used to solve the allocation of multi robot system through formulating the MRTA problem as mTSP. In [56] [57] simulated annealing incorporated with other heuristic approaches were used to allocate a set of tasks to a number of processors in computer system problems.

Another different optimization approaches were also used for solving the task allocation problem. For example population-based approaches such as the genetic algorithm was used in [58] for providing a feasible solution for a group tracking system which is capable of tracking several targets rather than individual targets. Genetic algorithm was also used in [59] to provide a solution for the time extended task allocation of multi robots in a simulated disaster scenario. Ant colony optimization, another technique of the population based optimization approaches was used in [60] to solve the task allocation problem of MRS. In [61] ant algorithm was used in the context of multi-robot cooperation for the aim of solving the task allocation problem.

The task allocation problem was also solved using hybrid optimization approaches such as the tabu search with random search method in [57] and tabu search with noising method in [62]. In [63] a simultaneous approach for solving the path planning and task allocation problems for a MRS is proposed, where simulated annealing and ant colony optimization approaches were investigated and applied for solving the problem.

The focus of the work in this thesis is to solve the MRTA problem using a metaheuristic optimization-based approach.

2.2.6 Organizational Paradigm

2.2.6.1 Centralized Architecture

In centralized architecture approaches the decision making takes place in a central unit or on a central agent. This central agent must be connected to all other agents in the system to be able to collect all the required data about every unit involved in the system. The other agents send their information to the central agent on which the control algorithm is implemented which processes the gathered information to generate the output solution. Then the central agent sends the suitable commands to all the entities in the system to control their actions [38].

The centralized approach is commonly used in various research and industrial applications and it is one of the most widely reported approaches in the literature of task allocation problems [64]. The centralized approach is relatively simpler in implementation than other approaches and is characterized by its higher efficiency and being more economical in time and cost saving. On the other hand the centralized approach is inefficient and not practical in terms of scalability since as the number of used agents in the system increases the performance of the system decreases in terms of computational time requirements [1]. The main disadvantage of the centralized approach is the unreliability of the system in situations of system failure or malfunction. In such cases, if the central agent fails, this will lead to complete failure of the whole system until the replacement or fixation of this central agent.

In [65] the centralized approach was used in the context of sensor networks, the authors proposed a centralized algorithm to solve the MRTA problem in order to assign tasks to mobile robots to extend the life time of the sensor network. Also in [66], a centralized approach was proposed and tested for solving the MRTA for the inspection problem in an industrial plant.

2.2.6.2 Decentralized Architecture

Decentralized architecture is an approach in which the role of the central controller is divided between the agents of the system. There are different configurations of the decentralized architecture depending on the type and level of communication between the agents. In some configurations all the agents can communicate with each other and thus all the information of the system is available as in centralized controller. Other configurations only include neighborhood communication between the agents and thus not all the information of the system is globally available to other agents of the system.

One of the advantages of the decentralized control architecture is the flexibility of the whole system. This flexibility appears in the scalability of the system as the addition of new agents will not cause degradation of the performance of the whole system as in centralized architecture. The main advantage in this control architecture is the enhanced reliability of the system due to the participation of several control systems on the contrary to centralized architectures. Thus if any of the agents encountered a sudden failure, the rest of the system will be able to continue functioning till the main task is achieved [1].

In [67] the authors proposed a decentralized implementation of the Hungarian method proposed in [20] in order to solve the MRTA problem formulated as an instance of the OAP. In [68] two decentralized auction based approaches, the consensus-based auction algorithm and the consensus-based bundle algorithm were proposed for solving the MRTA problem of a fleet of autonomous mobile robots. Also an evolutionary computation decentralized approach was proposed for solving the MRTA problem using genetic algorithm in [69]

2.3 Concluding Remarks

In this chapter a brief introduction about MRS was given including the definition of MRS and the motivations of using them. The MRTA problem was also defined and some main aspects of the MRTA problem such as MRTA scheme, planning of the problem, organizational paradigms used, previous MRTA formulations and the previously reported solving approaches were also discussed.

Chapter 3

Metaheuristic Optimization

This chapter reviews metaheuristic optimization and starts by discussing optimization in general in 3.1 and provides a classification of optimization approaches in 3.2. Metaheuristics optimization is introduced in 3.3 followed by presenting the motivations of using metaheuristics 3.4.Metaheuristic algorithms are discussed in more details in 3.5.

3.1 Optimization

Optimization is an important and a fundamental process that has been used extensively in solving daily life problems in the sake of finding the optimal state through finding a solution that minimize or maximize the cost of an objective function, and therefore it can be considered as one of the oldest sciences in the world. Nowadays several optimization techniques are being widely used in solving industrial and engineering problems as well as economics related problems and applications. Also various and new optimization techniques are currently innovated since this research area is of a great interest and importance.

Optimization problems can be categorized into single-objective and multi-objective problems. Single-objective problems are relatively easier when compared to multi-objective problems. Since multi-objective problems require optimizing different objectives simultaneously and these different objectives can be contradicting, also multi-objective optimization problems are more difficult when it comes to satisfying the many different constraints applied to the problem domain which increases the complexity of these types of optimization problems [70].

The optimization problem can also be classified into continuous optimization problems such as real mathematical functions optimization, where the solution of the problem is a set of numerical values that when substituted in the set of variables of the mathematical function gives the minimum value for this function. Another type of optimization problems is the discrete optimization problems where the solution includes discrete valued variables. One of the well known types of discrete optimization problems is the combinatorial optimization problems where solving the problem includes finding a certain combination of the set of the finite discrete objects that defines the problem. Any combination of the set of objects that satisfies the set of constraints of the problem is considered a candidate feasible solution where the combination or grouping of objects that minimizes the cost of the objective function is the optimal solution [70].

In real world applications, most of the targeted problems that need to be optimized are multi-objective combinatorial optimization problems of an increased complexity that is characterized by a large search space and are categorized as NP-hard and NP-complete problems. Due to this increased complexity, classical optimization techniques failed in solving these types of complex problems and therefore researchers have focused on new search techniques and developed innovative optimization methods such as the metaheuristic algorithms in order to solve these difficult optimization problems [71].

3.2 Optimization Approaches Classification

Optimization approaches can be generally divided into two main categories, deterministic and stochastic approaches. In Figure 3.1, a simple taxonomy of optimization approaches is presented.

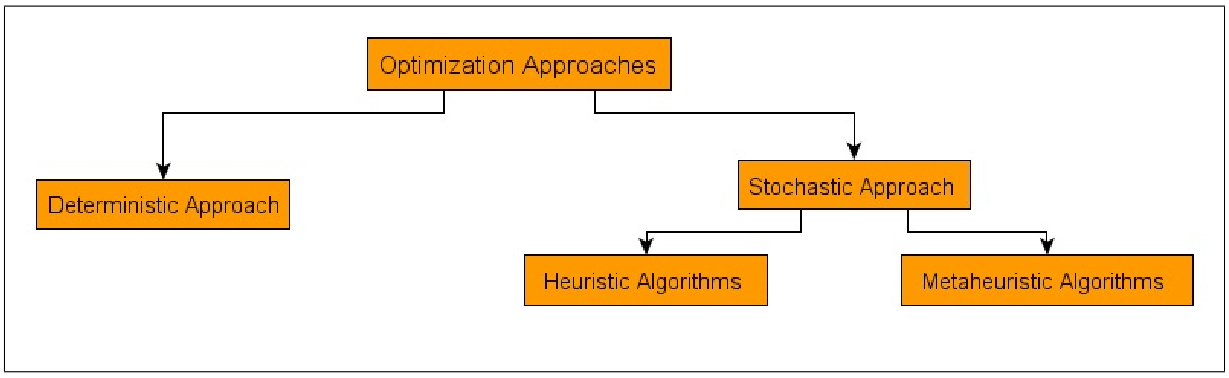


Figure 3.1: Optimization Approaches Simple Taxonomy

3.2.1 Deterministic Approaches

Most of the classical optimization approaches are of a deterministic nature. These deterministic algorithms have a rigorous nature which implies that when these algorithms are used to solve a certain problem, the algorithm will follow the same path and will produce the same output for the same input parameters whenever this algorithm was used whether today or tomorrow. Deterministic algorithms are one of the oldest and most studied optimization approaches and it was proven to be practical and efficient when solving small to medium scale problems such as P class problems which can be solved in polynomial time scale such as the Minimum Spanning Tree problem. The main drawback of such algorithms is that they become computationally expensive when solving large scale or difficult problems such as NP-hard and NP-Complete class problems. Well known examples of deterministic approaches are linear programming, non-linear programming, gradient based, gradient free, branch and bound, hill climbing and A* algorithms [70].

3.2.2 Stochastic Approaches

The other category of optimization algorithms is the stochastic algorithms. Stochastic processes consist of a sequence of random events and therefore a stochastic process can be viewed as a process of unpredictable state. Optimization algorithms that are stochastic based are the ones that make use of probabilistic methods in the building of the algorithm in order to find the optimal solution. The use of probabilistic methods through the search algorithm has several significances especially when dealing with real word applications that are characterized by their vast search space and noisy environment. One of the main advantages of using probabilistic methods is the capability of stochastic algorithms to escape local optimum and to find the global optimum within several local optimums found in the search space. Another important advantage of stochastic algorithms is that they have higher potential for exploring new search areas in the search space through the injecting of randomness to the algorithm variables. Stochastic algorithms also have an enhanced performance when dealing with noisy input data [72] [73]. The stochastic algorithms are mainly divided into two main sub categories, heuristic and metaheuristic algorithms.

3.3 Introducing Metaheuristics

The term metaheuristic is divided into two words "meta" and "heuristics". Heuristic originally comes from the Greek word heuriskein which means "to find" or "to search" and thus heuristics indicates the study and application of search methods and techniques to solve problems. The other word meta is also a Greek word which means "post" therefore adding the suffix meta to the word heuristics indicates the use of higher level methodologies of search techniques in order to solve an optimization problem. Unfortunately until now there has been no agreed definition of heuristics and metaheuristics in the literature. Although many researchers have proposed definitions for both terms, but none is considered an official definition of either. In the literature, some authors used heuristics and metaheuristics interchangeably due to the fact that the difference is small [70] [71].

3.3.1 Heuristics

Before the introduction of metaheuristics researchers tried to solve optimization problems by the use of what is called heuristic algorithms. Heuristic algorithms attempt to solve an optimization problem by trial and error methods. Because of their stochastic nature they use random and probabilistic techniques through the algorithm. They also incorporate local search methods with the probabilistic methods in order to find the optimal solution of the problem. Researchers in the early literature tried to solve complex optimization problems using specialized heuristic algorithms. Although the developed algorithms yielded very promising results over a wide range of applications, but however due to the lack of general applicability of these heuristic algorithms, researchers had to design new algorithms for every new tackled problem and even for variations in the previously solved problems [74] [75]. These drawbacks of heuristic algorithms required the finding of a new approach for solving complex optimization problems which was one of the main motivations for the emergence of metaheuristics.

3.3.2 Metaheuristics

The term metaheuristics was first introduced in 1986 in [76]. Metaheuristics are an extension of the previous heuristic algorithms and can be considered as general purpose upper methodologies that guide the basic underlying heuristics during the search for the optimum solution. Metaheuristic algorithms are characterized by their general applicability to various and different optimization problems and it has been claimed in the literature that metaheuristics can be used to solve almost any optimization problem. Although one of the main challenges of using metaheuristics is to adapt the chosen optimization techniques or algorithms to the problem being optimized, however it is yet considered an enhanced approach then developing new specially designed heuristics to solving new optimization problem [75].

Metaheuristic algorithms were developed to tackle complex optimization problems when other classical optimization techniques either failed to solve such complex problems or solved them inefficiently. They are considered as approximate search methods aiming at solving problems for global optimum through integrating local search methods with random search and probabilistic methods. Although metaheuristics are intended to solve optimization problems for global optimum, however they do not provide any guarantee to reaching optimality when applied to optimization problems. One of the most important features of metaheuristic algorithms is their capability of generating acceptable near optimal solutions in reasonable time and that they are less computationally expensive than other classical approaches [77]. This important feature of metaheuristics explains the major interest in this field of research especially in industrial and scientific applications which became noticeable through the significant number of published papers and books in this field [78].

More details of the metaheuristic algorithms and their classification is introduced in 3.5

3.4 Motivations of Using Metaheuristics

In order to understand the motivations of using metaheuristic optimization and to be able to make correct decision about when to use and when not to use metaheuristics, it is essential to analyze all the information available about the problem being optimized and to have a general understanding of this tackled problem. Therefore in this section, the different aspects of any optimization problem will be first discussed before getting into the motivations of using metaheuristic optimization algorithms.

3.4.1 Aspects of Optimization Problem

In Figure 3.2, the discussed ingredients of the optimization problem that affects the choice of the optimization algorithm is introduced.

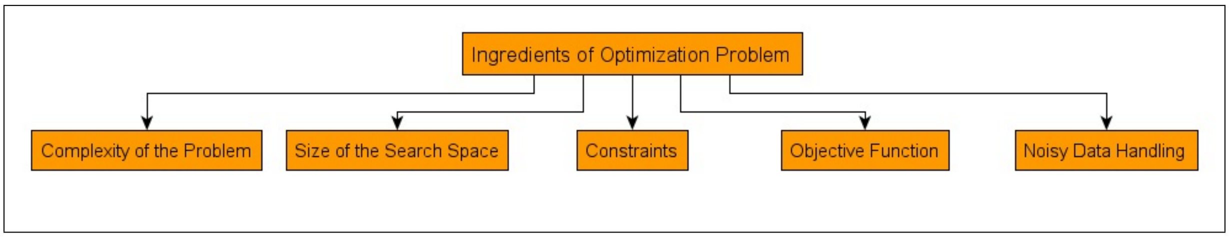


Figure 3.2: Aspects of Optimization Problem

3.4.1.1 Complexity of the Problem

Optimization problems can be categorized according to their complexity and furthermore the complexity of the problem is one of the major factors affecting the choice of the optimization algorithm.In computational complexity theory, P class problems are problems that can be solved in polynomial time, which indicates the presence of well established deterministic algorithms that can solve this type of optimization problems in polynomial time. An example of P class problems is the Minimum Spanning Tree optimization problem. The Kruskals algorithm is an example of a deterministic polynomial time algorithm that is capable of solving this problem in polynomial time.

Another complexity class of optimization problems is the NP class which refers to a set of problems whose optimal solution cannot be found using deterministic algorithms in polynomial time. However if an initial guess or solution to the problem is found, this solution can be verified as the optimal solution or rejected using polynomial time algorithms. NP-hard class is a further class of computation complexity theory classes, NP-hard problems can be informally defined as problems that are at least as hard as the hardest NP problem. A well known example of the NP-hard class is the TSP which is one of the most famous optimization problems. The last complexity class is the NP-complete class which refers to problems that are both NP and NP-hard. In real word applications, almost the majority of the problems are either NP-complete or NP-hard combinatorial problems that imply the lack of well known established algorithms that are capable of solving such problems in polynomial time.

3.4.1.2 Size of the Search Space

In optimization problems, the aim is to find the optimal solution from a set of all feasible solutions of the problem. Therefore the size of the search space of the problem that includes all possible feasible solutions is an essential factor influencing the choice of the optimization algorithm. The number of feasible solutions of the problem depends mainly on the nature and size of the problem and therefore it is a must to determine the nature of the problem before choosing the type of algorithm to be used in order to solve it. Since the problem of interest which is discussed in this thesis as well as most of the real world problems are of a combinatorial nature, therefore in this section four types of combinatorial problems will be enlightened. Figure 3.3 demonstrates the four types of combinatorial problems that will be introduced.

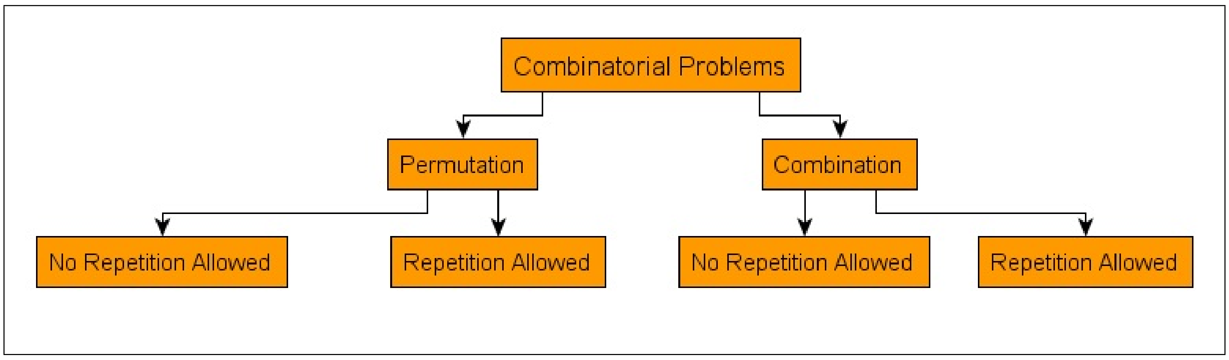


Figure 3.3: Combinatorial Problems Classification

Combinatorial problems can be divided into two types, permutation and combination problems. Permutation can be defined as the arrangement of a set of objects in a certain order. From this simple definition it could be inferred that the order of objects does matter in permutations, while in combinations the order does not matter. For example a fruit salad is a combination of apples, bananas and grapes, if the order of the fruits is altered, it will not cause a difference for fruit salad. However, if the combination of the lock password for a safe is 1234 it will not unlock if 1324 is provided instead, although the same numerical elements of the password were provided but in a different order. In this case the password for the lock is identified as a permutation not a combination since the order did matter. It is important to determine the nature of the problem in order to calculate the number of permutations or number of combinations. This calculation also depends upon the size of the problem.

Another aspect of combinatorial problems whether a combination or a permutation, is whether repetition is allowed or not. An example of no repetition allowed permutation is the TSP, in this problem the order of the cities does matter in the quality of the solution and also repetition of cities is not allowed and therefore the TSP is a no repetition allowed permutation problem. A repetition allowed permutation example is the safe lock password example previously explained where the password of the safe lock can be 1010 which includes repetition of elements and at the same time is a permutation, since the password 0101 will not gain access to the safe although including same elements.

Furthermore in combination, repetition could be allowed or prohibited. An example of no repetition allowed combination is lottery numbers. The order of lottery number does not matter but no repetition could be allowed or this will yield more than one winner which contradicts the rule of the game. An example of allowed repetition is a problem where 3 coins should be chosen from a set of coins containing 5 different types of coins each is found twice. For example if the set contains the following coins “1, 1, 5, 5, 10, 10, 15, 15, 20, 20, 25 and 25”. In this problem, repetition of coins is allowed and the order of coins in the solution will not impose a difference therefore such a problem is a repetition allowed combination problem.

After determining the type of the problem, the next step is to calculate the number of feasible solutions through calculating the number of permutations or combinations depending upon the type and the size of the problem. For example, if the encountered problem is an asymmetric TSP of 5 cities. Since the problem is categorized as a no repetition allowed permutation problem, therefore the number of permutations can be calculated using equation (3.1).

$$\text{Number of Feasible Solutions} = \frac{n!}{(n-r)!} \qquad \text{where n = r = 5} \tag{3.1}$$

Another example is to find the correct combination of four digits that represent the lock of a safe. In this example, the problem is categorized as a permutation with repetition problem and thus the number of permutation can be calculated as in equation (3.2)

$$\text{Number of Feasible Solutions} = n^r \qquad \text{where n = 10, r = 4} \tag{3.2}$$

It is obvious that knowing the category of the problem as well as its size strongly affects the number of feasible solutions and therefore affects the size of the search space of the problem. This knowledge of the search space of the tackled problem strongly affects the type of the optimization algorithm chosen for solving the problem.

3.4.1.3 Constraints

In real word applications the optimization problems are not only complex and have large search space, but also are heavily constrained. These applied constraints to the optimization problem can make it very difficult to find feasible solutions within the search space of the problem. This encountered difficulty in constructing a feasible solution makes it very difficult and time consuming to find the optimal solution of such a heavily constrained problem. And although the optimal solution could be found, however the required time and memory resources will be enormous and thus not practical. Another important challenge when targeting optimization problems is that some optimization problems are real time constrained. This indicates that the solution of the problem must be found in real time. Although such type of problems might not be complicated, but the fact that the optimal solution must be found in real time imposes a difficulty to the problem. In such real time applications, it is more important for the user to get a near optimal solution to the problem in real time than to get the optimal solution after time consuming processes [71] [75].

3.4.1.4 Objective Function

In optimization problems the main aim of applying the optimization algorithm is to optimize (minimize or maximize) the objective function of the problem where the objective function is used to test the quality of the provided feasible solutions.In real world applications the objective function of some optimization problems might be difficult and/or complicated and therefore time consuming. Although such problems might not be that complicated or difficult, but the objective function itself requires enormous time and consuming computations.

3.4.1.5 Noisy Data Handling

Another major problem facing the optimization of real world applications is the noisy and uncertain information and the handling of such noises. These sorts of problems appear because the model of the problem being optimized might not be as exact as the real problem, since it includes some assumptions or approximations. Also the objective function of the problem might be noisy or is varying with time. Therefore the generated optimal solution cannot be accepted without further investigation of this alleged optimality [75].

3.4.2 When to Use Metaheuristics

As previously discussed, the main purpose of using optimization algorithms is to find the optimal solution of a problem through iteratively evaluating the quality of the found solutions using the objective function of the problem till the optimum solution is found. The performance of any optimization algorithm includes two main points. First its capability of generating a solution for the optimization problem, and whether this solution is the optimal solution or not, and if it is not the optimal solution how far is this solution from optimality. Second how many resources this algorithm utilized in order to find this generated solution, and certainly it must be noted that one of the most important resources used is the amount of time consumed to find this solution. Thus these two factors, quality of solution and time consumed to find this solution, will be the main two factors used in presenting the motivation of using metaheuristics.

First, the complexity of the problem is considered. As previously explained the P class optimization problems are relatively simple and polynomial time algorithms have already been developed to solve such kind of problems. Therefore the use of metaheuristics is unwise and inefficient when solving a P class optimization problem, because metaheuristics will not guarantee finding the optimal solution. On the contrary, deterministic algorithms will be able to find the optimal solution for the problem. On the other hand, when tackling optimization problems of the NP-complete and NP-hard classes the deterministic algorithms tend to be inefficient and time consuming. At this case the use of metaheuristics would be the intelligent choice since metaheuristics will be capable of producing near optimal accepted solutions in much less time than deterministic algorithms.

Second, the search space of the problem is considered. It was previously agreed that deterministic algorithms are more efficient when encountering P class optimization problems because of their ability to solve these types of problems in polynomial time. Unfortunately this is not the case when the size of the problem increases and thus the number of feasible solutions increases causing the increase of the search space of the problem. Therefore, although the deterministic algorithm will be capable of finding the optimal solution for such a problem, however due to the exhaustive nature of the algorithm finding the optimal solution in such a large search space will become very time consuming.

Consequently when targeting P class problems with large search space, metaheuristics would be a suitable choice in order to avoid consuming a lot of time to find the optimal solution. For example, the TSP with 5 cities will yield 120 possible solutions while a TSP with 52 cities will yield approximately 8×10^{67} possible solutions. For the 5 cities problem the use of a polynomial time deterministic algorithm would be a suitable choice for finding the optimal solution although the tackled problem is an NP-hard problem due to the limited search space of this instance of the problem. On the other hand, for the 52 cities problem the use of metaheuristics would be the best choice to solve such a huge search space combinatorial problem.

Third, the level of constraints of the problem is considered. As previously discussed, there are some optimization problems which are heavily constrained and although such problems may be one of the P class problems, but the fact that they are heavily constrained make it very difficult and time consuming to solve such problems using deterministic approaches. In such cases metaheuristics had emerged to be the most suitable approach to solve such heavily constrained problems. In addition, metaheuristics are extensively used to solve real time constrained optimization problems because of the fact that metaheuristics are capable of finding near optimal solutions much faster than deterministic approaches. Therefore in applications that require working with real time constraints, a solution with slight divergence from the optimal solution is still accepted as long as it has been generated in real time.

Finally, the objective function is considered. In some optimization problems, regardless of their level of complexity, the objective function used to evaluate the quality of the solution is complicated, noisy and sometimes varying with time. In such applications the quality of the solution must be investigated through the generation of a set or population of solutions through various evaluations of the objective function in order to verify the quality of the solution. This process may become time consuming and therefore the use metaheuristics with such applications is justified in order to save time.

In conclusion, one can sum up the use of metaheuristics in solving optimization problems in the following cases:

1. Complex optimization problem with average or large search space
2. Simple optimization problem with a large search space
3. Heavily constrained and real time constrained optimization problems
4. Optimization problems of multi-modal and time consuming objective functions
5. Optimization problems with noisy and time varying models

3.5 Metaheuristic Algorithms Classification

In the last two decades, various numbers of metaheuristic algorithms have been developed. In the literature, different researches tried to provide several classifications of metaheuristic algorithms. Although there is not an official classification of metaheuristics, but there are several points in common between different classifications. For example, researches classified metaheuristic algorithms depending on memory usage, number of neighborhoods used, dynamicity of the objective function and original source of inspiration [79].

In this section, different metaheuristic algorithms are introduced and briefly explained. The introduced algorithms are classified according to their nature, whether trajectory-based or population-based. Figure 3.4 illustrates the simple classification of metaheuristic algorithms introduced and the algorithms explained in this section.

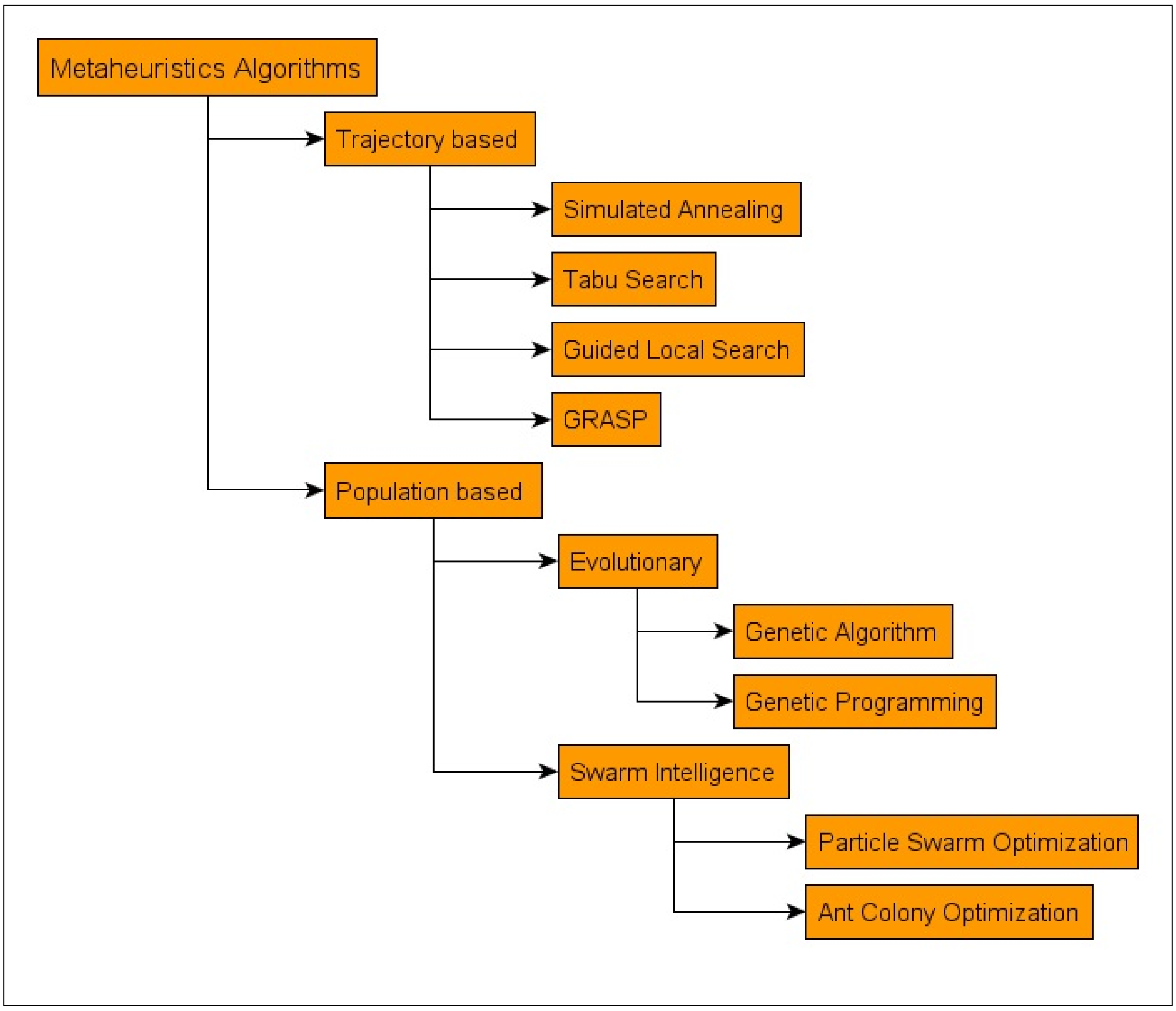


Figure 3.4: Taxonomy of Metaheuristic Algorithms

3.5.1 Trajectory-based Metaheuristics

Trajectory based metaheuristic algorithms are the algorithms that construct a trajectory in the search space of the problem in order to find the optimal solution. The form and the characteristics of the designed trajectory depend upon the algorithm strategy used in finding the optimal solution which differs from one metaheuristic algorithm to another. In trajectory based algorithms, the algorithm works with only one solution per iteration. Simulated annealing, tabu search, iterated local search, guided local search and GRASP are various examples of trajectory based metaheuristic algorithms [79]. Trajectory based algorithms are more exploitation oriented, which means that they tend to intensify the search in the neighborhood of current solution rather than explore other neighborhoods [75].

3.5.1.1 Simulated Annealing

Simulated annealing is a metaheuristic algorithm based on the physical annealing processes where heating of materials occurs to very high temperatures and then cooled slowly according to a cooling schedule. When the frozen state is reached at temperature equals zero, the system reaches its minimum energy state. If the cooling process does not happen slowly, defects may occur to the material being annealed. Simulated annealing was first introduced in 1953 in [80]. It was used as an optimization algorithm in 1983 in [81]. In simulated annealing the algorithm starts by a random solution of the search space and then the solution is enhanced through a certain number of iteration depending upon the cooling schedule. In simulated annealing, the minimum of the objective function is analogous to the minimum energy state of the system in the physical annealing process and thus the optimal solution is supposed to be found at the minimum temperature of the cooling schedule. Simulated annealing has been used in various applications such as non-linear function optimization, solving the travelling salesman problem, scheduling problem and various other applications. One of the main advantages of simulated annealing is the ease of use and ease of implementation. On the other hand simulated annealing requires a lot of computer time for many runs and computations and it also requires tuning a lot of parameters. Finally simulated annealing has been proved for providing acceptable solution for a wide range of problems [78] [74] [82].

3.5.1.2 Tabu Search

Tabu search is one of the most used metaheuristic algorithms in solving combinatorial optimization problems. It was first introduced in [76]. Although tabu search is considered a simple algorithm, it has been very effective and powerful for solving difficult combinatorial optimization problems. Tabu search is the implementation of local search integrated with memory structures. The memory structures are used to escape local optimum solution and to diversify the search space of the problem so that unvisited areas of the search space are explored. A simple tabu search algorithm include two types of memory, a short term memory (tabu list) in which recently visited solution are added so that they are not revisited for a certain number of iterations and a long term memory (frequency based memory) which gives an indication about how often a search point has been visited. Tabu search has many applications, it has been used in solving scheduling problems such as academic scheduling, job shop scheduling and sequencing and batching and it was also used in solving the vehicle routing problem, quadratic assignment problem and layout planning. The main advantages of tabu search are their ability to solve large scale difficult problems and to provide acceptable results that is often better than previously provided solution by other algorithms. On the other hand tabu search requires too many parameter setting which affects the quality of the algorithm and its capability of reaching the global optimum solution [78] [74].

3.5.1.3 Guided Local Search (GLS)

Another variation of the local search is the guided local search metaheuristic algorithm. It was first introduced in 1996 in [83]. In guided local search the algorithm is composed out of two parts. The first is a local search algorithm which searches in the neighborhood of the current solution for better solutions. The second part is a dynamic change of the objective function of some of the visited solutions in order to escape local optimum and to diversify the search. At first the local search is applied until the algorithm gets trapped into a local optimum solution. Then the objective function of the local optimum solution and its neighborhood is changed in order to decrease the attraction of the local search to this local optimum through penalizing the objective function of such solutions. Finally local search is applied again to find a better solution. Guided local search has been used in solving various optimization problems such as vehicle routing, travelling salesman problem, scheduling problem and quadratic assignment. One of the important advantages of the guided local search is its simplicity and ease of parameters tuning [82] [84].

3.5.1.4 Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP is a simple metaheuristic algorithm of the trajectory-based family. GRASP is used for solving combinatorial optimization problems. The algorithm consists of two phases, the first phase is the construction and the second phase is enhancement through local search and therefore GRASP can be considered as a multi-start iterative algorithm where at each iteration of the algorithm the two phases are executed. At the first phase, a solution is constructed one element at a time and the elements are chosen from a list of candidate elements according to a certain semi-greedy heuristic to allow construction of improved quality solutions. In order to allow diversification of the starting solution provided from the construction phase, the next element is chosen according to a probabilistic randomization function from the best elements in the candidates list. At the second phase of the GRASP algorithm, local search is applied in order to find the local optimum solution in the constructed solution neighborhood. GRASP was first introduced in 1989 in [85], it has been used in solving several optimization problems such as scheduling, vehicle routing, resource allocation and optimal assignment problems [86]. The main advantage of GRASP algorithm is its ability to explore large search space optimization problems and to escape local optimum because of its global search nature. However, in GRASP algorithms each new iteration is a start of a new search, which indicates that GRASP algorithms do not make use of previous search experiences.

3.5.2 Population-based Metaheuristics

Population-based metaheuristics are algorithms that consists of a population of solutions or individuals that are either manipulated or used to manipulate the solution respectively throughout the algorithm in order to find the optimal solution. In every iteration the algorithm is dealing with a set or a population rather than a single solution which explains the diversifying nature of the population-based algorithms. This indicates that this type of algorithms explore the search space of the optimized problem [75] [82]. Population-based algorithms are divided into two main sub-categories. First, evolutionary-based algorithms such as genetic algorithm, genetic programming, evolutionary programming and evolution strategy. Second, swarm intelligence-based algorithms such as ant colony optimization, particle swarm optimization and artificial bee colony.

Evolutionary-based Algorithms

The evolution process can be considered as an optimization process of living organisms in order to enhance the abilities of these organisms to survive the change of the environment. The evolution process includes natural selection of individuals, survival of the fittest and reproduction. In metaheuristics evolutionary algorithms can be considered as an artificial model of the evolution process. Evolutionary based algorithms normally start with a population of solutions and then a selection process occurs to select fittest individuals among the population which simulates survival of the fittest in the evolution process. Finally genetic operators such as combination (crossover) and modification (mutation) of fit individuals occur in order to generate a new population with better traits [75] [79].

3.5.2.1 Genetic algorithms

Genetic algorithms are a class of metaheuristic optimization algorithms that are inspired by natural evolution process. They start by a population of candidate solutions for the problem being optimized, and then use genetic inspired operators such as crossover and mutation over specially selected individuals according to their fitness function. Then selection criteria are applied in order to select the new population for the next iteration. The quality of the solution is enhanced through the enhancement of the population from one generation to another. It was initially developed by John Holland and introduced in the seventies in [87]. It has been widely used in various applications of economics, operations research, engineering, industrial and real world applications. Genetic algorithms are characterized by their ability to solve multimodal discrete or discontinuous functions and they were also applied and successfully solved large space difficult combinatorial optimization problems. The main advantages of genetic algorithms is that they suitable for multi-objective function problems, large search space problems and are suitable for noisy environments. The main disadvantage of genetic algorithms is that they require a lot of time to converge for the optimal solution [88].

3.5.2.2 Genetic Programming

Genetic programming (GP) is an another evolutionary-based approach based on Darwinian theory of evolution which rely upon the survival of the fittest individuals in the current population to the next iteration. GP is similar to GA, the algorithm starts with a random population of individuals or candidate solutions and genetic operators are iteratively applied on the current population. Then the next population is chosen according to a certain selection strategy after the evaluation of the fitness of the individuals of the current population using the fitness function. The main variation in GP is that the individuals themselves are programs instead of fixed length strings. GP has been the goal of several researchers since it has the potential of giving computers the ability to automatically generate computer program in order to solve the assigned task. The main disadvantage of GP is that they are computationally expensive as they have an explosive growth rate [75].

Swarm Intelligence-based Algorithms

Swarm intelligence refers to the behavior of a system that is made up of a decentralized self-organizing population (swarm) of agents that are spatially distributed coordinating their actions in order to reach a complex collective behavior such as bees, ants, fish and birds. Swarm intelligence is used to solve optimization problems. The term swarm intelligence was first introduced in 1989 in [89]. They key features to swarm intelligence is the use of a large finite number of simple processing elements that work together through the neighborhood communication without centralized control. There have been several metaheuristic algorithms that depend upon the swarm intelligence approach. For example, ant colony optimization, particle swarm optimization, firefly optimization and harmony search [75].

3.5.2.3 Particle Swarm Optimization

Particle Swarm Optimization was developed by Kennedy and Eberhart in [90]. PSO is a population-based stochastic algorithm inspired by the social behavior of natural organisms. PSO mimics the social behavior of the individuals cooperating in a swarm such as birds in bird flocking swarm or fishes in fish schooling swarm. The word particles refer to the individuals or the candidate solutions in the optimization problem. Each individual of the swarm decides his next step or next position depending on three main search strategies. The first strategy depends upon the local intelligence and the past experience of each particle where this search strategy is a local search strategy. The second strategy depends upon the global intelligence of the swarm such that each particle tends to move to the position where global success was found through other particles. In this strategy the particle is emulating the success of other particles and therefore it is a global search strategy. The last strategy depends upon the introducing of some randomness to the search such that each particle have some tendency to move randomly in order to be able to explore new search areas of the search space and to avoid getting stuck in local optimum. The PSO is relatively simpler than other population-based approaches such as the genetic algorithm since it requires the adjusting of few parameters and it does not include the use of crossover and mutation operators. PSO had many applications such as function optimization, fuzzy system control and artificial neural network training [75] [70].

3.5.2.4 Ant Colony Optimization

Ant colony optimization is recently introduced swarm intelligence based metaheuristic optimization algorithm. It was first introduced in 1991 in [91]. The ant colony optimization mimics the behavior of real ants living in colonies that are capable of solving complex problems through the indirect communication and cooperation between simple agents (ants). The communication between the agents in the algorithm also mimics the indirect communication between real ants which occurs through the dynamic change in the environment done by each ant and felt by other ants, which is known for stigmergic communication. The ant colony optimization has been used in several applications and optimization problems such as the travelling salesman problem, scheduling problem, vehicle routing problems. Ant colony optimization was also used in solving dynamic combinatorial optimization problems such as routing in telecommunication networks [92].

3.6 Conclusion

In this chapter, metaheuristic as an optimization based approach has been introduced. To sum metaheuristics are higher level methodologies that are used to guide the search process for the optimal solution in an optimization problem. They are problem are global optimum independent abstract algorithms that try to explore the search space efficiently in order to find near optimal acceptable solutions.

Metaheuristics integrate local search with probabilistic randomization in order to have a balance between search space diversification and intensification so that they can explore the search space of the problem and to avoid getting trapped in local optimum solutions. Metaheuristics in contrast to deterministic algorithms do not guarantee finding the optimal solution but they have the capability of finding near optimal solutions in accepted time.

Finally metaheuristics are flexible, suitable for global optimization difficult problems and can be used in real world practical applications. On the other hand they do not guarantee optimality, they lack theoretical basis and the provided solutions are affected by the tuning of multiple search parameters.

Chapter 4

Metaheuristic Optimization-based Approach to MRTA

As previously discussed, the MRTA problem is one of the most important problems in the field of mobile robotics research. Generally the MRTA problem concerns about allocating a number of mobile robots to a number of tasks such that this allocation would lead to the most efficient performance of the whole system. Although it may appear that it is a simple task to allocate a number of robots to a number of tasks, however the MRTA is one of the most difficult problems which can be anticipated from the excessive number of algorithms and approach proposed in the literature to solve this problem. The complexity of the problem comes from different angles such as the type of robots to be allocated, the type of tackled tasks and the level of constraints bounding the allocation process. In this chapter, a metaheuristic based approach is proposed to solve the MRTA problem. The proposed approach is concerned with solving the MRTA problem with its different complexity levels and at a generic level.

The rest of the chapter is organized as follows. The problem formulation is introduced and the basic concepts behind this formulation are also discussed in 4.1. The type of tackled robots, tasks and constraints and the adaptation of the mTSP formulation to the MRTA problem is presented in 4.2. Finally the proposed algorithms are discussed in 4.3 and a summary of the chapter is given in 4.4.

4.1 Multiple Traveling Salesman Problem Formulation

The mTSP can be considered as the generalization of the original TSP which is used as a platform for the study of general methods that can be applied to a wide range of discrete optimization problems. In TSP, n cities are given and a traveling salesman must visit these n cities and return back home, making a round trip. He would like to travel in the most efficient way which could be the cheapest way, the fastest route, the shortest route or some other criterion.

Similarly in the mTSP a number of nodes n and the distances between them are given. The main difference in the mTSP is that instead of a single salesman, a number of salesmen m are given. The salesmen are required to cover all the available nodes and return back to their starting node such that each salesman make a round trip. The mTSP can be formally defined on a graph $G = (V, A)$ where V is the set of n nodes and A is the set of arcs. Let $C = (c_{ij})$ be the distance matrix associated with A. Assuming the more general case which is an asymmetric mTSP, thus $c_{ij} \neq c_{ji}\ \forall\ (i, j) \in A$. The mTSP can be formulated as follows [32]:

$$x_{ij} = \begin{cases} 1 & \text{if arc (i,j) is used in the tour} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

$$\text{minimize } \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \times x_{ij} \tag{4.2}$$

$$\sum_{j=2}^{n} x_{1j} = m \tag{4.3}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{4.4}$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 2, ..., n \tag{4.5}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 2, ..., n \tag{4.6}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |SubTour| - 1, \quad \forall SubTour \subseteq V \backslash \{1\}, SubTour \neq \phi \tag{4.7}$$

$$x_{ij} \in \{0, 1\}, \forall (i, j) \in A \tag{4.8}$$

Where equation (4.2) represents the objective function which is the summation of the total distance traveled, (4.3) and (4.4) ensures that exactly m salesmen departed their starting node and returned back. Equations (4.5), (4.6) and (4.8) are the usual assignment constraints. Finally, (4.7) is the subtour elimination constraint.

4.2 Multirobot Task Allocation Problem Formulation

As previously discussed in Chapter 2, researchers have tried to solve the MRTA problem as an instance of one of the several well-known problem formulations such as the optimal assignment problem, vehicle routing problem, mTSP, scheduling problem and many other formulations. In this thesis, the mTSP problem formulation is used to solve the MRTA problem. The strong analogy between the MRTA problem and mTSP was one of the main motivations for using such a formulation. It can be simply observed that the robots that will be used to execute the tasks can be considered as the multiple traveling salesmen who must visit all cities while the tasks that should be executed can be considered as the cities that will be visited. Also the presence of well known established approaches and algorithms to solve the mTSP was another important motivation for choosing to formulate the MRTA problem as an instance of the mTSP.

Since the proposed approach mainly aims to solve the task allocation problem of MRS in real world applications. Therefore through the different phases of the development of the introduced approach, the central target was to introduce a generic approach that is capable of solving various MRTA problems of different features and challenges. This target had to be taken into consideration during the formulation of the problem and therefore the use of the mTSP formulation as presented in section 4.1 was not suitable enough to solve the task allocation problem in real world MRS applications. Therefore the previously presented formulation had to be extended and adapted in order to be used as a formulation for the MRTA problem. In order to appropriately adapt the mTSP formulation to be used as a formulation for the MRTA problem, one must properly categorize the MRTA problem in interest. By revising the proposed MRTA taxonomy introduced in Figure 2.1, it is important to categorize the problem according to the MRTA scheme discussed in 2.2.2.

In this thesis, the solving approach intends to solve MRTA problems that include heterogeneous ST robots, heterogeneous SR tasks and time extended task assignment. After the categorization of the MRTA problem in interest, the mTSP formulation must be adapted to be used for solving the MRTA problems. This adaption is done through extending the mTSP formulation and the addition of extra features to the forming ingredients of the mTSP. Figure 4.1 explains the extension of the mTSP formulation to accommodate the requirements of the MRTA problem.

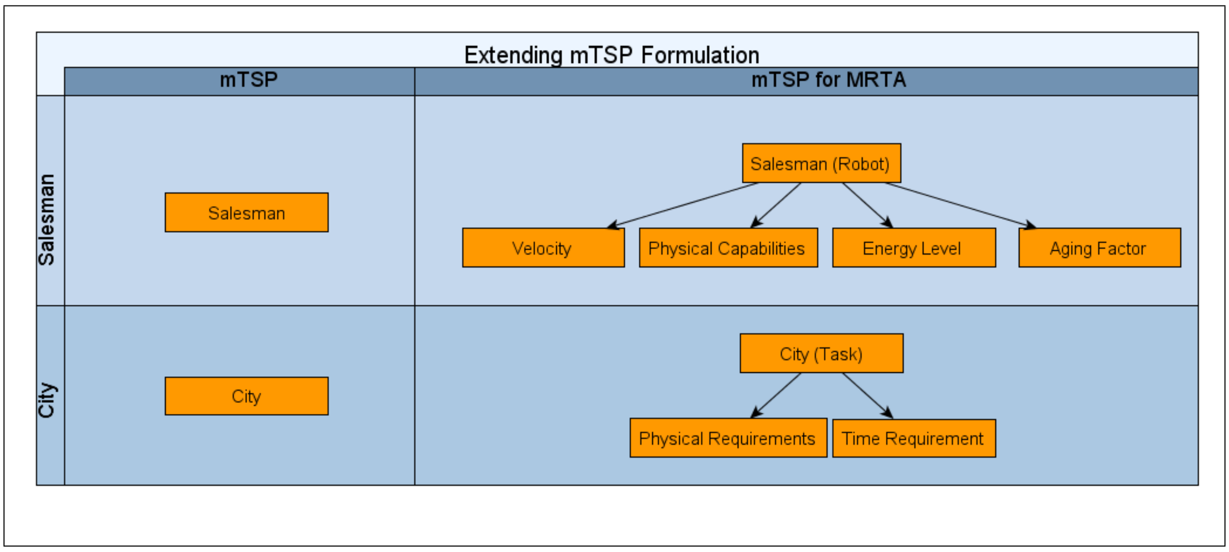


Figure 4.1: Extending mTSP formulation for MRTA

Since most real MRS applications require heterogeneous robots of different capabilities, therefore it was a must to consider the heterogeneity of the robot in the proposed approach. Four main features of the robot were considered and thus were added to the traveling salesman in the implementation phase. The four features are as follows:

- Velocity of the robot
- Robot capabilities
- Energy level of the robot
- Aging factor (efficiency)

The first feature is the velocity of the robot, since in some applications the used robots might be of different velocities therefore it was a must to have traveling salesmen of different traveling velocities. The second feature is the level of the capabilities of the robot or the additional equipment that is added to the robot which are an extra sending or acting elements that are added to the robot. In MRS, different robots cooperating together in real world applications must have different capabilities. For example in search and rescue applications the robots should be equipped with grippers, heat sensors, cameras, laser range finders and water hoses which are all additional equipment that define the level of capability of the robot. In order to model this heterogeneity of robots, the traveling salesmen were defined with a certain level of capability that is equivalent to the level of capability of the real robot. The third feature is the amount of energy that is available in the robot at the time of allocation, this energy can be represented as the remaining time in the battery of each robot. In real world applications that require dynamic allocation of robots, at some point of the process the level of energy of the robot is different than the starting point and thus not all robots are capable of doing the same tasks. Therefore this heterogeneity had to be modeled through adding the battery level feature to the traveling salesmen. The fourth and last feature is the aging factor of a robot, which is a factor that represents the current efficiency of the robot. The current efficiency of the robot indicates whether a robot is capable of finishing the assigned task in the minimum required time or will it require the robot more time than the minimum time of the task in order to be able finish it. The aging factor feature was added to the traveling salesmen in the mTSP problem formulation.

In the same manner, the mTSP formulation for solving the MRTA problem needed to be adapted to handle the heterogeneity of the tasks and therefore it was a must to add extra features to the cities. The added features to the cities are as follows:

- Task requirements
- Minimum time required to finish the task

The first feature that represents the heterogeneity of the task is the requirements of this task. In real world applications, although the main task is divided into subtasks that when are combined together forms the main task, however these subtasks are different, diverse and each of them has its own requirements that must be provided in order for the main task to be successfully executed. In the proposed approach, each city was defined with one or more requirement in order to handle the heterogeneity of the tasks. The second feature that defines the heterogeneity of the task is the time it requires in order to be successfully executed. Whether it is an instantaneous task that requires only to be visited by the robot or it is an extended time task with certain time requirement which means that the robot must stay in this task for a certain amount of time in order to execute it. This feature was also added to the city to extend the mTSP formulation.

4.2.1 Solution Construction

After the formulation of the MRTA problem as an instance of the mTSP, and after accommodating the mTSP model to be suitable with the real MRTA problem. The next step is that the solution must be constructed. The solution will be a list including all robots and all tasks in a certain order which defines the quality of the solution according to the objective function. For example a problem with three robots and five tasks, one of the candidate solutions will be as follows:

$$\text{Candidate Solution} = [R1\ T1\ T2\ R2\ T3\ T4\ R3\ T5] \tag{4.9}$$

Any combination of this list would present a candidate solution, and since the TSP is a permutation problem thus the order of this list does affect the quality of the solution measured by the objective function. The appearance of a robot in the solution means that all the tasks that are found after this robot are the tasks that will be executed by this robot. In this solution task 1 and 2 will be executed by robot 1 while task 3 and 4 will be executed by robot 2 and task 5 will be executed by robot 3. Another possible candidate solution is represented as follows:

$$\text{Candidate Solution} = [R1\ T1\ T2\ T3\ T4\ T5\ R2\ R3] \tag{4.10}$$

In this solution, the first robot is the one that will be executing the five available tasks, and the other two robots will not be used to execute any tasks.

Since the proposed approaches in this thesis are of a stochastic nature, therefore the construction of the solution is totally random. The only applied constraint is that the first element of the solution is a robot not a task. The method of constructing the random solution of the problem is presented in Algorithm 1.

Algorithm 1: Candidate Solution Construction

```
   Input: Tasks list $tasksList$, Robots list $robotsList$
   Output: Candidate allocation $candAlloc$
1  Define: Tasks list size $tasksListSize$, Robots list size $robotsListSize$,
2  $randNum \leftarrow$ generateRandomNumber(1,$robotsListSize$) % A function to
   generate random number between minimum and maximum threshold given
3  $tempRobot \leftarrow$ Remove($robotsList$,$randNum$) % A function to move an element
   from the given list at the given index
4  Add($candAlloc$,$tempRobot$,) % A function that adds to the given list, the given
   element
5  $robotsListSize \leftarrow robotsListSize - 1$
6  for $i \leftarrow 1$ to $tasksListSize + robotsListSize - 1$ do
7      if $tasksList$ is Empty then
8          $randNum \leftarrow$ generateRandomNumber(1,$robotsListSize$)
9          $tempRobot \leftarrow$ Remove($robotsList$,$randNum$)
10         Add($candAlloc$,$tempRobot$,)
11         $robotsListSize \leftarrow robotsListSize - 1$
12     else if $robotsList$ is Empty then
13         $randNum \leftarrow$ generateRandomNumber(1,$tasksListSize$)
14         $tempTask \leftarrow$ Remove($tasksList$,$randNum$)
15         Add($candAlloc$,$tempTask$,)
16         $tasksListSize \leftarrow tasksListSize - 1$
17     else
18         $decideRandNum \leftarrow$ generateRandomNumber(0,1)
19         if $decideRandNum$ == 0 then
20             $randNum \leftarrow$ generateRandomNumber(1,$robotsListSize$)
21             $tempRobot \leftarrow$ Remove($robotsList$,$randNum$)
22             Add($candAlloc$,$tempRobot$,)
23             $robotsListSize \leftarrow robotsListSize - 1$
24         else
25             $randNum \leftarrow$ generateRandomNumber(1,$tasksListSize$)
26             $tempTask \leftarrow$ Remove($tasksList$,$randNum$)
27             Add($candAlloc$,$tempTask$,)
28             $tasksListSize \leftarrow tasksListSize - 1$
29         end
30     end
31 end
```

At the beginning of the algorithm there are two lists which are the robots list that include all robots that need to be allocated. The other list is the tasks list which includes all tasks that need to be executed. At the end of the algorithm a solution listed is provided which contains all elements of both the tasks list and the solution list added in a random order. At each iteration of the algorithm an element of the tasks list or the robots list is added to the solution list through generating a random number and based on this number the next element is either chosen from the robots list or the tasks list. After randomly deciding the source list of the next element to be added to the solution list, the element itself is chosen randomly from the source list through generating a random number and choosing the element depending upon the generated number. Thus at each run of this algorithm, a solution list is generated that is totally random.

4.2.2 Objective Function

The main purpose of an optimization process is to find the optimal solution to a certain problem through maximizing or minimizing a certain function. This function measures the quality of the provided solution and it is generally called the objective function of the optimization problem. Since in this thesis, the main aim of the proposed approach is to find the optimal solution for the MRTA problem and therefore an objective function is required to evaluate the quality of the solutions provided by the suggested algorithms. Although the MRTA problem is formulated as an instance of the mTSP, however the same objective function of the mTSP previously explained in (4.2) cannot be straightforwardly used as the objective function for the MRTA problem. Therefore, some variations had to be introduced to the objective function of the mTSP in order to be effectively used for the MRTA problem.

There are two main variation of the MRTA problem objective function than the mTSP objective function. The two proposed variations are:

- The variable to be minimized is the total time rather than the total distance
- A multi-objective function instead of a single objective function

The first variations is minimizing the time instead of the distance. In the mTSP, assuming that all salesmen are using same methods of transportation and thus maintain same navigation speed. The traveling distance between any two cities will not be altered by replacing the salesman navigating between the two cities by another one and therefore the distance in the mTSP was a valid choice for the objective function. On the other hand, in the MRTA problem, the different velocities of the robot will definitely alter the traveling time between any two cities. If the navigating robot between the two cities was replaced with another one having a different velocity, the navigation time will change although the traveling distance is the same. Therefore it is more suitable to use the total time instead of the total distance to calculate the objective function of any candidate solution.

Before trying to calculate the traveling time, the form of candidate solutions in the proposed formulation is revised. The candidate solution as previously shown in equation (4.11) consists of a number of robots and a number of tasks that are randomly ordered. In order to calculate the total traveling time, the candidate solution must be divided into a number of subtours equal to the number of robots in the candidate solution where each subtour contains the tasks assigned to each robot. For example, in order to calculate the total traveling time of the candidate solution in (4.11), the solution will be divided into subtours as follows:

$$\text{Candidate Solution} = [R1\ T1\ T2\ R2\ T3\ T4\ R3\ T5] \tag{4.11}$$
$$\text{Subtour 1} = [T1\ T2] \tag{4.12}$$
$$\text{Subtour 2} = [T3\ T4] \tag{4.13}$$
$$\text{Subtour 3} = [T5] \tag{4.14}$$

Then for k subtours and t tasks in each subtour, the total traveling time is calculated as follows:

$$\text{total traveling time} = \sum_{j=1}^{k} \frac{\sum_{i=1}^{t-1} \text{distance between}(\ subtour_{j_i}, subtour_{j_{i+1}}\ )}{\text{Velocity of Robot}_j} \tag{4.15}$$

Another significant variation between the mTSP objective function and the MRTA proposed objective function is that the MRTA objective function is a multi-objective function. This change occurred due to the addition of the total execution time as a new variable. The total execution time is the time taken by each robot to finish the tasks assigned to it. This new variable depends upon the properties of both the robots and the tasks. It depends upon the minimum time required by each task in order to be executed and upon the aging factor of the robot which indicates its current efficiency. For example, in solution (4.11) if task 1 and task 3 have a minimum time requirements of 10 and 5 minutes respectively. Robot 1 and robot 2 have an aging factor of 1 and 0.9 respectively which indicates that their current efficiency is 100% and 90%. Then the total time required for robot 1 to finish task 1 is exactly 10 minutes while the total time required for robot 2 to finish task 3 will be more than five minutes due to degradation of robot 2 efficiency.

Similar to (4.15) The total execution time will be calculated as follows:

$$\text{total execution time} = \sum_{j=1}^{k} \frac{\sum_{i=1}^{t} \text{task execution time}\ (\ subtour_{j_i}\ )}{\text{Aging factor of Robot}_j} \tag{4.16}$$

As previously mentioned in Chapter 2 that one of the main advantages of MRS is enhancing the performance of the system through decreasing the amount of time required for task execution. This can be done through applying the concept of parallelism since more than one robot will be working together. And therefore the total traveling time and total execution time should not be the summation of the time of each subtour but the maximum time of the available subtours. For example, in solution (4.11) if subtour 1 total time is 30 minutes, subtour 2 takes 20 minutes subtour 3 takes 10 minutes then the total objective function will not be the summation of the three number but will be the maximum number out of them which is 30 minutes. Since at the time subtour 1 is finished, subtour 2 and subtour 3 would have been finished earlier.

Thus finally the proposed objective function for evaluating the solutions provided by the proposed approaches to solve the MRTA problem will be as follows:

$$\begin{aligned} \mathrm{f}(\mathbf{x}) = &\arg\max_{j\in\{1,2,\ldots k\}} \frac{\sum_{i=1}^{t-1} \text{distance between}(\ \mathit{subtour}_{j_i}, \mathit{subtour}_{j_{i+1}}\ )}{\text{Velocity of Robot}_j} \\ &+ \\ &\arg\max_{j\in\{1,2,\ldots k\}} \frac{\sum_{i=1}^{t} \text{task execution time}\ (\ \mathit{subtour}_{j_i}\ )}{\text{Aging factor of Robot}_j} \end{aligned} \tag{4.17}$$

4.2.3 Constraints and Solution Feasibility

Although the previously shown solutions, (4.9) and (4.10) are candidate solutions that are feasible solutions of the mTSP, however this does not guarantee that these solutions are feasible solutions for the MRTA problem. Therefore some constraints must be applied on the solution provided to make sure that these solutions are feasible solutions that could be used for solving the MRTA problem.

For a candidate solution of the mTSP to be a feasible solution for the MRTA problem, the candidate solution must be checked for two main constraints which are:

- Robot capabilities and task requirements matching
- Robot available battery time and tasks required time matching

First constraint is the matching of robot and task requirements. For a candidate solution to be a feasible solution, the physical requirements of all the tasks that are assigned to a certain robot must be matching with the available capabilities of this robot. The reason such a condition must be checked is that if one of the assigned tasks have a certain requirement that is not part of the robot capabilities, this task will not be successfully executed which implies the failure of the main task execution. This malfunction of the whole MRS performance will be due to the allocation process. For example, let's assume that task 2 in (4.9) is object recognition for a certain body, this object recognition task requires a camera, then this task must be defined as a city with a camera requirement during the formulation phase. In the candidate solution (4.9) task 2 has been allocated to robot 1, in this case if robot 1 was not defined in the formulation phase as a traveling salesman with a camera capability then this candidate solution cannot be accepted as a feasible solution. Therefore, in order to accept a candidate solution as a feasible solution, all the requirements of the tasks allocated must be checked if they are matching the capabilities of the robots which are supposed to execute such tasks.

The second constraint is the available battery level of the robots. As previously explained, a defining feature of robots is the amount of energy remaining in this robot, which is represented by the remaining time in the battery of the robot. Thus each traveling salesman was defined by a variable indicating the battery life of this salesman. On the other hand each city is defined with a certain time requirement that defines how much time a salesman (robot) must spend in this city before moving to the next city. Therefore for a solution to be accepted as a feasible solution, it was a must to check the compatibility of the battery time of the robot with the time requirements of the cities. Also the traveling time of the robot between tasks must be taken into consideration, since if the traveling distance between the tasks is time consuming it will affect the quality and the feasibility of the solution. For example if task 1 and task 2 each have a time requirement of five minutes and assuming a traveling time between them is zero. In candidate solution (4.9), if the remaining battery time of robot is less than ten minutes then the robot will not be able to execute both tasks which means a failure of the main task execution and at this case candidate solution (4.9) cannot be accepted as a feasible solution. Also if the traveling time between task 1 and task 2 is five minutes. In this case candidate solution (4.9) will not be feasible unless robot 1 remaining battery time was at least fifteen minutes in order to be capable of executing task 1 then navigating to task 2 and finally executing task 2. Both examples are visually explained in Figure 4.2.

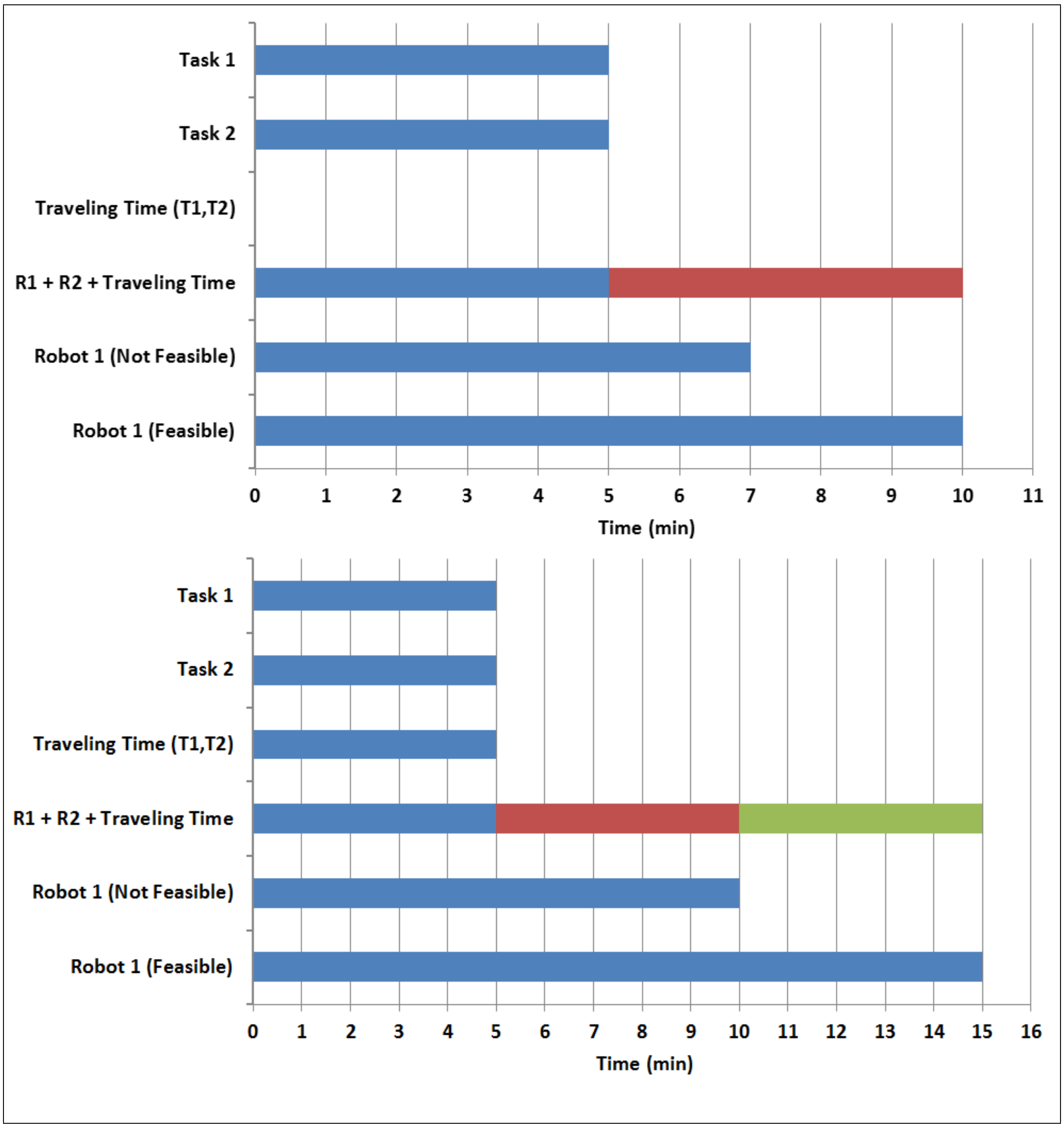


Figure 4.2: Time Requirement Constraint Examples

These applied constraints strongly affect the search space of the problem through decreasing the number of candidate solutions that can be accepted as feasible solutions. One may think that this decrease of the number of feasible solutions among the candidate solutions may make it easier for the applied algorithm to find the optimal solution than the case without the constraints. On the other hand, these applied constraints in fact make it more difficult and more time consuming to find the optimal solution. This is mainly because the algorithm will be visiting a large number of solutions that are candidate solutions of the mTSP but are not feasible to solve the MRTA problem.

4.3 Proposed Algorithms

The aim of the proposed approach in this thesis was to introduce a framework that is capable of solving the MRTA problem in a generic way. This objective was explicitly taken into consideration during the formulation phase of the problem so that the framework is capable of handling and formulating any instance of the MRTA problem into a suitable mTSP model that is when solved produces appropriate results for the addressed MRTA problem. Moreover, in order to introduce a fully generic framework, the proposed algorithms used for solving the MRTA problem had to have a generic feature of being able to solve any problem regardless of the details of the problem being solved. Therefore one of the main motivations of using metaheuristics is that they are general approaches that do not depend on the problem being solved, which is an important feature that is suitable to the purpose of solving any MRTA problem. Another motivation for using metaheuristics is their ability to solve large search space complex problems, produce near optimal acceptable solutions and that they are less computational expensive than other approaches.

4.3.1 Trajectory-based Approach

The first introduced algorithm is the SA algorithm which is a metaheuristic algorithm of the trajectory-based approaches family. The trajectory-based family of metaheuristic algorithms is the family of algorithms that uses a single solution throughout the algorithm in order to find the optimal solution.

The SA algorithm is used to solve the mTSP model that has been produced after the formulation phase of the MRTA problem. The inputs to the algorithm is a list of the available traveling salesmen or robots with their defined features, a list of all cities or tasks with their defined features and a list of the distances between the tasks current locations. At each iteration of the algorithm a solution must be constructed to be evaluated. At the start of the algorithm an initial solution is generated that is completely random. The initial solution vector is always filled with random elements of the tasks list and the robots list until both lists are empty; the only constraint is that the start of the solution must be an element of the robots list not the tasks list. After the construction of the initial candidate solution or any other neighboring solution through the algorithm iterations, the feasibility of this solution must be checked. As previously explained two checks are made. The first is checking for robots and tasks capabilities matching, the second check is the robot battery remaining time matching with the assigned tasks.

As the algorithm progress, neighboring solutions of the current solution must be generated in order to explore the search space of the problem. There are several proposed methods in the literature to generate neighbor solutions such as swapping, deletion and insertion and other methods. These methods depend on the type of formulation of the problem being solved. In this proposed approach four methods were implemented to generate candidate solutions. The four methods are as follows:

- Swapping

- Deletion and insertion
- Inversion
- Scrambling

The swapping operator choses two random elements of the solution list and swaps them with each other. Deletion and insertion operator chooses a random element and delete it from its current position and randomly insert it in a new position. The inversion operator chooses two random positions and inverts the order of elements between these two positions. The scrambling operator picks two random positions and scrambles the elements between these two positions. In Figure 4.3 an illustrative example of all the used operators is presented.

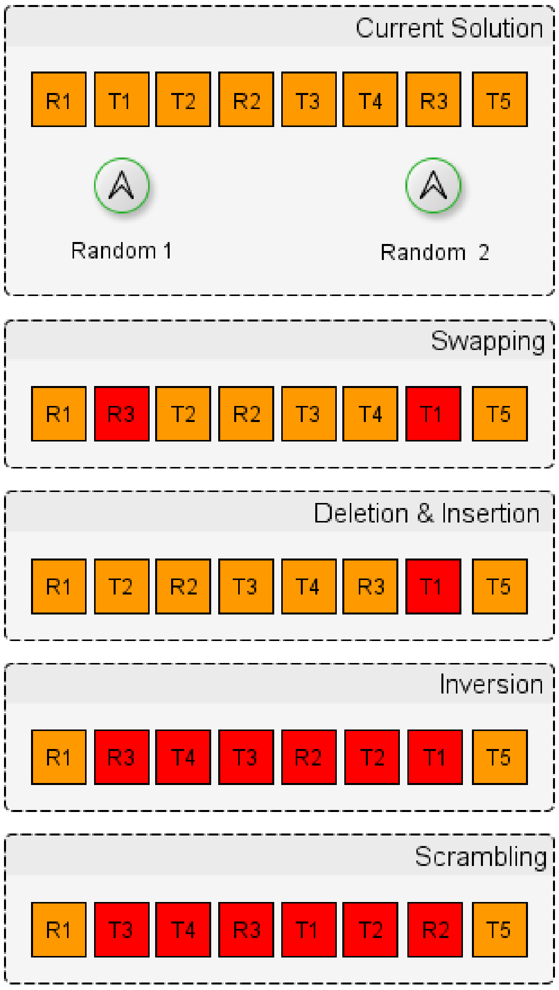


Figure 4.3: SA Neighboring Solution Generation

At each iteration one of the four methods is randomly chosen in order to generate a neighboring solution of the current solution. The four methods vary in their diversification and intensification effect on the generated neighboring solution. For example, the swapping and the deletion and insertion operators do not introduce a severe change on the current solution and therefore can be considered as exploitative operators. While the inversion and the scrambling operators can cause severe change to the current solution and thus can be considered as explorative operators. The random choice of the used operator gives the algorithm both the explorative and exploitative features that are useful in escaping local minimum and finding better solution through searching in the neighbors of elite solutions respectively.

Before the start of the SA algorithm, the cooling schedule must be defined. The cooling schedule consists of four variables which are as follows:

- Initial temperature
- Final temperature
- Temperature decrement
- Number of iterations per temperature

The initial temperature is the starting temperature of the algorithm. The choice of the value of this variable is crucial as it affects the behavior of the algorithm. If the starting temperature is so high this will make the rate of acceptance of worst solutions very high which increases the running time of the algorithm which might be not necessary, while if the starting temperature is too low this will not allow the diversification of the search space of the problem being solved and the algorithm might prematurely converge and stuck in local minimum. The final temperature of the algorithm is the temperature at which the algorithm becomes a greedy algorithm and stops accepting worse solutions. Theoretically this final temperature must be set to zero, however practically if the temperature is set to zero, this will unnecessarily increase the running time of the algorithm although the acceptance rate at very low temperature is almost the same as zero. The temperature decrement or sometimes called annealing schedule is the variable that decides how the temperature is decremented from the initial temperature to the final temperature throughout the algorithm iterations. There are several annealing schedules used in the literature, one of the most commonly used annealing schedule is the geometric cooling schedule at which the current temperature is calculated as follows:

$$T(t) = T_o \alpha^i, \qquad i = 1, 2, ..., i_f \text{ and } 0 < \alpha < 1 \tag{4.18}$$

The final variable of the SA cooling schedule is the number of iterations per temperature. At each temperature a number of iterations must be allowed in order to allow the system to stabilize at this temperature through several exploration of the search space at approximately same acceptance ratio of worst solutions. The value of this variable could be either static or dynamic; usually a constant value is used through the algorithm.

The SA as any metaheuristic algorithm combines a greedy strategy to search for better solutions and a stochastic strategy in order to escape getting stuck in a local minimum. Therefore at each iteration of the SA the neighboring solution is directly accepted if and only if this neighboring solution quality is better than the previous solution which represents the greedy part of the SA. On the other hand, if the neighboring solution is worse than the current solution, this solution non-improving is probabilistically accepted which represents the stochastic strategy used to escape local optimum in the SA algorithm. The acceptance rate of worse solutions depends upon the current temperature of the algorithm. If the current temperature is high, the acceptance rate is high which means more worse solutions would be accepted giving the algorithm the chance to explore the search space of the algorithm and to escape local minimum. On the other hand, if the current temperature is low, the acceptance rate is low which means that less worse solutions would be accepted giving the algorithm the chance to intensify the search in neighborhoods of elite solutions where there is a higher potential of finding the global minimum.

Algorithm 2 is the proposed algorithm used to solve the MRTA problem in this thesis using the SA metaheuristic algorithm.

4.3.2 Population-based Approach

The second introduced algorithm is the Genetic Algorithm (GA) which is an evolutionary algorithm of the population-based family of metaheuristic algorithms. Population-based is the family of algorithms that iteratively transforms a population of solutions throughout the algorithm in order to generate a new population of solutions in the aim of finding the optimal solution.

The inputs to the algorithm are the list of the available travelling salesmen or robots with their defined features, a list of all cities or tasks with their defined features and a list of the distances between the tasks current locations. At the start of the GA, a population of solution with a certain population size must be constructed; the population is a list that includes a number of feasible solutions. A very big population size of the GA usually does not improve the performance of the algorithm and was reported to consume much time till reaching the optimal results. In the literature, several variations of the population size were introduced that includes dynamic changing of the population size throughout the algorithm iterations. Researchers also suggested that the population size of the algorithm should depend upon the size of the list representing the solution. In the proposed approach the population size is constant throughout the algorithm, and at the start of the algorithm the population is filled with completely random solutions as previously explained in the SA algorithm 2.

As the GA progress, the population is transformed into a next generation through applying operators that are patterned after naturally occurring genetic operations such as the crossover and the mutation operators. The crossover operator in GA is the mimicking of the biological recombination between chromosomes, when some portion of the genetic material is swapped between chromosomes producing a new offspring chromosome. A number of well-known used crossover operators are the 1-point, n-point and the uniform crossover operators which are used with binary coded solutions. While the single arithmetic, simple arithmetic and whole arithmetic are used with real value solutions. The second genetic operator is mutation, which is the introducing of a new genetic material into an existing chromosome to impose a change in the property of this chromosome. Mu-

Algorithm 2: SA-based MRTA

```
Input: Tasks list $tasks$, Robots list $robots$, Distances between tasks $distances$
Output: Optimal allocation $optAlloc$
1  Define: Initial temperature $inTemp$, Final temperature $finTemp$, Iterations per
   temperature $iterTemp$, Current temperature $currTemp$, Geometric coefficient $\alpha$,
   Transition probability $transProb$, Current allocation $currAlloc$, Neighbor
   allocation $neighborAlloc$, Current cost $currCost$, Neighbor cost $neighborCost$,
   Optimal cost $optCost$
2  $currAlloc \leftarrow$ generateValidSolution($tasks$, $robots$, $distances$)
3  $currCost \leftarrow$ getAllocationCost($currAlloc$)
4  $optCost \leftarrow currCost$
5  while $currTemp < finTemp$ do
6      for $i \leftarrow 1$ to $iterTemp$ do
7          $neighborAlloc \leftarrow$ generateNeighborSolution($currAlloc$)
8          $neighborCost \leftarrow$ getAllocationCost($neighborAlloc$)
9          if $neighborCost < currCost$ then
10             $currAlloc \leftarrow neighborAlloc$
11             $currCost \leftarrow neighborCost$
12             if $neighborCost < optCost$ then
13                 $optAlloc \leftarrow neighborAlloc$
14                 $optCost \leftarrow neighborCost$
15             end
16         else
17             Generate: A random number $randNum$
18             $transProb = \exp(-\frac{neighborCost - currCost}{currTemp})$
19             if $transProb > randNum$ then
20                 $currAlloc \leftarrow neighborAlloc$
21                 $currCost \leftarrow neighborCost$
22             end
23         end
24     end
25     $currTemp = currTemp * \alpha$
26 end
```

tation in binary encoded solutions can be done through flipping the value of a randomly chosen binary bit in the solution list for binary encoded solutions while for real value solutions mutation can be done through the introduction of random Gaussian noise.

Since the MRTA problem in consideration has been previously formulated into an instance of the mTSP and therefore the solution is represented as a permutation of robots and tasks. Therefore the previously discussed mutation operators could not used in the proposed approach. Instead, four mutation operators were implemented and used, which are the same operators used in the SA algorithm to generate neighboring solutions. The mutation operators are the swapping, deletion and insertion, inversion and scrambling operators previously explained in Figure 4.3. Similarly the crossover operators previously discussed cannot be used with permutation problems and thus were not implemented in the proposed approach and therefore two other crossover methods were implemented. The two implemented crossover operators are the partially mapped crossover and the order crossover. They are both genetic crossover operators used for permutation problems.

At each iteration of the GA proposed in this thesis, a number of solutions from the current population (parents list) must be chosen for crossover or mutation in order to generate the offsprings (children list). Since the GA mimics the genetic reproduction process and is based on the concepts of natural selection, therefore in this algorithm the crossover operator is applied over the best parents hoping to produce more fit solutions of higher quality while the mutation operator is applied on less fit parents in the aim of producing a more fit solution through the mutation of the less fit one. The crossover rate and mutation rate are used to decide the number of parents that are going to be used for crossover and the number of parents that are going to be used for mutation respectively. After several testing of both the crossover and the mutation operators in the proposed algorithm, it was found that the mutation operator had a lower probability of exploring the search space while the crossover had a higher probability of diversifying and exploring the search space of the problem. Based on this conclusion, the crossover rate and mutation rate had been iteratively set through various runs of the algorithm in order to reach the most suitable results. After generating the new offspring solutions through applying the crossover and the mutation operators, the solutions are grouped together in the children list.

The new population is the list of solutions that survived from the current iteration to be used as the parents list in the next iteration of the algorithm. This new population is composed of three types of solutions, which are the best solutions in parents list, the best solutions in the children list and some random solutions. Since the GA is one of the metaheuristic algorithms that attempt to reach a near optimal solution in relatively short time, therefore the proposed GA had to incorporate two search strategies, an explorative and an exploitative strategy. The explorative strategy is used to allow the algorithm to explore the vast search space of the problem and to prevent the algorithm of prematurely converging to sub optimal results, while the exploitative strategy is used to allow the algorithm to intensify the search for the optimum solution within high quality solutions. Those strategies had been implemented through dividing the number of iterations of the algorithm into four quarters.

In each quarter, the algorithm will apply both the explorative and the exploitative strategies on the search space of the problem being solved but with different ratios. In the first quarter, the next population will consist of 20 % of the best solutions in the parent list and 20 % of the best solutions in the children list and the remaining 60 % of the new population will be newly generated random solutions. During the second quarter, the next population will consist of 30 % of the best parents and 30 % of the best children and the remaining 40 % is newly generated random solutions as the first quarter. It can be noticed that in the first two quarters that the explorative and exploitative features are both applied. Exploration of the search space is realized through the acceptance of a large number of newly generated random solutions to the next population, while the exploitative feature is realized through the addition of some of the fittest solutions from both the parents and the children lists. Although both features were implemented, however the explorative feature is more dominating in the first two quarters in order to be capable of appropriately exploring the search space of the problem in the aim of finding the optimal solution and to avoid premature convergence of the solution.

In the third quarter, 40 % of the best parents and 40 % of the best children are added to the next population, the remaining 20 % of the next population includes a combination of less fit solutions from both the parents list and the children list, as well as newly generated random solutions. In this quarter, the greediness of the algorithm increased relative to the previous two quarters, and therefore the algorithm tends to be more exploitative in this quarter. On the other hand, the algorithm still realizes the explorative feature through accepting less fit solution from both the parents list and the children list or newly generated random solutions and adding these solutions to the next population. In the fourth and last quarter, the algorithm becomes completely greedy and thus the next population only includes 50 % of the best parents and 50 % of the best children. During this last quarter the algorithm is intensifying the search for the optimal solution within the best elite solutions without accepting any less fit or any randomly generated solutions.

Algorithm 3 is the proposed algorithm used to solve the MRTA problem in this thesis using the GA.

4.4 Concluding Remarks

In this chapter, the mTSP was introduced and discussed. Moreover an extension of the mTSP problem formulation was proposed to be used to formulate the MRTA. The solution construction method, the objective function and the constraints of the problem were also presented in this chapter. Finally a trajectory-based algorithm, SA and an evolutionary-based algorithm, GA were proposed to solve the MRTA problem.

Algorithm 3: GA-based MRTA

```
   Input: Tasks list $tasks$, Robots list $robots$, Distances between tasks $distances$
   Output: Optimal allocation $optAlloc$
1  Define: Parents list, $parentList$, Children list $childList$, Next generation list
   $nxtGenList$, Number of iterations $numIter$, Elitism percent $elitismPerc$,
   Population size $popuSize$, Current allocation $currAlloc$, Optimal allocation
   $optAlloc$
2  for $i \leftarrow 1$ to $popuSize$ do
3      $parentList \leftarrow$ generateValidSolution($tasks$, $robots$, $distances$)
4  end
5  $currAlloc \leftarrow$ Minimum($parentList$)
6  for $i \leftarrow 1$ to $numIter$ do
7      if $i < 25\%$ of $numIter$ then
8          $elitismPerc = 20\%$
9      else if $i < 50\%$ of $numIter$ AND $i > 25\%$ of $numIter$ then
10         $elitismPerc = 30\%$
11     else if $i < 75\%$ of $numIter$ AND $i > 50\%$ of $numIter$ then
12         $elitismPerc = 40\%$
13     else
14         $elitismPerc = 50\%$
15     end
16     $childList \leftarrow$ crossover(The minimum 10% of $parentList$)
17     $childList \leftarrow$ mutation(The maximum 90% of $parentList$)
18     $nxtGenList \leftarrow$ The minimum $elitismPerc$ of $parentList$
19     $nxtGenList \leftarrow$ The minimum $elitismPerc$ of $childList$
20     for $j \leftarrow 1$ to $(100\% - elitismPerc \times 2)$ of $numIter$ do
21         $nxtGenList \leftarrow$ generateValidSolution($tasks$, $robots$, $distances$)
22     end
23     if Minimum($parentList$) $< currAlloc$ then
24         $currAlloc \leftarrow$ Minimum($parentList$)
25     end
26     $parentList \leftarrow nxtGenList$
27 end
28 $optAlloc \leftarrow currAlloc$
```

Chapter 5

Experimental Results

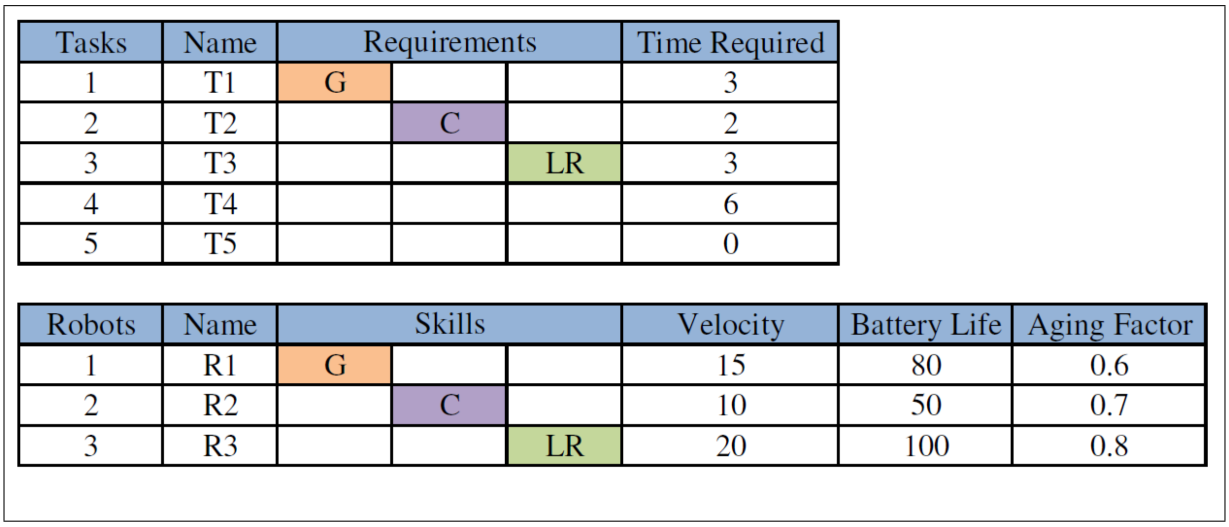
In Chapter 4 SA-based and GA-based approaches were proposed to solve the MRTA problem. In this chapter the proposed approaches are extensively tested over a number of different MRTA problem scenarios where the testing results are reported in detail. In section 5.1 the experiment setup is introduced and explained. The evaluation metrics used to report the quality of the algorithms are explained in section 5.2. The MRTA problem scenarios used to test the algorithms are presented in section 5.3. In section 5.4 the results of both algorithms are reported in detail and a comparative study between the proposed algorithms is conducted. Finally in section 5.5 a conclusion of the chapter is given.

5.1 Experiment Set up

In order to test the proposed algorithms, a number of MRTA problem scenarios are presented. In each scenario both algorithms are applied and the results of both algorithms are reported through the evaluation metrics. Any MRTA problem scenario has three main inputs which are the robots, the tasks and the distance matrix between the tasks. Since the planning of the robots paths throughout the environment is out of the scope of this thesis, thus it is assumed that the shortest free of obstacle path between the tasks is already given in the distance matrix. The expected output for solving the MRTA problem is the optimal allocation that maximizes the performance of the system and minimizes the cost.

5.1.1 Experiment Inputs

As previously mentioned, the inputs of any MRTA problem scenario are the robots, the tasks and the distance matrix between the tasks. Since in the formulation phase previously discussed in chapter 4, each robot and each task has been previously defined with a number of features to describe. Therefore it is a must to consider the features of both the robots and the tasks as inputs to the experiment as well. In the experimentation phase of this thesis, all the inputs of any scenario are defined in a Microsoft Excel sheet file. When starting the experiment, the testing environment starts by reading all the information found on the Excel sheet and then load it to predefined variables in the test environment such that all the defined inputs can be used throughout the algorithm without going back to the Excel sheet file. In Figure 5.1 an example of the Excel sheet file used is presented where the inputs are defined.



Tasks	Name	Requirements			Time Required
1	T1	G			3
2	T2		C		2
3	T3			LR	3
4	T4				6
5	T5				0

Robots	Name	Skills			Velocity	Battery Life	Aging Factor
1	R1	G			15	80	0.6
2	R2		C		10	50	0.7
3	R3			LR	20	100	0.8

Figure 5.1: Example of Inputs in Excel Sheet

It can be noticed that the inputs are the tasks with their defining features which are the physical requirements and the time requirements. The other input is the robots with their defining features which are the physical capabilities (skills), their velocities, the battery life and the aging factor. It must be mentioned that in the undertaken experiments the physical capabilities of the robots as well as the requirements of the tasks are only limited to one or a combination of more than one of the three additional skills which are the gripper, the camera and the laser range finder which are denoted by G,C and LR respectively in the inputs table in the Excel sheet. In order to ensure the consistency of units in the calculations it is assumed that all the distances between the tasks are in meters while the velocities of the robots are in meter per minute and finally the time requirements of the tasks are in minutes.

It was first proposed that all the inputs are given to the testing environment through a graphical user interface panel. But for more simplicity and flexibility the inputs were defined in an Excel file so that it can be faster to edit any detail concerning the inputs of the scenario or to easily alter the whole scenario through adding and deleting the inputs defined in the Excel sheet file. This can be easily done by adding a new robot to the table of robots or a new task to the table of tasks or changing any of the values of the features defining both the robots and the tasks in Figure 5.1.

5.1.2 Implementation and Testing Environment

The two proposed algorithms SA and GA are implemented using Java. Java is not only used in the implementation of both algorithms, but also used to test them through applying both algorithms over the proposed scenarios and calculating the results of the evaluation metrics for both algorithms. The reported results of the evaluation metrics in this thesis are the average values of 10 runs. All experiments are conducted on a Microsoft Windows operating system on a device whose specifications are presented in table 5.1.

Table 5.1: System Specifications

Processor	1.70 GHz
Installed Memory (RAM)	6.00 GB
System Type	64-bit Operating System

The SA and GA parameters were tuned experimentally using an iterative method. Both algorithms were applied over Berlin-52 problem which is a well-known traveling salesman testbench problem taken from TSPLIB [93]. Different experiments were carried on where SA and GA were applied several times on the Berlin-52 problem while changing the parameters of both algorithms. After an enormous number of experiments the numerical values reported in Table 5.2 and in Table 5.3 were used as the parameters for both the SA and the GA respectively.

Table 5.2: SA Parameters

Initial Temperature	Final Temperature	Iterations per Temperature	α
10^4	10^{-4}	1000	0.85

Table 5.3: GA Parameters

Population Size	Number of Iterations
20	4×10^3

Both algorithms were tested over the Berlin-52 with these parameter setting and the results were reported in Table 5.4. These promising results for the Berlin-52 problem indicates that these parameter settings for the SA and the GA will have the probability of yielding prominent results when applied for solving the MRTA problem.

Table 5.4: SA and GA Results for the Berlin-52 Problem

Berlin-52 Optimal	7542
Berlin-52 SA Best of 10 runs	7544
Berlin-52 SA Average of 10 runs	7876
Berlin-52 GA Best of 10 runs	7916
Berlin-52 GA Average of 10 runs	8181

Since one of the main objectives of the proposed approach in this thesis is to be generic and can be used to solve any MRTA problem. Therefore, the parameter settings reported in Table 5.2 and Table 5.3 were used as the default settings for the SA and the GA when used to solve the MRTA problems in the different test scenarios of this work.

5.2 Evaluation Metrics

In section 5.3 the proposed algorithms are tested over a number of scenarios that include different instances of the MRTA problem. In order to be able to test the quality of the proposed algorithms, four evaluation metrics were used. Each one of both algorithms was used to solve the proposed problem in each scenario and then the results and the quality of the solution provided was evaluated through the four evaluation metrics. The four metrics used to test both algorithms are as follows:

- The optimal allocation found in terms of the objective function [*OptAllocCost*]
- Time taken to optimal allocation [*AvgTime*]
- Deviation of the optimal allocation cost from random solution cost [*DevRandom*]
- Deviation from optimal allocation after a certain percentage of maximum time [*DevAlloc*]

The first evaluation metric is the cost of the optimal allocation found which is the best solution in terms of the objective function previously explained in chapter 4. The optimal allocation reported for each problem in the thesis is the average of the optimal allocations found after 10 runs of the applied algorithm to solve the tackled problem. Although the actual optimal allocation (ground truth) is not known for the tackled problems, however the proposed metric is important to indicate the best solution found by the algorithm and therefore gives an indication of the quality of solutions provided by the algorithm. The optimal allocation is the solution that minimize the objective function previously explained in equation (4.17).

The second metric is the average time required by the algorithm until finishing the total number of iterations. When the total number of iterations of the algorithm is reached, the algorithm stops and the elapsed time is reported. It was one of the main objectives of this thesis to introduce an approach that is capable of solving the MRTA problem and does not consume much time to reach the optimal or near optimal acceptable results. Therefore the time metric is an important factor in measuring the quality of the proposed approaches.

The third metric is the deviation of randomly generated solutions form the optimal allocation found by the proposed algorithms. Although the optimum solution found by each algorithm was reported in the first metric, however the optimal allocation (ground truth) for each problem being solved is unknown. Thus the opportunity to compare the results provided by the proposed algorithms with the ground truth could not be achieved. Therefore it was important to consider the deviation of randomly generated solutions from the optimal solution found by the proposed algorithms. This was done to compare the results of the proposed algorithms with the worst case scenario which is not applying any optimization algorithm on the problem being solved, and hence to be able present the quality of the solutions provided by the proposed algorithms. The random solution reported is the average of 10 random solutions *RandomCost*. The deviation is calculated in equation (5.1).

$$DevRandom = \frac{RandomCost - OptAllocCost}{OptAllocCost} \tag{5.1}$$

The last metric represents the deviation from optimal allocation of the cost of solutions provided by the algorithm after a certain percentage of the time taken to reach the optimal allocation *CostAtPercMaxTime*. For example, if one of the proposed algorithms requires 30 seconds till reaching the optimal allocation of a certain problem. Thus the deviation metric will measure how much a solution provided by this algorithm is far from the optimal allocation when generated after 10 seconds instead of 30 seconds. The importance of this evaluation metric appears in the context of dynamic task allocation in real world applications. In such applications, the MRTA problem must be solved several times in real time during the application since it is not sufficient enough that the algorithm provides only an initial single allocation at the start of the application. Therefore if the algorithm has enough time to generate an optimal allocation at the start of the application, it may not have that amount of time to generate an optimal allocation during the run time of the application. Therefore it is a must to measure how far is the solution provided by the algorithm from the optimal allocation if there was no enough time to be given to algorithm. In this thesis, the deviation from optimal allocation is calculated for solution generated after 10%, 30% and 50% of the maximum time taken to reach the optimal allocation. The deviation is calculated in equation (5.2).

$$DevAlloc = \frac{CostAtPercMaxTime - OptAllocCost}{OptAllocCost} \tag{5.2}$$

5.3 Evaluation Scenarios

In this section a number of test scenarios are used to test the algorithms proposed in this thesis. Before getting into the results of the different test scenario, a simple demonstration is made to show how a MRS problem can be modeled in order to solve the task allocation problem of this problem using the approaches presented in this thesis. In Figure 5.2 a simple search and rescue scenario is presented.

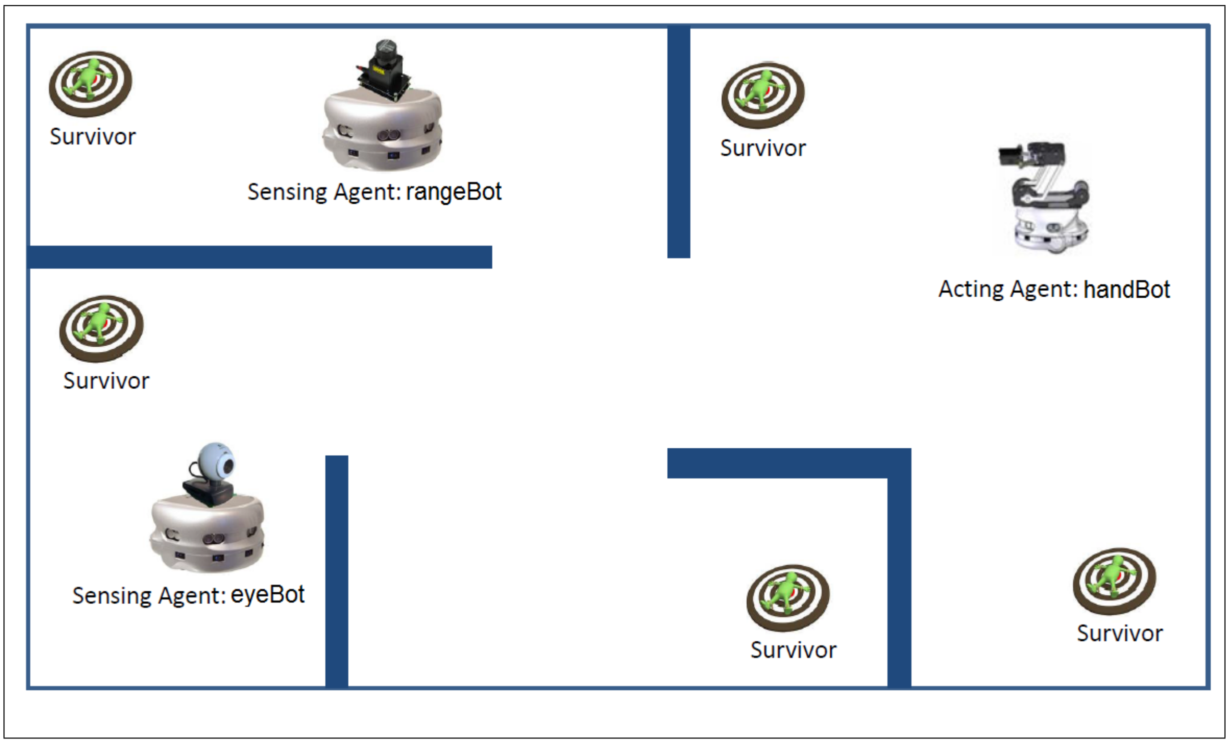
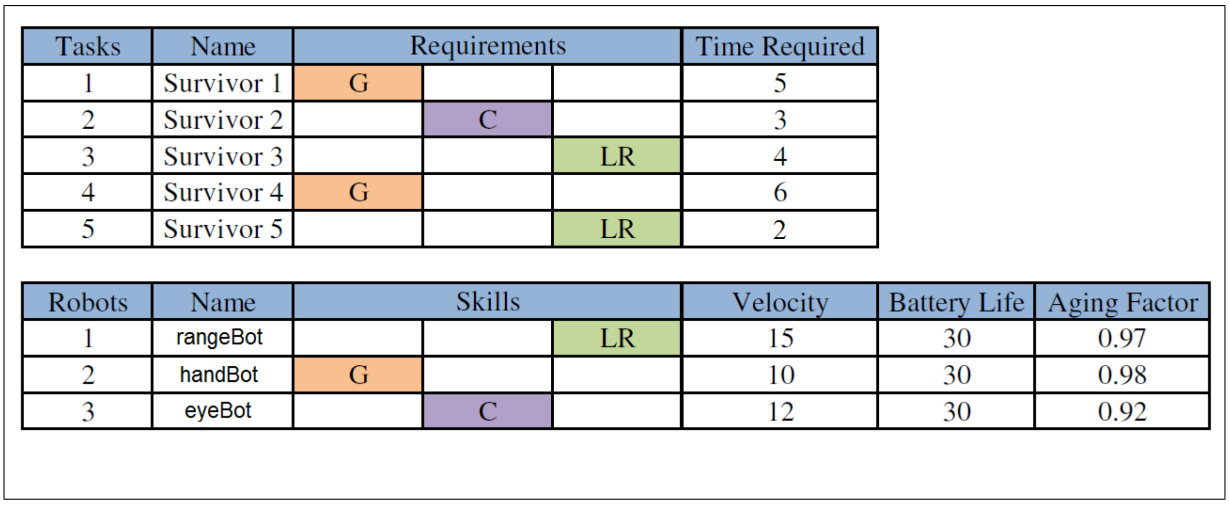


Figure 5.2: MRS search and rescue scenario

In this search and rescue scenario, five survivors are found in the search area. These survivors have different requirements. Also a number of robots are found in the area of the disaster where the search and rescue mission should be accomplished. The number of robots is limited to only three robots where each robot has different sensing and acting capabilities. In this problem, the robots used are Khepera III robots, where the first robot is equipped with a laser rangefinder and denoted the name rangeBot, the second robot is equipped with a gripper and denoted the name handBot, the third and last robot is equipped with a camera and denoted the name eyeBot. The search and rescue problem presented in Figure 5.2 must be modeled in order to be able to find the optimal allocation of each task (survivor) to the available robots. The model of this problem is illustrated in Figure 5.3.



Tasks	Name	Requirements			Time Required
1	Survivor 1	G			5
2	Survivor 2		C		3
3	Survivor 3			LR	4
4	Survivor 4	G			6
5	Survivor 5			LR	2

Robots	Name	Skills			Velocity	Battery Life	Aging Factor
1	rangeBot			LR	15	30	0.97
2	handBot	G			10	30	0.98
3	eyeBot		C		12	30	0.92

Figure 5.3: Search and rescue problem model

The five survivors in the problem are modeled as five tasks where each task has both physical and time requirements. The robots are also modeled with their physical skills, operating velocities, their current energy level and their efficiencies. After modeling of the problem is done, the next step is to pass the input file of the problem modeling to the proposed algorithms presented in this thesis. The expected output of the algorithms is shown in Figure 5.4 which represents the optimal allocation of the tasks to the robots.

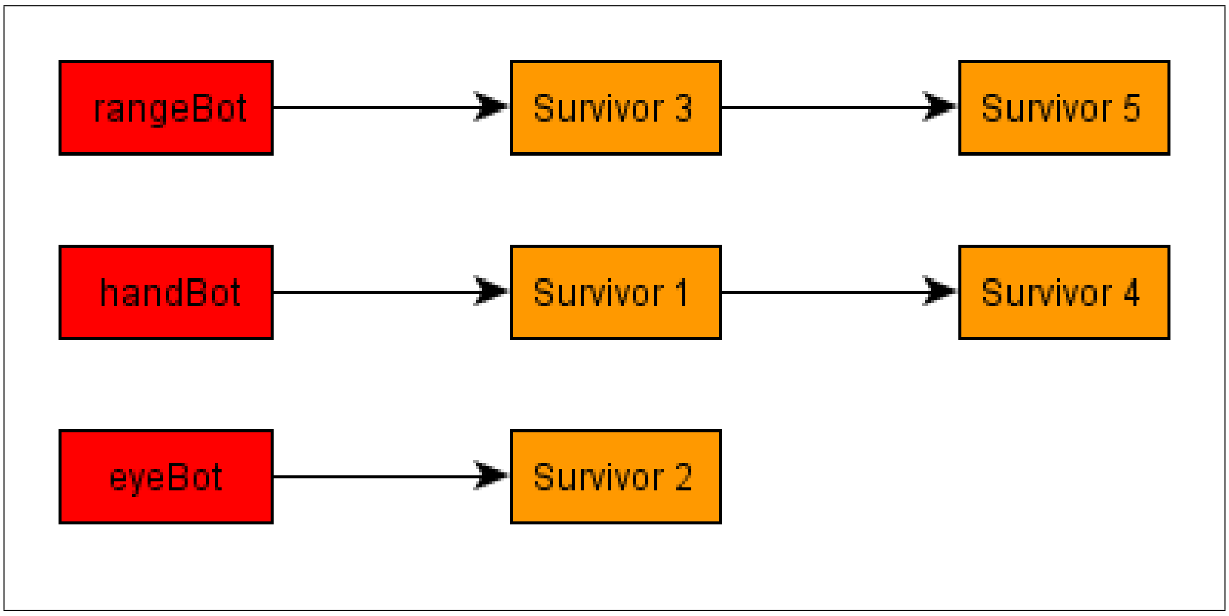


Figure 5.4: Search and rescue problem task allocation

As it can be seen in the Figure that survivor 3 and survivor 5 that required a laser rangefinder were assigned to the rangeBot robot while survivor 1 and survivor 4 that required a gripper were assigned to the handBot and finally survivor 2 that required a camera was assigned to the eyeBot. In Figure 5.5 a time chart of the optimal allocation found is illustrated.

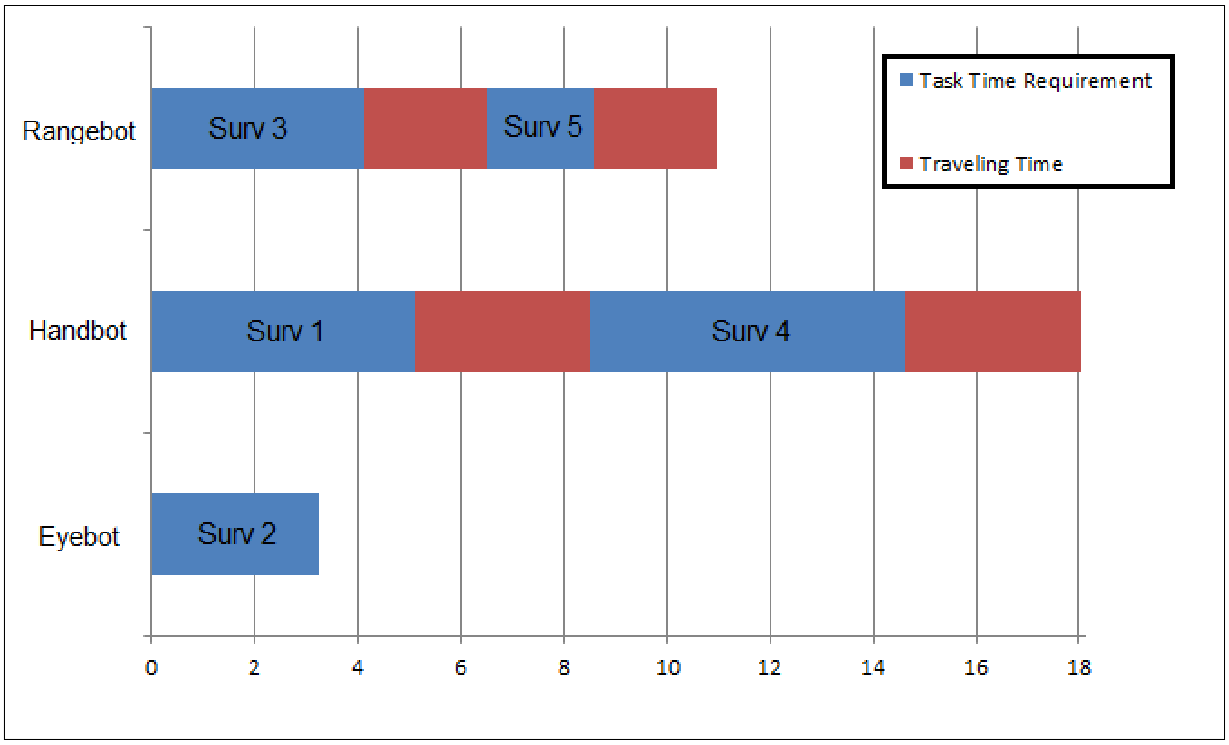


Figure 5.5: Search and resecure allocation time chart

The time chart in Figure 5.5 shows the time taken by each robot to finish its assigned tasks. Also the total time taken for the whole mission to be accomplished will be the maximum time of the three robots since the robots will be working in parallel. It can be noticed that the time taken by each robot includes both the time taken for finishing the assigned tasks as well as the traveling time between the tasks.

5.3.1 Scalability Test Scenarios

As previously discussed in chapter 2, MRS have a higher potential of being used in various and different applications where these targeted applications are of a complex nature. Also it must be noticed that the complexity of the targeted problems as well as the complexity of the multi-robot teams is increasing with time. And since one of the main sources of complexity of MRS applications is the size of the robot team as well as the number of tasks to be executed. Therefore it was essential when solving the MRTA problem to propose an algorithm that is capable of handling complex MRTA problems of large number of tasks and robots.

In this section the scalability of the algorithm is under question. Analysis of the scalability of the algorithm is done through applying both the SA and the GA over a number of scenarios at which the number of tasks and the number of used robots increases over each scenario. The evaluation metrics previously defined in section 5.2 are used to report the quality of the proposed algorithms when solving the MRTA problems suggested in the scalability test scenarios. It must be mentioned that in these scenarios there are no constraints applied on the domain of the problems being solved. The main reason of eliminating the constraints is to eliminate any other source of complexity to the problem rather than the increased number of robots and tasks. Where eliminating other sources of complexity to the problem enables the adequate testing of only the scalability of the proposed algorithms with no other factors affecting the problem.

5.3.1.1 Small-scale MRTA Problem

The first MRTA problem scenario is a small-scale problem. The number of tasks and robots in this scenario is five and three respectively. The maximum distance between the tasks is 54 meters, the task with the maximum time requirement needs 12 minutes. From the reported numbers it can be concluded that the tackled problem is a small-scale MRTA problem with limited number of tasks and robots. After applying the SA and the GA on this small-scale problem, the evaluation metrics were used to test the quality of both algorithms. The results of applying both algorithm are reported in Table 5.5.

Table 5.5: Small-scale MRTA scenario Results

Algorithm	*OptAllocCost*	*AvgTime*	*DevRandom*	*DevAlloc 10%*	*DevAlloc 30%*	*DevAlloc 50%*
SA	12.76	**1.04**	167%	3%	0%	0%
GA	12.76	1.58	167%	**0**%	0%	0%

The reported results show that both algorithms converged to the same allocation cost while the total time consumed by the SA is less than the GA. The deviation of the optimal allocation found from random allocation is roughly 170% for both algorithms which show that both algorithms succeeded in enhancing the quality of the randomly found solutions. The GA provided improved allocations than SA when both algorithms were applied for only 10% of the maximum time taken by each algorithm. Figure 5.6 and Figure 5.7 demonstrate the performance of SA and GA respectively versus number of iterations and time when both are used to solve the small-scale MRTA problem.

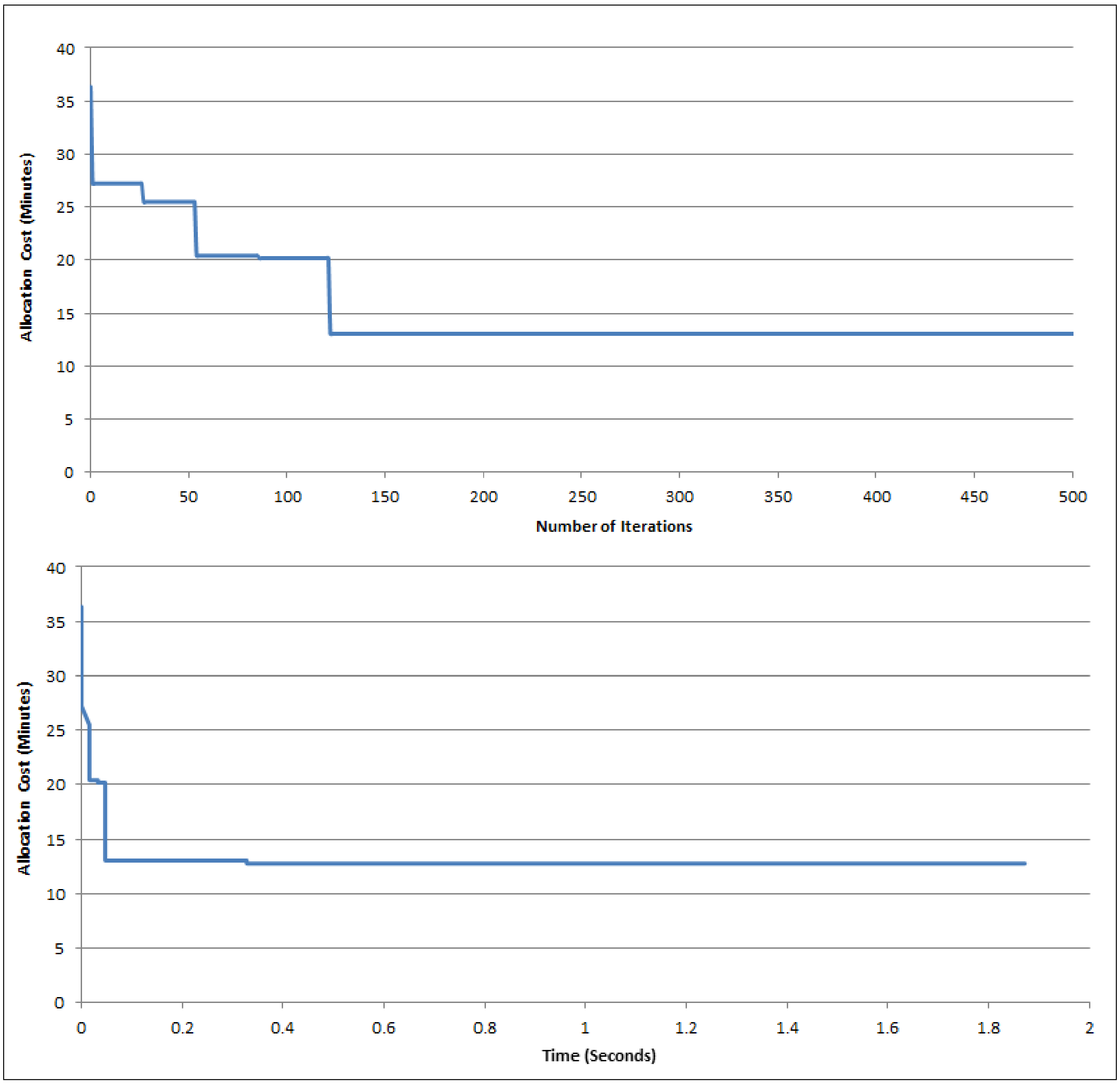


Figure 5.6: SA Performance for Small-scale MRTA Scenario

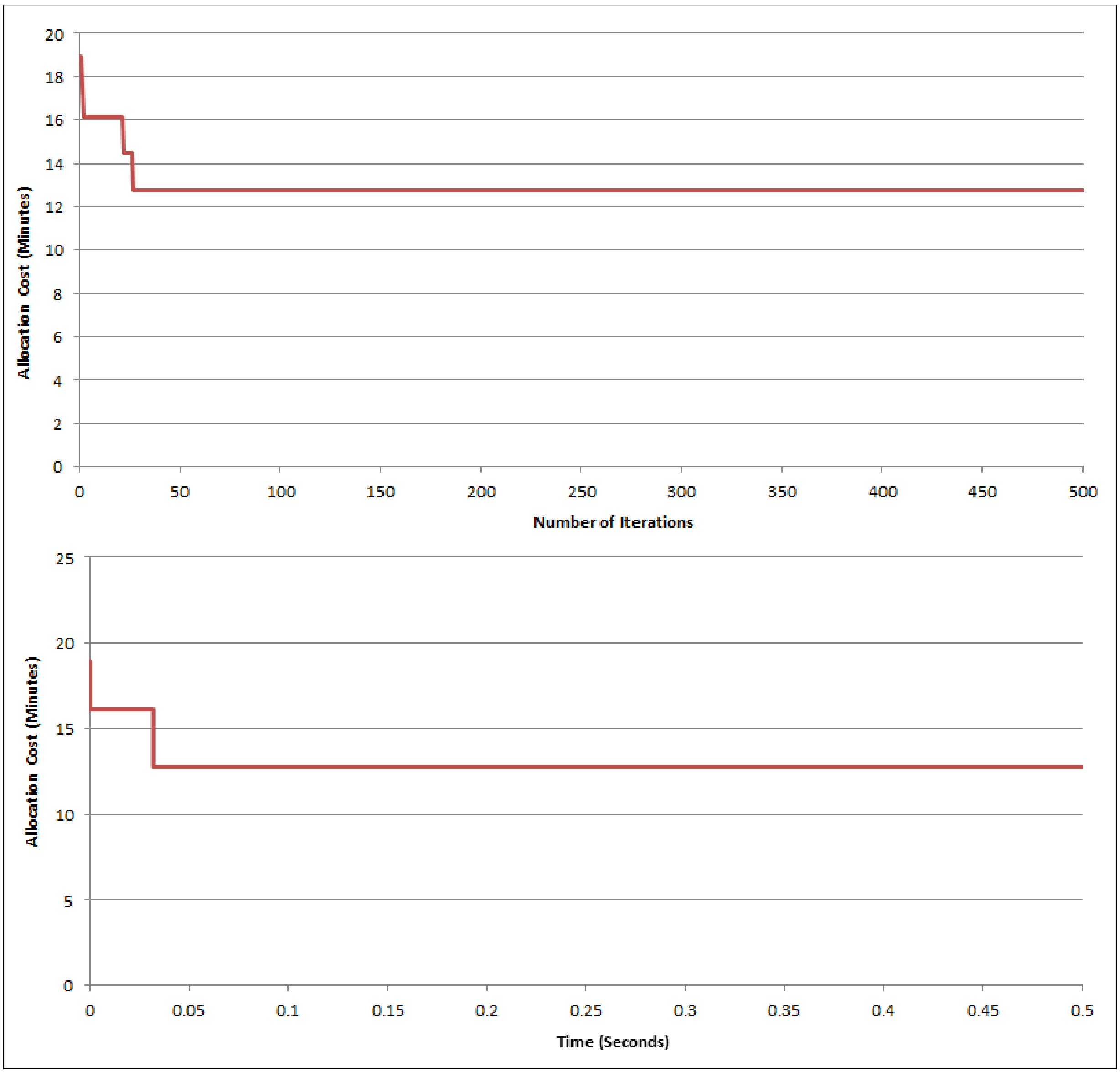


Figure 5.7: GA Performance for Small-scale MRTA Scenario

Finally, the deviation of the SA and GA from the optimal allocation when both are applied for only 10%,30% and 50% of the maximum time taken to reach the optimal allocation is demonstrated in Figure 5.8 and Figure 5.9.

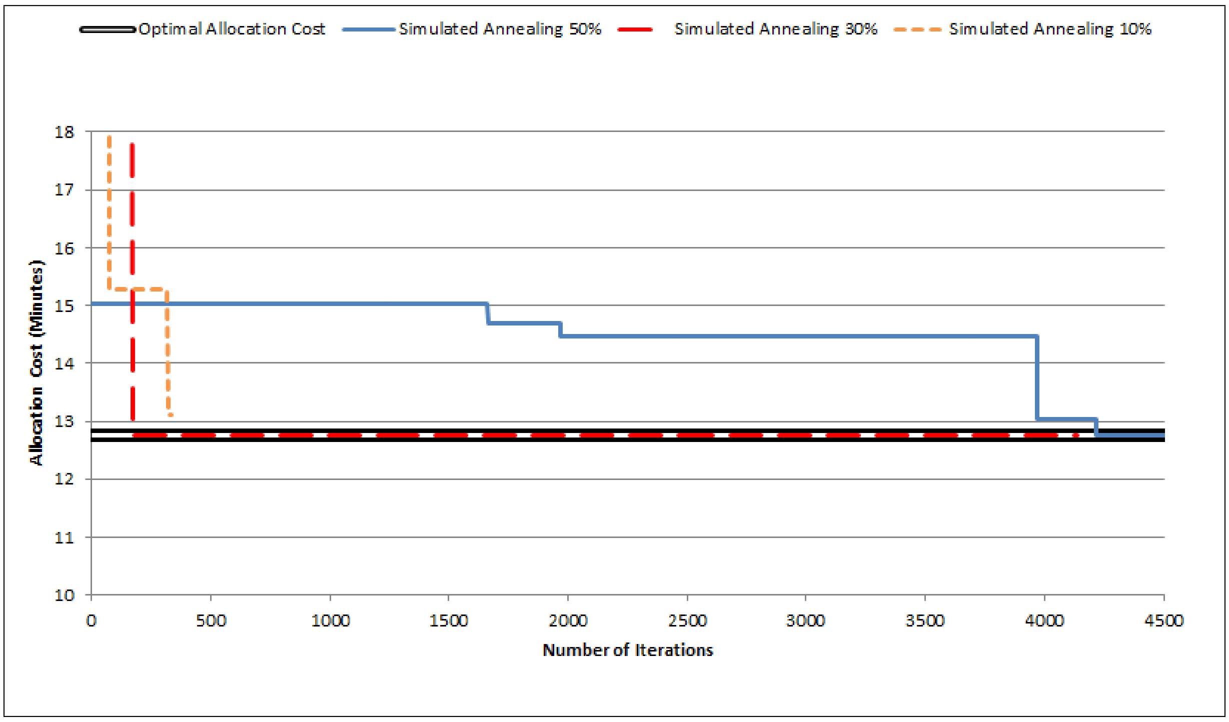


Figure 5.8: SA Deviation for Small-scale MRTA Scenario

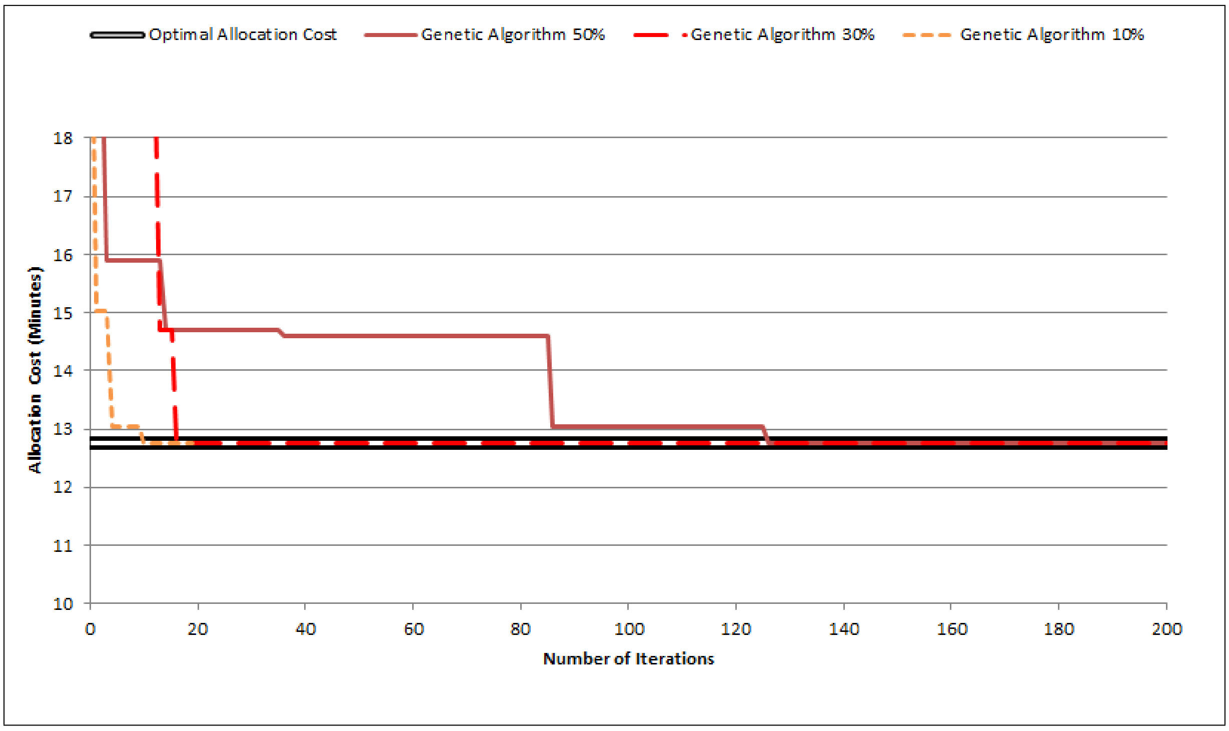


Figure 5.9: GA Deviation for Small-scale MRTA Scenario

5.3.1.2 Medium-scale MRTA Problem

The second MRTA problem scenario is a medium-scale problem. The number of tasks and robots in this scenario is fifteen and five respectively. The maximum distance between the tasks is 180 meters, the task with the maximum time requirement needs 18 minutes to be finished. From the reported numbers it can be concluded that the tackled problem is a medium-scale MRTA problem with intermediate number of tasks and robots that is relatively higher than the first scenario. The results of applying both algorithm are reported in Table 5.6.

Table 5.6: Medium-scale MRTA scenario Results

Algorithm	*OptAllocCost*	*AvgTime*	*DevRandom*	*DevAlloc 10%*	*DevAlloc 30%*	*DevAlloc 50%*
SA	**28.5**	**3.67**	**304%**	45%	28%	5%
GA	29.47	6.06	291%	**15%**	**6%**	**3%**

The reported result shows that SA provided improved allocations than GA when both algorithms reach the total number of iterations. This improved performance of the SA can also be noticed from the deviation of the allocation of each algorithm from random solution where the SA results roughly deviate 300% from random allocations while GA allocations deviates 290% from random allocations. This indicates that that the solutions of the SA are further away from random allocations than the solutions of the GA. Another improvement of SA over GA is that the total time consumed by the SA is less than the total time consumed by the GA till reaching the total number of iterations. On the other hand, the GA delivered better results than SA when both algorithms were applied for only 10%, 30% and 50% of the maximum time taken by each algorithm. This indicates that GA is capable of finding near optimal solutions earlier than SA.

Since the number of tasks and robots in the medium-scale MRTA problem increased over the small-scale problem and thus the problem complexity increased. Therefore the allocation cost and the total time consumed by each algorithm reported in Table 5.6 are higher than the values reported in Table 5.5. Although there is an increase in the total time consumed by each algorithm to reach the optimal results, however the reported time is still low and thus practical to be used in medium-scale MRS applications. Figure 5.10 and Figure 5.11 demonstrate the performance of SA and GA respectively versus number of iterations and time when both are used to solve the medium-scale MRTA problem.

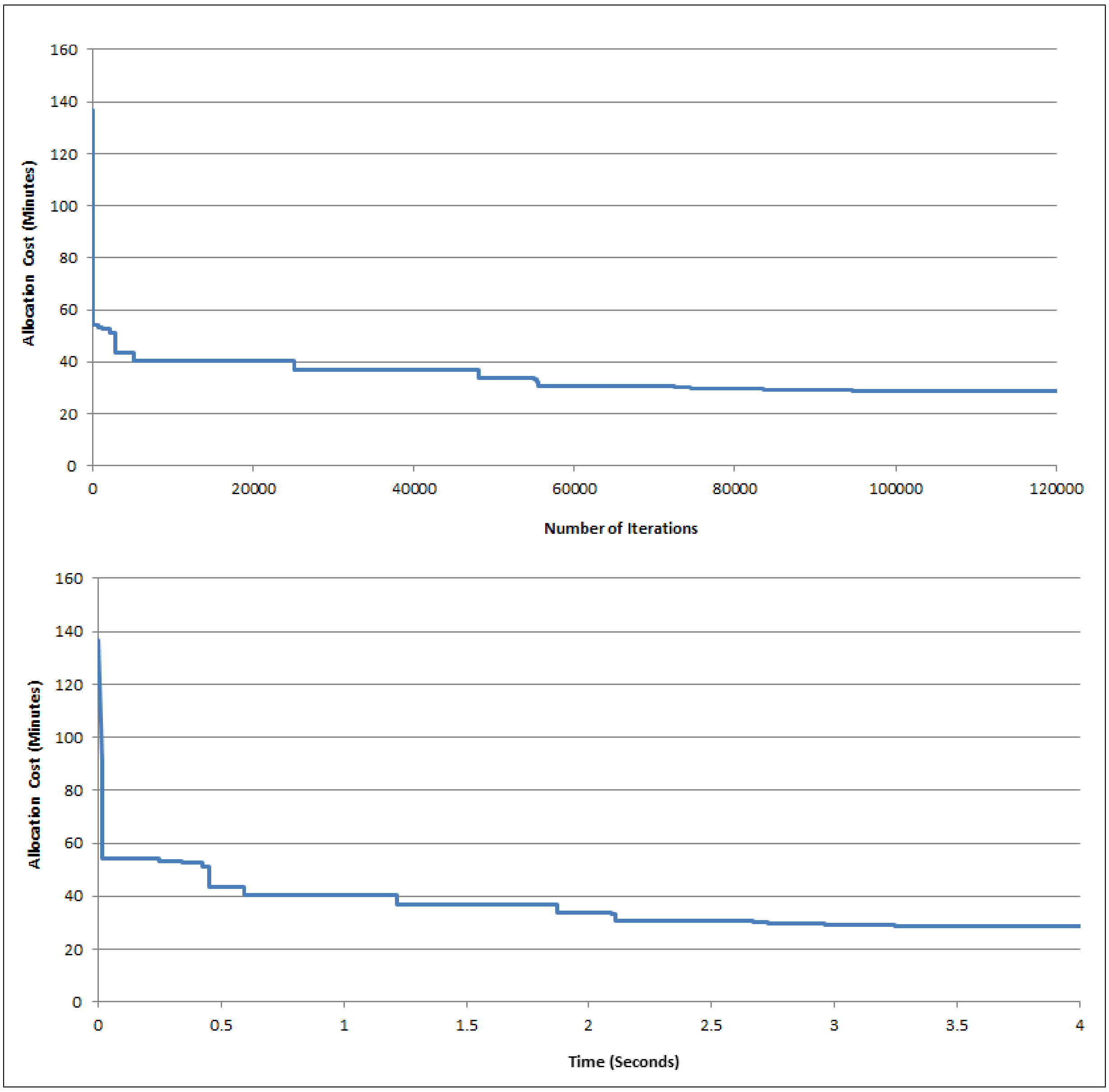


Figure 5.10: SA Performance for Medium-scale MRTA Scenario

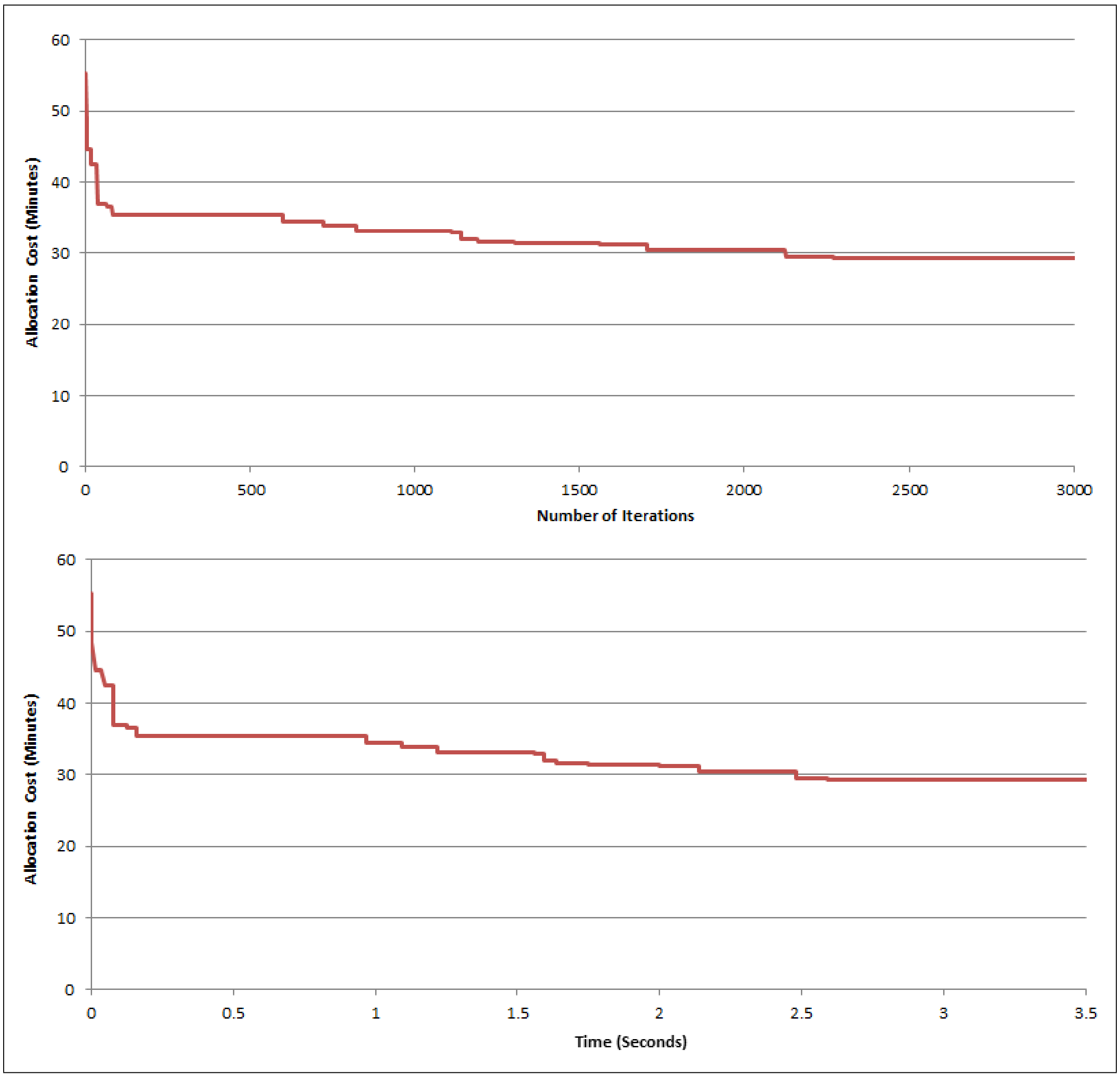


Figure 5.11: GA Performance for Medium-scale MRTA Scenario

Finally, the deviation of the SA and GA from the optimal allocation when both are applied for only 10%,30% and 50% of the maximum time taken to reach the optimal allocation is demonstrated in Figure 5.12 and in Figure 5.13.

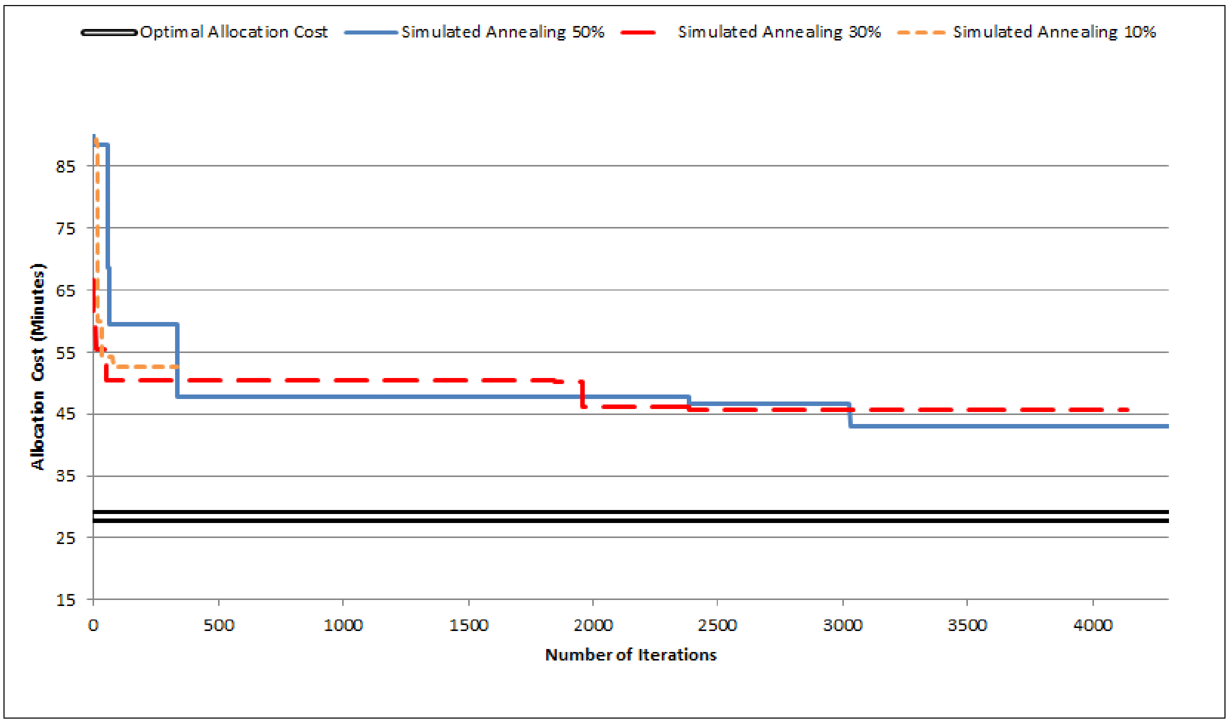


Figure 5.12: SA Deviation for Medium-scale MRTA Scenario

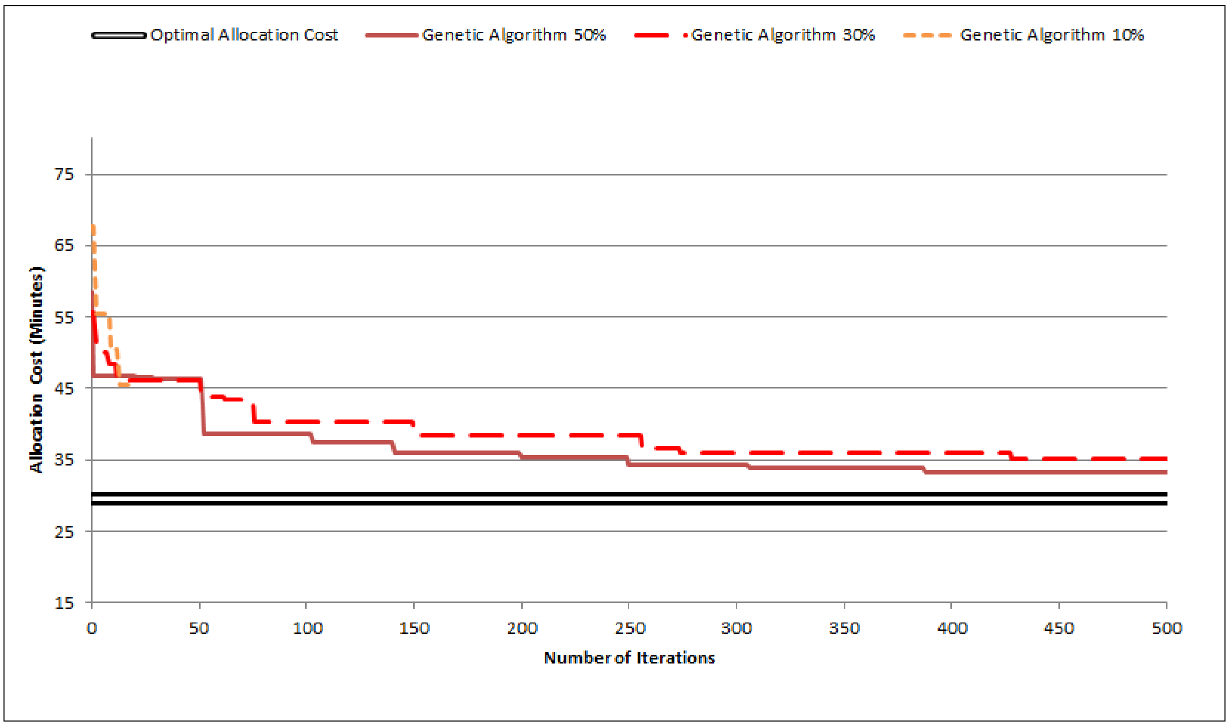


Figure 5.13: GA Deviation for Medium-scale MRTA Scenario

5.3.1.3 Large-scale MRTA Problem

The third MRTA problem scenario is based on the famous berlin52 traveling salesman benchmark problem and therefore this scenario is a large-scale problem. The number of tasks and robots in this scenario is fifty two and fifteen respectively. where the the maximum distance between the tasks is 1716 meters, the task with the maximum time requirement needs 99 minutes to be finished. From the reported numbers it can be concluded that the tackled problem is a large-scale MRTA problem with an extended number of tasks and robots that is much higher than both the first and the second scenarios. The results of applying SA and GA are both reported in table 5.7

Table 5.7: Large-scale MRTA Scenario Results

Algorithm	*OptAllocCost*	*AvgTime*	*DevRandom*	*DevAlloc 10%*	*DevAlloc 30%*	*DevAlloc 50%*
SA	**338.06**	**22.21**	**1116%**	116%	72%	20%
GA	378.82	39.1	985%	**45%**	**23%**	**14%**

The results reported in Table 5.7 shows that SA provided better allocation cost than GA in much less time. The difference in the consumed time between the two algorithms significantly increased relative to the previous scenarios due to the increase of the problem complexity. The results provided by SA and GA deviated by 1116% and 985% respectively which is also considered a significant increase in the deviation from random allocation in this scenario rather than other scenarios. These results reveals the importance of optimizing the allocation problem especially in large-scale MRS applications. Similar to the previous scenarios, GA outperformed SA in deviation from the optimal allocation when both algorithms were applied for only 10%, 30% and 50% of the maximum time taken by each algorithm. These results show that GA will probably provide better results than SA when there is no enough time for the algorithm to reach the total number of iterations. Figure 5.14 and Figure 5.15 demonstrate the performance of SA and GA respectively versus number of iterations and time when both are used to solve the large-scale MRTA problem.

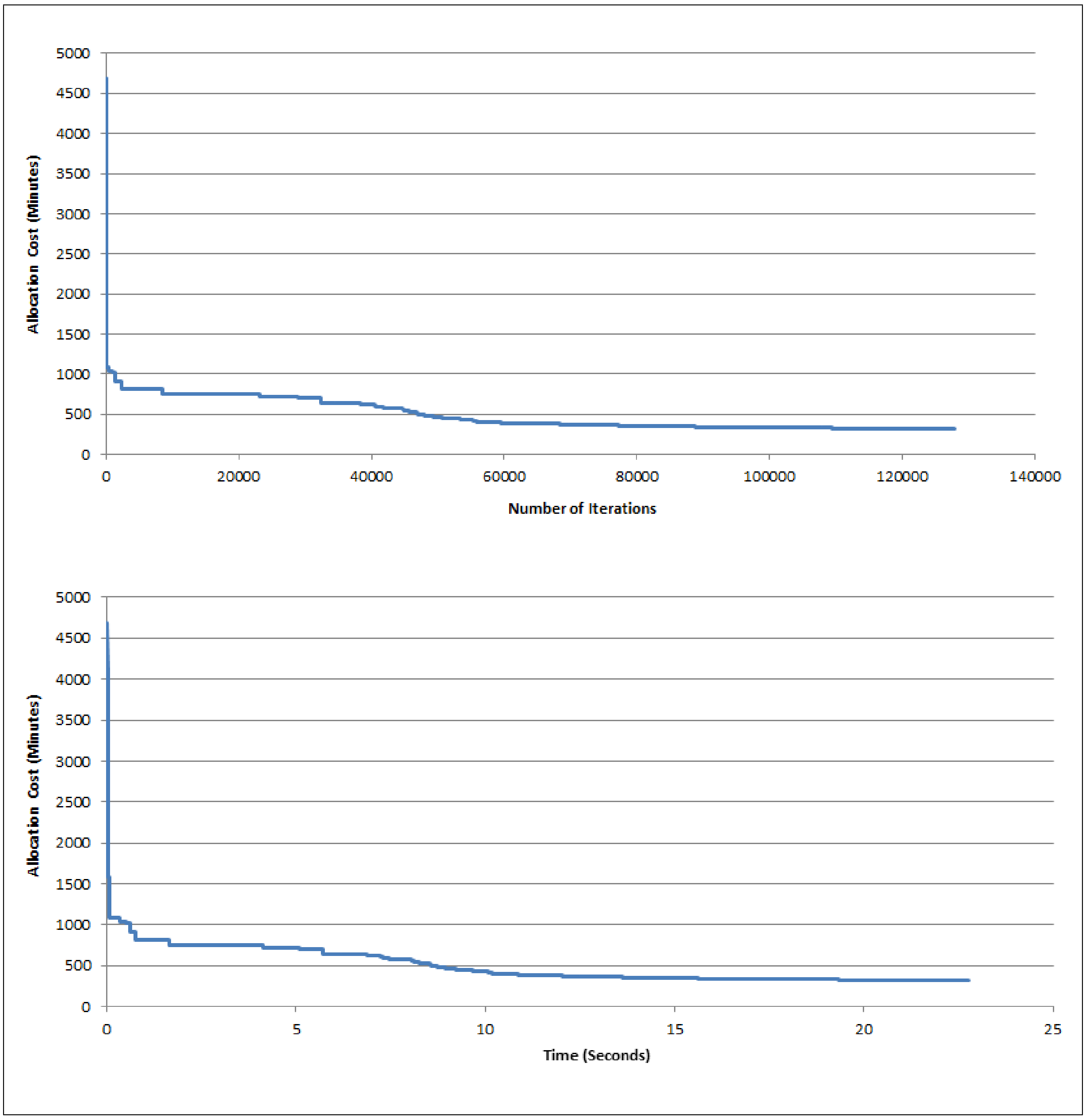


Figure 5.14: SA Performance for Large-scale MRTA Scenario

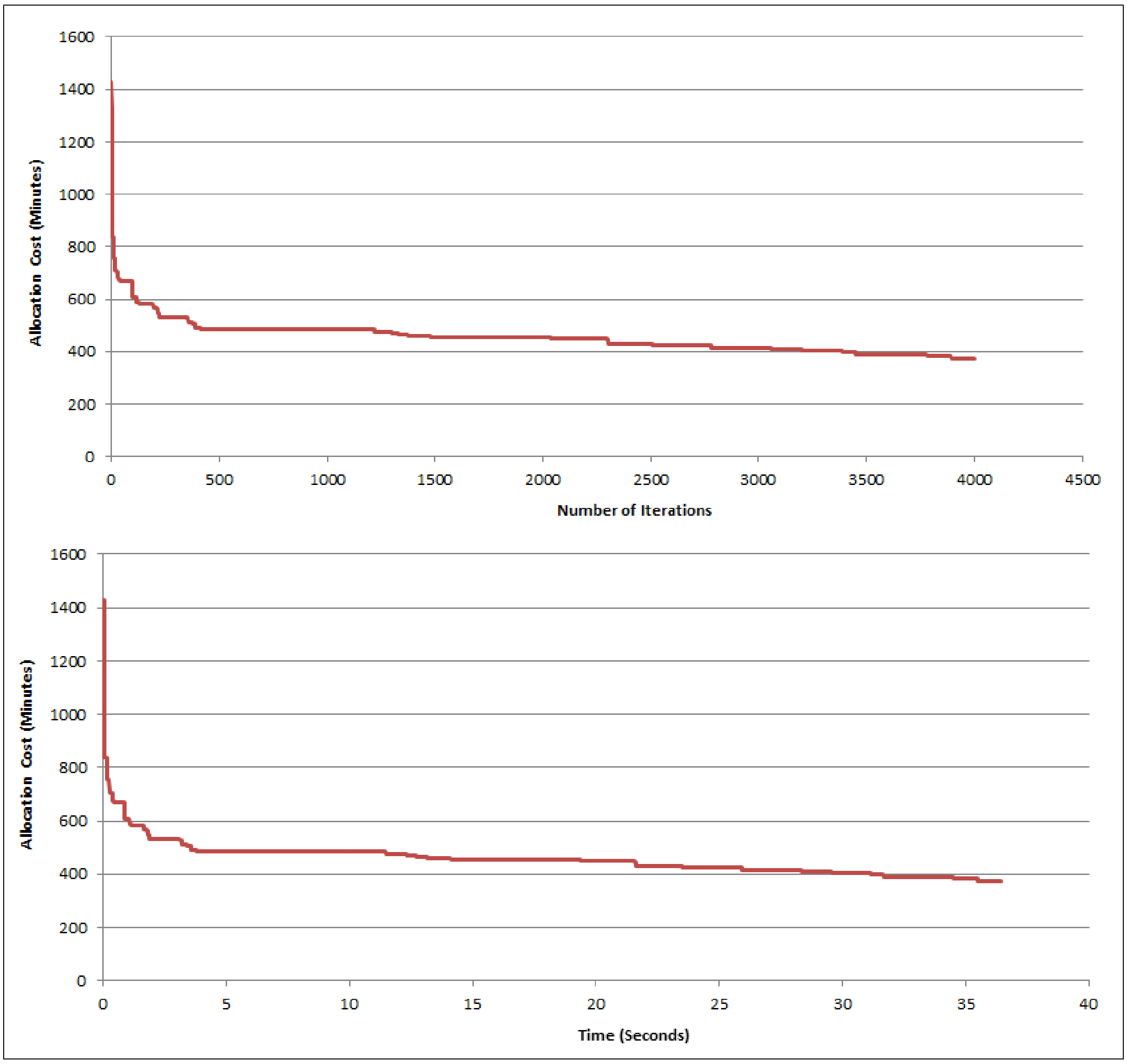


Figure 5.15: GA Performance for Large-scale MRTA Scenario

Finally, the deviation of the SA and GA from the optimal allocation when both are applied for only 10%,30% and 50% of the maximum time taken to reach the optimal allocation is demonstrated in Figure 5.16 and in Figure 5.17.

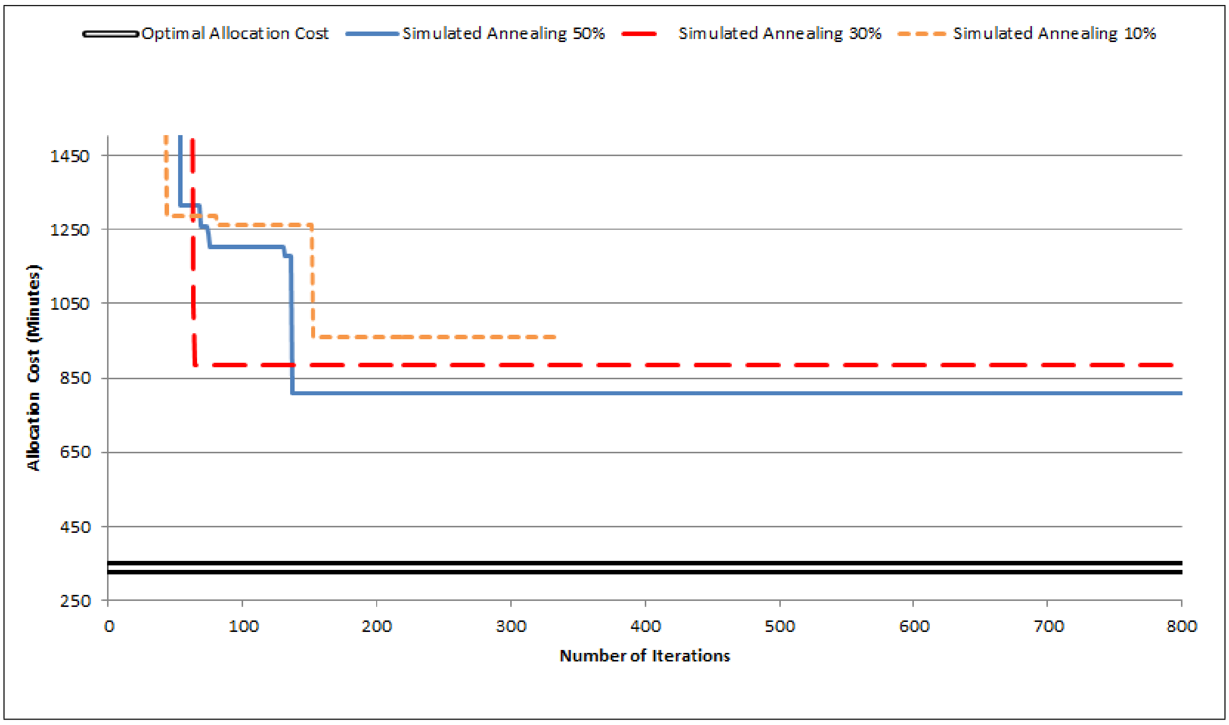


Figure 5.16: SA Deviation for Large-scale MRTA Scenario

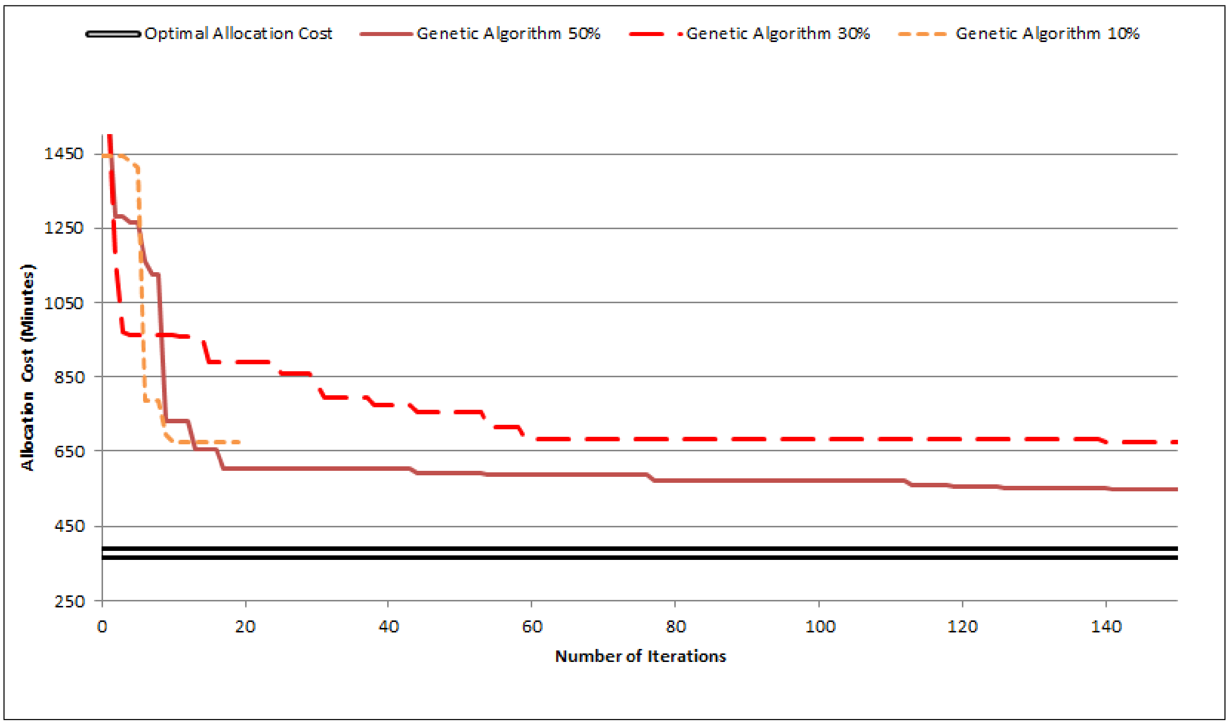


Figure 5.17: GA Deviation for Large-scale MRTA Scenario

5.3.2 Constraints-handling Test Scenarios

Another significant cause of complexity of the newly tackled MRS applications is that the problems are highly constrained. The constraints of MRS applications mainly arise because of two main reasons. The first source of constraints is the introduction of the heterogeneous robots in the multi-robot team as well as the heterogeneity of the tasks being executed. The second source of constraints is the energy level of the used robots and the time requirements of the tasks to be executed. As previously discussed in chapter 3 that the applied constraints significantly increases the MRTA problem complexity and thus make it harder for the solving algorithm to find the optimal allocation. Also the increased complexity of the MRTA problem through the increase of the number of applied constraints make the solving algorithm consumes much time in order to reach the optimal allocation. Therefore in this thesis, when proposing an approach to solve the MRTA problem it was important to propose an algorithm that is capable of handling the extended constraints of the MRTA problem such that the algorithm is capable of generating feasible solutions of acceptable near optimal costs in relatively short time.

In this section the ability of the algorithm to handle the applied constraints on the problem is under question. Analysis of the ability of the algorithms to handle problem constraints is done through applying both the SA and the GA over a number of scenarios at which the level of constraints increases over each scenario. The evaluation metrics previously defined in section 5.2 are used to report the quality of the proposed algorithms when solving the MRTA problems suggested in the constraint-handling test scenarios. It must be mentioned that in these test scenarios the number of used robots and tasks is constant and not increasing over each scenario. The proposed number of robots and tasks in these constraint handling scenarios is relatively low with respect to the scalability scenarios proposed in section 5.3.1. The main reason of reducing the number of used robots and tasks is to eliminate any other source of complexity to the problem rather than the level of applied constraints on the problem. Where eliminating other sources of complexity to the problem enables the adequate testing of only the ability of the proposed algorithms to generate acceptable near optimal solutions during the presence of difficult tight constraints. In these test scenarios the number of tasks and robots is ten and five respectively. The chosen number of robots and tasks is close to the number of robots and tasks in the second scenario of the scalability test scenarios and therefore the problem can be considered as a medium-scale MRTA problem. The number of robots and the tasks as well as the distances between the tasks will not be altered through the three constraints-handling scenarios.

5.3.2.1 Capabilities Matching Constrained MRTA Problem

The fourth proposed scenario is the first constraint matching scenario. The main aim of this scenario is to test the ability of the proposed algorithms of handling capabilities matching constraints. As previously discussed that one of the main constraints in the MRTA problem is that the capabilities of the robots assigned to execute certain tasks must match with the requirements of these tasks. In this scenario the robots and tasks are defined with different capabilities and requirements respectively. The robots are assumed to have a battery level of infinity since in this scenario only the capabilities matching is under test. The results of applying both algorithm are reported in Table 5.8.

Table 5.8: Capabilities Matching MRTA Scenario Results

Algorithm	*OptAllocCost*	*AvgTime*	*DevRandom*	*DevAlloc 10%*	*DevAlloc 30%*	*DevAlloc 50%*
SA	27	**7.82**	315%	63%	14%	8%
GA	27	23.79	315%	**14%**	**5%**	**3%**

The reported results in Table 5.8 shows that both algorithms converged to the same allocation cost which is deviated by 315% from random allocations. Thus both algorithms were capable of providing acceptable results for a capabilities matching constrained problem and therefore enhanced the overall performance through decreasing the allocation cost. Similar to the previous scalability scenarios, SA consumed less time than GA till reaching the total number of iterations. Although the number of robots and tasks in this scenario is almost similar to the number of robots and tasks in the medium-scale scenario discussed in subsection 5.3.1, however the consumed time by SA in the capabilities matching scenario was roughly doubled relative to the medium-scale scenario. Similarly, the consumed time by GA in the capabilities matching scenario increased about four times than the consumed time in the medium-scale scenario. This increase of consumed time by both algorithms is due to the presence of the capabilities matching constraints which decrease the number of feasible solutions and thus increase the total time taken by each algorithm till finding the optimal solution. Although the time consumed by both algorithms increased, however the increase of the time consumed by the GA is considerably higher than the increase of the SA consumed time which indicates that SA can outperform GA in constrained MRTA problems.

Similar to the previous scenarios, GA provided better results than SA in terms of deviation from optimal allocation when both algorithms where applied for only 10%, 30% and 50% of the maximum time taken by each algorithm. These results indicate that although the total time consumed by SA is less than the total time consumed by GA, however GA can provide better results than SA when both are applied for limited time. Figure 5.18 and Figure 5.19 demonstrate the performance of SA and GA respectively versus number of iterations and time when both are used to solve the capabilities matching constrained MRTA problem.

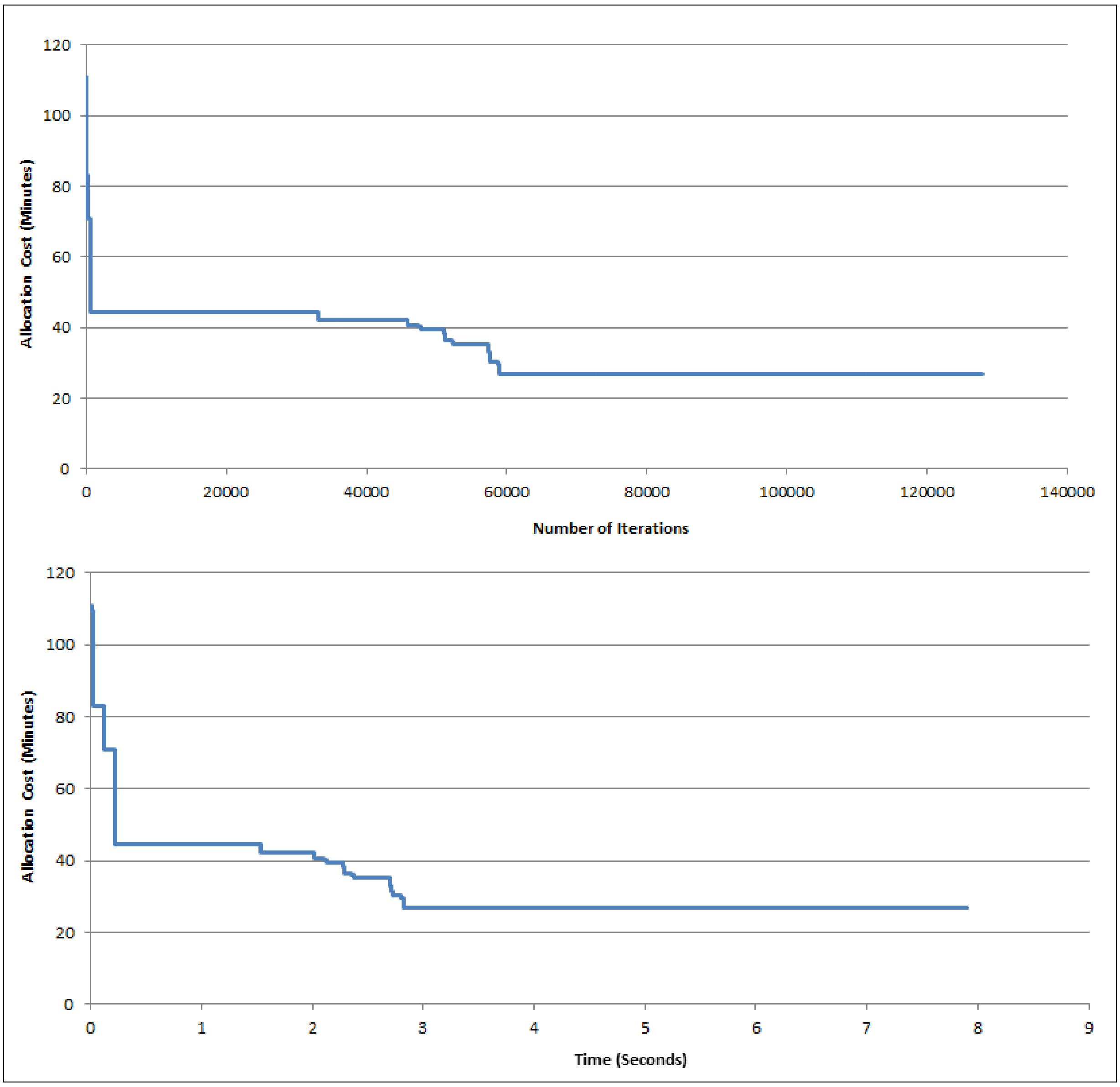


Figure 5.18: SA Performance for Capabilities Matching MRTA Scenario

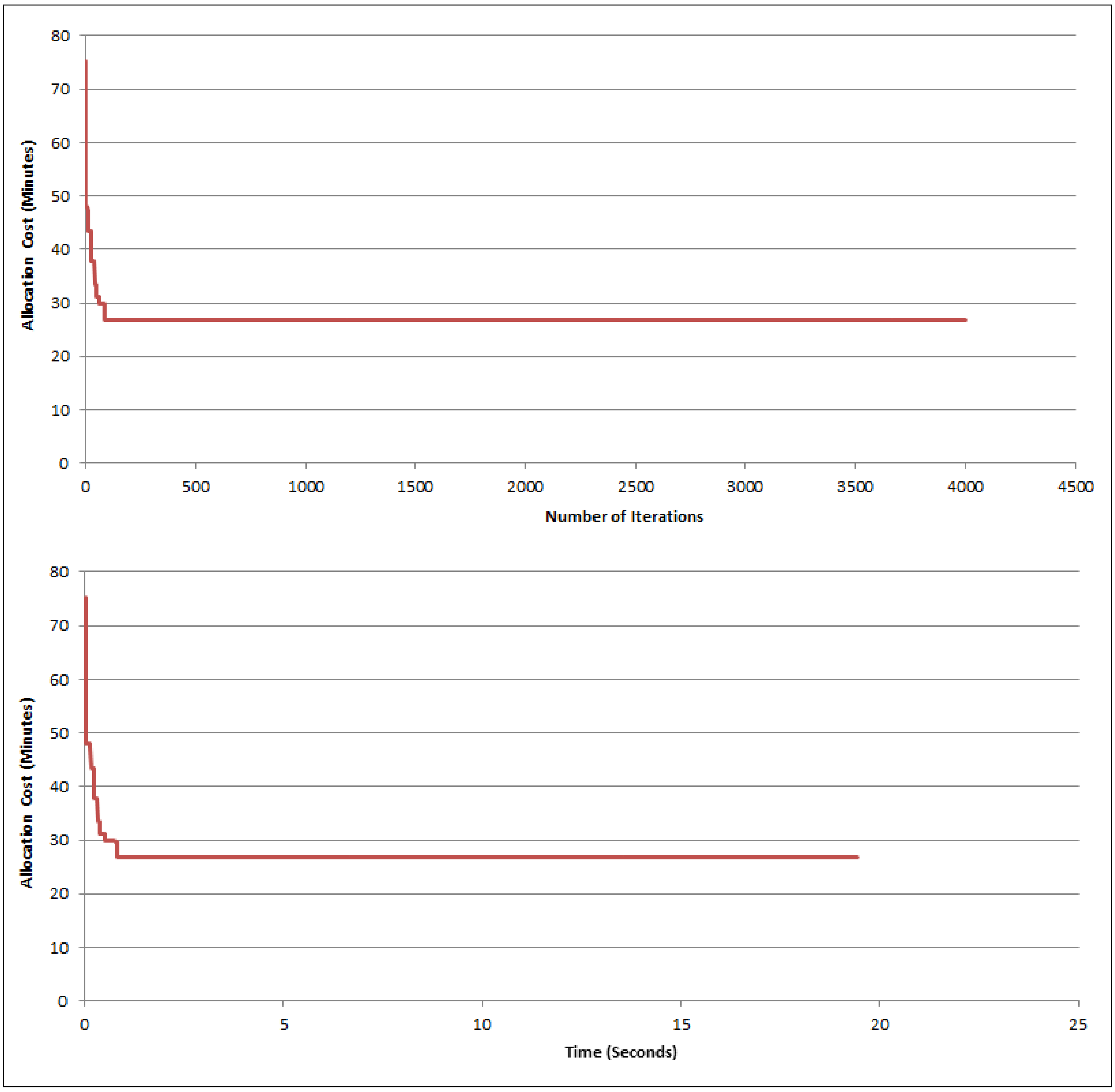


Figure 5.19: GA Performance for Capabilities Matching MRTA Scenario

The deviation of the SA and GA from the optimal allocation when both are applied for only 10%,30% and 50% of the maximum time taken to reach the optimal allocation is demonstrated in Figure 5.20 and in Figure 5.21.

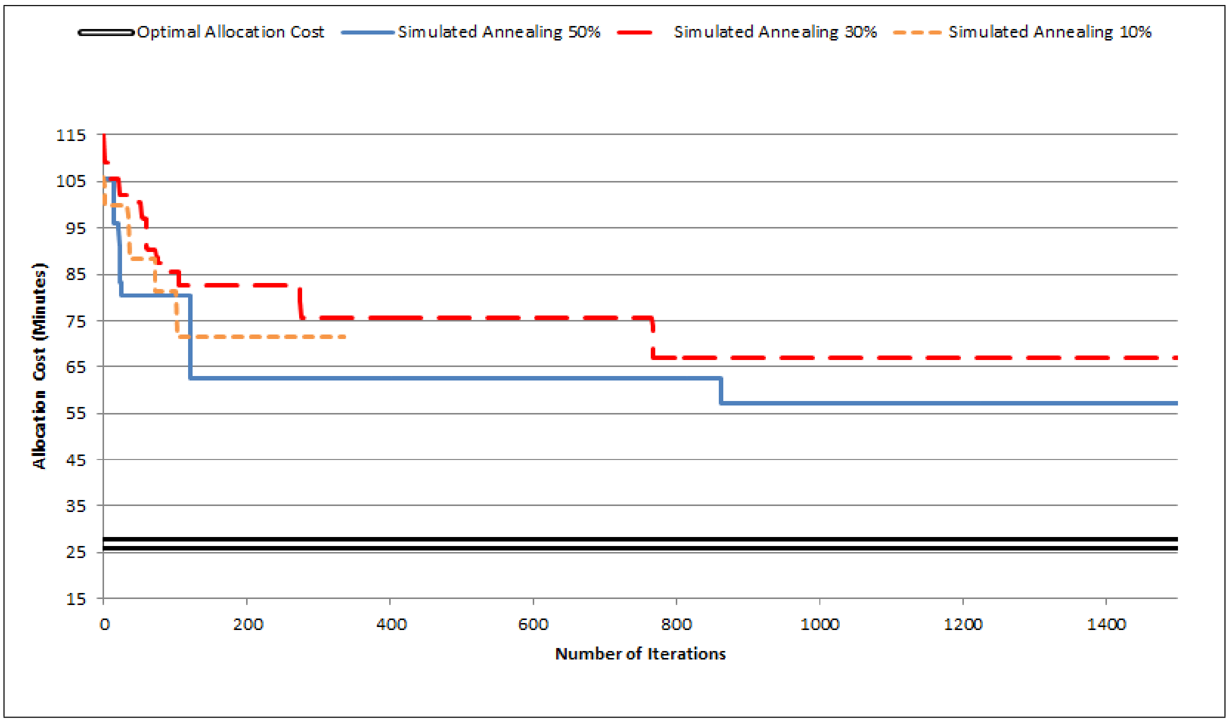


Figure 5.20: SA Deviation for Capabilities Matching MRTA Scenario

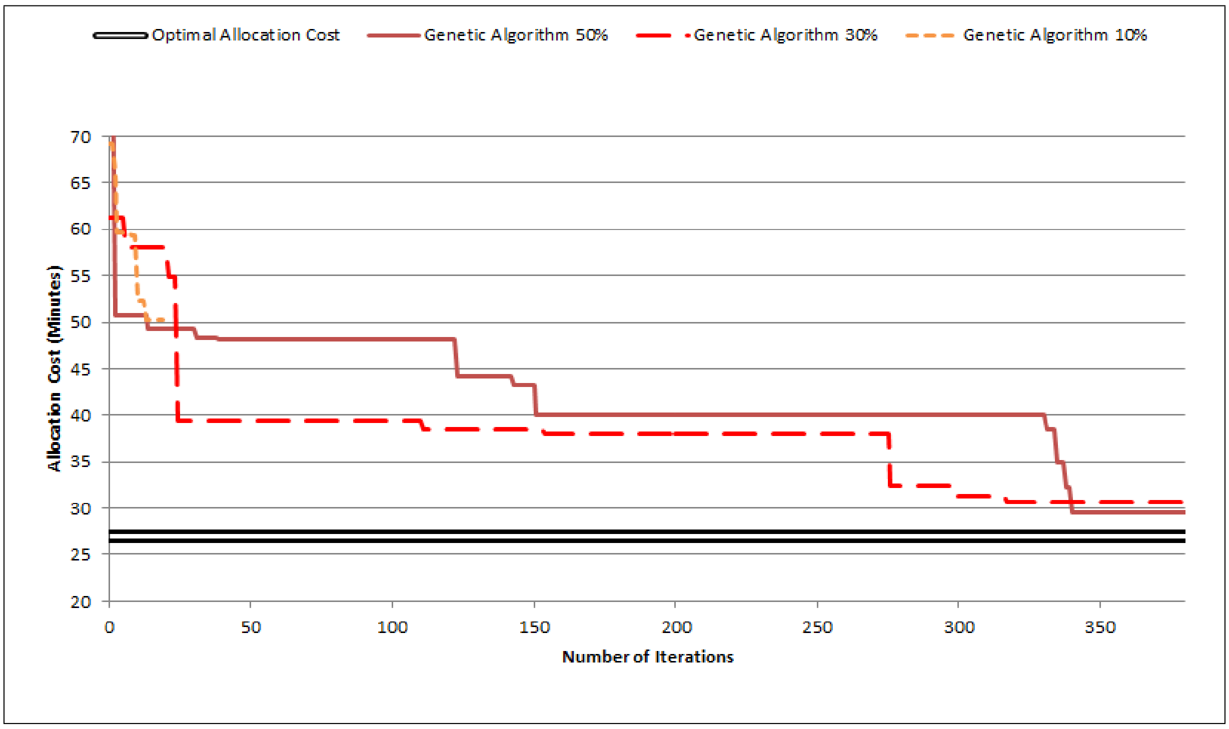


Figure 5.21: GA Deviation for Capabilities Matching MRTA Scenario

5.3.2.2 Time Matching Constrained MRTA Problem

The fifth proposed scenario is the second constraint matching scenario. The main aim of this scenario is to test the ability of the proposed algorithms of handling the time requirements constraints. As previously discussed that one of the main constraints in the MRTA problem is the available battery level of the used robots and if this battery level is compatible with the time requirements of the tasks and navigation time required between the tasks. Therefore in this scenario the robots are defined with limited battery levels such that not all candidate solutions can be accepted as feasible solutions. The robots and the tasks are defined without any physical capabilities or physical requirements but are defined in order to only test the ability of the proposed algorithms to handle the time matching constraints. The results of applying both algorithms are reported in Table 5.9.

Table 5.9: Time Matching MRTA Scenario Results

Algorithm	*OptAllocCost*	*AvgTime*	*DevRandom*	*DevAlloc 10%*	*DevAlloc 30%*	*DevAlloc 50%*
SA	**22.26**	**5.97**	**112%**	18%	14%	7%
GA	22.73	23.15	107%	**8%**	**3%**	**2%**

The results reported in Table 5.9 shows that the allocation cost provided by SA and GA are almost the same for the time matching scenario. Both algorithms are capable of enhancing the quality of the allocation cost and to provide solutions that deviate from random allocations by 110% roughly. Similar to previous scenarios, the total time consumed by SA is less than GA. However, GA tends to provide better allocation costs that less deviate from allocations provided by SA when both algorithms are only applied for only 10%, 30% and 50% of the maximum time taken by each algorithm. Figure 5.22 and Figure 5.23 demonstrate the performance of SA and GA respectively versus number of iterations and time when both are used to solve the time matching constrained MRTA problem.

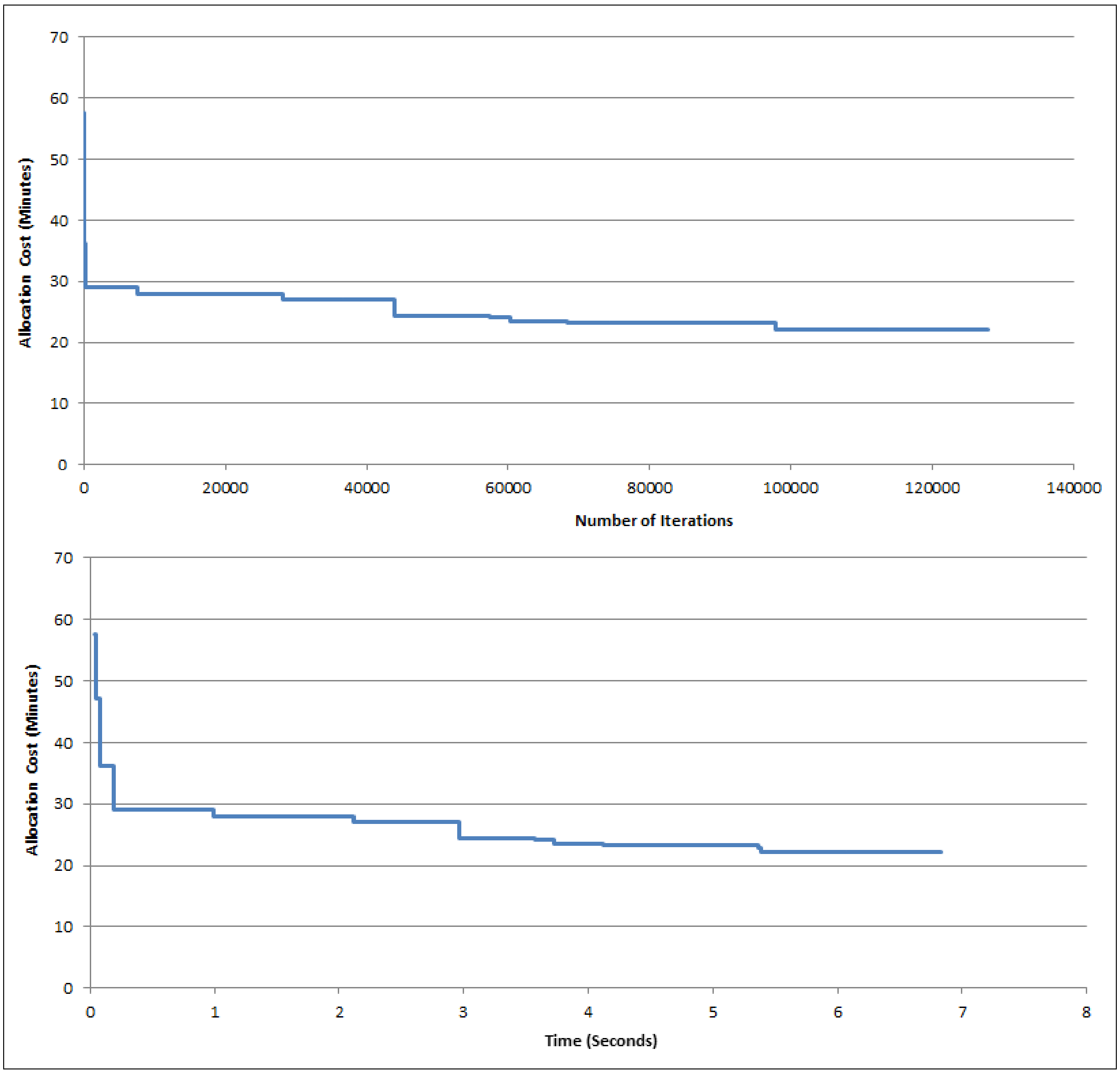


Figure 5.22: SA Performance for Time Matching MRTA Scenario

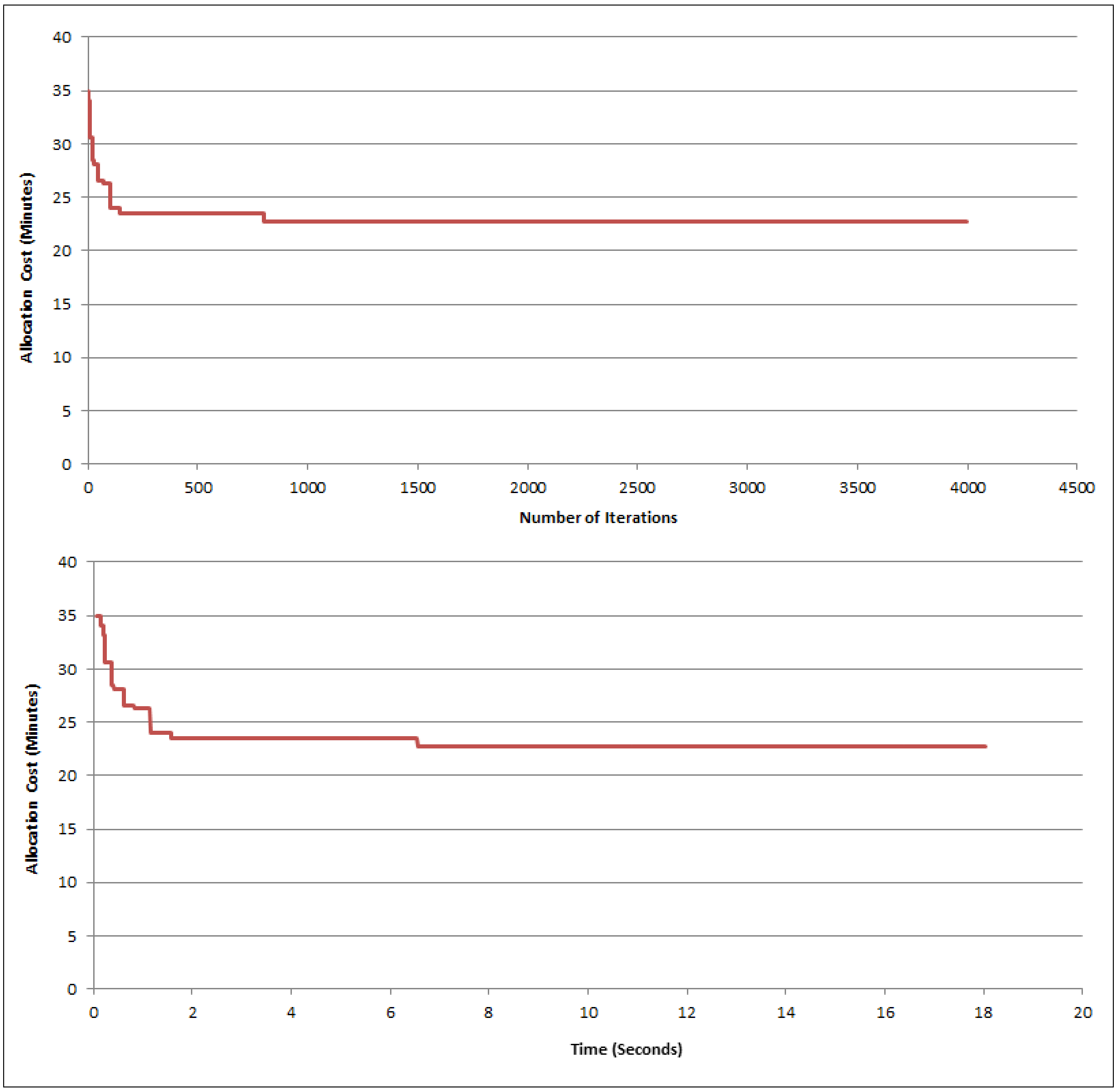


Figure 5.23: GA Performance for Time Matching MRTA Scenario

Finally, The deviation of the SA and GA from the optimal allocation when both are applied for only 10%,30% and 50% of the maximum time taken to reach the optimal allocation is demonstrated in Figure 5.24 and in Figure 5.25.

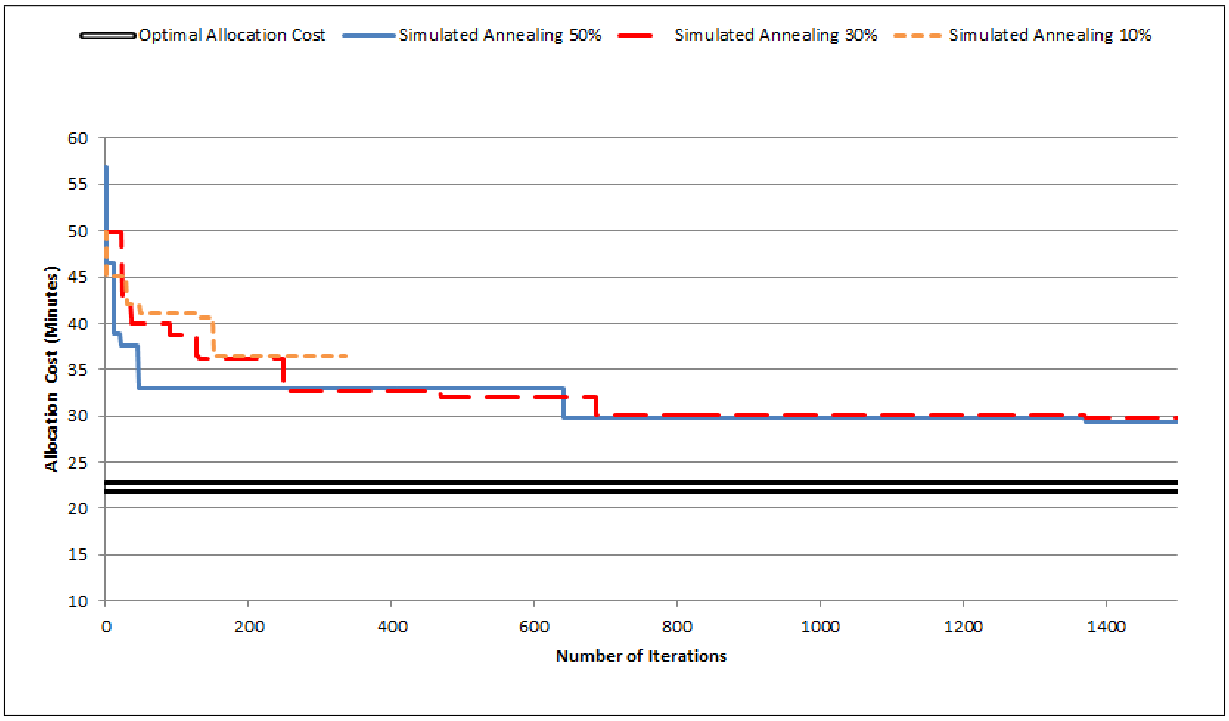


Figure 5.24: SA Deviation for Time Matching MRTA Scenario

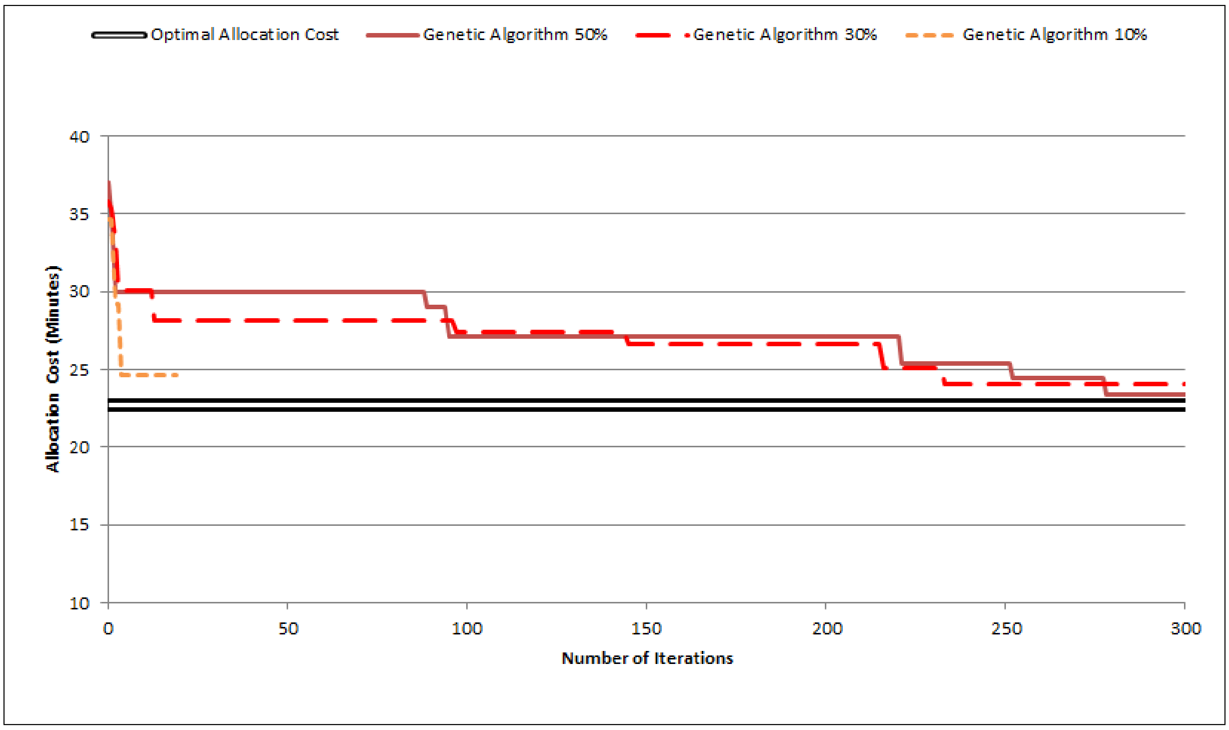


Figure 5.25: GA Deviation for Time Matching MRTA Scenario

5.3.2.3 Heavily Constrained MRTA problem

The last scenario is the combination of the capabilities matching and time matching scenarios. In this scenario the MRTA problem is a heavily constrained problem, where the problem is constrained by the matching of robot capabilities and task requirements as well as the matching of the battery level of the robots with the task time requirements. The results of applying the SA algorithm are reported in Table 5.10.

Table 5.10: Scenario 6 SA Evaluation Metrics

Algorithm	*OptAllocCost*	*AvgTime*	*DevRandom*	*DevAlloc 10%*	*DevAlloc 30%*	*DevAlloc 50%*
SA	28.85	17	93%	31%	20%	4%

The reported results shows that SA algorithm was capable of enhancing the quality of random allocation costs roughly by 90% the total time consumed by SA algorithm in this scenario shows a significant increase in the total time consumed relative to the previous scenarios. This significant increase indicates the high level of constraints applied over the problem domain and that it became more difficult for the optimization algorithm to find the optimal allocation under these tight constraints. However, total time consumed by SA is still low relative to the increased complexity of the problem and thus can still be practical to be used in tightly constrained MRS applications. Concerning the GA, there were no reported results for scenario 6. The failure of reporting any results for scenario 6 using the GA is because of the fact that the problem is a heavily constrained one which make it harder and more time consuming for the algorithm to generate a feasible solution. GA was tested to solve the problem introduced in scenario 6 with only one iteration. After 10 runs, the average time taken for one iteration of the GA was 15 seconds. Since the number of iterations used for the GA in this thesis is 4000 iteration, therefore if 4000 iteration was multiplied by 15 seconds for one iteration this will yield nearly 17 hours which is roughly the total time needed for the GA to finish solving the MRTA problem in this scenario. Therefore the GA algorithm failed to solve the heavily constrained problem proposed in the last scenario. Figure 5.26 demonstrates the performance of SA versus number of iterations and time when it is used to solve the heavily constrained MRTA problem.

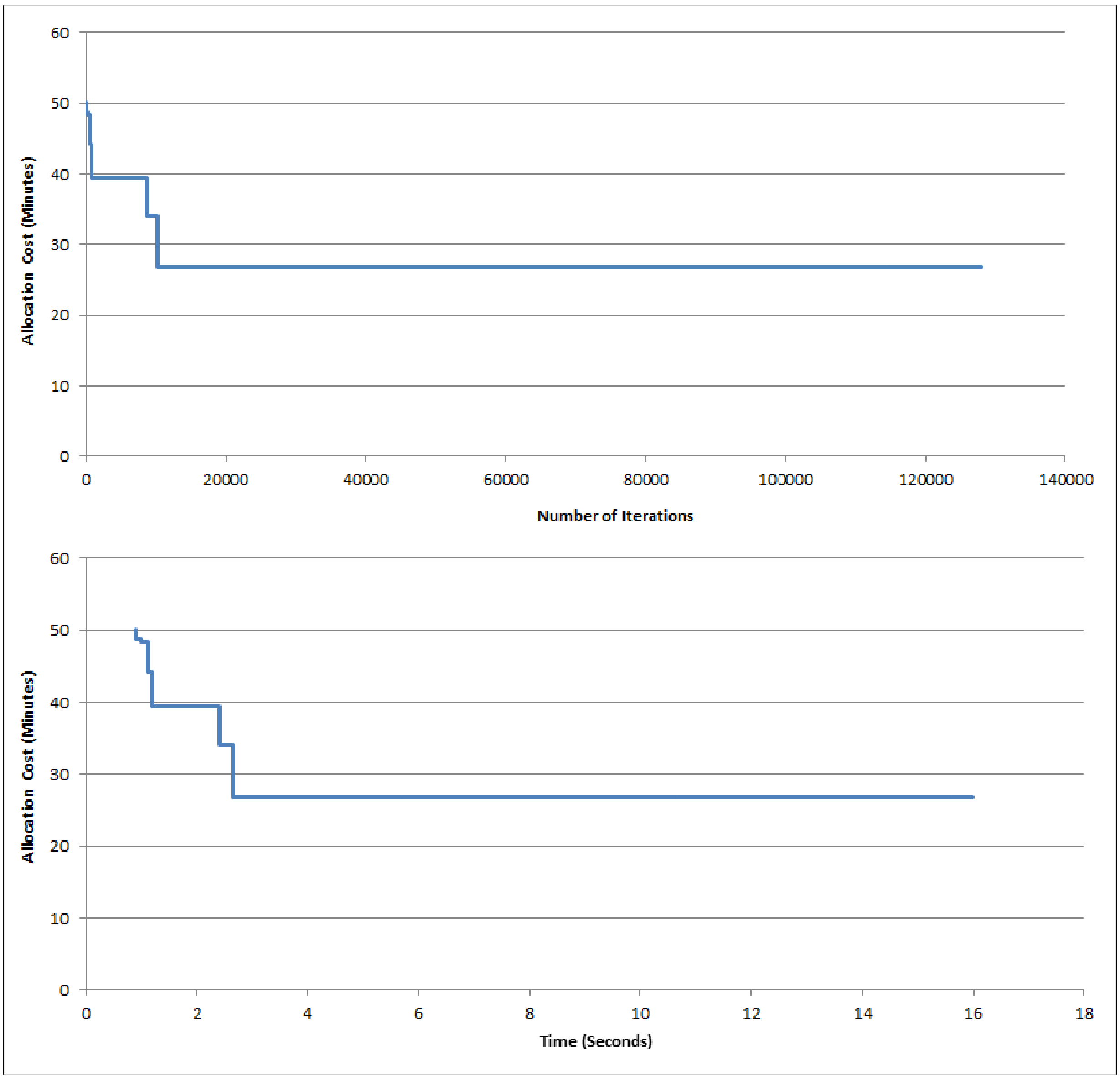


Figure 5.26: SA Performance for Heavily Constrained MRTA Scenario

Finally, The deviation of the SA from the optimal allocation when it is applied for only 10%,30% and 50% of the maximum time taken to reach the optimal allocation is demonstrated in Figure 5.27.

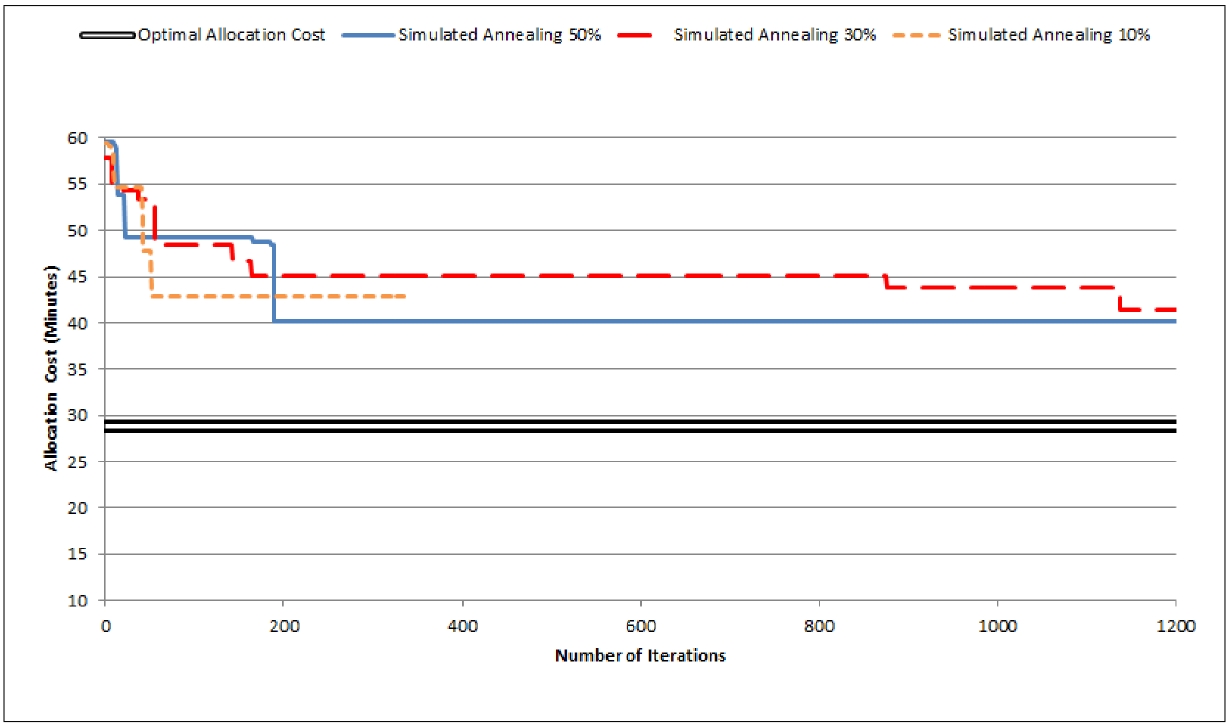


Figure 5.27: SA Deviation for Heavily Constrained MRTA Scenario

5.4 Comparative Study

In this section a comparative study between the two proposed algorithms SA and GA is conducted over the six previously proposed scenarios in section 5.3. The results of the two proposed algorithms is illustrated in Table 5.11. Moreover the two proposed algorithms, SA and GA are also compared to the results provided by a market-based approach for solving the MRTA prblem. The market-based approached developed in [94] was applied to solve the same six test scenarios proposed in section 5.3.

Table 5.11: Comparative Study of Proposed Algorithms Results

Points of Comparison	Small-scale		Medium-scale		Large-scale	
	SA	GA	SA	GA	SA	GA
Allocation Cost (min)	12.76	12.76	**28.5**	29.47	**338.06**	378.82
Average Time Consumed (Sec)	**1.04**	1.58	**3.67**	6.06	**22.21**	39.1
Deviation From Optimal 10%	3%	**0%**	45%	**15%**	116%	**45%**
Deviation From Optimal 30%	0%	0%	28%	**6%**	72%	**23%**
Deviation From Optimal 50%	0%	0%	5%	**3%**	20%	**14%**

Points of Comparison	Capabilities Matching		Time Matching		Heavily Constrained	
	SA	GA	SA	GA	SA	GA
Allocation Cost (min)	27	27	**22.26**	22.73	28.85	-
Average Time Consumed (Sec)	**7.82**	23.79	**5.97**	23.15	17	-
Deviation From Optimal 10%	63%	**14%**	18%	**8%**	31%	-
Deviation From Optimal 30%	14%	**5%**	14%	**3%**	20%	-
Deviation From Optimal 50%	8%	**3%**	7%	**2%**	4%	-

Although the six proposed scenarios are diverse and different as they include different number of robots and tasks as well as different constraints. However, the results demonstrated in the previous table shows that SA outperformed GA in terms of the optimal allocation found by each algorithm in nearly all of the six test scenarios. SA either provided allocations of lower cost than GA or either both algorithms provided the same results. Another advantage of SA over GA is that in all of the test scenarios the total time consumed by SA is less than the total time consumed by GA. Although Table 5.2 and 5.3 indicates that the total number of iterations of SA is more than the total number of iterations of GA. However, if both algorithms are allowed to run till reaching the total number of iterations SA will finish sooner than GA and thus provide earlier results.

Although SA outperformed GA in all test scenarios in terms of optimal allocation cost as well as total time consumed. However, GA provided better results than SA in all test scenarios in terms of the deviation of provided allocations from optimal allocation when both algorithms were applied for only 10%, 30% and 50% of the total time taken to reach the optimal allocation. These results indicates that GA is capable of providing better allocation costs than SA when there is no enough time for running the algorithm till reaching the total number of iterations.

5.4.1 Small-scale MRTA Scenario

Figure 5.28 illustrates the difference in the performances of the three approaches in comparison SA, GA and the market-based approach versus time when used to solve the small-scale MRTA scenario.

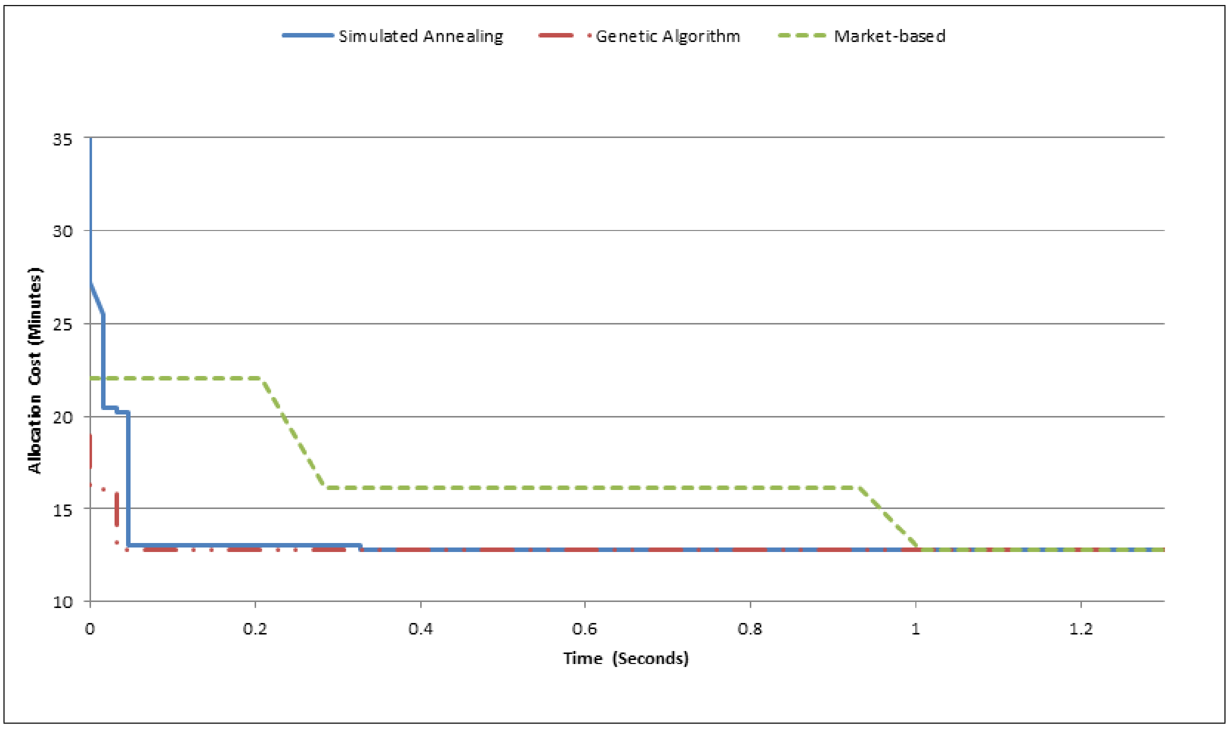


Figure 5.28: Small-scale MRTA Scenario Comparative Study

The three approaches were capable of converging to the same allocation cost in the small-scale scenario. GA was the fastest to reach the optimal allocation followed by SA and then the market-based approach.

5.4.2 Medium-scale MRTA Scenario

Figure 5.29 illustrates the difference in the performances of the three approaches in comparison SA, GA and the market-based approach versus time when used to solve the medium-scale MRTA scenario.

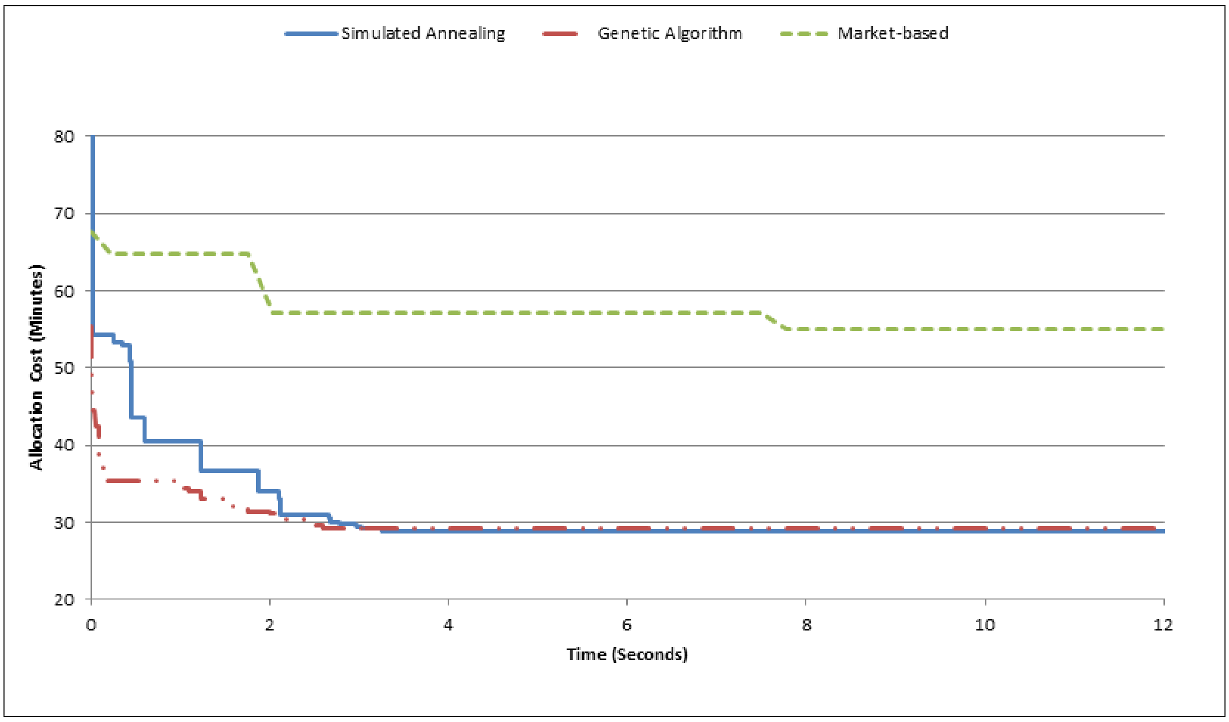


Figure 5.29: Medium-scale MRTA Scenario Comparative Study

The performance of of SA and GA in Figure 5.29 is consistent with the values reported in Table 5.11. SA was capable of converging to better allocation cost than GA at the end of both algorithms. However, GA was faster in converging to better allocations than SA at the beginning of the algorithms and thus if both algorithms are stopped early before reaching the total number of iterations, the results of the GA are expected to be better than the results of SA. Finally both the SA and GA provided better allocation costs than the market-based approach.

5.4.3 Large-scale MRTA Scenario

Figure 5.30 illustrates the difference in the performances of the three approaches in comparison SA, GA and the market-based approach versus time when used to solve the large-scale MRTA scenario.

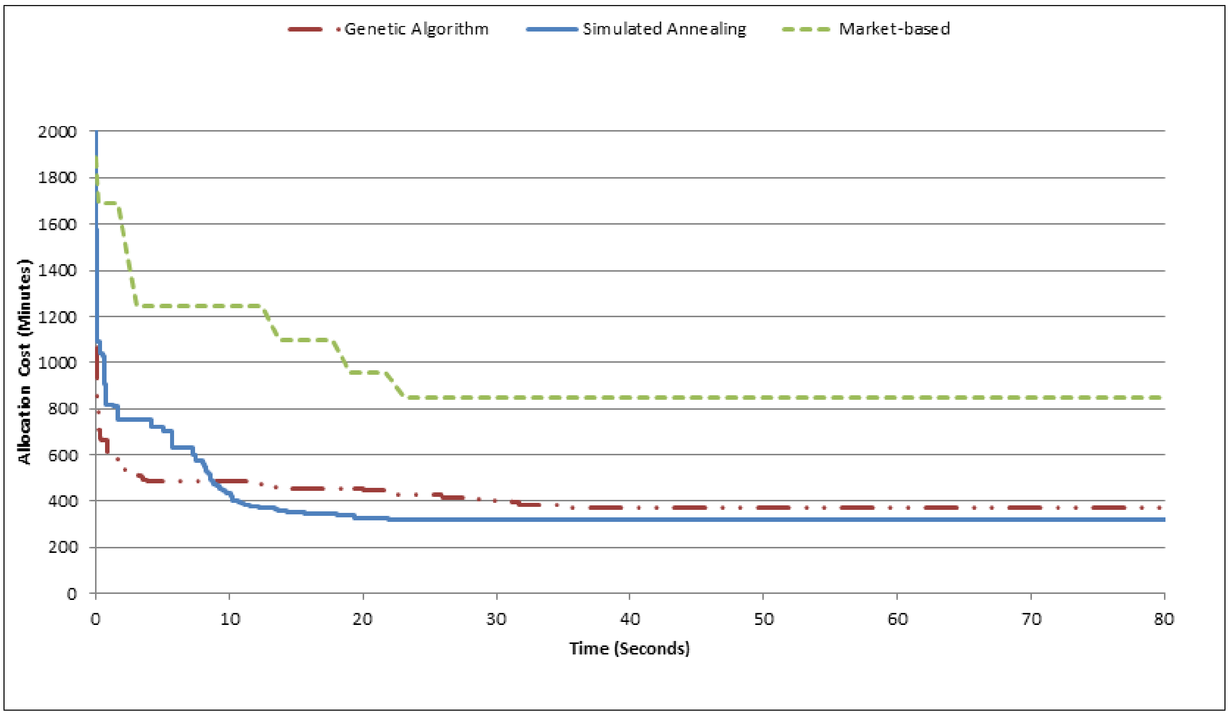


Figure 5.30: Large-scale MRTA Scenario Comparative Study

As reported in Table 5.11, the results in Figure 5.30 shows that the performance of SA is better than GA in terms of allocation cost. Also the results in the figure shows that GA is capable of converging to better results faster than SA when both algorithms were applied for only 10%, 30% and 50% of the maximum time taken by each algorithm. However, as more time is given for both algorithms SA converges to better allocations than GA. Finally, comparing the results of SA and GA with the market-based approach results shows that the metaheuristic approaches SA and GA significantly outperformed the market-based approach in terms of allocation cost provided for the large-scale problem. This major difference in the allocation cost between the metaheuristic approaches and the market-based approach in solving the large-scale MRTA scenario indicates that market-based approaches might not be the suitable approach for MRS applications with extended number of tasks and robots. On the other hand, the metaheuristic approaches have proven to provide acceptable results in medium and large-scale MRTA scenarios with extended number of tasks and robots.

5.4.4 Capabilities Matching MRTA Scenario

Figure 5.31 illustrates the difference in the performances of the three approaches in comparison SA, GA and the market-based approach versus time when used to solve the capabilities matching MRTA scenario.

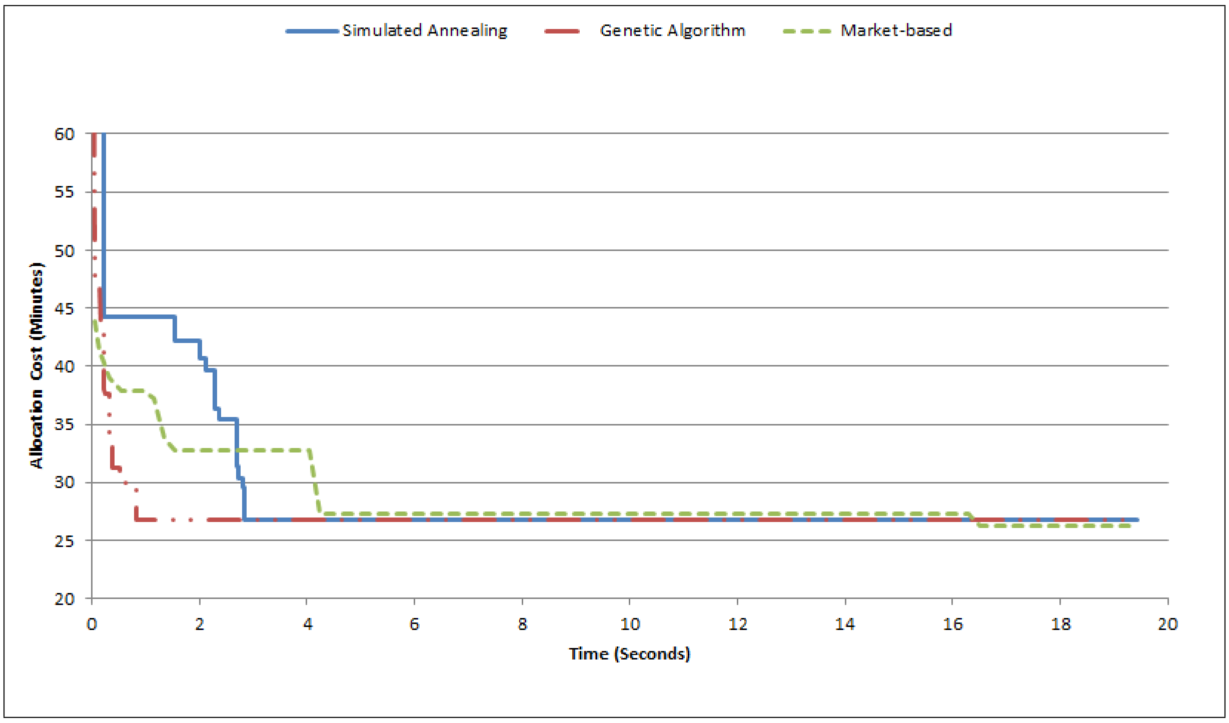


Figure 5.31: Capabilities Matching MRTA Scenario Comparative Study

The performance curves of SA and GA in Figure 5.31 demonstrates that at the end of both algorithms the allocations found are of the same cost for both algorithms. On the other hand, GA algorithm was capable of converging to better allocations of lower costs earlier than SA. Thus if there was no enough time for both algorithms to reach the total number of iterations, the GA would be the suitable algorithm and is expected to provide better results than SA. Although in this MRTA scenario, the metaheuristic approaches converged earlier than the market-based approach, however the market-based approach eventually provided better allocation costs than both metaheuristic algorithms.

5.4.5 Time Matching MRTA Scenario

Figure 5.32 illustrates the difference in the performances of the three approaches in comparison SA, GA and the market-based approach versus time when used to solve the time matching MRTA scenario.

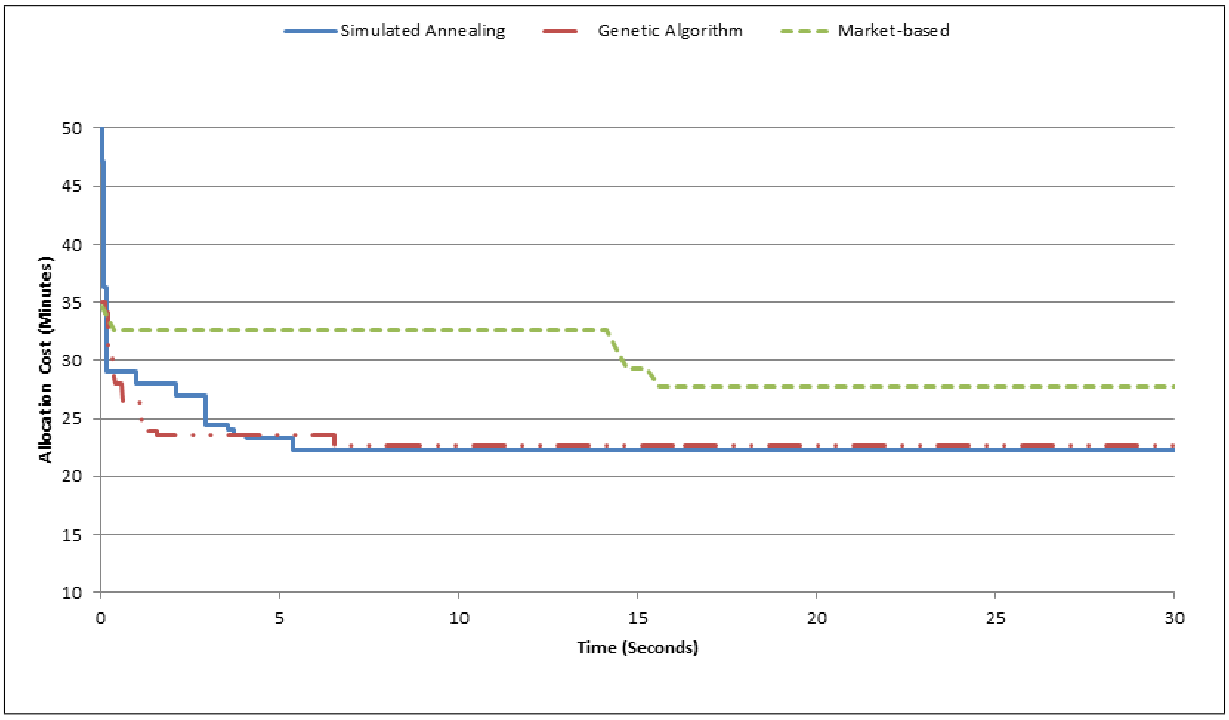


Figure 5.32: Time Matching MRTA Scenario Comparative Study

Figure Figure 5.32 shows that SA provides slightly better allocations than GA, however the the total time consumed by SA is significantly lower than total time consumed by GA. Moreover the performance curves show that GA converges to better solutions faster than SA and thus if both algorithms are stopped at only 10% or 30% or 50% of maximum time, the GA will probably provide better solutions than SA. The reported results in Figure 5.32 demonstrates that both metaheuristic approaches provided better allocation costs than the market-based approach. Also the metaheuristic approaches converged faster than the market-based approach.

5.4.6 Heavily Constrained MRTA Scenario

Figure 5.33 illustrates the difference in the performances of two approaches in comparison SA and the market-based approach versus time when used to solve the time matching MRTA scenario. There were no reported results for the GA as it consumed a large amount of time and thus was not practical.

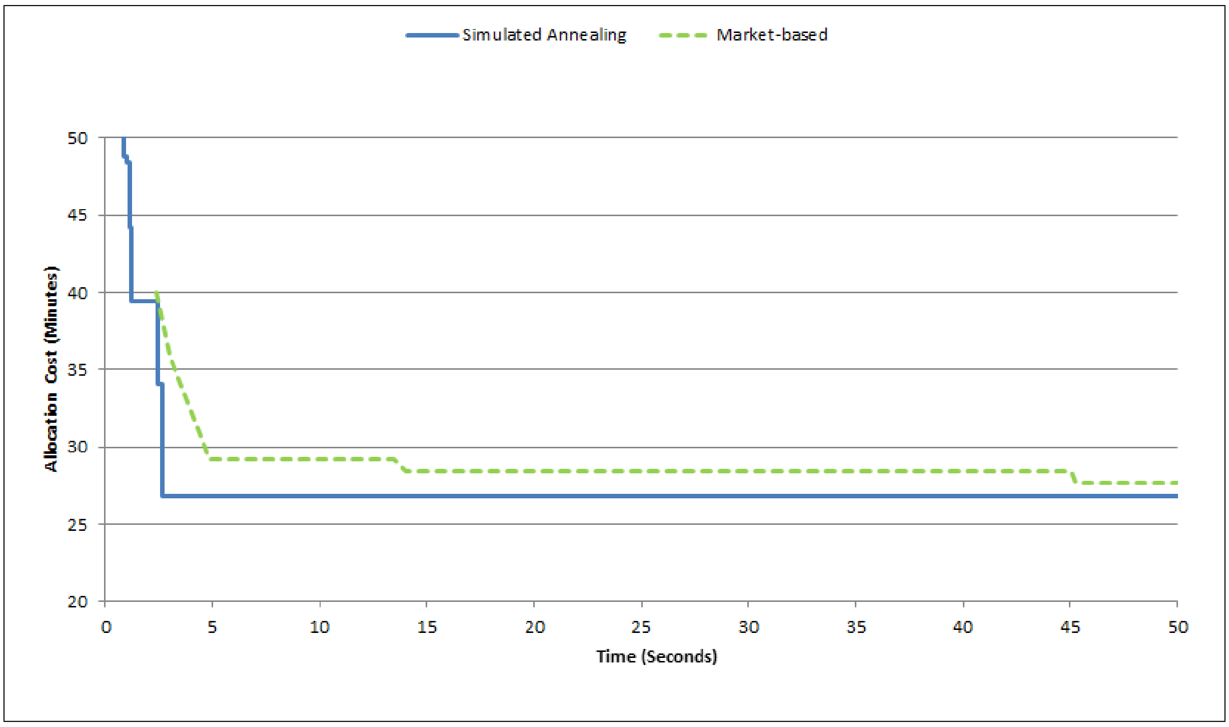


Figure 5.33: Heavily Constrained MRTA Scenario Comparative Study

The reported results in Figure 5.33 show that SA provided better results than the market-based approach although the difference between the two approaches in terms of allocation cost is minimal. However SA converged much faster than the market-based approach.

5.5 Concluding Remarks

In this chapter the experimental results of this thesis were presented. The experiments proposed algorithms were implemented and tested using Java. Four evaluation metrics were proposed in order to be able to test the quality of the two proposed algorithms. Six evaluation scenarios were used to test the proposed algorithms. The first three scenarios tested the scalability of the algorithms while the last three tested the ability of the algorithms to handle the problem constraints and to generate feasible acceptable solutions under these results. Both the SA and the GA produced good results over the first five scenarios. The GA failed to produce results in the last scenario which was heavily constrained.

Finally a comparative study was conduced between the metaheuristic approaches and a market-based approach. The metaheuristic approaches outperformed the market-based approach in the scalability test scenarios where the results provided by the market-based approach significantly degraded as the number of tasks and robots increased. However both approaches approximately provided similar results in the constraints handling scenarios.

Chapter 6

Conclusion

6.1 Conclusion

In recent years, MRS has received an increased consideration from various researchers in the field of robotics research. This significant consideration is due to the fact that MRS can be used in various different applications such as search and rescue, mobile surveillance systems, humanitarian demining. Moreover micro MRS can be used in medical applications such as operations of organ transplants and eliminating cancer cells [95]. MRS has numerous advantages and therefore researchers mainly agreed that MRS are better than single robot systems. There are several motivations for developing MRS such as that the task complexity of targeted applications is very high for a single robot to accomplish alone and the fact that a team of robots will perform a task faster than a single robot and other several motivations. One of the main challenges of MRS is the allocation of tasks to the members of the multi-robot team. The task allocation problem in MRS is one of the problems that is under extensive research by various researchers in the academia, industrial research departments and military laboratories. This widespread focus on MRTA is because that the optimization of this problem will lead to the proper utilization of available resources and thus increasing the overall system performance and decreasing the overall system costs. Because of this prominent significance of the MRTA problem, several approaches and algorithms have been proposed and developed to solve this problem.

The main intentions of this thesis was to propose a generic approach for solving the MRTA problem. The developed approach will be responsible for providing a solution that is not only a feasible solution but an optimized one. Since the optimized solution will enable the appropriate use of the available resources and thus increasing the overall system performance and decrease the costs. Another objective is to propose an approach that is capable of handling real world application constraints such as time constraints, robot capabilities and task requirements matching constraints. The proposed approach must be able to also handle swarm robotics applications where the number of robots and number of tasks are overextended. In this work, a metaheuristic optimization-based approach was proposed for solving the task allocation problem in MRS. Simulated annealing (SA) and genetic algorithm (GA) were the metaheuristic optimization algorithms implemented.

Several test scenarios were implemented in order to test the proposed algorithms. The test scenarios intended to test the scalability of the algorithms and the quality of the provided solutions to applications that include a large number of robots and tasks. Also the test scenarios were used to test the ability of both algorithms to handle real world applications that are tightly constrained.

The analysis of the experimental results showed that SA and GA were both successful in handling various scenarios of increased number of robots and tasks and of difficulty constraints. Both algorithms were able to generate enhanced quality solution relative to the randomly generated solutions. Finally a comparative study was conduced between the proposed algorithms in this work and a market-based approach. The metaheuristic approach and the market-based approach were tested over the same test scenarios and the results were reported. These results showed that metaheuristic approach outperformed the market-based approach in a number of aspects such as the optimality of the found allocations as well as the the total time taken to reach the optimal allocation.

6.2 Future Work

While the proposed approaches were successful in solving the presented scenarios of the MRTA problem, however there were some aspects of the problem that were assumed for simplification. Theses assumptions could be handled in the future work in order to increase the reliability of the proposed approach and to incorporate the proposed algorithms with the other future work in order to present a complete platform for MRS application. Some of the future work recommendations would be as follows:

1. The input distance matrix in this work was assumed to include the shortest obstacle free paths between the position of the tasks. The proposed approach could be further expanded in order to plan the paths of the robots in the environment before the allocation process through incorporating our previous work introduced in [77] with the proposed approach in this thesis. Through this combination, the path planning algorithm proposed in [77] will be responsible for filling the input distance matrix with the shortest obstacle free distance between the positions of the tasks. Then the algorithms proposed in this thesis will be responsible for finding the optimal allocation of the tasks to the multi-robot team as explained in the thesis.

2. Although the performance of both proposed algorithms were extensively tested over a number of test scenarios, however the proposed algorithms are planned to be tested using Khepera III real robots in a newly built testbed arena for MRS simulations and experiments in the Robotics and Autonomous Systems (RAS) laboratory at the German University in Cairo.

3. In this thesis, the complex tasks are assumed to be divided by an experienced human operator into a number of subtasks that are autonomously allocated to the members of the robot team. For the aim of complete autonomy of the system, further research may investigate proposing an algorithm that is responsible for dividing the complex tasks into a number of subtasks that are then allocated using the algorithms in this

thesis. The importance of this point emerges from the fact that incorrect division of the complex tasks may lead to losing the optimal solution to the problem.

Bibliography

[1] A. Elmogy, *Market-based Framework for Mobile Surveillance Systems.* PhD thesis, University of Waterloo, 2010.

[2] L. E. Parker, "Multiple mobile robot systems," in *Springer Handbook of Robotics*, pp. 921–941, Springer, 2008.

[3] A. Sgorbissa, "Multi-robot systems and distributed intelligence: The ethnos approach to heterogeneity," *Mobile Robotics, Moving Intelligence*, pp. 423–446, 2006.

[4] B. Gerkey, *On Multi-robot Task Allocation.* PhD thesis, Faculty of the Graduate School University of Southern California, August 2003.

[5] Y. Han, D. Li, J. Chen, X. Yang, Y. Hu, and G. Zhang, "A multi-robots task allocation algorithm based on relevance and ability with group collaboration," *International Journal of Intelligent Engineering and Systems*, vol. 3, no. 2, 2010.

[6] A. Farinelli, L. Iocchi, and D. Nardi, "Multi robot systems: A classification focused on coordination," *IEEE Transactions on System Man and Cybernetics, part B*, vol. 34, no. 5, pp. 2015–2028, 2004.

[7] T. Arai, E. Pagello, and L. Parker, "Guest editorial advances in multirobot systems," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 655–661, 2002.

[8] Y. Cao, A. Fukunaga, A. Kahng, and F. Meng, "Cooperative mobile robotics: Antecedents and directions," in *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1, pp. 226–234, 1995.

[9] B. Gerkey and M. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[10] A. M. Khamis, A. M. Elmogy, and F. O. Karray, "Complex task allocation in mobile surveillance systems," *Journal of Intelligent & Robotic Systems*, vol. 64, no. 1, pp. 33–55, 2011.

[11] E. Manisterski, E. David, S. Kraus, and N. R. Jennings, "Forming efficient agent groups for completing complex tasks," in *5th Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 834–841, 2006.

[12] R. Zlot and A. Stentz, "Market-based multirobot coordination for complex tasks," *The International Journal of Robotics Research*, vol. 25, no. 1, pp. 73–101, 2006.

[13] U. Endriss, "Lecture notes on fair division," *Institute for Logic, Language and Computation*, 2010.

[14] T. Walsh, "Online cake cutting," in *Algorithmic Decision Theory* (R. Brafman, F. Roberts, and A. TsoukiÃ s, eds.), vol. 6992 of *Lecture Notes in Computer Science*, pp. 292–305, Springer Berlin Heidelberg, 2011.

[15] I. Kash, A. D. Procaccia, and N. Shah, "No agent left behind: Dynamic fair division of multiple resources," 2012.

[16] R. Zivan and E. Segev, "Trust based efficiency for cake cutting algorithms," *Trust In Agent Societies (Trust-2011)*, p. 141, 2011.

[17] P. . Tannenbaum and R. Arnold, *Excursions in Modern Mathematics.* Prentice Hall, 2000.

[18] F. S. Roberts, *Discrete Mathematical Models with Applications to Social, Biological, and Environmental Problems.* Pearson, 1976.

[19] A. Sahu and R. Tapadar, "Solving the assignment problem using genetic algorithm and simulated annealing," *IAENG International Journal of Applied Mathematics*, vol. 36, pp. 37–40, 2007.

[20] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83–97, 1955.

[21] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009.

[22] B. Kaleci and O. Parlaktuna, "Market-based multi-robot task allocation using energy-based bid calculations," *International Journal of Robotics and Automation*, vol. 27, no. 4, p. 396, 2012.

[23] L. Liu and D. A. Shell, "Multi-level partitioning and distribution of the assignment problem for large-scale multi-robot task allocation," *Robotics: Science and Systems VII*, 2012.

[24] S. H. Liu and Y. Zhang, "Multi-robot task allocation-based on swarm intelligence," *Journal of Jilin University*, vol. 40, no. 1, pp. 123–129, 2010.

[25] B. Kaleci, O. Parlaktuna, M. Ozkan, and G. Kirlik, "Market-based task allocation by using assignment problem," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 135–141, 2010.

[26] N. Atay and B. Bayazit, "Mixed-integer linear programming solution to multi-robot task allocation problem," *Washington Univ., St. Louis, Tech. Rep. WUCSE-2006-54*, 2006.

[27] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[28] L. E. Parker, "Alliance: An architecture for fault tolerant multi-robot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 4, pp. 220–240, 1998.

[29] R. Brooks, "Planning is just a way of avoiding figuring out what to do next," *Technical report, MIT Artificial Intelligence Laboratory*, 1987.

[30] L. E. Parker, "L-alliance - a mechanism for adaptive action selection in heterogeneous multi-robot teams," *Center for Engineering Systems Advanced Research*, 2007.

[31] B. P. Gerkey and M. J. Matric, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," *IEEE International Conference on Robotics and Automation (ICRA 2003)*, pp. 3862–3868, 2003.

[32] T. Bektas, "The multiple traveling salesman problem: An overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[33] A. Mosteo and L. Montano, "Simulated annealing for multi-robot hierarchical task allocation with minmax objective." Workshop on Network Robot Systems: Toward intelligent robotic system integrated with environments, IROS'06, 2006.

[34] Z. Jian, P. Zhihong, and L. Bo, "Multi-task allocation of ucavs considering time cost and hard time window constraints," in *Control Conference (CCC), 2012 31st Chinese*, pp. 2448–2452, 2012.

[35] R. M. Zlot, *An Auction-based Approach to Complex Task Allocation for Multirobot Teams.* PhD thesis, Carnegie Mellon University, October 2006.

[36] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesman problem," *Robotics Automation Magazine, IEEE*, vol. 17, no. 4, pp. 70–77, 2010.

[37] S. Sarieltalay, T. Balch, and N. Erdogan, "Multiple traveling robot problem: A solution based on dynamic task selection and robust execution," *Mechatronics, IEEE/ASME Transactions on*, vol. 14, no. 2, pp. 198–206, 2009.

[38] M. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[39] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 758–768, 2002.

[40] A. Stentz, M. B. Dias, R. Zlot, and N. Kalra, "Market-based approaches for coordination of multi-robot teams at different granularities of interaction," *Robotics Institute - Carnegie Mellon University*, 2004.

[41] B. P. Gerkey and M. J. Mataric, "A market-based formulation of sensor-actuator network coordination," in *AAAI Spring Symposium on Intelligent Embedded and Distributed Systems*, 2002.

[42] G. Zlotkin and J. S. Rosenschein, “A domain theory for task oriented negotiation,” *In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993.

[43] T. Sandholm, “An implementation of the contract net protocol based on marginal cost calculations,” *In Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, 1993.

[44] T. Sandholm and V. Lesser, “Issues in automated negotiation and electronic commerce: Extending the contract net framework,” *In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS 95)*, 1995.

[45] A. Chavez, A. Moukas, and P. Maes, “A multi-agent system for distributed resource allocation,” *In Proceedings of the First International Conference on Autonomous Agents*, 1997.

[46] M. P. Wellman and P. R. Wurman, “Market-aware agents for a multiagent world,” *Robotics and Autonomous Systems*, vol. 24, 1998.

[47] H. Kose, U. Tatlidede, C. Mericli, K. Kaplan, and H. L. Akin, “Qlearning-based market-driven multi-agent collaboration in robot soccer,” *Proceedings of the Turkish Symposium on Artifcial Intelligence and Neural Networks*, pp. 219–228, 2004.

[48] T. Sandholm and S. Suri, “Improved algorithms for optimal winner determination in combinatorial auctions and generalizations,” *Proceedings of the Seventeenth National Conf. on Arti*

cial Intelligence, pp. 90–97, 2000.

[49] M. B. Dias, *A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments.* PhD thesis, Robotics Institute, Carnegie Mellon University, 2004.

[50] M. Golfarelli, D. Maio, and S. Rizzi, “A task-swap negotiation protocol based on the contract net paradigm,” *Technical Report 005-97, CSITE (Research Center for Informatics and Telecommunication Systems)*, 1997.

[51] R. G. Smith, “The contract net protocol: High-level communication and control in a distributed problem solver,” *IEEE Transactions on Computers*, 1980.

[52] R. Horst, P. Pardalos, and N. Van Thoai, *Introduction to Global Optimization.* Nonconvex Optimization and Its Applications, Springer, 1995.

[53] M. Darrah, W. Niland, and B. M. Stolarik, “Multiple uav dynamic task allocation using mixed integer linear programming in a sead mission,” *American Institute of Aeronautics and Astronautics*, 2005.

[54] A. R. Mosteo and L. Montano, “Simulated annealing for multi-robot hierarchical task allocation with flexible constraints and objective functions,” in *Workshop on Network Robot Systems: Toward Intelligent Robotic Systems Integrated with Environments. IROS*, 2006.

[55] A. R. Mosteo, *Multi-robot Task Allocation for Service Robotics: From Unlimited to Limited Communication Range*. PhD thesis, Universidad de Zaragoza, 2010.

[56] D. Juedes, F. Drews, L. Welch, and D. Fleeman, "Heuristic resource allocation algorithms for maximizing allowable workload in dynamic, distributed real-time systems," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, IEEE, 2004.

[57] W. Kmiecik, M. Wojcikowski, L. Koszalka, and A. Kasprzak, "Task allocation in mesh connected processors with local search metaheuristic algorithms," in *Intelligent Information and Database Systems* (N. Nguyen, M. Le, and J. Swiatek, eds.), vol. 5991 of *Lecture Notes in Computer Science*, pp. 215–224, Springer Berlin Heidelberg, 2010.

[58] P. J. Shea, K. Alexander, and J. Peterson, "Group tracking using genetic algorithms," *Proc. of the International Society Information Fusion*, 2003.

[59] E. G. Jones, M. B. Dias, and A. Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints," *Autonomous robots*, vol. 30, no. 1, pp. 41–56, 2011.

[60] J. Wang, Y. Gu, and X. Li, "Multi-robot task allocation based on ant colony algorithm," *Journal of Computers*, vol. 7, no. 9, pp. 2160–2167, 2012.

[61] Y. Ding, Y. He, and J. Jiang, "Multi-robot cooperation method based on the ant algorithm," in *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, pp. 14–18, 2003.

[62] W. Chen and C. Lin, "A hybrid heuristic to solve a task allocation problem," *Computers & Operations Research*, vol. 27, no. 3, pp. 287–303, 2000.

[63] D. K. Liu and A. K. Kulatunga, "Simultaneous planning and scheduling for multi-autonomous vehicles," in *Evolutionary Scheduling*, pp. 437–464, Springer, 2007.

[64] K. Al-Yafi and H. Lee, "Centralized versus market-based approaches to mobile task allocation problem: State-of-the art," in *EMCIS 09. European and Mideterranian Conference on Information Systems*, 2009.

[65] B. Coltin and M. Veloso, "Mobile robot task allocation in hybrid wireless sensor networks," in *Proc. on IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2932–2937, 2010.

[66] C. Liu and A. Kroll, "A centralized multi-robot task allocation for industrial plant inspection by using a* and genetic algorithms," in *Artificial Intelligence and Soft Computing* (L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, eds.), vol. 7268 of *Lecture Notes in Computer Science*, pp. 466–474, Springer Berlin Heidelberg, 2012.

[67] S. Giordani, M. Lujak, and F. Martinelli, “A distributed algorithm for the multi-robot task allocation problem,” in *Trends in Applied Intelligent Systems* (N. Garcia-Pedrajas, F. Herrera, C. Fyfe, J. BenÃtez, and M. Ali, eds.), vol. 6096 of *Lecture Notes in Computer Science*, pp. 721–730, Springer Berlin Heidelberg, 2010.

[68] H. C., L. Brunet, and J. How, “Consensus-based decentralized auctions for robust task allocation,” *Robotics, IEEE Transactions on*, vol. 25, no. 4, pp. 912–926, 2009.

[69] P. G., Z. C., and L. Y., “Evolutionary computation approach to decentralized multi-robot task allocation,” in *Natural Computation, 2009. ICNC ’09. Fifth International Conference on*, vol. 5, pp. 415–419, 2009.

[70] X. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*. Wiley, 2010.

[71] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. Springer, 2000.

[72] J. C. Spall, “Stochastic optimization,” in *Handbook of Computational Statistics* (J. E. Gentle, W. K. Hardle, and Y. Mori, eds.), Springer Handbooks of Computational Statistics, pp. 173–201, Springer Berlin Heidelberg, 2012.

[73] N. V. Sahinidis, “Optimization under uncertainty: State-of-the-art and opportunities,” *Computers & Chemical Engineering*, vol. 28, no. 6, pp. 971–983, 2004.

[74] Y. Shimizu, Z. Zhang, and R. Batres, *Frontiers in Computing Technologies for Manufacturing Applications*. Springer, 2007.

[75] E.-G. Talbi, *Metaheuristics: From Design to Implementation*, vol. 74. Wiley, 2009.

[76] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[77] A. Hussein, H. Mostafa, M. Badrel-din, O. Sultan, and A. Khamis, “Metaheuristic optimization approach to mobile robot path planning,” in *Engineering and Technology (ICET), 2012 International Conference on*, pp. 1–6, 2012.

[78] S. Olafsson, “Metaheuristics,” in *Handbooks in Operations Research and Management Science, Simulation* (S. G. Henderson and B. L. Nelson, eds.), vol. 13, pp. 633–654, North Holland, 2006.

[79] M. Birattari, L. Paquete, T. Strutzle, and K. Varrentrapp, “Classification of metaheuristics and design of experiments for the analysis of components tech. rep. aida-01-05,” tech. rep., Darmstadt University of Technology, November 2001.

[80] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of state calculations by fast computing machines,” *The journal of chemical physics*, vol. 21, p. 1087, 1953.

[81] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, “Optimization by simulated annealing,” *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[82] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[83] C. Voudouris and E. Tsang, "Partial constraint satisfaction problems and guided local search," *Proc., Practical Application of Constraint Technology (PACT'96), London*, pp. 337–356, 1996.

[84] C. Voudouris and E. Tsang, "Guided local search and its application to the traveling salesman problem," *European journal of operational research*, vol. 113, no. 2, pp. 469–499, 1999.

[85] T. A. Feo and M. G. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations research letters*, vol. 8, no. 2, pp. 67–71, 1989.

[86] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[87] H. Holland John, "Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence," *USA: University of Michigan*, 1975.

[88] C. R. Reeves, "Genetic algorithms and neighbourhood search," in *Evolutionary Computing*, pp. 115–130, Springer, 1994.

[89] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics?*, pp. 703–712, Springer, 1993.

[90] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942–1948, 1995.

[91] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142, pp. 134–142, 1991.

[92] M. Dorigo and T. Stützle, "The ant colony optimization metaheuristic: Algorithms, applications, and advances," in *Handbook of metaheuristics*, Springer, 2003.

[93] "Tsplib." `http://elib.zib.de/pub/Packages/mp-testdata/tsp/tsplib/tsplib.html`.

[94] A. Hussein, "A market-based approach to multi-robot task allocation problem," Master's thesis, German University in Cairo, 2013.

[95] A. Shehata, "Adaptive group formation in multi-robot systems," Master's thesis, German University in Cairo, 2013.