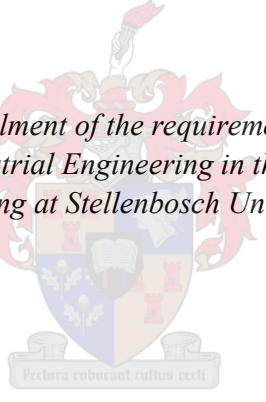
Developing a Dynamic Mission-ready Resource Problem Solution Model

by
Bernard Cornelius Leuvennink



Thesis presented in fulfilment of the requirements for the degree of Master of Industrial Engineering in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof James Bekker

March 2020

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2020

Abstract

The mission-ready resource allocation (MRRA) problem is well-known in Operations Research. It is a combinatorial problem which quickly assumes a large decision space as the problem is extended to practical situations. Efforts to solve the MRRA problem were aimed at given scenarios, and once a solution is found, the solution process was terminated. In this study, the question of a dynamic, stochastic MRRA (DSMRRA) problem is investigated. In this problem, a series of missions that are distributed over time require resources and the availability of resources vary. The objective is to find good solutions for all the missions; the allocation of a given mission will influence the available resources of a subsequent mission.

In this study, simulation and bi-objective optimisation were used to demonstrate that the DSMRRA problem can be solved. The static MRRA problem originated in the military, but in this study, it is demonstrated that the DSMRRA problem can also be applied to business operations.

Opsomming

Die missie-gereed hulpbrontoekenningsprobleem (MRRA) is reeds bekend in die veld van Operasionele Navorsing. Dit is ‘n kombinatoriese probleem wat vinnig tot groot besluitruimtes lei wanneer die probleem in die praktyk toegepas word. Oplossings word gemik op spesifieke scenarios en sodra ‘n oplossing gevind is, word die proses gestop. Hierdie studie fokus op ‘n dinamiese stogastiese MRRA(DSMRRA). In hierdie probleem benodig ‘n reeks missies wat oor tyd verspreid is toedeling van hulpbronne waarvan die beskikbaarheid wissel. Die doel is dan om goeie oplossings te vind vir al die missies; toedeling van hulpbronne aan ‘n sekere missie sal toekomstige missies beïnvloed.

In hierdie studie was simulasie en bi-doelwit optimering gebruik om te demonstreer hoe die DSMRRA opgelos kan word. Die statiese MRRA het sy oorspong in militêre agtergrond, maar in hierdie studie word daar gewys hoe die DSMRRA ook in besigheids probleme gebruik kan word.

Acknowledgements

Firstly, I would like to thank my supervisor for his constant support and guidance with the topic. Secondly, I would like to thank my family for their love and support throughout. Lastly, but certainly not the least, I would like to thank God for giving me the ability to do this study and for keeping me motivated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter serves as an introduction to the research objectives and research strategy. It provides background to the Mission-ready Resource Allocation (MRRA) problem and states the shortcomings of current solutions. This leads to the research objectives of this study. The methodology that will be followed to reach each desired objective is also discussed in this chapter.

1.1 Background to mission-ready resource allocation problems

MRRA solution methods are used to solve resource allocation problems and minimise cost. These problems allow decision makers with certain resources at their disposal to use these resources to complete tasks in an optimal or near-optimal way. Like operational research itself and most operational research problems, MRRA problems originated during the Second World War (Kirby & Capey, 1998). The original use was to help a battle commander to allocate resources in an optimal or near-optimal way to successfully complete required missions. The battle commander would have resources at his disposal, that could be utilised to complete missions, but each resource would have a cost associated with using it for that mission. The aim was to allocate the resources in such a way that it would minimise the cost of resources used.

A *mission-ready resource* (MRR) is defined as a combination of resources, such as equipment, ammunition, personnel and fuel. Since different MRRs are more or less suitable for a certain task, each MRR would have their own task suitability for each task. This task suitability is an indication of the degree to which the MRR is suited for that specific task (Wakefield, 2001). In essence, it shows how well the resource is expected to perform when tasked with that specific mission. This task suitability is important when considering how the resources should be allocated, since it shows the capabilities of the resources to complete missions.

As stated before, each MRR also has a cost associated with it. This is the cost of using that specific resource during a mission and it is known as the *lift cost*. The aim of solving the MRR problem is to minimise this cost, while still successfully completing all required missions.

It is usually assumed that the number of MRRs available is infinite or at least enough that the number of missions to be completed is the constraining factor (Scholtz, 2014). This means

that the problem would have to be adapted if only a limited number of MRRs are available for a specific problem.

This infinite number of MRRs that is available also provide another challenge, because it creates the opportunity for a very large decision space (Scholtz, 2014). According to Wakefield (2001), even if there are only 16 tasks that require MRR assignment, with 20 units of each MRR available, then the feasible solutions would be about 630 000 out of a total of $21^{15} = 6.8 \times 10^{19}$ possibilities. This translates to only about one in every 1.04×10^{14} solutions being feasible.

This is an extremely low probability and it leads to the problem that search algorithms might have trouble just to find feasible solutions, let alone optimal or near-optimal solutions. This issue will only increase further as the problem becomes larger when used in the industry.

Therefore, it is clear from these theoretical numbers that, even if the decision is made to have a limited number of resources, another challenge still remains, since it is then very difficult to find the feasible solutions. This leads to the need to use some constraint handling technique to help find those feasible solutions.

It can be seen that the original MRRA problem is a static problem (Wakefield, 2001). A solution is found at a specific point in time and the problem ends there. In many real-life scenarios this is not the case, because in most cases it is often not a once-off problem that needs solving. Resources might be required for further missions at a later stage and the cost implications of future missions often need to be considered from the start.

For example, the battle commander of an army base might need to plan the usage of resources over a longer period of time with several attacks that need to be fended off. This could mean that there would be multiple missions at different time periods. Resources that were used to complete missions might take time before they become available again after being used. These factors would complicate the decision-making that the battle commander needs to consider.

This more complex problem leads to the need for a dynamic MRRA solution. This would enable decision-makers to make better decisions by incorporating the effect of possible future missions into their current solution and their final decision.

In order to achieve this, a good understanding of multi-objective metaheuristics will be necessary, since certain trade-offs might have to be made to allow the possibility of future missions to be considered. Multi-objective meta-heuristics have risen in popularity (Jones *et al.*, 2002) and could be used in this case to balance the minimisation of lift costs and the readiness of resources for possible future missions.

As stated previously, the MRRA problem was first introduced and used within a military context. Many new operational research problems and problem-solving techniques were first used during wartime, but have since been adapted for the public sector (Kirby & Capey,

1998). Some aspects of the MRRA problem and the techniques used to solve it might also be useful within a business environment. However, the problem might have to be adapted and changed in some way to suit different industries.

It is important to consider how it might be possible to adapt the military version of the MRR problem to fit other industries. This is because similar problem-solving techniques can help businesses reduce waste and be more competitive by making better decisions (Kirby & Capey, 1998). Most organisations will want to optimise the use of their resources and minimise costs or improve customer satisfaction (Sabri & Beamon, 2000), so the problem is clearly not just found in the military environment. Similar problems exist within different types of businesses.

1.2 Example of a mission-ready resource allocation problem

As an example of a static MRRA problem, it might be that a battle commander has the following resources at his disposal:

- Four tanks
- Three fighter planes

In the MRRA problem, this battle commander wishes to attack an enemy base nearby. The enemy base has the following assets that the commander wishes to destroy:

- Control tower
- Ammunition storage area

Both of the enemy assets must be destroyed by using the available resources. Each military resource has a task suitability associated with it. This is an indication of the ability of that resource to destroy that specific threat if the resource is assigned to attack that asset. Let δ_{ij} be the task suitability when resource j is assigned to task i. The task suitability for each resource is shown in Table 1.1.

Table 1.1: Task suitability (δ)

	Tank	Fighter Plane
Control Tower	0.4	0.8
Ammunition Storage Area	0.5	0.7

The military commander's first objective is then to optimise the total task compatibility for the mission when assigning the resources to enemy assets. For m threats and n resources, the first objective function could therefore be formulated as

$$\text{maximise } f_1(x) = \sum_{j=1}^{m}\sum_{i=1}^{n} \delta_{ij} x_{ij} \tag{1.1}$$

where x_{ij} is the number of resources of type j assigned to task i. This is also the decision variable.

However, the problem is further complicated since each resource also has a lift cost (φ) associated with it. This is the cost of using that specific resource to complete a specific mission. The lift costs are shown in Table 1.2.

Table 1.2: Lift cost (φ)

	Tank	Fighter Plane
Control Tower	60	70
Ammunition Storage Area	40	80

The military commander also wants to minimise the cost associated with this mission. Therefore, the second objective function is to minimise the total lift cost of the mission:

$$\text{minimise} f_2(x) = \sum_{j=1}^{m}\sum_{i=1}^{n} \varphi_{ij} x_{ij}. \tag{1.2}$$

Furthermore, it could be that the battle commander wants to assign exactly two resources to each enemy asset. It is also important to note that the number of resources of type j that is assigned cannot exceed the available number of resources and that the decision variable (x_{ij}) values must be integers. The constraints would therefore be:

$$\sum_{j=1}^{2} x_{ij} = 2, \forall i, \tag{1.3}$$

$$\sum_{i=1}^{2} x_{i1} \leq 4, \tag{1.4}$$

$$\sum_{i=1}^{2} x_{i2} \leq 3, \tag{1.5}$$

$$x_{ij} \epsilon N_0. \tag{1.6}$$

The objective functions can now be optimised by using multi-objective optimisation techniques. The simplex algorithm is suited for solving multi-objective optimisation problems.

Since the problem is small, all eight feasible solutions can be considered for completeness. The feasible solutions are shown in Table 1.3.

Table 1.3: Feasible solutions

Feasible Solution	x_{11}	x_{12}	x_{21}	x_{22}	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$
1	1	1	1	1	2.4	250
2	2	0	1	1	2	240
3	0	2	1	1	2.8	260
4	1	1	2	0	2.2	210
5	2	0	2	0	1.8	200
6	0	2	2	0	2.6	220
7	1	1	0	2	2.6	290
8	2	0	0	2	2.2	280

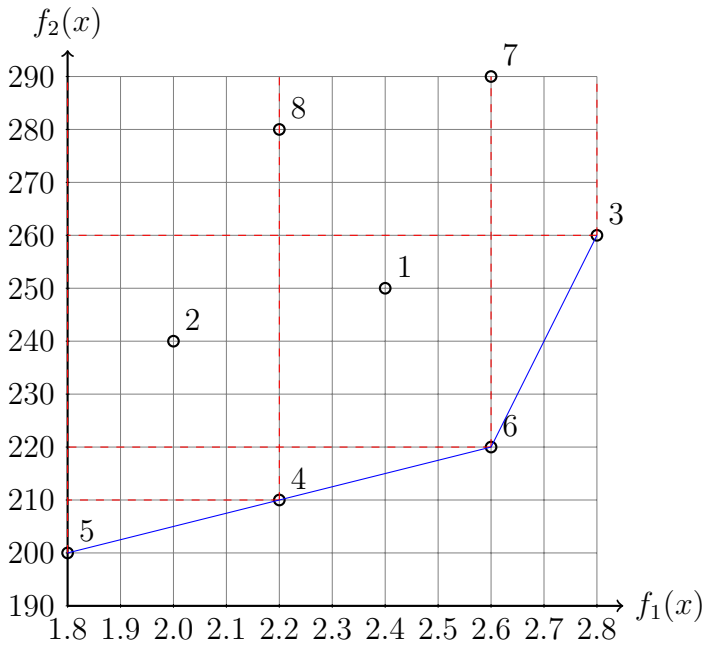


Figure 1.1: Feasible solutions of x

The objective functions are shown in Figure 1.1. It can be seen that only solutions 3, 4, 5 and 6 are not dominated by any other solutions. This set of non-dominated solutions is called the *Pareto set*. This means that the battle commander can choose any one of these four solutions and know that no other feasible solution combination will result in a better value for both objective functions.

The battle commander should, therefore, choose one of the following options:

1. The battle commander can assign two fighter planes to attack the control tower, as well as one tank and one fighter plane to attack the ammunition storage area. This will result in a total task suitability of 2.8 and a lift cost of 260.

2. The battle commander can assign one tank and one fighter plane to attack the control tower, as well as two tanks to attack the ammunition storage area. This will result in a total task suitability of 2.2 and a lift cost of 210.

3. The battle commander can assign two tanks to attack the control tower, as well as two more tanks to attack the ammunition storage area. This will result in a total task suitability of 1.8 and a lift cost of 200.

4. The battle commander can assign two fighter pilots to attack the control tower and two tanks to attack the ammunition storage area. This will result in a task suitability of 2.6 and a lift cost of 220.

In this example, one can also note how many non-feasible solutions there are. Four tanks and three fighter planes can be assigned to two enemy assets. Since no resources of a specific type could also be assigned to an enemy asset, it means that either 0, 1, 2, 3 or 4 tanks will be assigned to each enemy asset. This means that there are five different options when assigning the four tanks and, with the same logic, four different options when assigning the three fighter pilots. The result is that there are $(5 \times 4)^2 = 400$ different ways to assign these resources to the enemy assets. Out of these 400 possible combinations, only eight are feasible solutions to this specific problem. This adds to the complexity of the problem, since it is often difficult to find feasible solutions for larger problems.

1.3 Research problem

The example in Section 1.2 represents a small static MRRA problem. As seen in Table 1.4, this study will focus on the development of a solution technique to a dynamic stochastic MRRA problem. Table 1.4 will later be expanded to form Table 2.18 in Section 2.6.2.

Table 1.4: Problem classes: Introduction

	Static	**Dynamic**
Stochastic		This field will be explored during this study
Deterministic	Example problem from Section 1.2	

The techniques used to solve a static deterministic MRRA problem, like that in the example in Section 1.2, will be modified to allow for decisions over a longer period of time. For example, the battle commander from the example in Section 1.2 might want to attack another enemy base a week later. However, some of the resources might be damaged or maintenance might be required before the resource can be cleared for the next mission and, therefore, the resources might not be able to participate in both missions. The resources used during the first mission will influence the resources available for the next mission. This contributes to the dynamic nature of the problem.

Furthermore, stochastic elements can also be introduced to the problem. For example, if a resource of type j is used during the first mission, there can be a probability value associated with it being available for the second mission. A repair cost can also be used for damaged resources. For example, the cost of using the same resource for the second mission after it was already used during the first mission might be more expensive than if the resource was not used during the first mission.

Stochastic elements could also be introduced to the problem by associating probabilities to the task suitability of resources for future missions. This could be because the characteristics of the enemy assets during the next mission might not be available from the start.

A stochastic dynamic MRRA problem means that the problem will change over time and that there would be uncertainty associated with the problem. The problem will be dynamic over time, since a longer time period will be considered and it will have stochastic elements, since there will be probabilities associated with the lift cost of some resources for future missions.

Finding good feasible solutions for a dynamic MRRA problem will be difficult. As seen in Section 1.2, when the static MRRA problem is considered it is already only a small percentage of solutions that are feasible. This problem will only become worse in the dynamic MRRA problem, where there are multiple resource allocations that could be made at different points in time. The solution space will become much bigger and some sort of constraint handling method will be required to ensure that good feasible solutions are found.

1.4 Research objectives

The main objective of the study is to solve a dynamic version of the MRRA problem. How this problem could be adapted to yield solutions that would be fit for other industries, not just the military environment where the problem originated, will also be looked into. The objectives for the study are listed below.

Objectives:

1. *Develop* a solution to a dynamic MRRA problem by using multi-objective optimisation

and metaheuristics.

2. *Solve* MRRA problems that exhibit stochastic behaviour.

3. *Adapt* the solution technique developed for the dynamic MRR problem with stochastic elements to aid other business types. The solution technique should assist businesses to make strategic decisions in their business environment.

1.5 Proposed research approach

Firstly, in order to satisfy the objectives mentioned in the previous section, more background information will be required on current static MRRA problems. This will give an indication of what can be done with the knowledge of MRRA problems. The history and previous applications as well as the techniques used to find solutions can be useful when adapting the problem to be dynamic and fit new business environments.

Table 1.5 shows how each of the objectives will be met. It shows the steps that will be required to achieve each objective during the research.

Table 1.5: Research methodology

Objective	Steps
1	1) Obtain a good understanding of static MRRA problems 2) Conduct a literature study on metaheuristics and constraint handling 3) Develop a dynamic solution by using metaheuristics and constraint handling
2	1) Research appropriate programming software 2) Obtain a background understanding of basic statistics and simulation techniques 3) Use simulation to introduce stochastic elements to the problem 4) Develop a simulated solution
3	1) Adapt the original military MRRA problem to satisfy business needs 2) Apply the solution model to different business types by using examples

To achieve objective 1, the current solution to the static MRRA problem was used and adapted in order to determine solutions to dynamic MRRA problems. A literature study was

done on metaheuristics and constraint handling before this can be achieved. This gave an indication of how the trade-offs involved when creating a dynamic solution can be optimised. As stated before, an important trade-off that was addressed is minimising the cost of completing a certain mission and maximising the readiness of good resources for future missions.

To achieve objective 2, simulation can be used to address the stochastic needs of the problem and develop a solution. In order to achieve this, research was done on what the appropriate programming software might be for this specific problem. A background understanding of basic statistics and the implications of resources not being available was also required.

To achieve objective 3, information was required on the needs and problems that other common business environments experience and how they might relate to the original military-based problem. This knowledge will then be combined and used to find ways that the military version of the MRRA problem can be adapted to fit other business environments. The information will be used to adapt the MRRA problem appropriately so that it could benefit more businesses.

Furthermore, the need for a solution to a dynamic MRRA problem must be considered. Research was done to find out which industries, if any, would benefit from having a solution to the dynamic MRRA problem. From that a dynamic model was then developed, by using simulation, that provides a good solution to the dynamic MRRA problem. This will help decision-makers in the identified industries to make better decisions regarding resource allocation more quickly.

1.6 Benefits from this study

This study provides a way to find a solution to a dynamic MRRA problem. This solution to a dynamic MRRA problem is much more useful in the business environment than the current existing static problem, since many real-life problems are dynamic and continuous rather than static. This study provides a much needed platform for businesses to make good strategic resource allocation decisions by using the solution to a dynamic MRRA problem.

1.7 Conclusion to Chapter 1

In this chapter, the background and history of MRRA problems were briefly discussed in order to provide an overview of the abilities and shortcomings of the current static solutions to MRRA problems. An example of a typical MRRA problem was referenced to illustrate the properties of a static MRRA problem. This was then used to determine the research objectives of this study. The research objectives were defined and a methodology was presented on how

the objectives can be reached.

Chapter 2

Literature study

In this chapter, the history of operations research and Mission-ready Resource Allocation (MRRA) problems as well as previous literature on the subject and subjects related to this are discussed. This literature study provides background to the problem, highlight the work that has already been done in the field and show some techniques that can be used to build on during the study.

The need for different problem-solving techniques will be discussed. The different applications within different environments provides the motivation for the study and the motivation for the development of a dynamic MRRA problem-solving technique. Lastly, the literature study focusses on Multi-objective optimisation and the techniques used to implement it.

2.1 Operational research

In this section, Operational Research (OR) is discussed. The relationship between Operational Research, Operations Research and Operations Management (OM) will be clarified. The history of OR is discussed.

2.1.1 Operational research vs Operations research

Operational Research and Operations Research seem to have a similar general meaning. Operational Research originates in Europe, while Operations Research appears to be the American name for the same field of study. In this study, the term Operational Research is used simply because UK English spelling is used throughout this study.

2.1.2 Operational research vs Operations management

Many consider Operational Research and Operations Management to be similar things, whilst others see them as completely different fields. This may seem very confusing and although the history of these fields is intertwined and many original OR problems have direct managerial implications, there is a difference of focus.

Some scientists feel that OR focuses on mathematical modelling and optimisation, whereas OM focuses more on problems regarding processes and systems (Petrovic, Sanja; MacCarthy, 2014). No matter how one looks at it, OR and OM have one common end goal: To improve the efficiency of the entire system. It is therefore, very important that they work together to achieve that goal.

In this study, the differences and similarities between these two fields are not discussed further. Instead, the focus of the study is on moving towards that final goal of being more efficient.

2.1.3 History of operational research

The term *Operational Research* (also known as Operations Research in the United States and will be referred to as OR in this study), known today as the *Science of better*, was coined around 1940 as a description of the systematic application of quantitative analysis. At that stage it proved to be a radical change in the application of scientific techniques to solve problems that appeared during wartime. This change expanded the contribution that scientists could make to the military force by merely analysing the effectiveness of weapons to include advice to military commanders on how to employ these weapons in the most effective way (Kirby & Capey, 1997).

Although Operational Research was only formally established in 1941, it could be argued that some elements of quantitative analysis had been used to get a better understanding of victory and defeat in warfare from a much earlier time (Kirby & Capey, 1997). Even in prehistoric times, people must have questioned whether an enemy would be more easily defeated by a quick attack with many small stones or much slower attack with bigger stones that might cause more destruction.

During the First World War, there were many attempts to apply mathematical techniques to the analysis of military operations (Kirby & Capey, 1997). These techniques were, in some cases, very useful when attempting to outwit the enemy and gain a strategic advantage. F.W. Lanchester's papers, written in 1914, introduced his well-known N-square law. According to this law, the fighting strength of a group is proportional to the *square* of the group size (Johnson & MacKay, 2015). This was successfully used to quantify the relationship between victory and superiority in numbers in the deployment of air power (Kirby & Capey, 1997).

The use of scientists within military command structures proved to be crucial in the outcome of the Second World War. Although they had many great scientists working on military projects, Germany did not integrate scientists into strategic military decision making as well as the Allied forces did. Many believe that it is because of this difference that the allied forces ultimately managed to defeat the Germans during the Second World War (Kirby & Capey, 1997).

As time progressed and more complex problems required attention, proper problem structuring methods (PSMs) were required before people began to use mathematical principles to find solutions using Operational Research (Rosenhead, 2006). This allowed many problems to be mathematically examined to find good solutions.

One of these problems that was defined and mathematically solved is the MRRA problem.

This technique was used by battle commanders during the Second World War to allocate resources in a strategic manner. It influenced military decision-making during the war and for the years to come.

2.2 Problem classes

At this stage it is important to consider the different classes of resource or weapon-allocation problems, as seen in Table 1.4. This will give a better understanding of the existing solutions and the specific problem that will be addressed.

Firstly, a problem can be *static* or *dynamic* in nature and secondly, it could be *stochastic* or *deterministic*. These problems could thus be divided into four different classes, as seen in the list below. The differences between static and dynamic problems are discussed in the next section and the difference between stochastic and deterministic problems are discussed in Section 2.2.2.

1. Static stochastic
2. Dynamic stochastic
3. Static deterministic
4. Dynamic deterministic

At this stage it is also important to note the similarities and differences between MRRA problems and weapon-assignment problems. These are discussed in Section 2.2.3.

2.2.1 Characteristics of static and dynamic models

It is important to recognise the difference between static and dynamic models. This gives a good indication of the capabilities of each type of model. These capabilities can then be used to determine when static models are sufficient and when dynamic models might be required.

Static models are solved at a specific point in time. All the information and possible variables are available at this given point in time. The variable values are inserted into the static model and the model is solved for that moment.

Dynamic models are models that are used to solve problems that exist over a longer period of time, since they address different problems at multiple points in time. This allows the model to make more informed decisions. They allow the user to make decisions that might not be optimal at the current time, but that would allow better overall decisions over a longer time span. The main difference, therefore, is that the values of a static model remain the same over time, whereas the values of dynamic models change over time.

Dynamic models are very powerful when used to make decisions that will also influence future scenarios. This is often the case in the modern military or business environments, since the use of resources at a current time could influence their performance at a later stage. These resources might be exhausted or they might have to be repaired before they can be used again.

2.2.2 Stochastic models and deterministic models

As mentioned in Section 2.2, different types of optimum problems have different properties. One of these important properties is whether a specific problem is stochastic or deterministic. A problem is deterministic when all values are known exactly and it is stochastic when the values follow statistical distributions.

This is an important difference, since real-life problems are often stochastic, rather than deterministic. This is because most values in a simulation are estimates that cannot be known exactly. The model should therefore take into account the fact that these values will be given as a statistical distribution, rather than a specific deterministic value.

2.2.3 Mission-ready resource allocation problems vs weapon assignment problems

As stated previously, it is important to note the similarities and differences within the context of this study between MRRA problems and weapon-assignment problems. This is because the subtle differences require that the two sets of problems must be handled differently regarding specific aspects of the problem. On the other hand, the similarities between the problems prove useful when finding solutions to MRRA problems.

One will notice that the field of dynamic weapon-assignment problems is relatively well developed. And while there are many similarities between a dynamic weapon-assignment problem and an MRRA problem, like the fact that both are essentially resource allocation problems and that many different time stages should be taken into account, it is important to also consider the differences. The main differences are listed and discussed below.

1. **MRRA problems aim to minimise cost and maximise suitability, whereas weapon-assignment problems aim to reduce threats with a high probability and a low cost**

   Essentially, the aim of weapon-assignment problems is to assign weapons in such a way that threats are eliminated efficiently, whereas the aim of MRRA problems is to allocate resources in an effective way to complete all missions at a low cost. Weapon-assignment problems focus on the threats that must be eliminated, while MRRA problems focus on allocating suitable resources to the missions that need to be completed.

2. **MRRA problems aim to solve offensive tactical problems, whereas weapon-assignment problems aim to solve defensive tactical problems**

   Static weapon-assignment problems are similar to static MRRA problems, where resources need to be allocated to threats or enemy targets. The main difference, however, lies in the fact that weapon-assignment problems often deal with *defensive* tactics, while MRRA problems can often be seen as *attacking* resource allocation, and are therefore *offensive* tactics. This means that weapon-assignment problems aim to allocate resources in such a way that all threats will have the largest probability or certainty of being destroyed, depending on whether the problem is stochastic or deterministic. On the other hand, classic MRRA problems want to find a way to allocate available resources in such a way that the attacking mission has both a good probability of success and at a minimal cost.

3. **MRRA problems do not consider implications of surviving threats or failed missions, while weapon-assignment problems must reallocate resources to surviving threats during the next time stage**

   It is important to note that most weapon-assignment problems do not calculate the cost of using resources to disable threats. These problems are rather concerned with the implications of surviving threats, which MRRA problems do not consider. MRRA problems only consider the suitability of resources allocated to missions, not their success rate or probability of success.

4. **With MRRA problems, the state of the next time stage depends on the allocations made during the previous time stage, whereas with weapon-assignment problems, the state of the next time stage depends on the outcomes of the previous time stage and how many threats survived**

   Weapon-assignment problems tend to focus on the dynamic nature of a specific battle. This means that the time between different phases is often very short. It also means that the state of the next phase is often determined by the result of the previous stage. For example, the number of threats in time phase 2 usually depends on the number of threats that survived time phase 1. Therefore, the uncertainty of the state of the next phase depends on the success of the previous phase.

   MRRA problems, on the other hand, focus on the dynamic nature of allocating resources to missions effectively over a longer period of time, for example, multiple battles, where each battle is a time phase with its own corresponding outcome. The important difference is that, for MRRA problems, the state of the next phase does not directly depend on the number of threats that survived the previous battle. Instead it focuses on the

resources or weapons that were used during the previous battle, the recovery times of those resources and the time between those two phases or battles.

The important difference, therefore, is the focus of what is important for the next time phase. For weapon-assignment problems it is to eliminate as many threats as possible before the next assessment or time phase, while MRRA problems wish to have the best suitable resource available for the next mission or time phase.

It is important to notice this difference to understand these two problems and the solutions to each of them. Although this study focusses on MRRA problems, the approach to solving these weapon-assignment problems is also considered in the literature study for completeness and to investigate the techniques used. It is clear from the history that weapon-assignment problems played a large role in the development of all resource allocation problems and some of these techniques are useful when creating a dynamic MRRA problem solution.

2.3 Static mission-ready resource allocation problems

In this section, the characteristics and uses of a static MRRA problem are discussed. This problem is important to consider, since it is this problem that is expanded to solve dynamic MRRA problems. It is a problem that is already formulated and it forms the base knowledge of this study. An example was given in Section 1.2. The practical implications and uses for this problem will also be discussed in Section 2.3.2.

2.3.1 Characteristics of a static mission-ready resource allocation problem

Since an example was already discussed in Section 1.2, this section discusses the characteristics of the known static MRRA problem. As discussed in Section 2.2.1, a static problem is when the variables of the problem are independent of time. This means that a static MRRA problem is a resource allocation problem that is solved for a specific point in time.

2.3.2 Common modern uses for static resource-allocation problems

As stated before, the original use for MRRA problems was to aid battle commanders to make good strategic decisions. They had certain resources available and there were some missions that had to be completed using these resources. Military commanders still have the need to find some technique that will allocate their resources in such a way that they will still complete all the required missions with maximum suitability, but that cost would be kept at a minimum (Wakefield, 2001).

Outside of the military environment, these techniques are used to assign resources during disaster management (Caunhye *et al.*, 2012). In these cases it is also very important that quick decisions are made under very stressful conditions in order to save lives and other important assets by allocating the correct resources to them in a time of need. MRRA techniques are successfully used to aid in the decision-making of disaster management teams.

According to Van Wassenhove (2006) more than 500 disasters are estimated to strike the world every year. Around 75 000 people are killed and more than 200 million others are impacted by these disasters. These disasters often present great resource allocation and logistical challenges. The needs during these times are great and resources are often very limited.

The Operational Research field and the optimisation techniques that it produces give important decision-makers on the disaster management team the opportunity to make better decisions during these desperate times (Yi & Kumar, 2007). Operational research is especially helpful when aiming to tackle the logistical problems that exist during disasters, since the devastation is often on a large scale with many different areas that are in need of attention. This could be applied to attempt to minimise human suffering with the available resources (Yu *et al.*, 2018).

2.3.2.1 Example of a natural disaster resource-allocation problem

Suppose a natural disaster, like an earthquake or a tsunami, strikes a certain region. There are three specific areas that need attention, each with a priority value (V_j) shown in Table 2.1. The disaster management team has three teams that could be sent to these areas in need. However, the different teams have different team members and different skills, making them more- or less-suitable to help in different areas. In other words, the different teams have different chances of success in different areas. The task suitability or probability of success for each team in each area is shown in Table 2.2.

Table 2.1: Priority Values (V_j) of areas in need

j	Area 1	Area 2	Area 3
V_j	70	40	50

Table 2.2: Task suitability or probability of success (p_{ij}) for disaster response teams

	Team 1	Team 2	Team 3
Area 1	0.5	0.4	0.3
Area 2	0.8	0.9	0.7
Area 3	0.75	0.6	0.5

In order to help the disaster management team allocate these teams to the areas in need, resource allocation or weapon-assignment models can be used. A model very similar to the weapon-target assignment model from Manne (1958), discussed in Section 2.4, can be used. This will minimise the probability of not reaching important areas in need.

If the weapon-target assignment model is used to solve this disaster management-related problem, the objective function discussed in Section 2.4.2 is used. From (2.5) (in Section 2.4.2), the objective function is to

$$\text{minimise} \sum_{j=1}^{n} V_j \prod_{i=1}^{m} (1 - p_{ij} X_{ij}).$$

This objective function must then be minimised under the constraints from (2.6) and (2.7) in order to optimise the allocation of the teams.

2.3.2.2 Example of a business resource-allocation problem

There exist many business-related examples of resource allocation problems similar to the example mentioned in the previous section. For example, a logistician at a logistics company has to decide which trucks to use on different delivery routes. He has three trucks at his disposal and three delivery routes need to be supplied. Each truck has a different cost associated with driving each route. Some trucks are also better suited for delivering certain goods, like fresh food. This means that each truck has a suitability associated with each route.

The lift cost for each of the trucks are given in Table 2.3, while the task suitability is shown in Table 2.4. The lift cost, given in some monetary value (Rand or Dollars for example) is the cost of using that specific truck (or MRR) on that specific delivery route (or mission). The task suitability represents how well a specific truck is suited to carry the products that must be delivered on that specific delivery route.

Table 2.3: Lift cost (φ) of trucks

	Truck 1	Truck 2	Truck 3
Delivery route 1	800	750	700
Delivery route 2	600	600	550
Delivery route 3	700	650	600

Table 2.4: Task suitability (δ) of trucks

	Truck 1	Truck 2	Truck 3
Delivery route 1	0.90	0.80	0.85
Delivery route 2	0.80	0.75	0.60
Delivery route 3	0.80	0.70	0.75

Naturally, the logistician would like to minimise the total cost of sending the trucks on their respective delivery routes, but he would also like to maximise the suitability of the truck to the route that it is assigned to. These objectives are often contradicting. This problem is then very similar to the problem discussed in Section 1.2. The first objective, from (1.1), is then to

$$\text{maximise } f_1(x) = \sum_{j=1}^{m}\sum_{i=1}^{n} \delta_{ij} x_{ij}$$

while the second, from (1.2), is to

$$\text{minimise } f_2(x) = \sum_{j=1}^{m}\sum_{i=1}^{n} \varphi_{ij} x_{ij}.$$

The value of the decision variable x_{ij} must then be optimised in order to find a solution that satisfies the logistician's needs. This may be accomplished by using multi-objective optimisation techniques, since the lift cost must be minimised and the task suitability must be maximised.

2.4 Static weapon-assignment problems

In this section, four weapon-assignment problems from the literature are presented and discussed. This is to obtain better insight into the field of weapon-assignment problems. A good understanding of the techniques used to solve the different types of weapon-assignment problems will prove helpful when designing a solution to dynamic MRRA problems.

2.4.1 The weapon assignment problem by Orden & Goldstein (1952)

The complexity of weapon-assignment problems grew significantly after the Second World War to settle new needs in the field. In 1952, Orden & Goldstein (1952) formulated the so-called *classical assignment problem* . Suppose resources need to be allocated to tasks and each resource has a lift cost. This model aims to find a solution that would assign n agents to n tasks in such a way that it would minimise the overall cost.

Let the cost of assigning agent i to task j be c_{ij} and let x_{ij} be a binary decision variable that takes the value of 1 if agent i is assigned to task j, or 0 otherwise. The objective of the classical assignment problem is then to

$$\text{minimise} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

subject to constraints

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1, ..., n, \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1, ..., n, \tag{2.3}$$

$$x_{ij} \epsilon \{0, 1\}, \quad \begin{matrix} i = 1, ..., n, \\ j = 1, ..., n. \end{matrix} \tag{2.4}$$

The classical assignment problem could be modified to allow for more resources than tasks. *Dummy tasks* will have to be introduced. These are tasks that have a cost of 0 for all resources. The resources that do not have to be allocated, in other words, the extra resources, are then allocated to these dummy tasks.

Example of the weapon-assignment problem by Orden & Goldstein (1952)

An example of a weapon-assignment problem by Orden & Goldstein (1952) might be a problem where two different types of resources need to be allocated to two different tasks. The cost of assigning each resource to a task is given in Table 2.5.

Table 2.5: Cost of resource allocation for the classical assignment problem example

	Task 1	Task 2
Resource 1	20	25
Resource 2	25	30

Since this is a small example, and due to the constraints mentioned in (2.2) to (2.4) there are only two feasible solutions.

Solution 1 is when $x_{11} = 1, x_{12} = 0, x_{21} = 0$ and $x_{22} = 1$. This will result in an objective function value of $20 \times 1 + 25 \times 0 + 25 \times 0 + 30 \times 1 = 50$.

Solution 2 is when $x_{11} = 0, x_{12} = 1, x_{21} = 1$ and $x_{22} = 0$. This will result in an objective function value of $20 \times 0 + 25 \times 1 + 25 \times 1 + 30 \times 0 = 50$.

Since both these solutions result in the same objective function value, both solutions are optimal solutions to this example problem.

2.4.2 The weapon-target assignment model by Manne (1958)

This classical assignment formulation served as a stepping stone for many weapon-allocation models (Lötter, 2017). One of these models was the so-called *weapon-target assignment* model by Manne (1958). This model was based on the classical assignment problem, but it allowed the number of agents to differ from the number of tasks. In this model, the agents were replaced by m weapons and the tasks were replaced by n targets or threats. The objective was then to find a way to allocate the weapons to the threats in such a way that it would minimise the overall expected survival probabilities of the threats.

The model could be formulated as follows. Let V_j be the importance of eliminating threat j and let p_{ij} be the probability that weapon i will hit threat j with a single shot. Finally, let X_{ij} be the probability that weapon i is assigned to threat j. The objective function, according to Manne (1958) is then to

$$\text{minimise} \sum_{j=1}^{n} V_j \prod_{i=1}^{m} (1 - p_{ij} X_{ij}) \tag{2.5}$$

subject to constraints

$$\sum_{j=1}^{n} X_{ij} = 1, \; i = 1, ..., m, \tag{2.6}$$

$$X_{ij} \geq 0, \quad \begin{matrix} i = 1, ..., m, \\ j = 1, ..., n \; . \end{matrix} \tag{2.7}$$

The constraints ensure that the probability of assigning each weapon accumulates to 1 over all the threats and that the individual probabilities of assigning weapon i to threat j is greater than or equal to zero. This model assumes that the probability of each weapon to hit a threat are independent from each other.

Example of the weapon-target assignment model by Manne (1958)

An example of this weapon-target assignment model is as follows. Three weapons need to be allocated to two targets in such a way that the expected survival probability of the two threats are minimised. The importance rating of threats are given in Table 2.6.

Table 2.6: Importance rating of threats in weapon-target assignment model example

j	V_j
Threat 1	80
Threat 2	50

The probability that weapon i will hit target j is given in Table 2.7.

Table 2.7: Hit probabilities (p_{ij}) in weapon-target assignment model example

	Threat 1	Threat 2
Weapon 1	0.8	0.85
Weapon 2	0.75	0.6
Weapon 3	0.65	0.55

The decision variable is X_{ij}, the probability that weapon i will be allocated to target j. This must be chosen in such a way that the constraints in (2.6) and (2.7) are satisfied and the objective function in (2.5) is minimised.

A feasible solution is

$$X_{11} = 0.2, X_{12} = 0.8, X_{21} = 0.6, X_{22} = 0.4, X_{31} = 0.75 \text{ and } X_{32} = 0.25$$

with its corresponding objective function value of (from (2.5))

$$\sum_{j=1}^{2} V_j \prod_{i=1}^{3}(1 - p_{ij}X_{ij})$$

$$= V_1 \prod_{i=1}^{3}(1 - p_{i1}X_{i1}) + V_2 \prod_{i=1}^{3}(1 - p_{i2}X_{i2}).$$

Expanding the objective function yields

$$= V_1((1 - p_{11}X_{11})(1 - p_{21}X_{21})(1 - p_{31}X_{31})) + V_2((1 - p_{12}X_{12})(1 - p_{22}X_{22})(1 - p_{32}X_{32})).$$

Inserting numerical values from the example yields

$$\begin{aligned}
&= 80((1 - 0.8 \times 0.2)(1 - 0.75 \times 0.6)(1 - 0.65 \times 0.75)) \\
&+50((1 - 0.85 \times 0.8)(1 - 0.6 \times 0.4)(1 - 0.55 \times 0.25)) \\
&= 18.942 + 10.488 \\
&= 29.43
\end{aligned}$$

This is a value indicating the probability of threats surviving combined with the importance of those threats. The smaller the value, the smaller the probability of survival or the less important the threat. Therefore, this is the value that needs to be minimised.

Dynamic programming or a genetic algorithm can now be used to improve this solution further and find solutions that would give a smaller objective function value if that is required.

2.4.3 The weapon-target assignment model by DenBroeder *et al.* (1959)

After the introduction of the weapon-target assignment model, DenBroeder *et al.* (1959) aimed to find the number of weapons that should be assigned (m_j) that would maximise the expected value of destroying at least a certain number of threats. This meant that statistical analysis and stochastic behaviour were introduced to military decision-making and resource allocation problems.

Two important assumptions were made in the model made by DenBroeder *et al.* (1959). Firstly, the probability p_j of hitting threat j with a single shot from a certain defence resource is independent from the selected resource. Secondly, the chances of hitting a threat by various defence resources are also independent. This means that the survival probability of a threat, when resource y_j is allocated to it, is $q_j^{m_j}$, where $q_j = 1 - p_j$ (Lötter, 2017) and m_j is the number of weapons assigned to target j.

The problem by DenBroeder *et al.* (1959) was formulated as follows. Let a_d denote the probability of destroying exactly d threats where

$$a_d = \prod_{j=1}^{d} {q_j}^{m_j} \tag{2.8}$$

and

$$P_d = \sum_{j=d}^{n} a_d \tag{2.9}$$

denote the probability of destroying d or more threats. Then let V_d denote a priority value associated with eliminating exactly d threats. Then the objective of the weapon-allocation model of DenBroeder *et al.* (1959) is to

$$\text{maximise} \sum_{d=1}^{n} V_d a_d \tag{2.10}$$

subject to the constraints

$$\sum_{j=1}^{n} m_j = m, \tag{2.11}$$

$$m_j = N_0, \ j = 1, ..., n, \tag{2.12}$$

where n is the minimum number of threats that must be destroyed. (2.11) ensures that exactly m resources are assigned to the threats and (2.12) ensures that the number of resources assigned to threat j assumes a non-negative integer value (Lötter, 2017).

This model could be implemented to ensure that the expected value of destroying at least d threats will be maximised. This is important for battle commanders when planning for a defence. However, this remains a static problem which does not take the possibility of future attacks into account.

Example of the weapon-target assignment model by DenBroeder *et al.* (1959)

Suppose a battle commander has four weapons of equal value and ability at his disposal. There are multiple threats and the battle commander wants to know how many weapons must be assigned to maximise the probability of destroying at least two threats. Therefore, $m = 4$ and $n = 2$. The priority values of destroying exactly d threats (V_d) are given in Table 2.8.

Table 2.8: Priority value of destroying exactly d threats

d	1	2	3
V_d	0.8	0.6	0.5

The probability of hitting threat j with a single shot from a weapon (p_j) is given in Table 2.9.

Table 2.9: Probability of hitting threat j

Threat j	p_j	$q_j = 1 - p_j$
1	0.8	0.2
2	0.7	0.3
3	0.4	0.6

Assume that the battle commander decides to allocate two weapons to the first two threats. Therefore, $m_1 = 2$ and $m_2 = 2$. The probability of destroying exactly d threats is given by (2.8), therefore,

$a_1 = \prod_{j=1}^{1} {q_j}^{m_j} = 0.2^2 = 0.04$

and

$$\begin{aligned} a_2 &= \prod_{j=1}^{2} {q_j}^{m_j} \\ &= {q_1}^{m_1} \times {q_2}^{m_2} \\ &= 0.2^2 \times 0.3^2 = 3.6 \times 10^{-3}. \end{aligned}$$

The objective function, from (2.10), is then

$$\sum_{d=1}^{2} V_d a_d = V_1 \times a_1 + V_2 \times a_2 = 0.8 \times 0.04 + 0.6 \times (3.6 \times 10^{-3}) = \frac{427}{12500} = 0.03416.$$

This value indicates the probability of destroying d (in this case $d = 1$) threats and should then be maximised by changing the values of m_1 and m_2, while still conforming to the constraints from (2.11) and (2.12).

2.4.4 The weapon-target assignment model by Ahuja *et al.* (2007)

In 2003, Ahuja *et al.* (2007) adapted the weapon-target assignment model presented by Manne (1958) to make provision for the use of different types of weapons. In this model, let W_i be the number of weapons of type i available for assignment and x_{ij} is equal to 1 when weapon i is assigned to threat j, and 0 otherwise. The objective function of the model by Ahuja *et al.* (2007) is then to

$$\text{minimise} \sum_{j=1}^{n} V_j \prod_{i=1}^{m} q_{ij}^{x_{ij}} \tag{2.13}$$

subject to the constraints

$$\sum_{j=1}^{n} x_{ij} \leq W_i, \ i = 1, ..., m, \tag{2.14}$$

$$x_{ij} \epsilon \{0, 1\}, \quad \begin{matrix} i = 1, ..., m, \\ j = 1, ..., n, \end{matrix} \tag{2.15}$$

where the constraints ensure that no more than the available number of weapons of type i is assigned to threats.

Example of the weapon-target assignment model by Ahuja *et al.* (2007)

Suppose a battle commander has two weapons of type 1 and one weapon of type 2 at his disposal. Suppose he wants to allocate the weapons in such a way that it would minimise the survival probability of three threats with importance ratings (V_j) as given in Table 2.10. The hit probabilities of each type of weapon are given in Table 2.11.

Table 2.10: Importance ratings of threats (V_j) in the example of Ahuja *et al.* (2007)'s model

j	Threat 1	Threat 2	Threat 3
V_j	90	80	50

Table 2.11: Hit probabilities (p_{ij}) in the example of Ahuja *et al.* (2007)'s model

	Threat 1	Threat 2	Threat 3
Weapon type 1	0.9	0.85	0.8
Weapon type 2	0.75	0.7	0.8

Since it can be assumed that $q_{ij} = 1 - p_{ij}$, Table 2.12 shows the relevant values of q_{ij}.

Table 2.12: q_{ij} in the example of Ahuja *et al.* (2007)'s model

	Threat 1	Threat 2	Threat 3
Weapon type 1	0.1	0.15	0.2
Weapon type 2	0.25	0.3	0.2

Assume that the battle commander chooses to allocate the first weapon of type 1 to the first threat and the second weapon of type 1 to the second threat. The weapon of type 2 is then allocated to the third threat. Therefore, $x_{11} = 1, x_{12} = 1, x_{13} = 0, x_{21} = 0, x_{22} = 0$ and $x_{23} = 1$. The objective function, according to (2.13), is

$$\sum_{j=1}^{3} V_j \prod_{i=1}^{2} q_{ij}^{x_{ij}}$$

$$= \sum_{j=1}^{3} V_j \times q_{1j}^{x_{1j}} \times q_{2j}^{x_{2j}}$$

$$= V_1 \times q_{11}^{x_{11}} \times q_{21}^{x_{21}} + V_2 \times q_{12}^{x_{12}} \times q_{22}^{x_{22}} + V_3 \times q_{13}^{x_{13}} \times q_{23}^{x_{23}}$$

$$= 36.$$

This value can then be minimised by changing the values of x_{ij}, while still conforming to the constraints in (2.14) and (2.15).

2.5 Dynamic weapon-assignment problems

This section will discuss some models used in literature to solve weapon-assignment problems. As stated in Section 2.2.3, the techniques used in the models that are used to solve weapon-assignment problems are normally considered more defensive orientated problems, but the techniques can be useful to consider when solving dynamic MRRA problems.

2.5.1 Dynamic models used in weapon assignment

The first so-called *dynamic weapon-assignment problem*, also known as a *multi-stage problem*, was introduced by Hosein *et al.* (1988) in 1988. It involves the introduction of a temporal period T in the problem. This temporal period is usually divided into $\tau\epsilon\{0, ..., t\}$ shorter discrete time stages of equal duration. The aim is then to maximise the desired criterion at the end of the temporal period T by allocating weapon-threat pairs during each of the $t + 1$ discrete time stages.

In dynamic weapon-assignment models, the weapon-threat assignments are made sequentially and not simultaneously as in static models. This gives dynamic weapon assignment models the ability to take future events into account. In these models a weapon or resource may be reserved to be used at a later time where it might achieve a higher efficiency value.

According to Lötter (2017) there are two fundamentally different ways in which a dynamic weapon assignment problem could be formulated. The first is that it could be assumed that all the information about the number and location of all threats for each stage is known in advance.

The second is that the assumption has to be made that not all threats are known at the start of the temporal period. This would mean that the number of threats and their location are only known stochastically and must be calculated. The decision as to which one to use in each case will largely depend on the information available for that specific problem.

In the case that this study will look into, the uncertainty is whether or not a resource will be available for a mission if it has already been used during a previous mission. If it cannot be assumed that the resource recovery time is known in advance, it means that these recovery times are known only stochastically and must be calculated, as Lötter (2017) stated. Similarly, if the specific time that a future mission is going to happen might not be precisely known in advance, stochastic elements could be used to calculate it and solve the problem.

2.5.1.1 The dynamic weapon assignment model by Hosein *et al.* (1988)

Hosein *et al.* (1988)'s weapon-assignment model formulation is known in literature to use the so-called *shoot-look-shoot* strategy. It means that the outcome of every weapon-threat assignment pair is observed immediately before the model continues to solve the problem again for the next time period. This process is repeated for each time period τ with the aim of minimising the accumulated survival probabilities of the threats, weighted with their respective estimated threat priority values, at the end of the temporal period T.

This model makes the assumption that the number of threats and their positions is known in advance. The model also assumes that each weapon system could only be used once. Furthermore, the model requires that the outcome of every weapon-threat pair must be perfectly observed after each time stage. This means that the time stages must be long enough to allow

each weapon to attempt to destroy the threat and to calculate the outcome before the new weapon-threat pairs could be allocated for the next time stage.

The model could then be formulated as follows (Lötter, 2017):
Let $p_{ij}(\tau)$ be the probability that a weapon i will hit its assigned target j with a single shot during time slot τ. Let the corresponding survival probability then be $q_{ij}(\tau) = 1 - p_{ij}(\tau)$. A binary decision variable x_{ij} is then used that will take the value of 1 if weapon i is assigned to threat j, and zero otherwise. The *threat state* is then defined by an n-dimensional binary vector $\mathbf{u}\ \epsilon\{0,1\}^n$ with unit entries corresponding to the set of surviving threats, where n is the total number of threats. Similarly, the *weapon state* is defined by an m-dimensional vector $\mathbf{w}$ $\epsilon\{0,1\}^m$ with unit entries corresponding to the set of available weapons for assignment during the next time stage, where m is the number of weapons available at the beginning of the problem. These two vectors must be updated after the previous time stage, $\tau - 1$. Therefore, u_j will be 1 if threat j is still intact by the end of time $\tau - 1$, and 0 otherwise. Similarly, w_i will be 1 if weapon i has not been used by the end of time $\tau - 1$, and 0 otherwise.

The probability distribution of u_j depends on the probability that weapon i will hit threat j during the first time stage. Therefore, according to Lötter (2017), the probability that threat j survives or does not survive time stage $\tau - 1$ is given by

$$P(u_j = k_j) = k_j \prod_{i=1}^{m} [q_{ij}(\tau - 1)]^{x_{ij}(\tau)} + (1 - k_j)(1 - \prod_{i=1}^{m} [q_{ij}(\tau)(\tau - 1)]^{x_{ij}(\tau)}), \tag{2.16}$$

where $k_j \epsilon\{0,1\}$. Let $\mathbf{k} = [k_1, ..., k_m]$. The probability of observing the state vector $\mathbf{k}$ is then given by

$$P(\mathbf{u} = \mathbf{k}) = \prod_{j=1}^{n} P(u_j = k_j), \tag{2.17}$$

where the assumption is made that the state vector entries are independent. The objective during time stage $\tau\epsilon\{1, ..., t\}$ of the dynamic weapon-allocation model of Hosein *et al.* (1988) is to

$$\text{minimise} \sum_{\mathbf{k}\epsilon\{0,1\}^n} P(\mathbf{u} = \mathbf{k}) F_\tau^*(\mathbf{k}, \mathbf{w}) \tag{2.18}$$

subject to the constraints

$$\sum_{j=1}^{n} x_{ij}(\tau) = 1 - w_i,\ i = 1, ..., m, \tag{2.19}$$

$$x_{ij}(\tau)\epsilon\{0,1\}, \quad \begin{matrix} i = 1, ..., m, \\ j = 1, ..., n, \end{matrix} \tag{2.20}$$

where $F_T^*(k, w)$ denotes the optimal assignment cost for a $(\tau - 1)$-interval problem with an initial threat state of $\mathbf{k}$ and initial weapon state of $\mathbf{w}$. When $\tau = t$, the expected cost at the final stage

$$F_T^*(\mathbf{k}, \mathbf{w}) = \sum_{j=1}^{n} V_j u_j \tag{2.21}$$

is the accumulated threat priority values of the threats that survived to the final stage. The constraints ensure that each weapon could only be assigned once during the temporal period T and that all decision variables are binary. This is important, since a weapon or resource could not be partially assigned.

This model simply works on the basis that a dynamic model could be created by solving the static weapon-assignment model repeatedly (Huaiping *et al.*, 2006). This technique, however, is therefore considered to be very computationally expensive.

Once again, like in Lötter (2017)'s case, this model is concerned with the position of the threat and not whether or not the resource is available. However, this does not mean that similar techniques could not be used to solve a dynamic problem where the availability of the resource during the next time stage is unknown. The outcome of the first time stage is unknown before the start of that mission. This includes the time that the resource takes to recover and become available again. At the next time stage a similar *shoot-look-shoot* tactic could then be used to solve this dynamic problem if the new, more informed situation is assessed before making decisions again. In this case it would probably be more appropriate to call it an *assign-look-assign* tactic.

Section 2.5.2 will give more information on the history of stochastic behaviour in dynamic models and show how stochastic behaviour was introduced into the weapon-assignment models.

2.5.1.2 Example of the dynamic weapon assignment model by Hosein *et al.* (1988)

Suppose a battle commander has four weapons and needs to assign them in such a way that the overall survival probability of two threats (1 and 2) during time stage 1 and two threats (3 and 4) during time stage 2 are minimised.

During time stage 1, the probabilities of weapon i hitting target j are given in Table 2.13. Similarly, the probabilities of weapon i hitting target j are given in Table 2.14. As seen in the previous section, the corresponding survival probability is then $q_{ij}(\tau) = 1 - p_{ij}(\tau)$, where τ is the relevant time stage.

Table 2.13: Hit probabilities ($p_{ij}(1)$) for dynamic weapon-assignment problem during time stage 1

	Threat 1	Threat 2
Weapon 1	0.9	0.7
Weapon 2	0.5	0.8
Weapon 3	0.7	0.6
Weapon 4	0.7	0.8

Table 2.14: Hit probabilities ($p_{ij}(2)$) for dynamic weapon-assignment problem during time stage 2

	Threat 3	Threat 4
Weapon 1	0.8	0.8
Weapon 2	0.7	0.9
Weapon 3	0.8	0.6
Weapon 4	0.4	0.6

Suppose the initial decision variable is given in Table 2.15.

Table 2.15: Decision Variable ($x_{ij}(1)$) for dynamic weapon-assignment problem during time stage 1

	Threat 1	Threat 2	Threat 3	Threat 4
Weapon 1	1	0	0	0
Weapon 2	0	0	0	0
Weapon 3	0	0	0	0
Weapon 4	0	1	0	0

From the previous section, the probability that threat j survives time stage $\tau - 1$ is given by

$$P(u_j = k_j) = k_j \prod_{i=1}^{m} [q_{ij}(\tau - 1)]^{x_{ij}(\tau)} + (1 - k_j)(1 - \prod_{i=1}^{m} [q_{ij}(\tau)(\tau - 1)]^{x_{ij}(\tau)})$$

when $k_i = 1$.

Since it is already known that threats 3 and 4 will survive time stage 1 and only appear during time stage 2, the model is only concerned with finding the probabilities for threats 1 and 2. Therefore, the probability that threat j survives time stage $2 - 1 = 1$ is given by

$$P(u_j = 1) = 1\prod_{i=1}^{4}[q_{ij}(1)]^{x_{ij}(2)} + (1-1)(1-\prod_{i=1}^{4}[q_{ij}(2)(1)]^{x_{ij}(2)})$$
$$= \prod_{i=1}^{4}[q_{ij}(1)]^{x_{ij}(2)}.$$

The result is that

$$\begin{aligned} P(u_1 = 1) &= \prod_{i=1}^{4}[q_{i1}(1)]^{x_{i1}(2)} \\ &= [q_{11}(1)]^{x_{11}(2)} \times [q_{21}(1)]^{x_{21}(2)} \times [q_{31}(1)]^{x_{31}(2)} \times [q_{41}(1)]^{x_{41}(2)} \\ &= [0.1]^1 \times [0.5]^0 \times [0.3]^0 \times [0.3]^0 \\ &= 0.1 \end{aligned}$$

and

$$\begin{aligned} P(u_2 = 1) &= \prod_{i=1}^{4}[q_{i2}(1)]^{x_{i2}(2)} \\ &= [q_{12}(1)]^{x_{12}(2)} \times [q_{22}(1)]^{x_{22}(2)} \times [q_{32}(1)]^{x_{32}(2)} \times [q_{42}(1)]^{x_{42}(2)} \\ &= [0.3]^0 \times [0.2]^0 \times [0.4]^0 \times [0.2]^1 \\ &= 0.2. \end{aligned}$$

From this it can be calculated, from (2.17), that the probability of both threats surviving is

$$\begin{aligned} P(\mathbf{u} = \mathbf{k}) &= \prod_{j=1}^{4} P(u_j = k_j) \\ &= P(u_1 = 1) \times P(u_2 = 1) \times P(u_3 = 1) \times P(u_4 = 1) \\ &= 0.1 \times 0.2 \times 1 \times 1 \\ &= 0.02. \end{aligned}$$

Similarly, the probability that neither of the threats survive is

$$\begin{aligned} P(\mathbf{u}=\mathbf{k}) &= \prod_{j=1}^{4} P(u_j = k_j) \\ &= P(u_1=0) \times P(u_2=0) \times P(u_3=1) \times P(u_4=1) \\ &= 0.9 \times 0.8 \times 1 \times 1 \\ &= 0.72. \end{aligned}$$

The probability that only threat 1 survives is

$$\begin{aligned} P(\mathbf{u}=\mathbf{k}) &= \prod_{j=1}^{4} P(u_j = k_j) \\ &= P(u_1=1) \times P(u_2=0) \times P(u_3=1) \times P(u_4=1) \\ &= 0.1 \times 0.8 \times 1 \times 1 \\ &= 0.08 \end{aligned}$$

and the probability that only threat 2 survives is

$$\begin{aligned} P(\mathbf{u}=\mathbf{k}) &= \prod_{j=1}^{4} P(u_j = k_j) \\ &= P(u_1=0) \times P(u_2=1) \times P(u_3=1) \times P(u_4=1) \\ &= 0.9 \times 0.2 \times 1 \times 1 \\ &= 0.18. \end{aligned}$$

The objective is then to

$$\text{minimise} \sum_{\mathbf{k}\epsilon\{0,1\}^n} P(\mathbf{u}=\mathbf{k}) F_\tau^*(\mathbf{k},\mathbf{w}).$$

This value can be minimised at each time stage by changing the decision variable $(x_{ij}(\tau))$ at that time stage to find the optimal value for the objective function. This will result in a low overall survival probability of threats over all time stages.

2.5.2 Stochastic behaviour in a dynamic weapon-assignment model

In 2000, the first so-called *stochastic demand dynamic* weapon-assignment model was formulated by Murphey (2000). This model accommodated the fact that the position and number

of threats might not be known in advance. In this model, only a subset of the threats and their positions is known during each time stage of the temporal period.

The model must therefore find a balance between assigning weapons or resources to threats or missions that are known during the current time stage and reserving the weapons or resources for use on future threats or missions. It achieves this by incorporating cost coefficients that are monotonically increasing functions of time into the objective function. This penalises the prolonged assignment of weapons (Lötter, 2017).

Suppose $n(\tau)$ threats and their positions are known during time stage τ and $c(\tau)$ is a monotonically increasing cost function which aims to penalise the postponement of the assignment of a weapon. The objective function in the stochastic dynamic weapon-allocation model is to

$$\text{minimise} \sum_{r=1}^{t} c(\tau) \sum_{j=1}^{n(\tau)} V_j \prod_{i=1}^{m} q_{ij}(\tau)^{x_{ij}(\tau)} \tag{2.22}$$

subject to constraints

$$\sum_{\tau=1}^{t} \sum_{j=1}^{n(\tau)} x_{ij}(\tau) = 1, \ i = 1, ..., m, \tag{2.23}$$

$$\begin{aligned} x_{ij}(\tau)\epsilon\{0,1\}, \quad & i = 1, ..., m, \\ & j = 1, ..., n, \\ & \tau = 1, ..., t, \end{aligned} \tag{2.24}$$

so that each weapon is assigned exactly once during the entire temporal period and that the decision variables are binary.

This model could be applied to solve problems where all the threats or missions are not known in advance. It is useful in situations where the battle commander or decision maker must ensure that they allocate resources in such a way that is more or less flexible enough to accommodate future needs in the time phases to come.

2.6 Dynamic mission-ready resource allocation problems

In this section the characteristics of MRRA problems will be discussed and some examples will be given. This is to explain the details of what dynamic MRRA problems are and to show how they differ from dynamic weapon-assignment problems presented in the previous section.

2.6.1 The dynamic mission-ready resource allocation problem

As stated in Table 1.4, this study will focus on solving dynamic MRRA problems, where resource allocation must be done over a longer period of time for multiple missions.

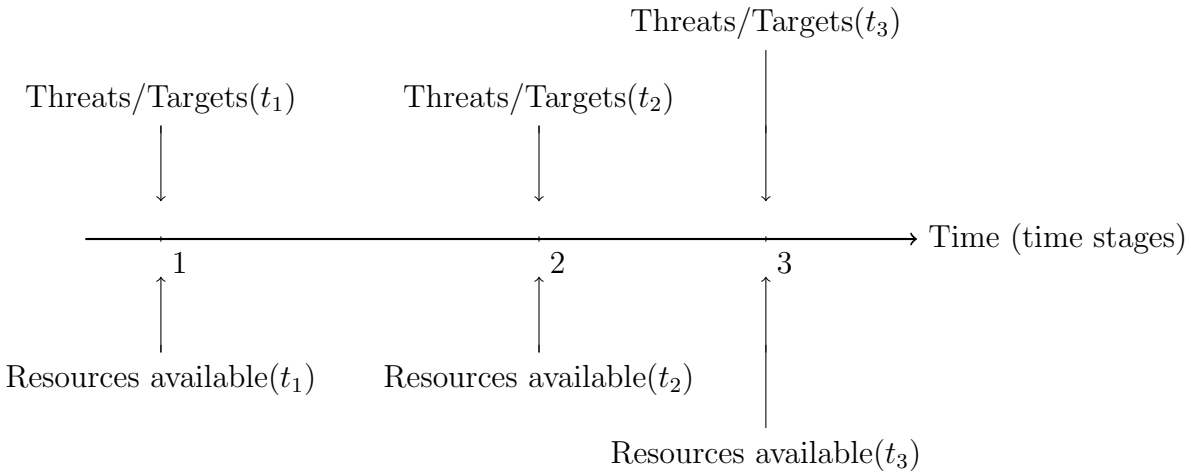


Figure 2.1: Timeline for the dynamic MRRA problem

During each time stage, resources are available and threats or targets must be destroyed by allocating sufficient resources to them. The objective remains to maximise the task suitability of each resource-threat allocation as well as minimise the lift cost of resources used, but over an extended time period. As seen in Figure 2.1, the resources available during time stage 2 may depend on the way that the resources were allocated during time stage 1. This means that decisions made at the current time stage will influence the resources available and, therefore, the options available at subsequent time stages.

Some resources may not be available for the next time stages after they have been involved in a mission. Other resources may never become available again after they have been used, whilst others may be more expensive to use more than once. This means that the lift cost will increase for subsequent time stages if those resources are used during current time stages.

Stochastic elements can be used to determine how long the resources used during time stage 1 will be out of action or unavailable for new missions. Similarly, the time stage lengths follow a distribution.

An example will be used to illustrate the dynamic MRRA problem. During time stage 1, at t_1 in Figure 2.1, the following enemy targets are to be neutralised:

1. Camp Alpha
2. Camp Bravo

During time stage 2, at t_2 in Figure 2.1, the following enemy target is to be neutralised:

1. Camp Charlie

During time stage 3, at t_3 in Figure 2.1, the following enemy targets are to be neutralised:

1. Camp Delta
2. Camp Echo

The following resources are available at the start of the timeline:

1. Team Beta (lift cost = 80)
2. Team Gamma (lift cost = 70)
3. Team Rho (lift cost = 68)
4. Team Epsilon (lift cost = 65)

The task suitability for each time phase is given in Tables 2.16 and 2.17. In this example, one team must be allocated to attack each camp, while a team cannot do two missions back to back. For example, if Team Beta is allocated to attack a camp at time stage 1, it cannot be used during time stage 2. Team Beta will then only become available again during time phase 3.

This is just a basic example, but in more complex problems the time that a resource or team is unavailable might differ, depending on the mission that the resource is assigned to or the characteristics of the resource itself. This could be changed in future examples and stochastic elements could be used to simulate the various probabilities and uncertainty of how long a resource will be unavailable after that team was assigned to a mission.

Table 2.16: Task suitability of time stages 1 and 2

	Camp Alpha	Camp Bravo	Camp Charlie
Team Beta	0.85	0.9	0.8
Team Gamma	0.7	0.65	0.75
Team Rho	0.75	0.65	0.5
Team Epsilon	0.6	0.5	0.7

Table 2.17: Task suitability of time stage 3

	Camp Delta	Camp Echo
Team Beta	0.7	0.9
Team Gamma	0.8	0.7
Team Rho	0.5	0.6
Team Epsilon	0.6	0.4

Assume now that the decision was made to allocate Team Beta to attack Camp Alpha and Team Rho to attack Camp Bravo. The lift cost is $80 + 68 = 148$ and the task suitability is $0.85 + 0.65 = 1.5$ for time stage 1. This would mean that Team Beta and Team Rho will not be available during time phase 2.

During time stage 2, in the absence of teams Beta and Rho, team Epsilon might be assigned to attack Camp Charlie. This would result in a lift cost of 65 and a task suitability of 0.7. This would also mean that Team Epsilon will not be available during time stage 3.

During time stage 3, Team Beta and Team Rho become available once again after being used during time stage 1. During this time stage, Team Gamma can be assigned to attack Camp Delta and Team Beta can be assigned to attack Camp Echo. This would result in a lift cost of $70 + 80 = 150$ and a task suitability of $0.8 + 0.9 = 1.7$ for time stage 3.

This is one possible solution to this example of the stated dynamic MRRA problem. This solution would result in a Total lift cost of $148 + 65 + 150 = 363$ and a Total task suitability of $1.5 + 0.7 + 1.7 = 3.9$ over the three time phases. The objective of this problem is to find a feasible solution, like this one, that would minimise the total lift cost and maximise the total task suitability.

The dynamic element of the problem stems from the fact that decisions made in a certain time stage *influence the resources available in future time stages*. As mentioned, this can be done in several ways. How this influence is calculated into the problem does not affect the dynamic state of the problem.

In this dynamic problem, a solution must be found that would optimise the objective functions over the entire temporal period of the problem. The temporal period must be specified at the beginning of the problem. In effect the task suitability must be maximised and the lift cost must be minimised over a long period of time.

2.6.2 The stochastic dynamic mission-ready resource allocation problem

Since all times are known deterministically, there are no stochastic elements in the case discussed in the previous section. It is therefore a deterministic dynamic MRRA problem. However, stochastic elements can also be introduced into the problem, thus making it a stochastic dynamic MRRA problem, as shown in Table 2.18. In the stochastic dynamic case, probabilities will be associated with the availability of resources during the next time stage if those resources are used during the current time stage.

Table 2.18: Problem classes

	Static	**Dynamic**
Stochastic	1) Static problems, like those presented by Wakefield (2001), but with uncertainty about the specific threats	1) The dynamic problem discussed in Section 2.6.1 2) Weapon-assignment problem by Murphey (2000), discussed in Section 2.5.2
Deterministic	1) Example problem from Section 1.2 on page 4 2) Static MRRA problem presented by Wakefield (2001) 3) Weapon-assignment problem by Manne (1958), discussed in Section 2.4 4) Weapon-assignment problem by DenBroeder *et al.* (1959), discussed in Section 2.4	Weapon-assignment problem by Hosein *et al.* (1988), discussed in Section 2.5.1

For example, the probability of resource 1 being available during time stage 2 after it was used during time stage 1 would be 0.9. This means that there is a chance that the resource might be available for both missions, but, unlike the case in Section 2.6.1, this will not be known before the end of time stage 1.

There might even be a more complex problem that needs stochastic elements to solve. Suppose there is a 10% chance that the resource will become available again after one week, a 70% chance that the resource will become available again after two weeks, a 10% chance that the resource will become available again after three weeks and a 5% chance that it will only become available again after four weeks. There might even be a 5% chance that the resource will never become available again, since there is a small chance that the resource might be heavily damaged or destroyed during the first mission.

If it is assumed that the time between t_1 and t_2 is three weeks, then it means that the resource will have a 90% chance of being available for the second mission after it was used during the first mission, since there is only a 10% chance that it would take longer than three weeks to become available again.

However, the exact time of the next mission might also be uncertain in some examples. This is often the case in real life, since it is often unknown exactly when enemy attacks will

happen or when the weather would be suitable for the mission. Time stage 2, t_2, might have a 30% chance of occurring two weeks after t_1, a 50% chance of occurring three weeks after t_1 and a 20% chance of occurring four weeks after t_1. This means that more combinations are possible and it becomes less intuitive to determine the probability that the resource will be available for the next mission.

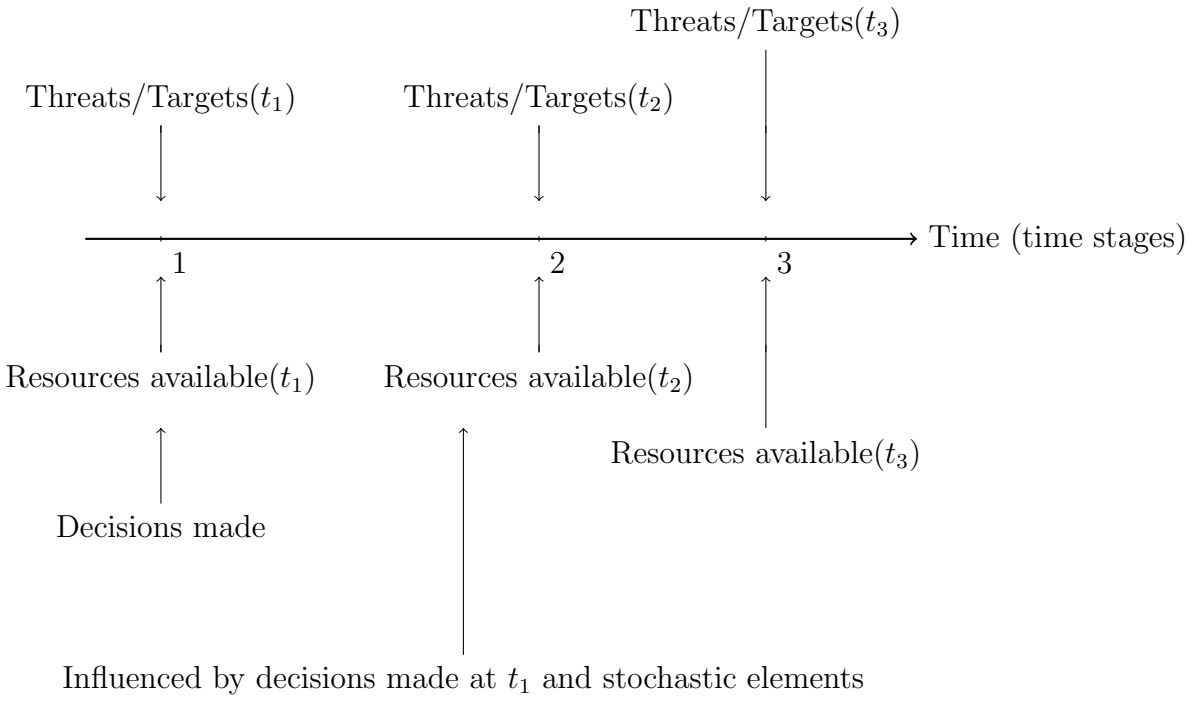


Figure 2.2: Timeline for the stochastic dynamic MRRA problem

2.7 The introduction of mission-ready resource problems in different environments

Now that a clear difference between the various problem classes is established, it is time to address the different roles that these problem classes play. Each problem class can be used to solve a unique type of problem. The goal of this section is to show where various problem classes are used within current business environments and to illustrate the need for a dynamic MRRA solution model. The role of the dynamic MRRA solution model within the bigger resource allocation optimisation field will be discussed.

2.7.1 The need for dynamic resource allocation

Static and dynamic models satisfy different needs. Static models were often good enough to satisfy the basic needs that military commanders required during the Second World War.

However, in modern warfare and in business resource allocation it is often important to consider the long-term effects of current decisions on the availability of future resources.

Unlike disaster management teams that need to solve a major problem at a specific point in time, many businesses need to consider future opportunities when making resource allocation decisions. They need to consider future opportunities that they might decline if they use a resource at the current point in time. Therefore, many business decisions are in need of a proper dynamic resource allocation technique or model to suit their dynamic needs. A dynamic MRRA problem-solving technique will aid them when assigning their resources in such a way to optimise their potential outcomes over a long period of time.

The effect when optimising at several individual time stages versus optimising the overall result with future decisions in mind by presenting a simple example problem is shown in Table 2.19. In this problem, there is only one of each type of resource available and each resource becomes unavailable for the next time stage when it is used. One mission that requires one resource must be completed at each time stage. The cost and suitability of each resource-mission combination is shown in Table 2.19. These costs and suitabilities vary from time stage to time stage, because the cost and suitability associated with a resource to complete different tasks naturally differ. The aim of the problem is that the cost over all four time stages must be minimised, while the total overall task suitability must be maximised.

Table 2.19: Deterministic example problem setup

Resource	Time stage 1		Time stage 2		Time stage 3		Time stage 4	
	Cost	Suitability	Cost	Suitability	Cost	Suitability	Cost	Suitability
A	200	0.6	80	0.8	180	0.6	190	0.6
B	120	0.6	170	0.8	170	0.3	120	0.5
C	100	0.6	80	0.7	170	0.2	190	0.5

It can be seen that if each time stage is optimised individually, a logical decision will be to choose resource C at time stage 1, since it is clearly the best solution at that time stage (cheapest, with same suitability). This means that resource C will become unavailable for time stage 2. The logical choice at time stage 2 will be to use resource A, since it is clearly cheaper than resource B and also produces a suitability of 0.8. This means that resource A will become unavailable for time stage 3. At time stage 3, resource B will then be chosen, since it has a better suitability than resource C for the same price (resource C is now back in contention). At time stage 4, resource B will be unavailable and resource A produces a better suitability than resource C at the same cost.

Table 2.20: Resource choices when each time stage is considered individually

Time stage	1	2	3	4	Total
Resource used	C	A	B	A	
Cost	100	80	170	190	540
Suitability	0.6	0.8	0.3	0.6	2.3

As can be seen in Table 2.20, the total cost is 540 and the total task suitability is 2.3 for all four time stages. This seems to be a good solution, but if sub-optimal solutions were selected at time stages 1 and 2, a better overall result can be found, as shown in Table 2.21.

Table 2.21: Resource choices when future time stages are taken into account

Time stage	1	2	3	4	Total
Resource used	B	C	A	B	
Cost	120	80	180	120	500
Suitability	0.6	0.7	0.6	0.5	2.4

This solution shows that by taking a more expensive option at time stage 1 and 2, the overall cost and suitability can be improved. This small example illustrates that static solution techniques cannot always be used to solve dynamic problems.

2.7.2 Mission-ready resource problem needs within different environments

It is known that the MRRA problem originated within a military environment. The techniques are, therefore, clearly very suitable for allocating military resources to their specific missions in such a way that the balance between the cost of these missions and the probability of successful completion of each mission are optimised. This development was largely due to the need for military efficiency during the Second World War.

However, in more modern times this is not the only need for resource-allocation problem solving. Many resource allocation problems now exist in business, social and political environments.

The static model is used more easily in current environments, since it is simpler and easier to apply. It is used to solve problems that exist at a specific point in time, like in the case of disaster management or logistics (discussed in Section 2.7.4).

Many logistical problems can also be solved by using MRRA problem-solving techniques. Missions need to be executed at a minimum cost. The MRRA problem is perfect for solving such problems. However, many logistical problems extend over a longer period of time with

several time-points where decisions must be made. This means that there is also a clear need for dynamic resource allocation techniques within logistics and other business environments. This will be discussed in the next section.

2.7.3 Resource allocation needs within a business

Businesses have the need to optimise the work done by their workers by allocating them in such a way that they complete an optimal number of tasks, or missions, at a minimum cost. This would allow the business to maximise their profits by maximising the utility of their work force.

Logistics within a business also require resource allocation. The correct mode of transport must be selected to ensure that the minimum cost is achieved, and that the resource suitability is acceptable or sufficient for the task. Most of these business-related resource allocation problems would benefit from a dynamic solution that would allow them to make good long-term decisions.

2.7.4 Resource allocation needs in disaster management

There are also many similar examples of resource allocation needs in disaster management. According to Caunhye *et al.* (2012) resource allocation techniques have become a very powerful tool when planning for disaster management, both before and after the occurrence of the disaster.

However, despite this, the logistics concerning many disasters, like the 2004 Indian Ocean tsunami and the 2010 Haiti earthquake were still not managed properly (Caunhye *et al.*, 2012). Proper planning and preparations were not in place.

During these disasters, there is a clear need to act quickly but correctly, in order to limit the damage or potential damage. It is also important to consider the cost implications of disaster management (Holguín-Veras *et al.*, 2013), since funds are often limited. Therefore, more emphasis should be placed on proper management of these disasters.

Before a disaster, there are often precautions that need to be taken by the disaster management team. This is often done with very limited funds and it is important to make good use of these funds by maximising the impact of the resources used. The correct use of resources could ensure that the proper precautions and preparations are in place to handle the potential disaster. This proper planning could potentially save money, time and even lives at a later stage.

According to Balcik & Beamon (2008) the most important challenges during emergency logistics planning, compared to business logistics planning, are the following:

1. Additional uncertainties

2. Complex communication and coordination
3. Efficient and timely delivery
4. Very limited resources compared to the problem at hand

From this it is clear that a proper stochastic MRRA solution model can at least help with the last two challenges on this list. The few resources that are at the disaster management team's disposal could then be applied more efficiently.

During a disaster or shortly after a disaster, the disaster management team must be able to act quickly with the available resources. They often have very limited resources available and many tasks or missions that require attention. Decisions need to be made on how to allocate these resources to the tasks. In order to do that effectively, they need a way to allocate their resources in an optimal way. A MRRA solution model can help disaster management decision-makers with the difficult decisions regarding the resource allocation dilemma (Altay & Green, 2006).

According to Pradhananga *et al.* (2016), resource allocation techniques are, in fact, used in some disaster management cases. These are used mostly as preparation in pre-disaster management. MRRA techniques are used to ensure that all resources are put in place as a precaution for disaster and to ensure that quick action is taken in the event of a disaster.

However, it can be seen from this section that the use of proper resource allocation is not always satisfactory in large disasters. These techniques should be used far more often during disaster management, especially on a larger scale.

2.7.5 Other environments that can still be further exploited by using MRRA problem-solving techniques

Previous sections have illustrated the impact that scientific decision-making had on strategic military decision-making and ultimately during the World Wars. It also showed that there might still be a need for MRRA problem-solving techniques in certain business sectors. It is clear that this technique might be underutilised and that there are possible other uses that should be explored.

It is clear that most resource allocation problems are solved as static problems. However, many business and social environments would benefit from a dynamic resource allocation problem that could allow the user to plan for a longer period of time and to take future events into account when allocating resources.

Dynamic resource allocation problems would help businesses to do medium- to long-term planning better. For example, project managers would be able to implement these techniques

to manage the resources at their disposal with more ease and to ensure that the required work gets done in time by a team that is well suited.

Another example of where dynamic MRRA problem-solving techniques can be useful in the business environment is the task allocation within a factory or distribution centre. Factory workers and forklift drivers can be allocated tasks in such a way that they work more efficiently. The greater efficiency can be achieved by choosing more suitable workers or drivers to do the tasks. The dynamic quality of the problem can help floor managers to plan the most efficient assignment decisions for an entire day or week, depending on their needs.

There is also a great need for efficient resource allocation within the medical environment. Hospitals would be able to serve the needs of their patients better if they made use of the MRRA techniques to allocate their resources. Nurses, doctors and hospital equipment are resources that need to be allocated to tasks or patients. This allocation must, once again, be done in a very similar way to the original MRRA problem. It must be done in such a way that the suitability of the resources to the tasks are maximised and the cost of the resources used is minimised. Since medical care is often expensive it is also important to consider the cost of using the hospital resources.

Similarly, government work is often far from optimal. This can be addressed partly by using dynamic MRRA problem-solving techniques to optimise the allocation of resources within these fields and ensure that the correct resources are allocated to important projects or missions.

2.8 Multi-objective optimisation

Many resource allocation problems have conflicting objectives. In the case of MRRA problems, the cost must be minimised while the task suitability must be maximised. These two are in direct conflict with each other and the result is that multi-objective optimisation (MOO) techniques must be used. This section will discuss MOO and some techniques that can be used to solve MOO problems.

Over the last few years the fields of optimisation and optimisation techniques have been popular research topics. The field of optimisation refers to the field where the best possible solution to a problem must be found, given certain limitations or constraints (Coello Coello, 2006). In single-objective optimisation problems the aim is to find the best solution in the feasible domain, where the feasible domain contains all feasible solutions that are subject to the constraints of the problem.

MOO problems, on the other hand, are problems where several conflicting objectives must be optimised. These conflicting objectives mean that MOO problems do not have only one 'best' solution, like with single-objective optimisation problems. The result is that differ-

ent solution techniques are required for MOO problems. In fact, according to Coello Coello (2006), the entire "notion of optimality changes when dealing with multi-objective optimisation problems".

Francis Ysidro Edgeworth introduced the notion that with MOO problems there is not only one single solution, but many 'optimal' solutions or a set of optimal solutions (Coello Coello, 2006). This set of optimal solutions comprises of the solutions within the feasible domain that do not have any other feasible solution that performs better than this solution with regards to all the objectives optimised in the problem. This optimal set is called the Pareto optimal set and will be discussed in Section 2.8.3.

Before that, it is important to consider the basic formulation of a generic MOO problem, which is discussed next.

2.8.1 Basic formulation of multi-objective optimisation problems

In order to discuss MOO problems and MOO solution techniques, one needs to consider a generic MOO problem with q objective functions $f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_q(\mathbf{x})$ that could be formulated as

$$\text{maximise } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_q(\mathbf{x})], \tag{2.25}$$

subject to

$$g_i(\mathbf{x}) \geq g_i^{\lim}, \ i = 1, ..., r, \tag{2.26}$$

$$h_j(\mathbf{x}) = h_j^{\lim}, \ j = 1, ..., s, \tag{2.27}$$

$$\mathbf{x} \epsilon \chi, \tag{2.28}$$

where $g_i(\mathbf{x})$ and $g_i^{\lim}$ for $i = 1, ..., r$ are the inequality constraint functions and their corresponding (non-zero) limiting values, respectively (Schlünz, 2016). Similarly, $h_j(\mathbf{x})$ and $h_j^{\lim}$ for $j = 1, ..., s$ are the equality constraint functions and their corresponding (non-zero) limiting values, respectively. The vector $\mathbf{f}(\mathbf{x})$ of the objective function values corresponding to a decision vector $\mathbf{x} \epsilon \chi$ is referred to as the *objective vector*. Let the total scaled constraint violation associated with the inequality constraints be

$$G(\mathbf{x}) = \sum_{i=1}^{r} \max \left\{ 0, \frac{g_i(\mathbf{x}) - g_i^{\lim}}{|g_i^{\lim}|} \right\} \tag{2.29}$$

and let

$$H(\mathbf{x}) = \sum_{j=1}^{s} \left| \frac{h_j(\mathbf{x}) - h_j^{\lim}}{h_j^{\lim}} \right| \tag{2.30}$$

be the total scaled constraint violation associated with the equality constraints (Schlünz, 2016). This is the formulation of MOOs and the techniques used to solve them are discussed in Section 2.8.4.

2.8.2 History of multi-objective optimisation

The Operations Research community has developed techniques to solve MOO problems since the 1950s (Coello Coello, 2006). Of this, multi-objective evolutionary algorithms (MOEAs) are considered to be the most common in modern use. David Shaffer is considered to be the first person to design a MOEA during the 1980s.

Shaffer's approach was called the Vector Evaluated Genetic Algorithm (VEGA) (Coello Coello, 2006). VEGA used a simple genetic algorithm with sub-populations that would be shuffled to create a new population that crossover and mutation algorithms could be used on.

After this, Goldberg & Holland (1988) seem to have been the first to hint at Pareto optimality in an evolutionary algorithm. Although Goldberg did not develop the implementation of his procedures, it is this work that most modern MOEAs are built on (Coello Coello, 2006). These include the Non-dominated Sorting Genetic Algorithm (NSGA) proposed by Srinivas and Deb (Srinivas & Deb, 1994) and the Niched-Pareto Genetic Algorithm proposed by Fonseca & Fleming (1994).

This notion of Pareto optimality will be discussed further in the next section.

2.8.3 Pareto sets and Pareto ranks

As stated before, in multi-objective optimisation problems, there are two or more conflicting objective functions that must be optimised simultaneously. This often means that there is no single solution that would be optimal with respect to every objective function. Trade-off solutions must be found to ensure that good values for every objective function are attained. This means that the so-called *non-dominated* solutions should be found.

A solution is said to be *dominated* if another solution exists that is better than this solution with respect to at least one objective, and no worse than this solution with respect to all other objectives. Consequently, a solution is then *non-dominated* if it is not dominated by any other solutions. The aim of multi-objective optimisation is therefore to find a set of solutions that is non-dominated. This set of non-dominated solutions is called the *Pareto set*. A decision-maker could then choose any of these solutions, based on the importance of the objective

functions in his/her mind, and be rest assured that it is a good solution with respect to all other objective functions as well. Therefore, it is important to understand these sets of solutions, since that, rather than a single optimal solution, is the format of the solutions to multi-objective optimisation problems.

Apart from the Pareto set, there could also exist secondary non-dominated subsets which solution vectors could be classified in. These are the sets of solutions that are not dominated by any solutions other than the ones in earlier fronts. For the maximisation example shown in Figure 2.3, all the solutions in the second set (dark circles in Figure 2.3) are the solutions that are non-dominated after the solutions that belong in the first Pareto set (white circles) are removed. Similarly, the third set (white triangles) consists of all the solutions that are non-dominated after the solutions of the first and second sets are removed.

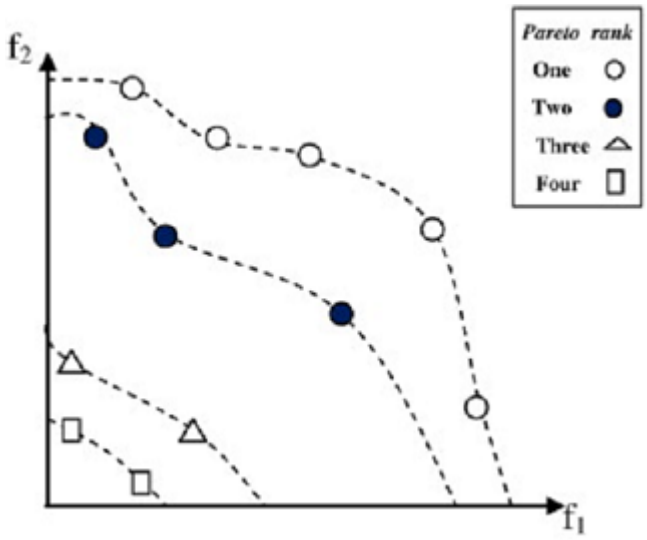


Figure 2.3: Examples of Pareto sets (Pirouzan *et al.*, 2014)

A solution vector could then be assigned a *set rank* or *Pareto rank* corresponding to the rank of the set that the solution belongs to. This rank gives an indication of how good the solution is. Solutions with a Pareto rank of one are part of the optimal set of solutions.

2.8.4 Multi-objective optimisation techniques

MOO problems are well known problems within operational research. Therefore, techniques exist within the field that can find good solutions to multi-objective optimisation problems. The aim is to find solutions that are part of the *Pareto set*. The different techniques will be discussed next. Note that these methods have been chosen because they all have fundamentally different approaches to finding the optimal set of solutions. Each method was chosen from a specific type of MOO method.

2.8.4.1 Genetic algorithms

GAs make use of certain techniques that resemble genetic evolution. In a basic GA a random starting population is chosen. This population is then altered with each iteration by allowing the better solutions from the previous generation to survive and perform *crossovers* that would yield a stronger overall population in the next generation. The crossover entails that the good solutions are combined to create new solutions. This is usually done by using part of one solution and part of another solution. The idea is to do many iterations of this crossover so that the best solution combinations will be found within the population.

GAs also make use of *mutations* to create a wider genetic pool and to reduce the chance that the algorithm gets stuck at local optimum solutions. Mutations are when a solution from the previous generation is taken and one decision variable within that solution is changed randomly for the next population. Mutations are usually set to happen at completely random times and for random solutions. As said before, this is to allow for genetic variety within the population.

The general idea of a genetic algorithm is to take a starting population, or a set of solutions, and improve it with every iteration. This is done by letting two good solutions perform a crossover that results in a new solution which will replace the weakest solution from the previous population. The pseudocode for MOGA is given in Algorithm 1.

Algorithm 1 Pseudocode for genetic algorithm

```
Create starting population
while Population improves do
    while Children not feasible solutions do
        Rank the solutions within the population
        Select 2 new "good" parents
        Perform crossover
        Mutate a chromosome of the new solution with a small probability, like 0.05
        Check feasibility of child
    end while
    if child dominates a solution within the original population then
        Let child replace that solution
    end if
end while
```

2.8.4.2 Simulated annealing

Simulated annealing is an optimisation approach that evaluates one solution at a time. A neighbouring solution is then chosen and accepted if it is either a better solution or with a

certain probability. The idea of simulated annealing is then that this probability is dependent on a certain temperature T, which decreases over time. This results in a decrease of the acceptance probability ensuring that fewer 'bad' solutions are accepted later during the simulation as the temperature decreases.

In this paper a technique called, *archived multi-objective simulated annealing* (AMOSA) will be used. This technique is discussed in depth in Bandyopadhyay *et al.* (2008). It is a technique that is based on the principles of simulated annealing, but also incorporates the dominated and non-dominated principles of multi-objective optimisation solutions.

The AMOSA technique has an acceptance probability, given by:

$$p = \frac{1}{1 + e^{\frac{-(E(q,T)-E(s,T))}{T}}}$$

where $E(q,T)$ and $E(s,T)$ are the corresponding energy values of the current and possible new state respectively. Essentially, this refers to the number of other solutions in the archive that dominates that solution.

The pseudocode for the AMOSA technique is given in Algorithm 2 (Bandyopadhyay *et al.*, 2008).

2.8.4.3 Multi-objective optimisation cross entropy method

Multi-objective Optimisation Cross Entropy Method (MOOCEM) is another type of MOO technique. The steps for coding MOOCEM are given in Algorithm 3. It depends on variance to create different solutions. It uses an archive to store 'good' solutions and uses them to improve the average with every iteration.

2.9 Constraint handling techniques

Multi-objective optimisation is usually computationally expensive as there are often many possible solution combinations and thus a large decision space. As discussed in Chapter 1, there are often really large numbers of possible solutions with few feasible solutions, and just to find a feasible combination could prove to be challenging. Because of this, there is a need for constraint handling techniques to find good feasible solutions. This section will discuss several constraint handling techniques, from Schlünz (2016), that can be used in those situations. The basic formulation of a MOO problem, from Section 2.8.1, is used to explain the constraint handling techniques.

Algorithm 2 Pseudocode for AMOSA

Set *Tmax, HL, SL, iter,* α, temp = Tmax
Initialize the Archive
$current - pt$ = random(Archive) /*Randomly choose a solution from the Archive to be*/
while temp> $Tmin$ **do**
  for $i = 0; i < iter; i++$ **do**
    $new - pt$ = perturb($current - pt$)
    Check the domination status of $new - pt$ and $current - pt$
    if $current - pt$ dominates $new - pt$ /* Case 1 */ **then**
      $\Delta dom_{avg} = \frac{(\sum_{i=1}^{k} \Delta dom_{i,new-pt}) + \Delta dom_{current-pt,new-pt}}{k+1}$
      /* k=total-no-of points in the Archive which dominate $new - pt$, $k \geq 0$. */
      $prob = \frac{1}{1+\exp(\Delta dom_{avg} * temp)}$
      Set $new - pt$ as $current - pt$ with probability = $prob$.
    end if
    if $current - pt$ and $new - pt$ are non-dominating to each other /* Case2 */ **then**
      Check the domination status of $new - pt$ and points in the Archive
      if $new - pt$ is dominated by $k(k \geq 1)$ points in the Archive /* Case 2a */ **then**
        $prob = \frac{1}{1+\exp(\Delta dom_{avg} * temp)}$
        $\Delta dom_{avg} = \frac{(\sum_{i=1}^{k} \Delta dom_{i,new-pt})}{k+1}$
        Set $new - pt$ as $current - pt$ with probability = $prob$
      end if
      if $new - pt$ is non-dominating with regards to all the points in the Archive /* Case 2b */ **then**
        Set $new - pt$ as $current - pt$ and add $new - pt$ to the *Archive*
        if $Archive - size > SL$ **then**
          Cluster Archive to HL number of clusters
        end if
      end if
      if $new - pt$ dominates $k, (k \geq 1)$ points of the Archive /* Case 2c */ **then**
        Set $new - pt$ as $current - pt$ and add $new - pt$ to the *Archive*
        Remove all the k dominated points from the Archive
      end if
    end if

if $new - pt$ dominates $current - pt$ **then** /* Case 3 */
Check the domination status of $new - pt$ and points in the $Archive$
if $new - pt$ is dominated by $k(k \geq 1)$ points in the $Archive$ /* Case 3a */ **then**
Δdom_{min} = minimum of the difference of domination amounts between the $new - pt$ and the k points
$prob = \frac{1}{1+\exp(-\Delta dom_{min})}$
Set point of the archive which corresponds to Δdom_{min} as $current - pt$ with probability = $prob$
else
Set $new - pt$ as $current - pt$
end if
if $new - pt$ is non-dominating with respect to the points in the Archive /* Case 3b */ **then**
Set $new - pt$ as $current - pt$ and add $new - pt$ to the $Archive$
if $current - pt$ is in the $Archive$ **then**
Remove it from the $Archive$
end if
else
if $Archive - size > SL$ **then**
Cluster $Archive$ to HL number of clusters
end if
if $new - pt$ dominates k other points in the $Archive$ **then**
Set $new - pt$ as $current - pt$ and add$new - pt$ to the $Archive$
Remove all the k dominated points from the $Archive$
end if
end if
end if
end for
temp = $\alpha *$ temp
end while
if $Archive - size > SL$ **then**
Cluster $Archive$ to HL number of clusters
end if

Algorithm 3 Pseudocode for MOOCEM

Choose some mean (μ) and some large standard deviation (σ). Set t = 1.
Choose d as a standard deviation that is small enough to stop.
Generate a sample $X_1, ..., X_N$ from the mean and the standard deviation by using a truncated normal distribution.
Use the same sample $X_1, ..., X_N$ and apply it to the problem.
Rank the solutions of $X_1, ..., X_N$ according to its Pareto Rank.
if Rank(Solution(X_N)) = 0 **then**
    Add X_N to set E
    Rank E and eliminate solutions that does not have a Rank of 0
    Reduce σ and set new μ equal to the mean of the decision variables in E.
    if $\sigma < d$ **then**
        Stop.
    end if
else
    Return to step 3.
end if

2.9.1 The constrained-dominated principle

This constraint handling technique is based on Deb *et al.* (2002)'s so called *constrained-domination principle* (CDP). It is a modification of the traditional domination principle. In this case a solution $\boldsymbol{x}$ is said to *constrained-dominate* a solution $\boldsymbol{y}$ if any of the following conditions holds:

1. Solution $\boldsymbol{x}$ is feasible and solution $\boldsymbol{y}$ is infeasible,
2. Solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ are both infeasible, but solution $\boldsymbol{x}$ exhibits a smaller overall constraint violation, or
3. Solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ are both feasible, but solution $\boldsymbol{x}$ dominates solution $\boldsymbol{y}$.

The overall constraint violation could be calculated as $G(\mathbf{x}) + H(\mathbf{x})$. This principle could be used instead of traditional domination to compare two solutions.

2.9.2 The multiplicative penalty function

The next constraint-handling technique is based on a multiplicative penalty function. In this technique, by Schlünz (2016), if a solution violates any constraint, a corresponding penalty value in line with the magnitude of that violation will be incurred. A scalar penalty value

is then created by using an exponential function that takes the constraint violation as the argument. The objective vector is then penalised by multiplication with this scalar value.

Let γ be the severity factor for the problem. The γ value might change, depending on the problem at hand. In this technique there is also a need to distinguish between pure maximisation objective functions and minimisation objective functions that were adapted to be maximisation objective functions, by multiplying them by -1. Let $f^{(+)}$ denote an objective function originally intended for maximisation and $f^{(-)}$ be an objective function that was originally intended for minimisation and that was subsequently multiplied by -1.

If there are u objective functions that were originally intended for maximisation and v objective functions that were originally intended for minimisation, the objective function could then be written as

$$\mathbf{f}(\mathbf{x}) = [f_1^{(+)}, ..., f_u^{(+)}, f_{u+1}^{(-)}, ..., f_v^{(-)}]$$

with $q = u + v$.

For the exponential penalty function ($P_m(\mathbf{x})$) it is also important to differentiate between objective functions originally intended for maximisation and objective functions originally intended for minimisation. Let $P_m^{(+)}$ be the penalty function for the case where the objective function was originally intended for maximisation and $P_m^{(-)}$ be the penalty function for the case where the objective function was originally intended for minimisation.

The exponential penalty function could then be defined as

$$P_m(\mathbf{x}) = \begin{cases} P_m^{(+)} = 2 - \exp(\gamma(G(\mathbf{x}) + H(\mathbf{x}))), \text{ for} f^{(+)} \\ P_m^{(-)} = \exp(\gamma(G(\mathbf{x}) + H(\mathbf{x}))), \text{ for} f^{(-)} \end{cases} \tag{2.31}$$

where γ is a strictly positive severity factor with range $0 \leq \gamma \leq 1$.

The penalised objective vector $\mathbf{f}_P$ could then be determined by multiplying the components of the objective vector by their corresponding penalty functions. Therefore, $\mathbf{f}_P(\mathbf{x}) = P_m(\mathbf{x})\mathbf{f}(\mathbf{x})$.

According to Schlünz (2016), this constraint handling technique effectively transforms a constrained multi-objective optimisation problem into an unconstrained problem. The objective vector can then be given by

$$\mathbf{f}_P(\mathbf{x}) = [P_m^{(+)}(\mathbf{x})[f_1^{(+)}(\mathbf{x}), ..., f_u^{(+)}(\mathbf{x})], P_m^{(-)}(\mathbf{x})[f_{u+1}^{(-)}(\mathbf{x}), ..., f_{u+v}^{(-)}(\mathbf{x})]]. \tag{2.32}$$

Traditional domination can then be done by using the penalised objective vector $\mathbf{f}_P(\mathbf{x})$. A disadvantage of using this technique is that there needs to be a differentiation between objective functions intended for maximisation and objective functions intended for minimisation. Advantages of this technique include the fact that all objective functions are penalised by using a singular scalar value, irrespective of the orders of their magnitude and that only one free parameter has to be tuned during the process (Schlünz, 2016).

However, penalty functions are not the only constraint-handling techniques available. For example, non-feasible solutions could be repaired to find feasible solutions that are similar to the original non-feasible solution.

2.9.3 Constraint transformations

These are constraint-handling techniques where the constraints are transformed into an extra objective function or multiple extra objective functions (Veldhuizen, 2007). The constraints are then replaced by these extra objective functions, essentially meaning that the multi-objective optimisation problem is now unconstrained.

When using these transformations, it is important to consider that they should be used with the aim of finding a single solution, called the *constrained global optimum* and not a set of solutions, like in traditional multi-objective optimisation problems (Veldhuizen, 2007).

Two types of transformations exist, namely *the bi-objective transformation* and *the multi-objective transformation*. Both of these transformations aim to reduce the total constraint violation when finding optimal solutions.

With the bi-objective transformation, one extra objective function is added. This is the sum of the constraint violations and this function must be minimised. The solution to this new multi-objective optimisation problem might still be infeasible, but because of the extra objective function it will be closer to being feasible compared to other solutions with similar original objective function values.

With the multi-objective transformation, every constraint is transformed into an objective function. This can then be used to find a solution with good trade-offs between all the constraints. However, this solution can still be infeasible, but it will be near-feasible.

2.10 Conclusion to Chapter 2

In this chapter, a literature review was presented on MRRA problems, the history of operational research and weapon-assignment problems. This provided background to the problem that this study will be focusing on. Information related to the problem, important terms and techniques that are important for the development of solutions to MRRA problems, specifically a dynamic stochastic MRRA problem, were discussed. Topics that were discussed include the history of resource allocation problems and multi-objective optimisation techniques.

The literature study also looked into the reasons for conducting this study on a dynamic MRRA problem. It concentrated on the uses of a dynamic MRRA problem solution model and the different environments that it can be used in.

2.11 Literature synthesis

This literature study highlighted the field surrounding the dynamic MRRA problem. It also looked into modern real-life problems and how they relate to the dynamic MRRA problem. By doing this, it also showed that there is a need for a solution model to this dynamic MRRA problem and that this solution model will be useful within different business environments.

This chapter, therefore, serves two important roles. Sections 2.2.1, 2.3.2, 2.6 and 2.7 serve to show the need for this specific solution model, while Sections 2.3, 2.4, 2.5, 2.8 and 2.9 show the work that inspired the development of the dynamic MRRA problem solution model.

Chapter 3

Selection of software and a multi-objective optimisation technique

In this chapter, the tools used to solve the stochastic dynamic MRRA problem in this study are discussed. This includes the decision-making regarding techniques and software that are used in the study. Different software and techniques are investigated and compared which will be used to solve the dynamic stochastic MRRA problem.

3.1 Choosing a programming software to solve a dynamic mission-ready resource allocation problem

The solution model to the dynamic MRRA problem must be coded in some way. To do this, some coding platform must be chosen. This section describes the decisions made regarding this programming platform.

Programming is a technique that can be used to create a dynamic MRRA solution model. Possible programs, including Matlab and Python, will be assessed to find a suitable programming platform that would fit the problem. These tools are common in the scientific field, and readily available to end users who wishes to optimise their resource allocations.

3.1.1 Advantages of Matlab

Matlab is a tool that is well known and widely used within the scientific community for its numerical analysis research capabilities. It is important to consider the reasons for possibly using Matlab to solve the dynamic MRRA problem. The following list include reasons to choose Matlab over Python.

1. Matlab is specifically designed for engineers and scientists. This is according to their official website, Mathworks (2018).

2. With Matlab, often very few or even no extra packages are required. When using Python, extra packages are often required to ensure that all functions can be performed.

3. Matlab has the capability of attending to a variety of different aspects of the problem. This means that Matlab can be used to program and simulate many different parts of a business environment, because of its wide range of capabilities.

4. Matlab runs faster than Python.

5. Matlab has a large scientific community. Many scientists and universities make use of it and, consequently the product is very developed and well known. This also means that help is often not difficult to find, if one is in need of it.

6. Matlab is simpler and easier to use for beginners. Python can prove to be more difficult to learn.

3.1.2 Advantages of Python

Python is also a programming tool that can be used to solve the dynamic MRRA problem. Python will also be considered. The following list includes reasons to choose Python over Matlab.

1. Python is free. The fact that Matlab is expensive influences the number of people that can make use of the software. It would be better to use software that can be used by more businesses.

2. Python is a proper programming language, where Matlab is mostly known just as a linear algebra package (Mahotas, 2013).

3. Python is growing faster and is becoming more popular than Matlab in modern times. This also results in Python allowing its users to program in a more modern way.

4. Python is open source. This means that one can see how many of the functions within Python have been implemented. This can help the coder to ensure that the functions were implemented correctly.

5. Python code is often much shorter than Matlab code. This makes the code much more compact.

6. Python is object orientated and has good syntax that helps to make the code clear.

7. Python has a helpful feature that allows the user to use LaTeX code in the code to create equations or plots in Python (Koepke, 2018).

3.1.3 Choosing a suitable simulation platform

It seems that Matlab would prove to be easier to learn, but that Python might be more valuable in the long run, because Python is open source and also an ever growing programming language. Therefore, being able to code in Python would prove to be a more valuable asset

when entering the industry. With this in mind, the choice is to use Python as a simulation platform.

3.2 Comparing different multi-objective optimisation methods

There are many suitable methods of multi-objective optimisation, as discussed in Chapter 2. Each of these methods follow a different algorithm to find the Pareto optimal solution or as close as possible to this for a specific problem. Because of these different methods, they perform differently, especially on larger or more complicated problems where the Pareto optimal set of solutions is not always easy to find.

Since these methods yield different results, they must be compared in some way to find the most suitable method and therefore, the most suitable answer or set of solutions. The two key performance indicators in this case are the quality of the solution and the time that the method takes to find that solution. The time to find a solution can be compared by measuring and comparing the computing times of the different methods. The faster the method can find the set of solutions, the better. However, comparing the quality of different sets of solutions is not as simple.

The Pareto optimal set of solutions, if it is known, might be a very large set of solutions and any method is highly unlikely to find all of those solutions. In fact, it is likely that some methods might not find any of these solutions exactly. However, that does not mean that these solutions are necessarily all bad solutions or bad methods, since the set of solutions can still be close to the Pareto set if the solution set resembles the Pareto set.

A close solution is, therefore, rather a set of solutions that have many solutions that do not differ much from solutions within the Pareto set. A good set of solutions would then have solutions that are close to a great variety of solutions within the Pareto set. From this it is clear that a way is required to quantify how close a set of solutions is to the Pareto set. The hyper area method is used for this.

3.3 The hyper area method

The hyper area method, or Hyper volume method if there are more than two functions that must be optimised, is a technique that can be used to calculate the performance of a multi-objective optimisation technique. If the true Pareto set of the problem is known, this technique calculates the area enclosed by the true Pareto set and some reference point (discussed in Section 3.3.1). It then compares this area to the area between the solution set of the multi-objective optimisation technique and the reference point. The smaller this difference, the

closer the proposed solution is to the true Pareto set. To quantify this difference, the following formula is used.

$$\text{HA}_{ratio} = \frac{\text{Area enclosed by the MOO technique solution}}{\text{Area enclosed by the real Pareto set}}$$

The closer the value of HA_{ratio} is to 1, the better the proposed solution, because if the value is close to 1, it means that the two solutions produce almost similar areas and, therefore, are very similar. A good set of solutions is a set of solutions similar to the Pareto set. For example, consider a minimisation problem where f_1 and f_2 must be minimised. If a proposed set of solutions does not cover all the solutions in the Pareto set, it will be given a larger area, as seen in Figure 3.1, where $(0, 0)$ was chosen as the reference point. The area under the straight lines that connect the solution set (the dots) is clearly going to be more than the area under the curve (used to represent the Pareto set). A set with too few solutions or a set that does not cover all the points in the Pareto set will therefore be penalised appropriately with a larger area.

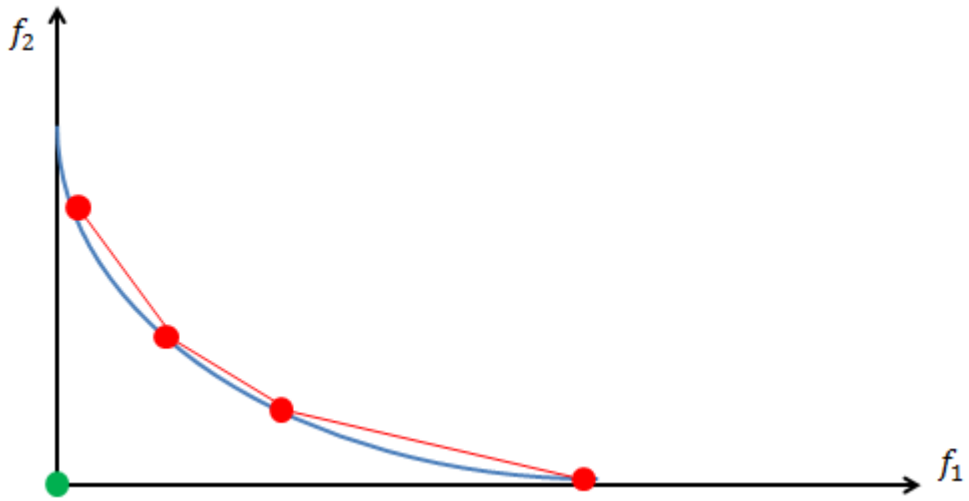


Figure 3.1: Hyper area of a solution set that does not cover all solutions from the Pareto set

Similarly, if there are many solutions and the solutions in the set are well distributed, but they are far from the Pareto optimal solutions, the area under the connected solutions will be much bigger than the area under the Pareto curve. This can be seen in Figure 3.2, where f_1 and f_2 must, once again, both be minimised.

A good hyper area, where the area under the test solution is very similar to the area under the Pareto set (Hyper Area ≈ 1), will only be possible if the solutions from the set are close to the Pareto set and well distributed. An example of this, for this minimisation example, can be seen in Figure 3.3. If the solutions in Figure 3.3 are connected by straight lines, the area under the solutions will only be slightly bigger than the area under the curve of the Pareto set. This will result in a hyper area ratio of close to 1, indicating a good solution.

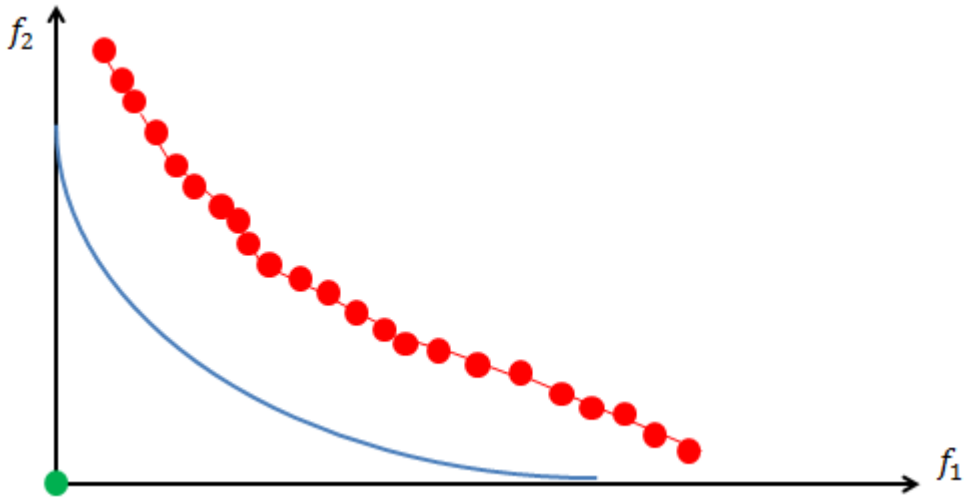


Figure 3.2: Hyper area of a solution set that is far from the Pareto set

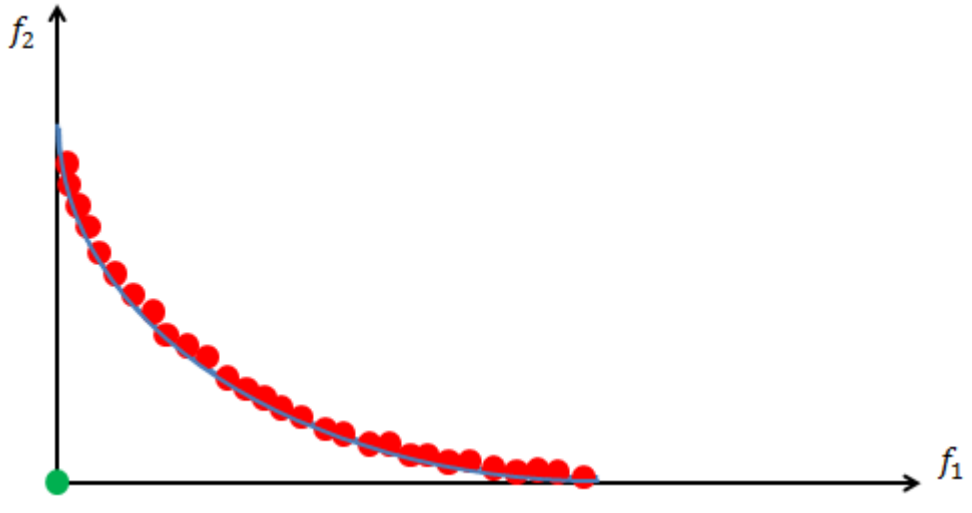


Figure 3.3: Hyper area of a good solution set

3.3.1 Hyper area reference point

The choice of the reference point is important, since it can influence the hyper area comparison. Ideally, according to Ishibuchi *et al.* (2018), each point on the Pareto front should contribute equally to the hyper area calculation. This means that the area that each individual point adds to the existing area (from all the other points), should be more or less similar. This concept is explained by considering Figures 3.4, 3.5 and 3.6.

In Figure 3.4, the reference point is chosen at $(0, 0)$. The areas that each point contributes to the hyper area are also shown in different colours in Figure 3.4. If, for example, point 1 were to be missing from the set, the green area (calculated as 8 square units) would not be added to the total area. Therefore, point 1's contribution to the hyper area is 8 squared units

in this case. Similarly, the red area (4 square units) would not be added if point 2 were to be missing and the yellow area (4 square units) would not be added if point 3 were to be missing. The purple area (8 square units) represents the area that would not be added if point 4 were to be missing from the set.

It can be seen that the result of this is that the points at the edges of the set (Points 1 and 4) carry more weight. These points will have a much larger effect on the hyper area calculation compared to the other points and thus give them a higher importance due to the size of the area that they contribute.

In Figure 3.5, the reference point is chosen at $(3, 3)$ and the result is that, in this case, the points at the end of the set carry less weight than the other points, contributing only two square units each, as shown by the green (for point 1) and purple (for point 4) areas in Figure 3.5. In this case the reference point was chosen too close to the Pareto set and now the points at the edges of the set have a much smaller effect on the hyper area calculation. Points 1 and 4 only contribute two square units to the hyper area, while points 2 and 3 contribute four square units each to the total area, as shown by the red and yellow areas respectively. Clearly this is also not ideal.

In Figure 3.6, the reference point is chosen at $(2, 2)$ so that each point in the Pareto set contributes equally to the hyper area calculation. In essence, if a method missed one point it should be penalised similarly to missing any of the other points. The reference point should therefore be chosen in such a way that each point on the Pareto front contributes more or less equally to the hyper area calculations. In Figure 3.6, the green, red, yellow and purple areas are all the same size (4 square units), meaning that if any of these points were missing the hyper area would be affected in the same way, regardless of which point is missing.

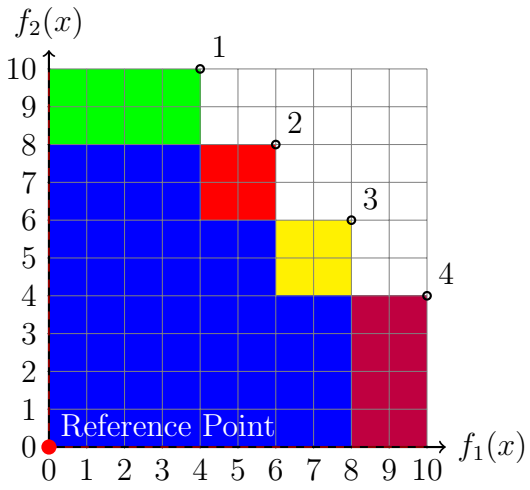


Figure 3.4: Hyper area with reference point at (0,0)

Naturally, this is not always possible, but that would be ideal and it is important to keep

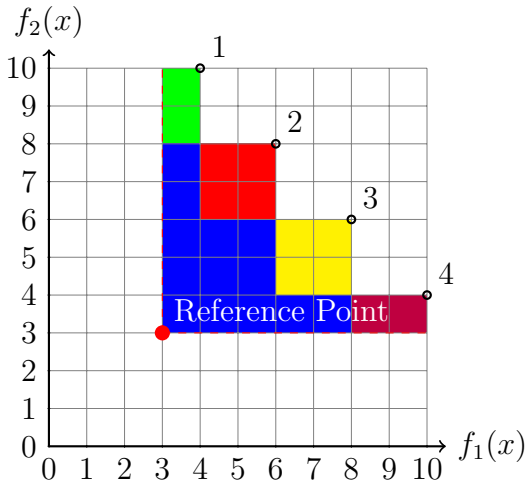


Figure 3.5: Hyper area with reference point at (3,3)

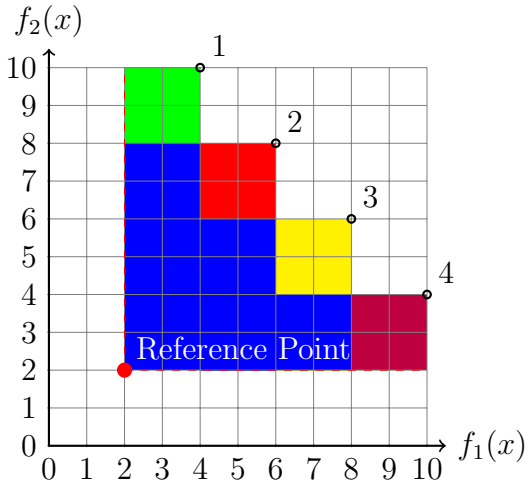


Figure 3.6: Hyper area with reference point at (2,2)

this idea in mind when choosing a reference point. It is also because of this that the reference point is normally chosen to be slightly further away than the Nadir point (Ishibuchi *et al.*, 2018). The coordinates of the Nadir point are defined as the best possible values that each function can have, if the other function values were to be ignored.

The offset that the reference point should be from the Nadir point can be calculated and is dependent on the number of solutions in the set. The formula for calculating the coordinates of the reference point (r_1, r_2) of a normalised Pareto set is

$$r_i = 1 + \frac{1}{\mu - 1}, \tag{3.1}$$

where μ is the number of solutions. From (3.1), it can be seen that the offset of the reference point on the f_i axis should be $r_i = l_i \times \frac{1}{\mu_i - 1}$, where l_i is the length of the set on the

f_1 axis. This means that the reference point should be moved r_i units away from the Nadir point.

3.3.2 Multi-objective optimisation methods tested

The hyper area method was used to compare the performance of MOO methods. The following three methods were chosen for comparison:

1. A simulated annealing based method called Archived Multi-objective Optimisation Simulated Annealing (AMOSA, Algorithm 2)

2. A population-based method called Multi-objective Genetic Algorithm (MOGA, MOGA 1)

3. A population-based method called Multi-objective Genetic Algorithm (MOGA, MOGA 1) as well as a variance-reduction based method called Multi-Objective Optimisation Cross Entropy Method (MOOCEM, Algorithm 3)

These three methods were chosen specifically because of their different approaches to solving MOO problems. AMOSA will sometimes accept slightly dominated solutions in an attempt to find the true optimal Pareto set of solutions. MOGA, on the other hand, will use the best solutions from a previous generation to create an offspring that might produce a good solution and, therefore, a better set of solutions with every iteration. Lastly, MOOCEM will create a new population with each iteration according to a certain mean and standard deviation that were adapted from the non-dominated solutions from the previous iteration.

3.3.3 Test methodology

The three MOO methods, discussed in the previous section, were programmed in Python. These then had to be tested, for two reasons. Firstly, to check that the code is correct and produces the correct expected results. Secondly, the different MOO methods must be compared to find the method that can be used to create a solution to the Dynamic MRRA problem.

In order to compare these three MOO methods, they were all coded in Python and used to solve five known multi-objective problems (MOPs), called MOP1 to MOP4 and MOP6 (Coello Coello *et al.*, 2002). These problems were chosen because the Pareto optimal sets of these problems were known beforehand and they can therefore be used to see how close the solutions from the tested methods are to the actual true Pareto set.

A suitable reference point was chosen for each MOP and the area between the reference point and the proposed solution from the tested method was calculated. That area value

was then compared to the area between the reference point and the actual Pareto set by calculating the hyper area, using those two values. The hyper area and the corresponding computing time that were required by the method to reach that solution set was then stored.

This process was repeated 1 000 times for each method on each MOP to get an average hyper area value and an average computing time to find a solution. The process had to be repeated 1 000 times to get good estimated values that can be compared with other methods, because many of these methods rely on random numbers to guide them through their quest to find an optimal set of solutions.

The paired t-test (Rosner, 1982), discussed in the next section, was then used to compare the different MOO methods. The 1 000 iterations were divided into groups of 50, resulting in 20 groups. For each MOO problem, the respective averages of the 50 values in each group was used to compare the different MOO methods. The paired t-test can only compare two sets of data. Therefore, it had to be done three times for each MOP. First, MOOCEM and AMOSA were compared, followed by AMOSA and MOGA, and lastly MOOCEM and MOGA.

3.3.4 The paired t-test

The paired t-test can be used to determine whether there is a statistically significant difference between two sets of data. In this study, the paired t-test is used to compare two different MOO methods, as stated in the previous section.

Firstly, to use the paired t-test, some assumptions must be made. Normality of the data is assumed, as well as equality of variance in standard deviation. To compare these two MOO methods using the paired t-test, the two sets of 20 data points each are subtracted from each other to generate a new set of 20 data points, called δ. The estimated mean ($\hat{\mu}$) and estimated standard deviation ($\hat{\sigma}$) of this set are calculated and a certain confidence level α is chosen (95% for this study).

$$\text{Confidence Interval} = \hat{\mu} \pm t_{n-1;1-\alpha/2}\frac{\hat{\sigma}}{\sqrt{n}} \tag{3.2}$$

If $n = 20$ and a 95% confidence is chosen, as in this case, the confidence interval can be calculated as follows,

$$\begin{aligned}\text{Confidence Interval} &= \hat{\mu} \pm t_{n-1;1-\alpha/2}\frac{\hat{\sigma}}{\sqrt{n}} \\ &= \hat{\mu} \pm t_{19;0.975}\frac{\hat{\sigma}}{\sqrt{20}}.\end{aligned}$$

The value for $t_{19;1-0.025}$ can be found in a graph for the t-distribution function. If this confidence interval includes 0, then it can be said with 95% confidence, that there is not a

statistically significant difference between the two values. If, however, the interval is entirely positive, it means that it can confidently be said that the first set is larger than the second. Similarly, if the interval is entirely negative, it means that the second set is larger than the first.

In the case of comparing different MOO methods, both the hyper area ratio and the computing time must be as small as possible. Therefore, if the confidence interval is entirely positive, it means that the second set is better than the first, and *vice versa*.

3.3.5 Results of Hyper Area Comparisons

The data from solving the test problems, from Coello Coello *et al.* (2002), were collected by the Python file that was coded for each MOO method. The test functions that were used to test the methods are given in Table 3.1, while the results are summarised in Table 3.2 and 3.3. As stated in the previous section, the paired t-test was used to compare hyper area ratios and computing time. The results of this are shown in Table 3.4.

Table 3.1: Multi-objective problems

MOP	Optimisation Functions	Constraints
1	$f_1(x) = x^2$ and $f_2(x) = (x-2)^2$	$-10^5 \leq x \leq 10^5$
2	$f_1(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2)$ and $f_2(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2)$	$-4 \leq x_i \leq 4;\ i = 1,2,3$
3	$f_1(x,y) = -[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2]$ and $f_2(x,y) = -[(x+3)^2 + (y+1)^2]$	$-\pi \leq x_i \leq \pi$ $A_1 = 0.5\sin 1 - 2\cos 1$ $+\sin 2 - 1.5\cos 2$ $A_2 = 1.5\sin 1 - \cos 1$ $+2\sin 2 - 0.5\cos 2$ $B_1 = 0.5\sin x - 2\cos x$ $+\sin y - 1.5\cos y$ $B_2 = 1.5\sin x - \cos x$ $+2\sin y - 0.5\cos y$
4	$f_1(x) = \sum_{i=1}^{n-1}(-10\mathrm{e}^{(-0.2)*\sqrt{x_i{}^2 + x_{i+1}{}^2}})$ $f_2(x) = \sum_{i=1}^{n}(\lvert x_i\rvert^a + 5\sin(x_i)^b)$	$-5 \leq x_i \leq 5; i = 1,2,3$ $a = 0.8$ $b = 3$

Table 3.2: Results of MOP 1 and MOP 2

MOP	Value	MOOCEM	AMOSA	MOGA
1	Min HAR	1.000011	1.002	1.016
	Q1 HAR	1.000014	1.328	1.03
	Median HAR	1.000017	1.599	1.06
	Q3 HAR	1.000022	1.925	1.081
	Max HAR	1.000093	2.413	2.77
	Average HAR	1.000021	1.626	1.171
	Standard Deviation HAR	0.000018	0.36	0.39
	Min Time	2.333	0.049	7.305
	Q1 Time	2.839	0.092	10.043
	Median Time	3.118	0.119	12.849
	Q3 Time	3.338	0.171	20.106
	Max Time	3.704	0.488	50.567
	Average Time	3.091	0.146	17.183
	Standard Deviation Time	0.323	0.081	11.514
2	Min HAR	1.073	1.45	1.06
	Q1 HAR	1.173	1.517	1.077
	Median HAR	1.194	1.517	1.089
	Q3 HAR	1.219	1.517	1.402
	Max HAR	1.899	1.519	4.507
	Average HAR	1.226	1.516	1.827
	Standard Deviation HAR	0.143	0.007	1.229
	Min Time	0.106	0.785	2.049
	Q1 Time	0.213	1.571	3143.145
	Median Time	0.215	2.738	3226.532
	Q3 Time	0.219	4.514	3764.639
	Max Time	0.366	18.928	4142.926
	Average Time	0.216	3.797	2855.858
	Standard Deviation Time	0.046	3.447	1344.531

Table 3.3: Results of MOP 3 and MOP 4

3	Min HAR	21.668	1.812	22.375
	Q1 HAR	24.912	4.976	30.632
	Median HAR	25.37	9.139	37.952
	Q3 HAR	25.805	15.709	38.573
	Max HAR	28.509	32.192	38.774
	Average HAR	25.355	10.695	5.821
	Standard Deviation HAR	0.763	6.894	33.413
	Min Time	0.122	0.093	0.768
	Q1 Time	0.189	0.14	321.562
	Median Time	0.193	0.162	13677.557
	Q3 Time	0.197	0.195	31953.265
	Max Time	0.291	0.688	117917.819
	Average Time	0.198	0.18	21802.041
	Standard Deviation Time	0.025	0.066	34321.289
4	Min HAR	1.0002	1.096	1.721
	Q1 HAR	1.003	1.871	2.119
	Median HAR	1.014	2.308	2.193
	Q3 HAR	2.383	5.463	2.361
	Max HAR	4.198	9.3	2.42
	Average HAR	1.553	3.362	2.182
	Standard Deviation HAR	0.766	1.869	0.191
	Min Time	0.142	0.204	673.631
	Q1 Time	0.328	0.275	2967.896
	Median Time	0.342	0.331	22009.648
	Q3 Time	0.356	0.56	53152.494
	Max Time	1.033	3.114	197315.868
	Average Time	0.368	0.452	35107.856
	Standard Deviation Time	0.127	0.28	56240.508

Table 3.4: Confidence intervals (95%) from the paired t-test for multi-objective problems

MOP	HA or Time	MOOCEM - AMOSA	AMOSA - MOGA	MOOCEM - MOGA
1	HA	[-0.781, -0.589] (MOOCEM better)	[0.363, 0.709] (MOGA better)	[-0.314, 0.016] (similar)
1	Time	[1.025, 1.087] (AMOSA faster)	[-12.344, -4.228] (AMOSA faster)	[-11.285, -3.175] (MOOCEM faster)
2	HA	[-0.344, -0.27] (AMOSA better)	[0.169, 0.423] (AMOSA better)	[-0.126, 0.104] (similar)
2	Time	[-2.64, -1.315] (MOOCEM faster)	[-2040.39, -656.192] (AMOSA faster)	[-2042.27, -658.266] (MOOCEM faster)
3	HA	[14.598, 17.812] (AMOSA better)	[-27.126, -21.276] (AMOSA better)	[-10.353, -5.638] MOOCEM better
3	Time	[-0.00083, 0.0425] (similar)	[-32931.5, -12347.8] (AMOSA shorter)	[-32931.5, -12347.8] (MOOCEM shorter)
4	HA	[-2.523, -1.25] (MOOCEM better)	[0.927, 2.049] (similar)	[-0.663, -0.134] (MOOCEM better)
4	Time	[-0.176, 0.785] (similar)	[-66101.7, -31001] (AMOSA better)	[-66101.4, -31000.7] (MOOCEM better)

The first test function is called MOP 1 and a description is given in Table 3.1. It is a relatively simple problem to test. $f_1(x)$ and $f_2(x)$ must be minimised.

The values for the average hyper area ratio for MOP1 for each method are given in Figure 3.7 and the values for the average computing time for MOP1 of each method are given in Figure 3.8. From the data, it can be seen that MOOCEM has the smallest hyper area ratio and therefore produces the best set of solutions, but AMOSA seems to be much faster. Clearly MOGA takes far too long to be a viable option.

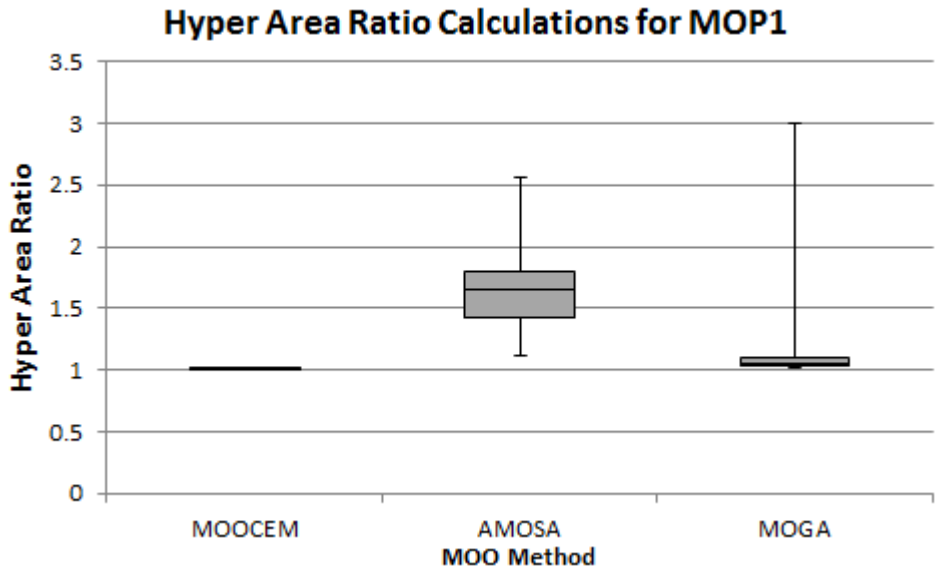


Figure 3.7: Hyper areas for MOP 1

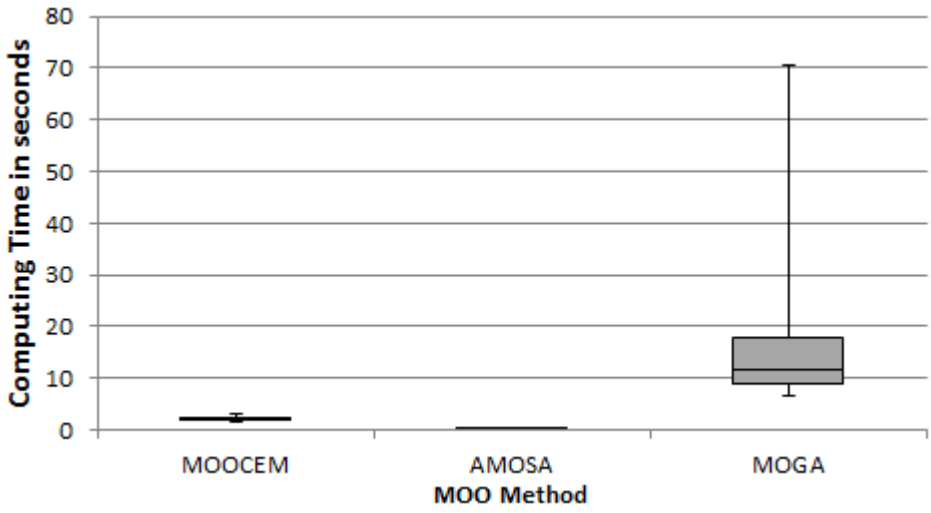


Figure 3.8: Computing times for MOP 1

The second MOP is chosen to compare how well methods cope with more decision variables. MOP 2 has three variables that can change while two functions must be optimised. The definition of MOP 2 is given in Table 3.1.

The values for the average hyper area ratio for MOP 2 for each method are given in Figure 3.9 and the values for the average computing time for MOP 2 for each method are given in Figure 3.10. From the data it is clear that MOOCEM outperformed AMOSA on MOP 2. MOOCEM was much faster on average and got to a better average solution in the end.

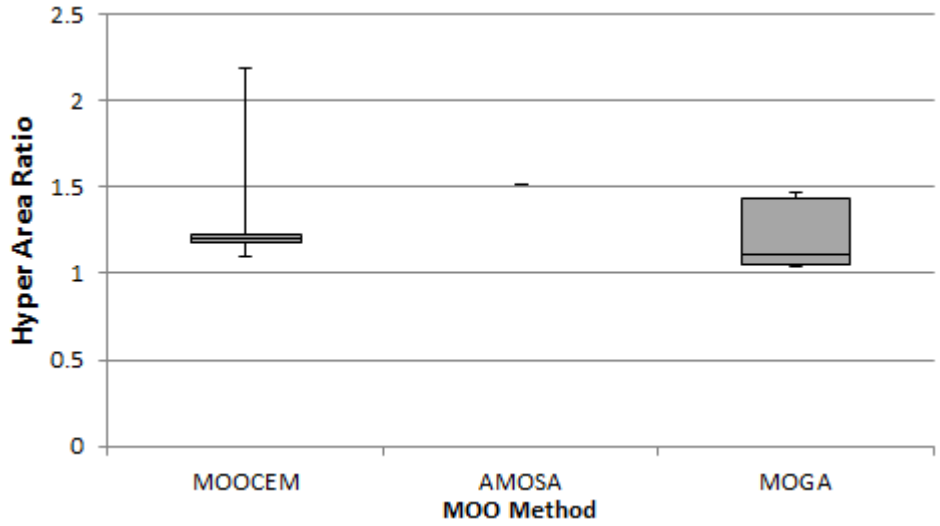


Figure 3.9: Hyper areas for MOP 2

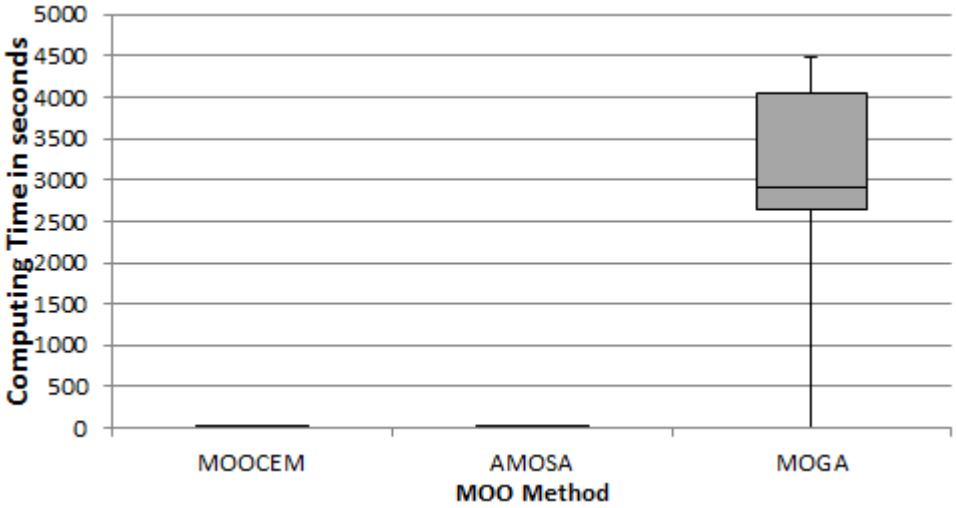


Figure 3.10: Computing times for MOP 2

From Figure 3.10 it is clear that MOGA takes much longer than MOOCEM or AMOSA. Figures 3.11 and 3.12 show the hyper area ratios and the computing times for MOP 2, which are better scaled without the dominant MOGA data.

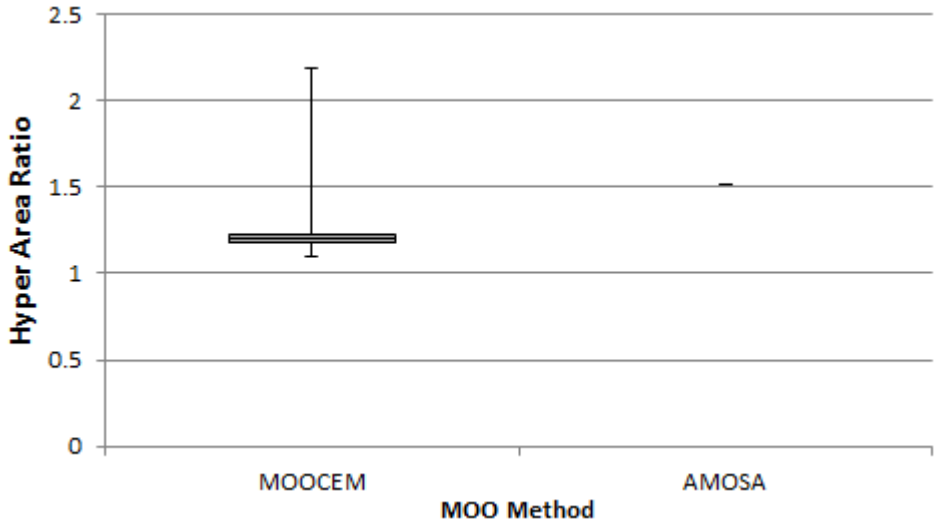


Figure 3.11: Hyper areas for MOP 2 without MOGA

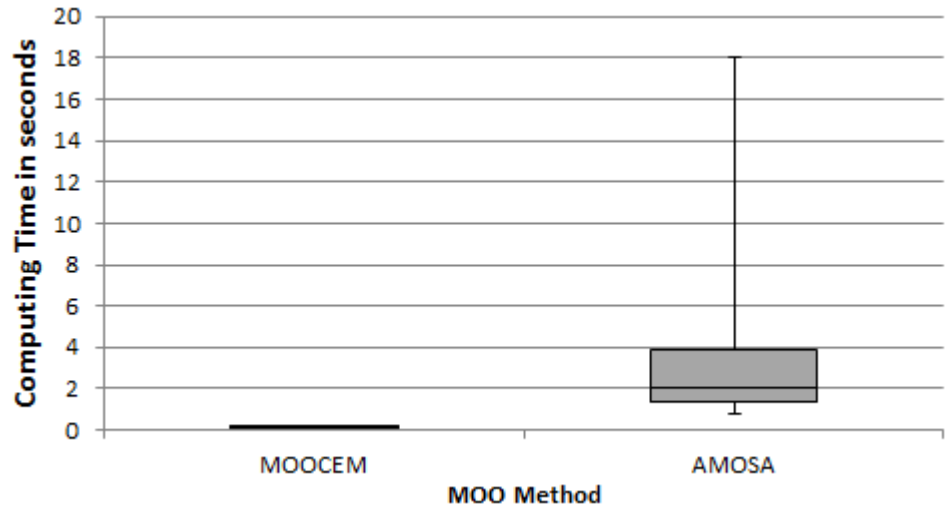


Figure 3.12: Computing times for MOP 2 without MOGA

MOP3 is a maximisation problem, with the description given in Table 3.1, that consist of two disconnected Pareto curves. This MOP is chosen to test a method's ability to find disconnected Pareto sets.

The values for the average hyper area ratio for MOP 3 for each method are given in Figure 3.13 and the values for the average computing time for MOP 3 of each method are given in Figure 3.14. Here, AMOSA seems to outperform MOOCEM in all categories. AMOSA is faster than MOOCEM and produces a better result, while it is noteworthy that neither produces a good set of solutions when being compared to the true Pareto optimal set.

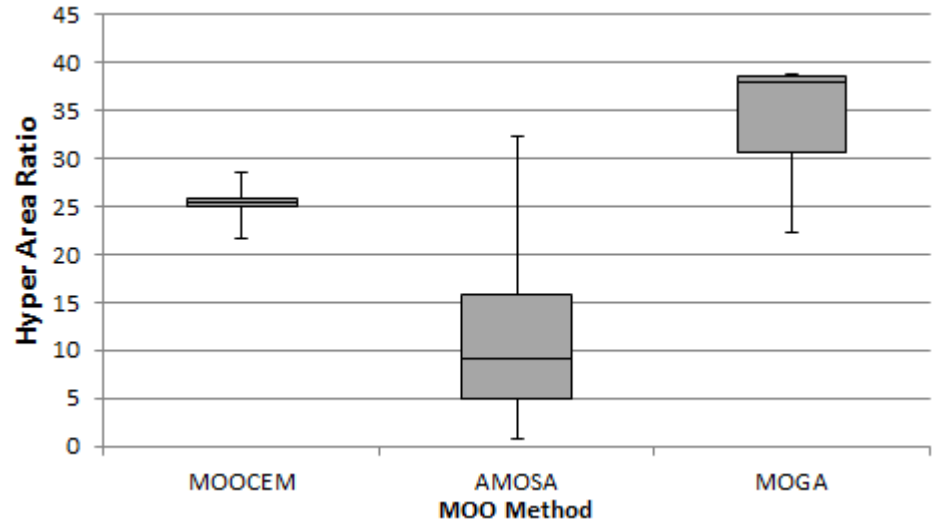


Figure 3.13: Hyper areas for MOP 3

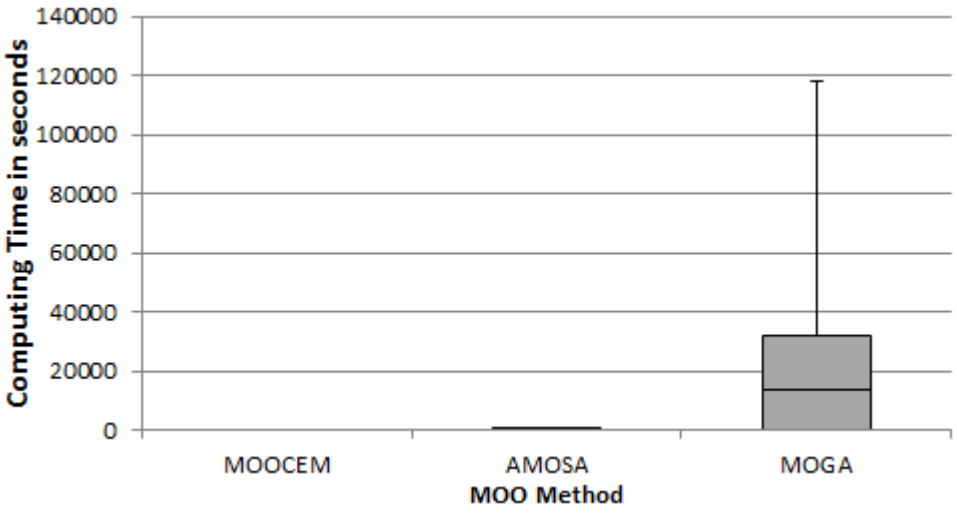


Figure 3.14: Computing times for MOP 3

Similarly to MOP 2, it is clear from Figure 3.14 that MOGA takes much longer than MOOCEM or AMOSA. Figures 3.15 and 3.16 depict the hyper area ratios and the computing times for MOP 3 without MOGA.

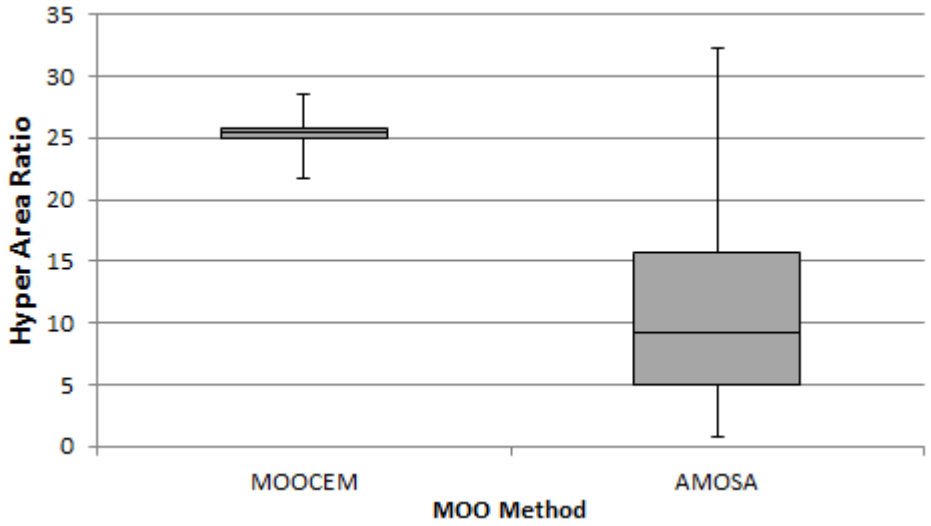


Figure 3.15: Hyper areas for MOP 3 without MOGA

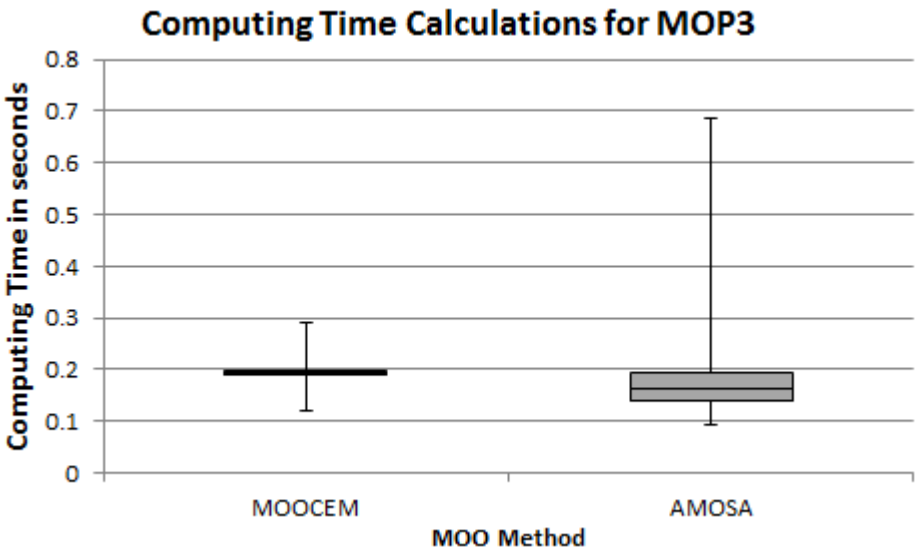


Figure 3.16: Computing times for MOP 3 without MOGA

Kursawe's MOP (MOP 4) is used, because it tests how well methods can perform when the optimisation functions and set of optimal solutions are disconnected (Coello Coello *et al.*, 2002). The definition of MOP 4 is given in Table 3.1.

The values for the average hyper area ratio for MOP 4 for each method are given in Figure 3.17 and the values for the average computing time for MOP 4 for each method are given in Figure 3.18. Here, AMOSA and MOOCEM seem to perform equally well in terms of time. MOOCEM does, however, produce a better hyper area ratio when compared to the actual Pareto set of solutions.

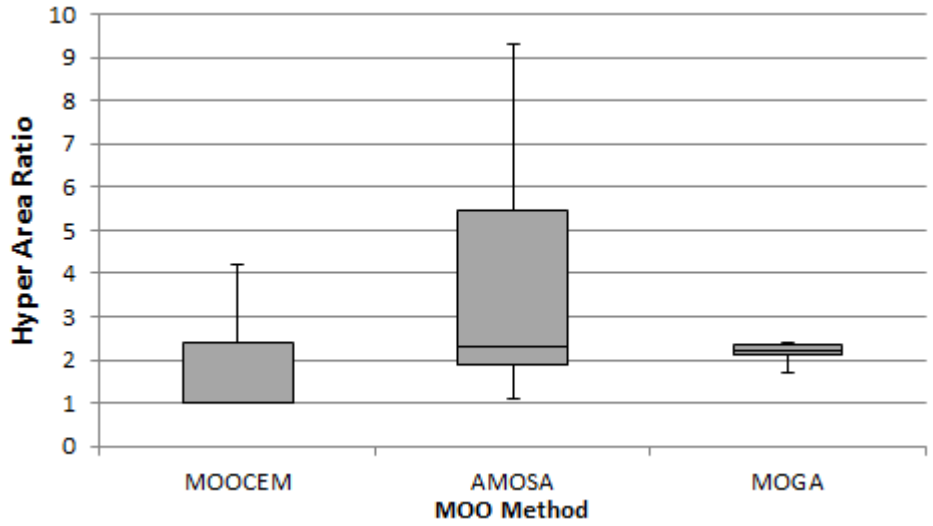


Figure 3.17: Hyper areas for MOP4

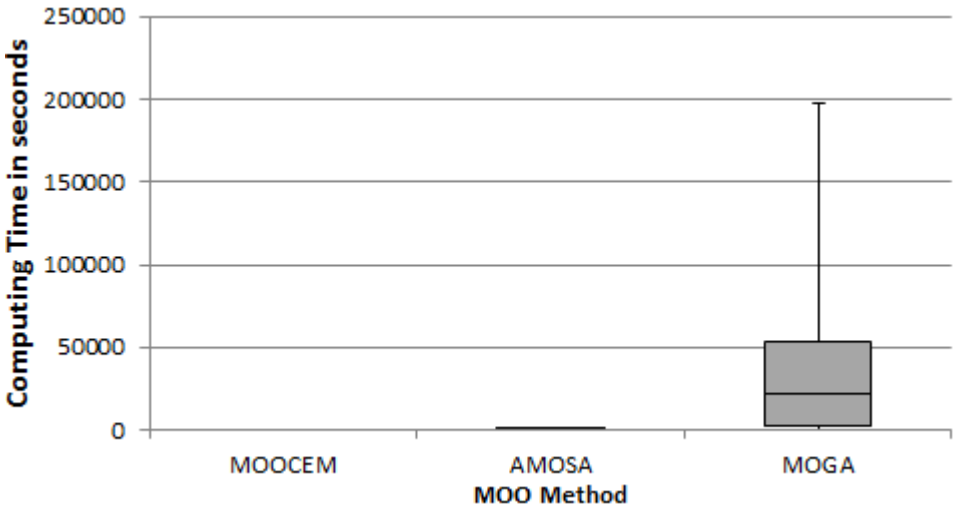


Figure 3.18: Computing times for MOP 4

Once again, it is clear from Figure 3.18 that MOGA takes much longer than MOOCEM or AMOSA. Figures 3.19 and 3.20 show the hyper area ratios and the computing times for MOP 4 without MOGA.

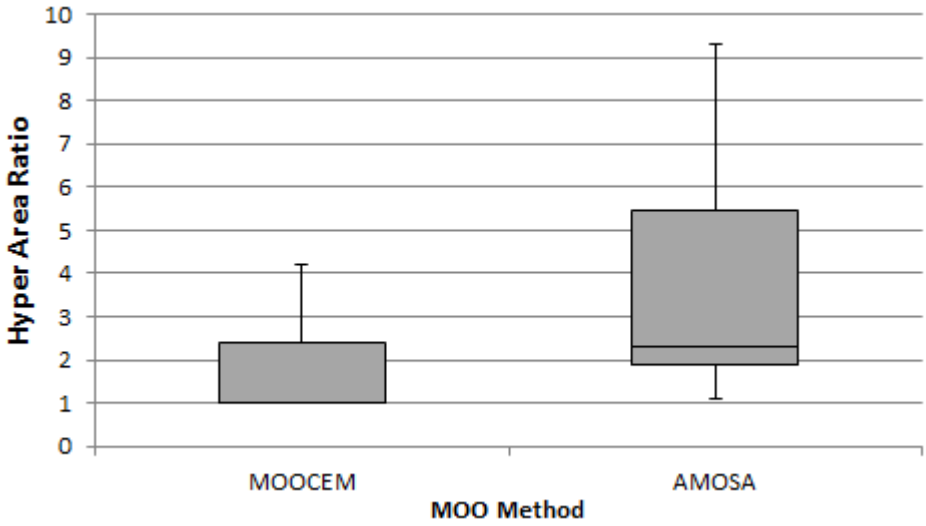


Figure 3.19: Hyper areas for MOP 4 without MOGA

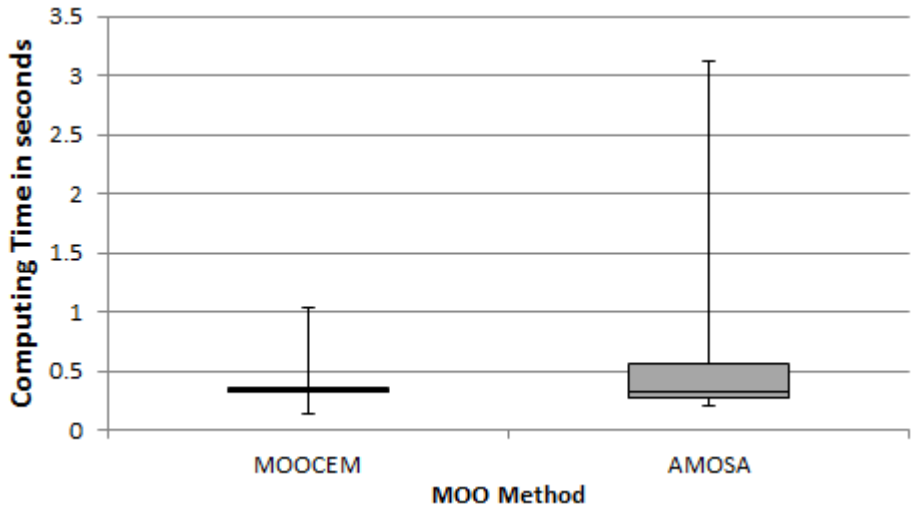


Figure 3.20: Computing times for MOP 4 without MOGA

3.4 Conclusion to Chapter 3

This chapter laid the groundwork to create a solution model for the Dynamic MRRA problem. Python was compared to Matlab and Python was chosen to be used in this study. Three different MOO methods (MOOCEM, AMOSA and MOGA) were coded in Python. They were then verified and compared using the hyper area method and the paired t-test. The results are shown in Table 3.4 and although MOOCEM and AMOSA outperformed each other on different occasions, it was clear that MOGA was much slower than the other two methods and often does not produce a better hyper area ratio. The result is that only AMOSA

and MOOCEM will be used for further development of the solution model of the Dynamic MRRA problem.

Chapter 4

Solution development

In this chapter the design of the Dynamic MRRA problem will be discussed. First, the basic steps of this design will be explained, showing the logic behind the solution model. Two toy problems will then be discussed and the solution model will be explained with regards to these toy problems. The first toy problem will be a Deterministic Dynamic MRRA (DDMRRA) problem. This toy problem will be used to illustrate the model's approach to the dynamic elements of the DDMRRA problem. After that, the extra complications of the Stochastic Dynamic MRRA problem will be discussed and the approach to solving Stochastic Dynamic MRRA (SDMRRA) problems will be revealed in the second toy problem.

4.1 Model description

A solution model is developed to solve MRRA problems. Figure 4.1 shows the basic building blocks of the solution model. In the first step, a decision variable is randomly populated with the assumption that all resources will be available during all the time stages, even though this is known not to be the case. This initial decision variable is then repaired to find a feasible solution, by using simulated annealing to optimise the penalty function (described in Section 4.2.1). Once a feasible solution is found, the MOOCEM algorithm is used to find a non-dominated set with the aim of minimising lift cost and maximising task suitability. The feasible solution becomes the starting average in MOOCEM, while the starting standard deviation used in MOOCEM is set to 10 000. Non-feasible solutions are repaired to feasible solutions by using simulated annealing and the penalty function to find the nearest feasible neighbour of any solution, resulting in an entirely feasible set of solutions. These solutions are then stored in an archive.

This process is iterated five times to ensure that different parts of the Pareto set is investigated. Once these iterations are complete, the dominated solutions are removed from the archive and the non-dominated solutions are presented to the user.

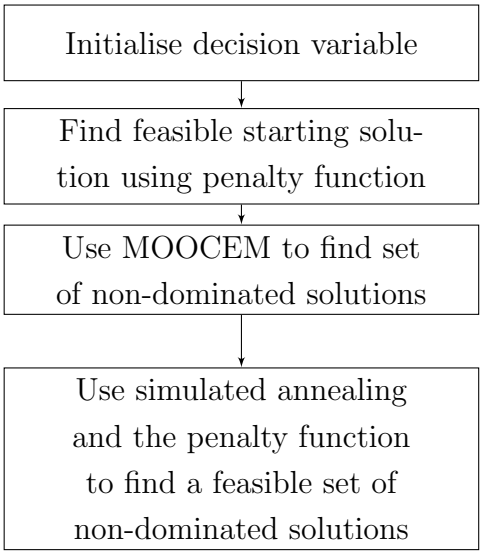


Figure 4.1: Solution model design

4.2 The deterministic dynamic MRRA problem

From Chapter 2 it is evident that a solution model to Dynamic MRRA problems will be helpful in decision-making environments. To simplify the process of finding a solution to a stochastic dynamic MRRA problem, a stepwise approach was taken. The static deterministic MRRA problem solution is already known, so the next step is to create a solution to a Deterministic Dynamic MRRA problem.

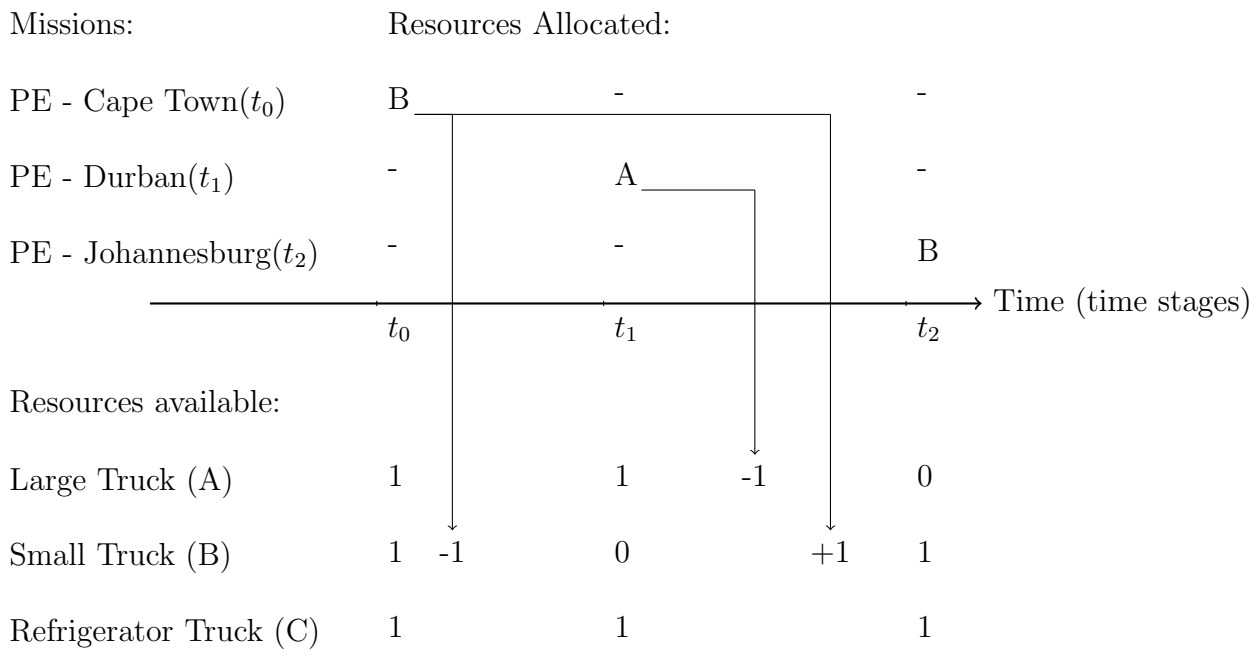
Figure 4.2 shows a toy deterministic dynamic MRRA problem with a feasible solution. In this example, a logistics company based in Port Elizabeth (PE) must decide which type of truck to allocate to various different routes (or missions). Three different types of trucks are available, while there are three routes, each of which requires one truck to complete. Each truck-route combination has its own specific task suitability and lift cost, given in Table 4.1 and Table 4.2 respectively.

The missions in this case are represented by the routes that must be travelled. These routes are all from PE to some destination (Cape Town, Johannesburg or Durban) and back to PE, where the company is based. In this problem, if a resource (truck) is used during any time stage, that specific resource becomes unavailable for the next time stage. This may be because the truck is still on its way back to PE and not ready in time for a next mission or require reparations before it can be used again.

For example, consider the proposed solution in Figure 4.2, where a small truck is allocated to the Cape Town route at t_0, a large truck is allocated to the Durban route at t_1 and a small truck is allocated to the Johannesburg route at t_2. At t_0, one small truck (Resource B) is used to satisfy the demand for a PE to Cape Town return route mission. Since this means

that this specific resource will then be unavailable for the next time stage, the result is that the number of available resources for Resource B at time stage t_1 must be decremented (as shown in Figure 4.2). However, that same resource will be back in time for t_2, so the number of available resources for small trucks at time stage t_2 must be incremented. This enables the company to re-use this small truck on the Johannesburg route at t_2.

With the same logic, allocating a large truck (Resource A) during t_1 results in one fewer resource of type A available at t_2. Since the problem ends at t_2, the problem will be over by the time this resource becomes available again, so in this case it does not have to be added again after t_2.



[H]

Figure 4.2: Feasible solution to a toy deterministic dynamic MRRA problem

Table 4.1: Toy problem task suitability

Truck	PE – Cape Town – PE	PE – Durban – PE	PE – Johannesburg – PE
Large (A)	0.7	0.8	0.8
Small (B)	0.7	0.6	0.5
Refrigerator (C)	0.8	0.9	0.9

Table 4.2: Toy problem lift cost

Truck	PE – Cape Town – PE	PE – Durban – PE	PE – Johannesburg – PE
Large (A)	300	330	450
Small (B)	250	280	300
Refrigerator (C)	320	380	400

A deterministic dynamic MRRA problem, naturally, has a much larger decision variable set than a deterministic static MRRA problem of similar size. This is simply because decisions must be made at future time stages as well, meaning that a dynamic MRRA problem consists of multiple static MRRA problem solution sets combined. This means that an extra dimension is given to the decision variable. This extra dimension represents the time stage in which that specific allocation decision must be made.

In addition to this extra dimension, there are some other complications with having to solve multiple static MRRA problems back to back: A penalty function must be used to determine whether solutions are feasible or not. The design of this penalty function will be discussed next.

4.2.1 Penalty function

The aim of using this penalty function is to identify solutions that are not feasible and to quantify how close any solution is to being feasible. This is important, since near-feasible solutions can ultimately be used to find feasible solutions that can be used in the final solution set.

The penalty function works like any other objective function. The function increases by one each time a resource is used when there are none available at that time stage or if the correct number of resources was not allocated to a specific mission. Basically, each time allocations are made in such a way that the solution does not satisfy all the constraints, the penalty function will increase. Naturally, this function must be minimised and if the penalty function of a solution returns a value of 0, it means that the proposed solution is feasible.

The penalty function must calculate the number of available resources for each time stage. This will differ for every solution, since it depends on the solution's earlier decisions, because the number of available resources for the next time stage depends on the number of resources used during the previous time stage. For each new solution, the penalty function will calculate the number of available resources of each type based on the decision variable of that specific solution. The pseudocode for the penalty function is given in Algorithm 4.

In the example in Figure 4.2, the penalty function of the example solution will increase if the number of allocated resources is more than the number of available resources. If, for

Algorithm 4 Pseudocode for penalty function

A solution (sol) is given to the function
Set Penalty = 0 , $j = 0$, $l = 0$ and $k = 0$
Let TimePoints = the number of time points in the problem
Let Missions = the number of missions in the problem
Let MRRs = the number of types of Mission-Ready Resources in the problem
while $j <$ TimePoints **do**
  while $l <$ Missions **do**
    while $k <$ MRRs **do**
      Check that the total number of resources allocated to each mission is sufficient
      if not sufficient **then** Increase *Penalty* by the difference
      end if
      $k \leftarrow k + 1$
    end while
    $k \leftarrow 0$
    while $k <$ MRRs **do**
      Check that there are enough resources of each type available for the total number of resources of that type allocated during time stage j.
      if not enough available **then** Increase *Penalty* by the difference
      end if
      $k \leftarrow k + 1$
    end while
    $l \leftarrow l + 1$
  end while
  $j \leftarrow j + 1$
end while

example, at time stage t_2, the decision was made to allocate a large truck (resource A) to the Johannesburg route, this would become an infeasible solution and the penalty function for this solution will become equal to 1 to indicate that it is not feasible.

4.2.2 Finding solutions for the deterministic dynamic MRRA problem

As stated in Chapter 1, even static MRRA problems often have many non-feasible solutions and only a few feasible solutions. Finding feasible solutions to the dynamic MRRA problem is now even more difficult than with the static MRRA problem, because of the added size and the adapted penalty function. Therefore, in order to commence with the process, the penalty function is first optimised entirely on its own by using single objective optimisation (Simulated Annealing in this case). This means that the first step is only to find a feasible solution, without taking the cost and task suitability into account. This helps to find a feasible solution faster. This solution can then be used to start the MOO process and find other feasible solutions that can potentially be better in terms of the other objective functions like lift cost and task suitability. A good MOO technique is required for this process. For this, MOOCEM is used, since it performed well when tested on the test functions MOP1, MOP2, MOP3, MOP4 and MOP6 in the previous chapter.

The toy problem is solved by maximising the task suitability ($f_1(x)$) and minimising the lift cost ($f_2(x)$). Since this is a small problem, all the feasible solutions can be considered for completeness. The results are shown in Table 4.3, while Figure 4.3 shows all the feasible solutions plotted. The Pareto front is represented by the blue lines. The optimal set, therefore, consists of Solutions 3, 9, 10 and 11.

Table 4.3: Results for toy problem

Solution Nr	Allocation combination			Lift Cost	Task Suitability
	PE – Cape Town – PE	PE – Durban – PE	PE – Johan-nesburg – PE		
1	A	B	C	980	2.2
2	A	C	B	980	2.1
3	B	A	C	980	2.4
4	B	C	A	1080	2.4
5	C	A	B	950	2.1
6	C	B	A	1050	2.2
7	A	B	A	1030	2.1
8	A	C	A	1130	2.4
9	B	A	B	880	2.0
10	B	C	B	930	2.1
11	C	A	C	1050	2.5
12	C	B	C	1000	2.3

4.3 The stochastic dynamic MRRA problem

After the solution to the DDMRRA problem was investigated, the stochastic dynamic MRRA problem was considered. This problem deals with scenarios where the availability of resources at specific time stages is uncertain. This can be because resources take longer than expected to return from previous missions or resources get damaged during previous missions or because missions must be done earlier than planned. It can also be a combination of these occurrences. However, for planning purposes, the model is only interested in whether or not a resource is available when a mission takes place. It is not concerned with the reason, only the fact that a planned allocation might not be possible because of some unforeseen event.

If there is a chance that certain resources might not be available by the time they are estimated to be available again, this is something that the decision-maker will be interested in. To show how this can affect the outcome of the eventual total lift cost and suitability, the worst-case scenario can be considered. For this, it is assumed that each resource only becomes available at the time stage after it is expected to become available. Each solution is then considered individually and if changes need to be made to accommodate for the resources that return later than expected, this is done to ensure that the solution will be feasible, even

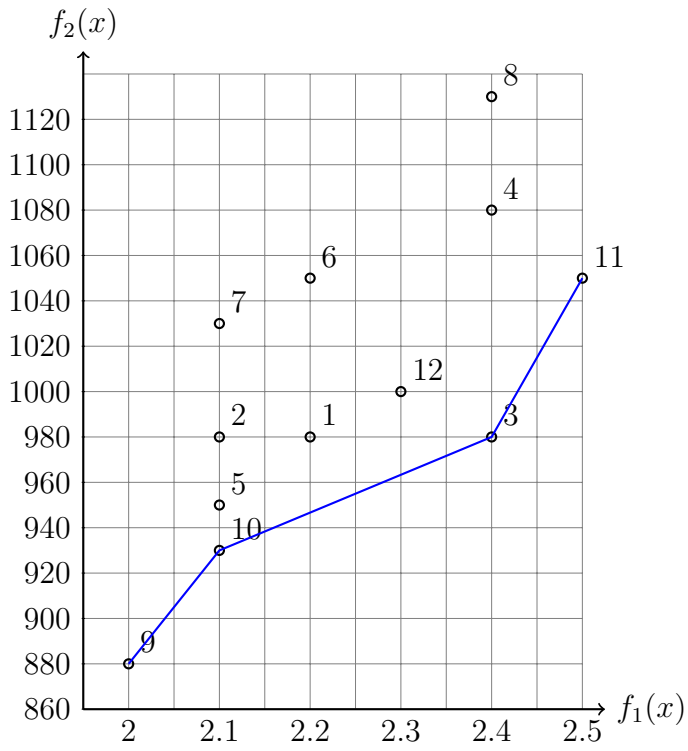


Figure 4.3: Feasible solutions to toy problem

if resources arrive one time stage later. This 'worst-case scenario' can then be used to see the risk of deciding on a solution that relies on resources returning on time.

As an example of a stochastic dynamic MRRA problem, the same toy problem from Section 4.2 was taken, but in this case there is uncertainty as to whether the resources will return when expected. In this problem, resources might not return one time stage later, as expected, but rather two time stages later.

Figure 4.4 shows how the solution from Figure 4.2 in Section 4.2 is affected when the small truck (Resource B) does not return in time for t_2. The result is that 0 of resource B will be available at t_2, so a small truck cannot be used to complete the PE – Johannesburg – PE route at t_2. The only available resource at t_2 is the refrigerator truck (Resource C) and this will have to be used. This change will mean that the total task suitability will increase from 2 to 2.4, but, since this is a more expensive option (cost will increase from 880 to 980), the company will have to pay more. Therefore, it is important for them to consider the risk of choosing this allocation and know that there is a chance that the overall cost may increase.

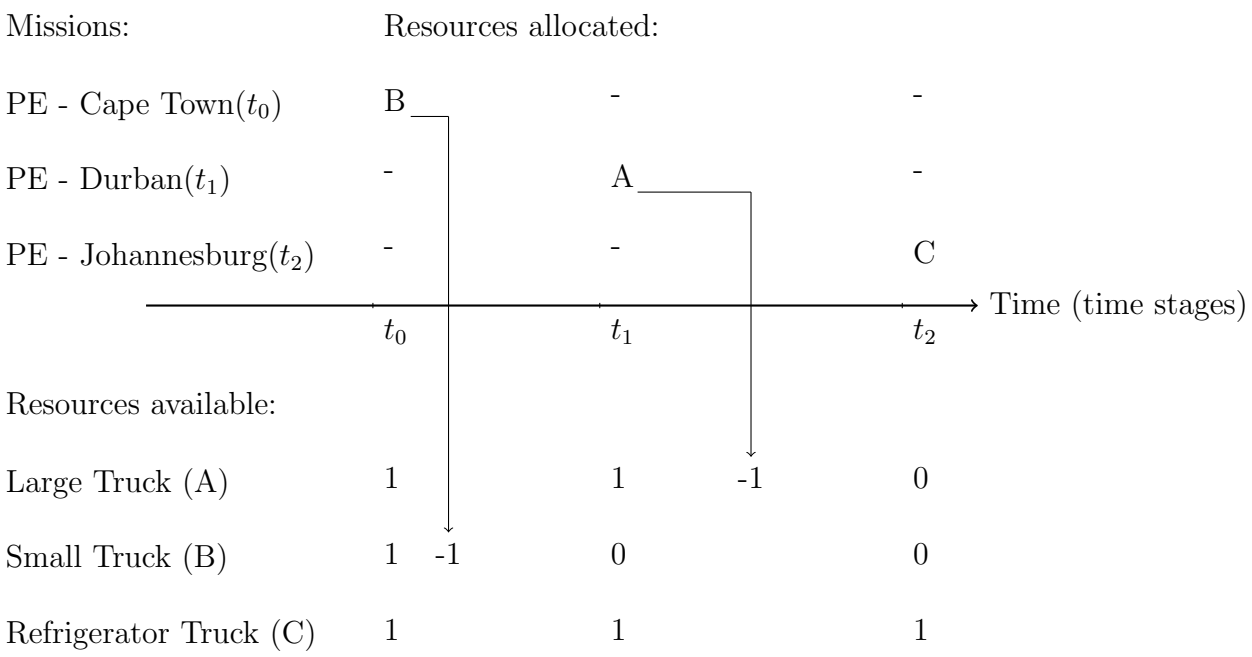


Figure 4.4: Feasible solution to a toy stochastic dynamic MRRA problem

If the company only has a budget of 950 and cannot afford to pay 980, they can consider solution number 5 from Table 4.3. This solution does not have any risk associated with it, since no allocations will have to be changed if resources do not return in time. In that case the company will know exactly what to expect, but it will cost them more than the risky original option that only costs 880 if resource B returns in time. This decision is left to the decision-maker of the company. The model only provides the options and identifies the risks.

Figure 4.5 shows how each non-dominated solution can be affected by a 'worst-case scenario'. Solution 3 stays feasible, solution 9 is forced to become solution 3, solution 10 is forced to become solution 4 and solution 11 is forced to become solution 5 if the correct resources do not arrive in time for their original allocations. The solution numbers, allocation details, lift costs and task suitabilities can be seen in Table 4.3.

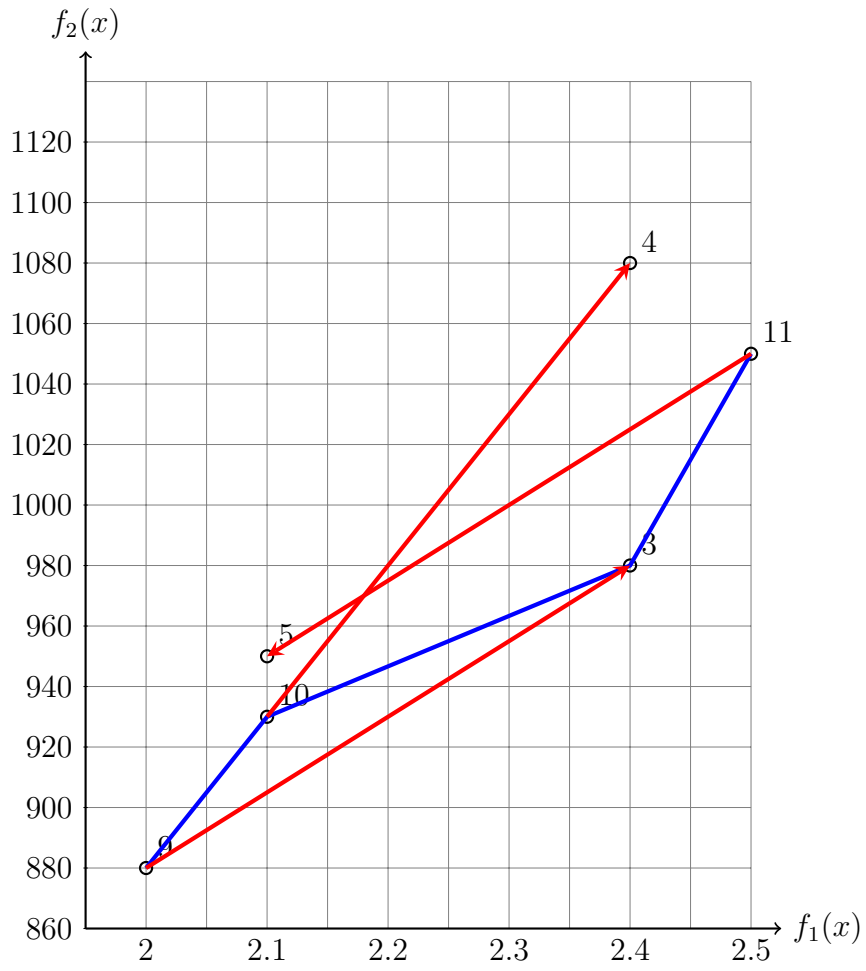


Figure 4.5: Feasible solutions to the stochastic toy problem

4.4 Conclusion to Chapter 4

This chapter looks into the design of the solution model. Two toy problems are used as examples to show the development of the solution model and how feasible solutions to dynamic MRRA problems can be found. The first toy problem is a deterministic dynamic MRRA (DDMRRA) problem. This problem is used to illustrate how the penalty function works and how it ensures that the model provides feasible solutions to the DDMRRA problem. The second toy problem is a stochastic dynamic MRRA (SDMRRA) problem. This toy problem shows how the model deals with uncertainty within the problem. Here it is seen that making decisions based on information that can potentially be different in the future is risky and plans can be forced to change if, for example, a resource does not return in time to start its next mission. This chapter shows how the model takes these risks into account and allows the decision-maker to make a more informed decision, based on all possible outcomes.

Chapter 5

Experimental results

In this chapter, the solution model will be tested and used to solve various larger example problems. To test this model thoroughly, it must be exposed to different types of problems, each designed to test specific important abilities of the model. The abilities that will be tested in this chapter include checking that the model is capable of finding the critical path if there are limited resources available and that the model is capable of handling more than one mission per time point as well as different allocation needs for each mission. Lastly, the model will also be tested on a SDMRRA problem.

In this chapter, a different industry is used for each example. This is to show that the deterministic and stochastic dynamic MRRA problem can be applied in different industries. An example of an MRRA problem at a logistics company was already discussed in the previous chapter, so this chapter will provide examples of a construction company, a hospital and an auditing firm.

5.1 Estimating the Pareto sets

As stated in the introduction, this chapter will be used to test the model on larger dynamic MRRA problems than before. Since these are larger problems, it is too time-consuming to manually determine the true Pareto sets of solutions. Therefore, to find a suitable estimated set of optimal solutions for each example, the model was executed 100 times. The solutions from these attempts were combined to create one optimal set of solutions for each example. The non-dominated solutions from that set will be used as the Pareto set to compare future solutions. Once the estimated Pareto set is found, a single set of solutions from the model can be compared to the Pareto set by using the hyper area ratio method (discussed in Chapter 3).

5.2 Finding feasible solutions when resources are scarce

It is important that the model is able to find feasible solutions successfully, even when the resources are relatively scarce. This is important, since most real-life problems do not have many idle resources waiting to be used. This means that finding feasible solutions is more difficult, since there are fewer. The first example in this chapter is used to show how this model can find the feasible solutions to a problem with few resources.

5.2.1 Construction company example problem

For this example of a deterministic dynamic MRRA problem, a construction company is considered. At this company, three different machines are available. There is one excavator, one bulldozer and one backhoe loader and there are nine jobs. One job must be executed each day over a nine-day period, each job at a different time. Each job can be at one of five different sites. Table 5.1 shows the site where the job needs to be completed at each time stage. One piece of machinery is required at each job and once that resource is used it becomes unavailable for the next day. This is to ensure sufficient time to return to the company storage area and recieve maintenance or repairs after completing the job.

Table 5.1: Jobs to be completed

Day	1	2	3	4	5	6	7	8	9
Job	Site 1	Site 2	Site 3	Site 4	Site 5	Site 1	Site 2	Site 3	Site 4

Table 5.2 shows the lift costs for each piece of machinery to visit the different sites to complete jobs, while Table 5.3 shows the task suitability.

Table 5.2: Construction example problem lift cost

	Site 1	Site 2	Site 3	Site 4	Site 5
Excavator	300	250	320	420	380
Bulldozer	330	280	380	430	370
Backhoe Loader	450	300	400	510	470

Table 5.3: Construction example problem task suitability

	Site 1	Site 2	Site 3	Site 4	Site 5
Excavator	0.7	0.7	0.8	0.8	0.5
Bulldozer	0.8	0.6	0.9	0.6	0.6
Backhoe Loader	0.8	0.5	0.9	0.7	0.7

The objective of this problem is to find a set of feasible non-dominated solutions that will minimise the total lift cost and maximise the sum of the total task suitability over all nine days (time stages). In the next section, the Pareto set for this problem will be estimated so that the performance of the solution model on this deterministic dynamic MRRA problem can be tested.

5.2.2 Construction company example Pareto set

As explained in Section 5.1, the Pareto set was estimated by repeating the optimisation 100 times and taking the non-dominated solutions from that set. Figure 5.1 shows the estimated Pareto set (red) and the combined solutions from the 100 repetitions (black). It can be seen that there are not many black solutions. This is because there are only a very limited number of feasible solutions to this example problem, many solutions within the 100 iterations overlapped, resulting in few black solutions.

This estimated Pareto set (red line in Figure 5.1) is assumed to be the best feasible non-dominated solutions to this problem. The estimated Pareto set can now be used to compare the performance of other solution sets. This will be done in the next section.

It can also be seen that the black solutions are grouped into three groups, each with its corresponding solution on the Pareto set. These groups are formed when the model is forced to decide between a less expensive option that would result in a smaller task suitability and a more expensive option that has a better task suitability.

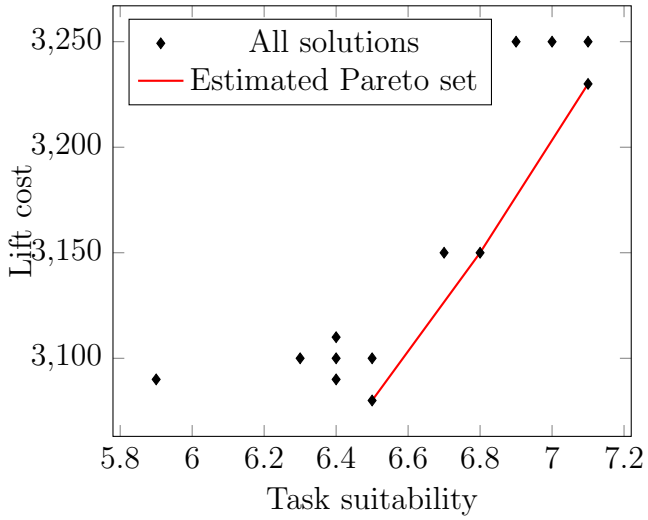


Figure 5.1: Construction company estimated Pareto front

5.2.3 Construction company example test results

To illustrate the performance of the model, it was used to calculate a single set of solutions, as the decision-maker will not use the model 100 times in practice. This single set was then compared to the Pareto set. The results of this specific set of solutions (blue) are plotted against the Pareto set (red) in Figure 5.2 and given in table 5.4. The decision-maker from the construction company must then decide on one of these approaches. If, for example, the decision-maker wishes to have a task suitability of 6.8 at a cost of 3 150 (solution 3 in Table 5.4), the decision variable for that solution is given in Table 5.5. This shows how that task

suitability and lift cost can be achieved.

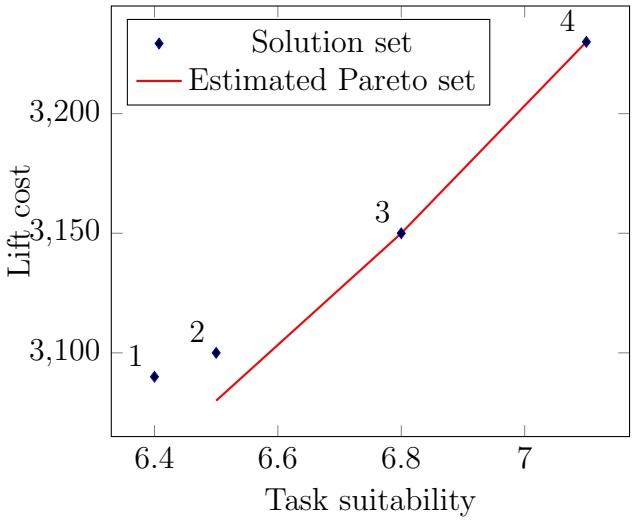


Figure 5.2: Construction company solution set

Table 5.4: Construction example solutions

Solutions Nr	Lift Cost	Task Suitability
1	3090	6.4
2	3100	6.5
3	3150	6.8
4	3230	7.1

Table 5.5: Construction example decision variable for solution 3

Day	Site	Machinery	Lift Cost	Suitability
1	Site 1	Bulldozer	330	0.8
2	Site 2	Excavator	250	0.7
3	Site 3	Bulldozer	380	0.9
4	Site 4	Excavator	420	0.8
5	Site 5	Bulldozer	370	0.6
6	Site 1	Excavator	300	0.7
7	Site 2	Bulldozer	280	0.6
8	Site 3	Backhoe Loader	400	0.9
9	Site 4	Excavator	420	0.5
Total			3150	6.8

The hyper area ratio comparison (from Chapter 3) was used to compare the single solution set to the estimated Pareto set. The result is a hyper area ratio of 1.109. This example shows that the model can find good feasible solutions, even if there are only limited resources available.

5.3 Problems with more complex allocation requirements

Up until this point, all of the example problems discussed have only had one mission per time stage and each mission only required one resource to be allocated. In this section, an example problem with more than one mission per time stage will be discussed. The number of resources required for each mission will also vary.

5.3.1 Hospital example problem

For this example, the scheduling of surgeons in a hospital is considered. In this deterministic dynamic MRRA problem, there are 5 types of medical staff (resources) that work for the hospital and five types of operations (missions) that can be required. The allocation requirements are given in Table 5.6. This example shows that more than one resource can be required for each mission and each time stage can have more than one mission.

Table 5.6: Number of resources required to complete each type of operation

Time Stage	1	2	3	4	5	6	7	8	9
Knee replacements	4	3	1	2	2	1	2	1	4
Hip replacements	3	1	2	2	3	3	2	3	2
Appendectomies	2	2	1	3	2	2	1	4	3
Cataract surgeries	4	3	4	2	3	1	2	1	1
Back surgeries	3	2	3	3	1	4	3	1	3

Table 5.7 shows the lift costs for each resource to complete each operation, while Table 5.8 shows the task suitability.

Table 5.7: Hospital example problem lift cost

	Knee replacement	Hip replacement	Appendectomy	Cataract surgery	Back surgery
Clinical Students	300	250	320	420	380
Nurses	330	280	380	430	370
Doctors	450	300	400	510	470
Surgeons	590	450	610	460	580
Assistants	360	270	350	360	410

Table 5.8: Hospital example problem task suitability

	Knee replacement	Hip replacement	Appendectomy	Cataract surgery	Back surgery
Clinical Students	0.7	0.7	0.8	0.8	0.5
Nurses	0.8	0.6	0.9	0.6	0.6
Doctors	0.8	0.5	0.9	0.7	0.7
Surgeons	0.9	0.9	0.9	0.9	0.8
Assistants	0.6	0.7	0.7	0.6	0.7

The objective of this problem is to find a set of feasible non-dominated solutions that will minimise the total lift cost and maximise the sum of the total task suitability over all time stages. In this example, resources will be unavailable for one time stage after being used. This is due to the duration of the operation procedure. There are five clinical students, thirteen nurses, six doctors, 14 surgeons and three assistants available at the hospital. In the next section, the performance of the solution model will be tested on this deterministic dynamic MRRA problem.

5.3.2 Hospital example Pareto set

As explained in Section 5.1, the Pareto set was estimated by executing the model 100 times and extracting the non-dominated solutions from that set. Figure 5.3 shows the combination of all thet solutions (in black), with the estimated Pareto front indicated by the red lines.

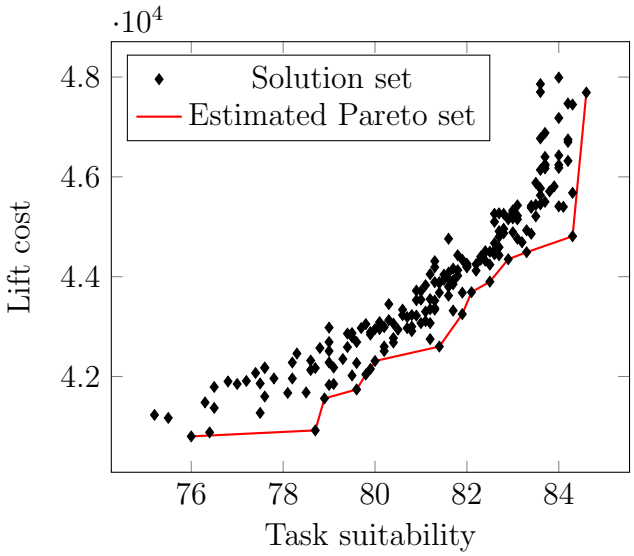


Figure 5.3: Hospital estimated Pareto front

5.3.3 Hospital example test results

The model was used to find a set of solutions that can be compared to the estimated Pareto set, using the hyper area ratio. Figure 5.4 shows this solution.

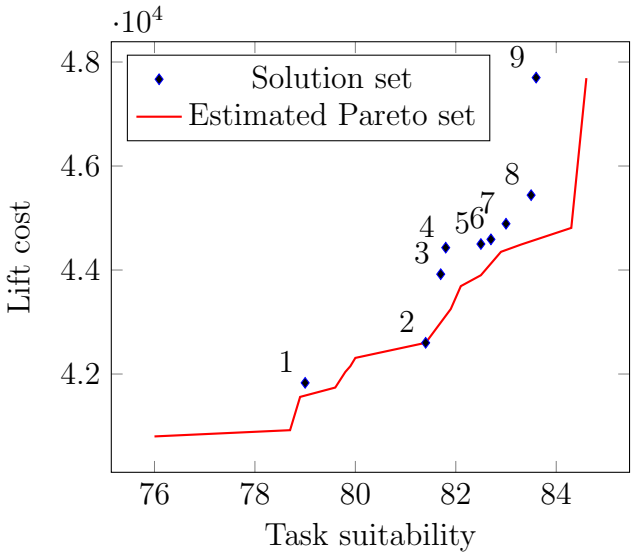


Figure 5.4: Hospital solution set

Table 5.9: Hospital example solutions

Solutions Nr	Lift Cost	Task Suitability
1	41830	75.5
2	42600	76.8
3	43920	77.6
4	44430	79.6
5	44500	80.5
6	44590	80.9
7	44890	81.0
8	45440	81.3
9	47700	82.0

This solution set can be used to advise the scheduling manager of the hospital. If, for example, the hospital has a budget of 43 000 to spend during the nine time stages, then the solution that would give them the best task suitability without spending more than the allocated budget is solution number 2 in Table 5.9. The result of this solution will be to spend 42 600 and have a total task suitability of 81.4 over nine time stages. The allocations that must be done to achieve this are shown in Tables 5.10 and 5.11.

Table 5.10: Hospital example decision variable for solution 2 from time stages 1 to 3

Time Stage	Operation	Students	Nurses	Doctors	Surgeons	Assistants
1	Knee replacement	0	4	0	0	0
	Hip replacement	0	2	1	0	0
	Appendectomy	0	0	1	0	1
	Cataract surgery	2	0	0	2	0
	Back surgery	0	2	0	1	0
2	Knee replacement	0	2	1	0	0
	Hip replacement	0	0	0	1	0
	Appendectomy	0	1	1	0	0
	Cataract surgery	0	0	0	3	0
	Back surgery	0	0	0	2	0
3	Knee replacement	0	0	1	0	0
	Hip replacement	0	1	1	0	0
	Appendectomy	0	1	0	0	0
	Cataract surgery	0	3	1	0	0
	Back surgery	0	3	0	0	0

Table 5.11: Hospital example decision variable for solution 2 from time stages 4 to 9

Time Stage	Operation	Students	Nurses	Doctors	Surgeons	Assistants
4	Knee replacement	0	1	1	0	0
	Hip replacement	0	0	0	0	2
	Appendectomy	3	0	0	0	0
	Cataract surgery	1	0	0	1	0
	Back surgery	0	3	0	0	0
5	Knee replacement	0	1	0	0	1
	Hip replacement	0	0	0	3	0
	Appendectomy	0	0	0	2	0
	Cataract surgery	0	0	0	3	0
	Back surgery	0	1	0	0	0
6	Knee replacement	0	0	1	0	0
	Hip replacement	2	0	0	0	1
	Appendectomy	0	2	0	0	0
	Cataract surgery	0	0	0	1	0
	Back surgery	0	4	0	0	0
7	Knee replacement	0	2	0	0	0
	Hip replacement	0	0	0	2	0
	Appendectomy	0	0	0	1	0
	Cataract surgery	0	0	0	2	0
	Back surgery	0	0	0	1	2
8	Knee replacement	0	0	0	1	0
	Hip replacement	2	0	0	0	1
	Appendectomy	0	1	1	2	0
	Cataract surgery	0	0	0	1	0
	Back surgery	0	0	0	1	0
9	Knee replacement	0	4	0	0	0
	Hip replacement	0	0	0	2	0
	Appendectomy	1	2	0	0	0
	Cataract surgery	0	0	0	1	0
	Back surgery	0	0	0	3	0

This solution set has a hyper area ratio of 1.14, when compared to the estimated Pareto set. It is an acceptable hyper area ratio, because the ratio is close to 1. This illustrates that the model is capable of solving more complex problems.

It is important to notice that the solution gives the best overall result over all the time

stages. This does not necessarily mean that the optimal solution at an individual time stage is found. This is because the model plans ahead and might decide to make a slightly worse allocation at one individual time stage in order to be able to make a much better allocation in the next time stage.

5.4 Stochastic dynamic MRRA problem

In this section the model is tested on a stochastic dynamic MRRA problem. This means that there is uncertainty regarding when a resource will be ready for use again after they are allocated. An example from an auditing firm is considered in this section.

5.4.1 Audit firm example problem

An auditing firm has five types of people working for them and five companies that must be audited, each at a different time stage. This is shown in Table 5.12. In this example, uncertainty around the number of time stages that resources will be unavailable for after being assigned and used exist. It will vary between one and two time stages. This will depend on how long the audits take and when the next time stage's audit is due to begin.

Table 5.12: Companies to be audited

Time Stage	1	2	3	4	5
Company	Company 1	Company 2	Company 3	Company 4	Company 5
Resources required	3	3	3	8	5

Five types of resources are available to the audit firm. These are two directors, eight managers, three third-year clerks, five second-year clerks and 10 first-year clerks. The lift cost and task suitability of each type of resource is given in Tables 5.13 and 5.14 respectively. The objective of this problem is to minimise the total lift cost and maximise the total task suitability over the five time stages.

Table 5.13: Audit firm example problem lift cost

Company	Company 1	Company 2	Company 3	Company 4	Company 5
1st year clerks	300	250	320	420	380
2nd year clerks	330	280	380	430	370
3rd year clerks	360	270	350	360	410
Managers	450	300	400	510	470
Directors	590	450	610	460	580

Table 5.14: Audit firm example problem task suitability

Company	Company 1	Company 2	Company 3	Company 4	Company 5
1st year clerks	0.7	0.7	0.8	0.8	0.5
2nd year clerks	0.8	0.6	0.9	0.6	0.6
3rd year clerks	0.6	0.7	0.7	0.6	0.7
Managers	0.8	0.5	0.9	0.7	0.7
Directors	0.9	0.9	0.9	0.9	0.8

5.4.2 Audit firm Pareto set

The Pareto set can be estimated in a similar way to the problems in the previous sections. The solutions from 100 solution sets were used to estimate the Pareto set, as seen in Figure 5.5. All solutions from the 100 solution sets are shown in black, while the red line highlights the estimated Pareto set.

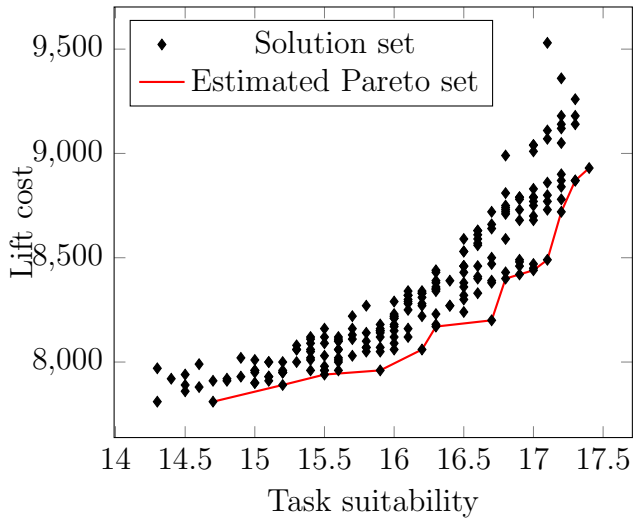


Figure 5.5: Audit firm estimated Pareto front

5.4.3 Audit firm test results

In this section, the performance of the solution model will be tested on this stochastic dynamic MRRA problem. One solution set is compared to the estimated Pareto set. This is shown in Figure 5.6. The different solutions in the set are shown in blue, while each solution's corresponding 'worst-case scenarios' are shown with a green arrow. The estimated Pareto set is, once again, shown in red.

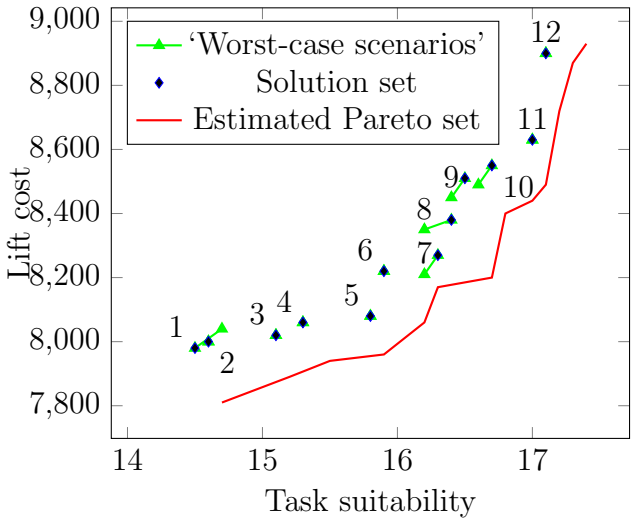


Figure 5.6: Audit firm solution set

The results from the different solutions in the solution set are shown in Table 5.15. The hyper area ratio of the solution set is 1.234, indicating that it is a relatively good solution, when compared to the estimated Pareto set. The 'worst-case scenarios' are also included in Table 5.15 to help the decision-maker to make a more informed decision. The 'worst-case scenario' aims to illustrate the results that the solution will yield if resources only become available two time stages after they were used.

Table 5.15: Audit firm example solutions

Solutions Nr	Lift Cost	Worst Cost	Task Suitability	Worst Suitability
1	7980	14.5	8040	14.7
2	8000	14.6	8000	14.6
3	8020	15.1	8020	15.1
4	8060	15.3	8060	15.3
5	8080	15.8	8080	15.8
6	8220	15.9	8220	15.9
7	8270	16.3	8210	16.2
8	8380	16.4	8350	16.2
9	8510	16.5	8450	16.4
10	8550	16.7	8490	16.6
11	8630	17.0	8630	17.0
12	8900	17.1	8900	17.1
13	9450	17.5	9370	17.3

It can be seen in Table 5.15 and Figure 5.6 that most of the 'worst-case scenario' solutions

only produce worse results in one of the two objectives (cost or suitability) or stay the same as the 'optimistic' solution that will be feasible only if each resource becomes available again after only one time stage. This is because the model must assign a different resource at some critical point. This resource is usually either more expensive and more suitable, or less expensive and less suitable than the one in the 'optimistic' solution. However, it remains important for the decision-maker to know and understand the chance of potentially having a different result. If, for example, the audit firm only has a budget of 8 000 and is looking for the cheapest option, it will be safer for them to choose solution 2, rather than solution 1 in Table 5.15. Although solution 1 is cheaper than solution 2, there is a risk that choosing solution 1 would result in a more expensive outcome in the end if resources take more than one time stage to become available again after being used.

A decision-maker from the Audit firm must then make a decision as to which solution from that set will suit their needs. If, for example, the decision-maker wishes to implement solution 9 in Table 5.15 and all resources become available again after one time stage, the decision variable is shown in Table 5.16. This will result in a lift cost of 8 510 and a task suitability of 16.5. If, however, some resources do not become available again in time, the decision variable can change. This other decision variable is shown in Table 5.17. This will result in a lift cost of 8 450 and a task suitability of 16.4.

Table 5.16: Audit firm example decision variable for solution 9

Time stage	Audit	1^{st} year clerks	2^{nd} year clerks	3^{rd} year clerks	Managers	Directors
1	Company 1	0	3	0	0	0
2	Company 2	3	0	0	0	0
3	Company 3	0	3	0	0	0
4	Company 4	2	2	2	0	2
5	Company 5	0	0	1	4	0

Table 5.17: Audit firm example 'worst-case scenario' decision variable for solution 9

Time stage	Audit	1^{st} year clerks	2^{nd} year clerks	3^{rd} year clerks	Managers	Directors
1	Company 1	0	2	0	1	0
2	Company 2	3	0	0	0	0
3	Company 3	1	2	0	0	0
4	Company 4	2	2	2	0	2
5	Company 5	0	0	1	4	0

Tables 5.16 and 5.17 show how the decision variable can change if the second year clerks working on the Company 1 audit do not finish in time to start working on the Company 3 audit. In this case, since there are only five second-year clerks available and three of them work on the Company 1 audit, only two will be available for the Company 3 audit. The result is that a first-year clerk must be used to satisfy the requirements of having three people working on the Company 3 audit. It is because of this change that the 'worst-case scenario' is cheaper but produces a worse suitability.

5.5 Conclusion to Chapter 5

In this chapter, the solution model was tested on larger problems. The solution to the construction company example showed that feasible solutions can be found even if there are limited resources available. This proves the model's capability of finding a critical path with feasible solutions.

Next, an example problem within a hospital was considered. The solution to this example problem illustrated that more than one resource can be allocated to each mission and also that the model is capable of finding solutions even when there is more than one mission per time stage.

An example of an auditing firm was also used to illustrate the solution to a larger stochastic dynamic MRRA problem. The outcome was that the model can provide useful solutions to a stochastic problem where it is uncertain how long resources will be unavailable for.

From these examples, it is clear that the solution model is reliable in finding a set of relatively good feasible solutions to various dynamic MRRA problems. This chapter shows that the solution model can be implemented and used to solve real-world problems in different business environments.

Chapter 6

Conclusion

6.1 Summary

This study identified that there is a need for this MRRA problem to be applied over a longer period of time. This was identified by conducting a literature review on resource allocation and weapon-assignment problems and finding solutions to these types of problems. Since many of these problems were solved at a time when the operations research community was focused on optimising war strategies, the need for solutions that could be implemented in various modern business environments became clear. It was highlighted that the development of a stochastic dynamic MRRA problem would be useful in these environments. This is because many business environments have problems that need to be optimised over a longer period of time and which take uncertainty into account.

The static MRRA problem is an established problem. This means that a solution to a dynamic MRRA problem must be found. This solution must aim to maximise total task suitability and minimise total cost over more than one time stage. Furthermore, this solution must also take into account the uncertainty of whether resources will be available. This model was developed in Python, using simulated annealing and MOOCEM to optimise objective functions.

These MOO techniques were chosen after comparing AMOSA, MOOCEM and MOGA on various test problems and using the hyper area ratio method to determine how good the solution set of each method is. The paired-t test was used to determine the statistical significance of the performance of each MOO technique in terms of hyper area ratios and computing time, when compared to one another. This indicated that the coding in Python is correct, while also providing a comparison between suitable techniques.

The solution model was designed to solve a dynamic MRRA problem. Feasible solutions were found by developing a penalty function that quantifies how 'infeasible' any solution is. In other words, the penalty function was developed to give an indication of how far solutions are from being feasible. The model combines this penalty function and simulated annealing to find feasible solutions that can be implemented in practice.

This thesis also discussed and showed how the model can be used to determine the 'worst-case scenario' for any solution in problems where there is uncertainty about whether or not resources will be available for certain missions after being used for earlier missions. The solution model provides an expected range of outcomes between the so-called 'optimistic' outcome and the 'worst-case scenario' outcome. This way the decision-maker can be prepared for different outcomes even when the problem is stochastic of nature.

Once the model was developed and coded in Python, it was tested on a variety of test examples. These examples were used to test whether the model is capable of solving problems in different environments. The results of these tests show that the model works correctly and can be used in the industry.

6.2 Self-reflection

During this study the student learnt about multi-objective optimisation (MOO) and the different techniques to ensure feasible solutions in problems where there is an overwhelming number of infeasible solutions. As stated in Chapter 1, MRRA problems often have many non-feasible solutions and only a few feasible solutions. The result is that, in order to solve the dynamic MRRA problem, special techniques are required to ensure that feasible solutions are found. In the study this was done with the so-called 'penalty function' within the model.

The student also improved his programming skills while working on this project. He learnt Python and used it to build a solution model and various MOO techniques, as well as the final solution model. Solving a dynamic MRRA problem with Python improved the student's analytical thinking abilities.

Something that could have been improved is the solution to the stochastic element of the problem. A 'worst-case scenario' was used in this study, simply because every single potential scenario could not be investigated. However, if this could have been done, it would have provided the decision-maker with an even more informed decision.

6.3 Future work

As stated in the previous section, the stochastic part of the problem can be investigated further. For example, one solution from a set can be taken and each possible outcome for that proposed solution can then be investigated. All the different possibilities can then be looked at and taken into account. The various possibilities could then contribute to a weighted average, with the probability of each outcome determining the weight. This would provide one expected lift cost and one expected task suitability for each solution within a set. This, possibly still combined with a best- and worst-case scenarios, would give the decision-maker a better idea of what lift cost and suitability to expect.

The decision-maker could also be helped further if the model did not produce a set of non-dominated solutions where the decision-maker must still choose one solution from a whole set of solutions, but rather if only one solution can be given to the decision-maker. This can be based on the utility of the decision-maker. This would mean that the utility value of the decision-maker must be determined, perhaps with some test questions to determine the utility

of their money accurately. The utility value is an indication of how much a person is willing to risk or spend for a better service in this case. For example, a person might be willing to spend R100 on a bottle of wine, but not R300 for a better bottle of wine, because at some point, in the eyes of the decision maker, the better wine will not be worth paying more for. How much a person values good wine will obviously depend on the person (a student might not be willing to pay R300 for any bottle of wine, no matter how good it is, because he or she does not value good wine as much). This utility value can then be incorporated into the model so that only one solution that is fit for the decision-maker is chosen from the non-dominated set.

Bibliography

Ahuja, R.K., Kumar, A., Jha, K.C. & Orlin, J.B. (2007). Exact and Heuristic Algorithms for the Weapon-Target Assignment Problem. *Oper. Res.*, **55**, 1136–1146. 25, 26

Altay, N. & Green, W.G. (2006). OR/MS research in disaster operations management. *Eur. J. Oper. Res.*, **175**, 475–493. 42

Balcik, B. & Beamon, B.M. (2008). Facility location in humanitarian relief. *Int. J. Logist. Res. Appl.*, **11**, 101–121. 41

Bandyopadhyay, S., Saha, S., Maulik, U. & Deb, K. (2008). A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA. *IEEE Trans. Evol. Comput.*, **12**, 269–283. 48

Caunhye, A.M., Nie, X. & Pokharel, S. (2012). Optimization models in emergency logistics: A literature review. *Socioecon. Plann. Sci.*, **46**, 4–13. 17, 41

Coello Coello, C. (2006). Evolutionary multi-objective optimization: a historical view of the field. *IEEE Comput. Intell. Mag.*, **1**, 28–36. 43, 44, 45

Coello Coello, C.A., Van Veldhuizen, D.A. & Lamont, G.B. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*, vol. 5 of *Genetic Algorithms and Evolutionary Computation*. Springer US, Boston, MA. 62, 64, 72

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, **6**, 182–197. 51

DenBroeder, G.G., Ellison, R.E. & Emerling, L. (1959). On Optimum Target Assignments. *Oper. Res.*, **7**, 322–326. 23, 24, 37

Fonseca, C.M. & Fleming, P.J. (1994). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization *. Tech. rep. 45

Goldberg, D.E. & Holland, J.H. (1988). Genetic Algorithms and Machine Learning. *Mach. Learn.*, **3**, 95–99. 45

Holguín-Veras, J., Pérez, N., Jaller, M., Van Wassenhove, L.N. & Aros-Vera, F. (2013). On the appropriate objective function for post-disaster humanitarian logistics models. *J. Oper. Manag.*, **31**, 262–280. 41

Hosein, P.A., Walton, J.T., Athans, M. & Athans, M. (1988). Dynamic weapon-target assignment problems with vulnerable C 2 nodes. *Lab. Inf. Decis. Syst.*. 27, 28, 29, 37

Huaiping, C., Jingxu, L., Yingwu, C. & Hao, W. (2006). Survey of the research on dynamic weapon-target assignment problem. *J. Syst. Eng. Electron.*, **17**, 559–565. 29

Ishibuchi, H., Imada, R., Setoguchi, Y. & Nojima, Y. (2018). How to Specify a Reference Point in Hypervolume Calculation for Fair Performance Comparison. *Evol. Comput.*, **26**, 411–440. 59, 61

Johnson, D.D. & MacKay, N.J. (2015). Fight the power: Lanchester's laws of combat in human evolution. *Evol. Hum. Behav.*, **36**, 152–163. 12

Jones, D., Mirrazavi, S. & Tamiz, M. (2002). Multi-objective meta-heuristics: An overview of the current state-of-the-art. *Eur. J. Oper. Res.*, **137**, 1–9. 2

Kirby, M. & Capey, R. (1997). The air defence of Great Britain, 19201940: an operational research perspective. *J. Oper. Res. Soc.*, **48**, 555–568. 12

Kirby, M. & Capey, R. (1998). The origins and diffusion of operational research in the UK. *J. Oper. Res. Soc.*, **49**, 307–326. 1, 2, 3

Koepke (2018). 10 Reasons Python Rocks for Research (And a Few Reasons it Doesn't) Hoyt Koepke. https://www.stat.washington.edu/~hoytak/blog/whypy. 56

Lötter, D.P. (2017). *Design of a weapon assignment subsystem within a ground-based air defence environment*. Ph.d. thesis, Stellenbosch University. 21, 23, 27, 28, 29, 33

Mahotas (2013). Mahotas: Open source software for scriptable computer vision. *J. Open Res. Softw.*, **1**, e3. 56

Manne, A.S. (1958). A Target-Assignment Problem. *Oper. Res.*, **6**, 346–351. 18, 21, 25, 37

Mathworks (2018). MATLAB vs. Python: Top Reasons to Choose MATLAB - MATLAB & Simulink. https://www.mathworks.com/products/matlab/matlab–v. 55

Murphey, R.A. (2000). Target-Based Weapon Target Assignment Problems. 39–53, Springer, Boston, MA. 32, 37

Orden, A. & Goldstein, L. (1952). Symposium on Linear Inequalities and Programming, Washington, D.C., June 14-16, 1951. http://agris.fao.org/agris–search/search.do?record. 19, 20

Petrovic, Sanja; MacCarthy, B. (2014). The OR Society: Blog - Operational Research versus Operations Management. 11

Pirouzan, D., Yahyaei, M. & Banisi, S. (2014). Pareto based optimization of flotation cells configuration using an oriented genetic algorithm. *Int. J. Miner. Process.*, **126**, 107–116. 46

Pradhananga, R., Mutlu, F., Pokharel, S., Holguín-Veras, J. & Seth, D. (2016). An integrated resource allocation and distribution model for pre-disaster planning. *Comput. Ind. Eng.*, **91**, 229–238. 42

Rosenhead, J. (2006). Past, present and future of problem structuring methods. *J. Oper. Res. Soc.*, **57**, 759–765. 12

Rosner, B. (1982). A Generalization of the Paired t-Test. *Appl. Stat.*, **31**, 9. 63

Sabri, E.H. & Beamon, B.M. (2000). A multi-objective approach to simultaneous strategic and operational planning in supply chain design. *Omega*, **28**, 581–598. 3

Schlünz, E.B. (2016). *Multiobjective in-core fuel management optimisation for nuclear research reactors*. Ph.D. thesis, Stellenbosch University. 44, 45, 48, 51, 52

Scholtz, E. (2014). *A comparative study on the value of accounting for possible relationships between decision variables when solving multi-objective problems*. Ph.D. thesis, Stellenbosch University. 1, 2

Srinivas, N. & Deb, K. (1994). Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evol. Comput.*, **2**, 221–248. 45

Van Wassenhove, L.N. (2006). Humanitarian aid logistics: supply chain management in high gear. *J. Oper. Res. Soc.*, **57**, 475–489. 17

Veldhuizen (2007). MOP Evolutionary Algorithm Approaches. In *Evol. Algorithms Solving Multi-Objective Probl.*, 61–130, Springer US, Boston, MA. 53

Wakefield, D.J. (2001). *Identification of Preferred Operational Plan Force Mixes Using a Multi-objective Methodology to Optimize Resource Suitability and Lift Cost*. Ph.D. thesis, Air Force Institute of Technology. 1, 2, 16, 37

Yi, W. & Kumar, A. (2007). Ant colony optimization for disaster relief operations. *Transp. Res. Part E Logist. Transp. Rev.*, **43**, 660–672. 17

Yu, L., Zhang, C., Yang, H. & Miao, L. (2018). Novel methods for resource allocation in humanitarian logistics considering human suffering. *Comput. Ind. Eng.*, **119**, 1–20. 17