THE GERMINAL CENTRE ARTIFICIAL IMMUNE SYSTEM

by

AYUSH JOSHI

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
July 2016

Abstract

This thesis deals with the development and evaluation of the Germinal centre artificial immune system (GC-AIS) which is a novel artificial immune system based on advancements in the understanding of the germinal centre reaction of the immune system. The key research questions addressed in this thesis are: can an artificial immune system (AIS) be designed by taking inspiration from recent developments in immunology to tackle multi-objective optimisation problems? How can we incorporate desirable features of the immune system like diversity, parallelism and memory into this proposed AIS? How does the proposed AIS compare with other state of the art techniques in the field of multi-objective optimisation problems? How can we incorporate the learning component of the immune system into the algorithm and investigate the usefulness of memory in dynamic scenarios? The main contributions of the thesis are:

- Understanding the behaviour and performance of the proposed GC-AIS on multi-objective optimisation problems and explaining its benefits and drawbacks, by comparing it with simple baseline and state of the art algorithms.
- Improving the performance of GC-AIS by incorporating a popular technique from multi-objective optimisation. By overcoming its weaknesses the capability of the improved variant to compete with the state of the art algorithms is evaluated.
- Answering key questions on the usefulness of incorporating memory in GC-AIS in a dynamic scenario.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Artificial immune systems are a class of bio-inspired algorithms which use the immune system of humans as the basis of their inspiration. They fall in the broader area of nature inspired computation which deals with computational models based on natural phenomenon. Some of the most established branches of nature inspired techniques include diverse models such as evolutionary computation (EC) based on Darwinian evolution [6], neural computation based on the working principle of the brain [62], swarm intelligence inspired by the swarm behaviour of birds [17], membrane computing based on the membrane structure of cells [122] and artificial immune systems. A review of nature inspired techniques can be found in [48].

Artificial immune systems (AIS) have been studied for over two decades and numerous algorithms inspired by immunological concepts have been developed and applied in several application areas. Certain problem domains can be associated with immune processes more than others, such as anomaly detection, clustering/classification, virus detection or computer security. However, the existing applications of AIS are not limited to only these domains. Successful application of AIS has been seen in diverse areas which can seem to be not directly related to immunological functions such as robotics, image processing and optimisation [43]. There are a few examples of AIS applications to challenging real

world problems or use within the industry but despite the successful applications of AIS the true potential of AIS has not yet been fully realized [82].

The opinion of some of the leading researchers in the field of AIS [82] has been that finding the right problem for AIS by associating the problem's features with AIS mechanisms, has been suggested as the way forward for AIS research in order to determine areas which can clearly benefit from immune inspired approaches. Following these guidelines, the research in this thesis explores a problem focused approach to present a novel AIS inspired by recent research in immunology, aimed towards multi-objective optimisation and provides an investigative experimental analysis on its behaviour and performance.

The structure of this chapter is as follows: Section 1.2 describes the motivation behind the research in this thesis. It comprises of sub-sections which deal with the motivation behind studying the specific problem classes investigated during the research namely, the set cover problem, the knapsack problem and the single time-step dynamic set cover problem. This is followed by the key research questions addressed in this thesis in section 1.3. The key contributions made in this thesis are listed in section 1.4 and an overview of the chapters in this thesis is provided in section 1.5. The publications arising from this thesis are listed in section 1.6 and finally a summary is presented in section 1.7.

1.2 Motivation

This thesis presents the development of a novel AIS called the *Germinal centre artificial immune system* (GC-AIS) which has been designed by taking inspiration from recent immunological findings, specifically understanding of the workings of the germinal centre reaction [163] in the immune system. Germinal centres are regions formed in the body which provide an immune response in order to eradicate specific invading pathogens which are not eliminated by the static immunity of the body which is non-specific and generic in nature [113].

1.2.1 The set cover problem and the immune system

In simple terms the set cover problem (SCP) can be stated as: if one is given a set of items called the universe set and a collection of subsets of these items, the goal of the set cover problem is to cover all the items of the universe using the least number of subsets from the collection of subsets.

The motivation to study SCP for AIS research comes from the fact that the immune processes of fighting against pathogens, in an abstract way can be seen as the immune system trying to to solve the SCP. At the onset of the disease caused by a pathogen invading the body, the immune system must prepare a counter attack towards the pathogen. One way it does so is by producing special cells called antibodies (Ab) which can bind themselves with the pathogen to eradicate it. Each pathogen has a special region called the *active site* to which the antibodies can attach themselves via their binding sites, which can be seen as the . At the onset of the disease the antibodies may be poor and barely bind with the pathogen, but they undergo several iterations of improvements in an evolutionary manner, during which they begin to get better and start binding first weakly and then strongly with the antigens. It is only when the binding is strong enough that they can eradicate the pathogen. In other words the immune system wins versus the pathogen only when its antibodies can bind strongly with the pathogens.

The shared characteristics of the immune behaviour in relation to the set cover problem can be explained in the following principles. Assume that every pathogen in nature is an instance of the SCP problem which the immune system must address at a given time. The binding sites of the antibodies can be seen as potential solutions to the problem, i.e. a collection of subsets of the universe. These antibodies must improve from barely binding (covering only a part of the universe, possibly using many subsets) to weak binding (covering the whole universe but not with least subsets) to strong binding (covering the universe with the least subsets) to the pathogen's *active sites* by the process of optimisation. Hence the immune system tries to solve the problem of finding the best matching between the antibodies and pathogens by randomised variations in the potential solutions

[93].

Even though the set cover problem is a single objective problem with a constraint it can be converted to a multi-objective problem by using the constraint as a secondary objective. According to [115] this multi-objective formulation can be more efficient than the single objective version in finding the optimal solution, when used by an an evolutionary algorithm. At the same time the authors of [82] mention that the immune system has many goals suggesting their application for multi-objective optimisation. As an example, apart from the obvious task of fighting invading pathogens, the immune system also takes part in other processes such as assisting in wound healing, promoting the development of the organs of the immune system [131]. This suggests that multi-objective optimisation as an application domain closely resembles the problems the artificial immune systems tackles.

More specifically germinal centre model of the immune system incorporates the two features that must be present in every multi-objective optimisation algorithm: guiding the search towards the global Pareto optimal region as well as maintaining diversity of solutions in the non-dominated front [53]. Guiding the search towards the optimal Pareto region can be seen in the germinal centres as the continuous process of improving the immune cells to fight against the invading pathogen. Maintaining diversity in the population can be seen during the communication and feedback of immune cells between germinal centres which provides selection pressure and helps maintain diversity as well as population size of the germinal centres.

In order to follow a problem focused approach, first a problem must be selected that needs to be addressed. In this case the problem is the set cover problem and the motivation to design a new AIS stems from associating the SCP with the processes of the immune system that fight pathogens. Another attribute of the problem that motivates the use of AIS to tackle this problem are is the fact that a multi-objective formulation of the problem can be better than the single formulation as suggested in [115] and as stated above AIS in general and germinal centre theory in particular has features that strongly

resemble multi-objective optimisation.

1.2.2 A real multi-objective problem

Though the initial motivation to design the GC-AIS stems from the need to tackle the multi-objective version of the SCP the resulting AIS designed is just like any other AIS for optimisation i.e. a is general purpose meta-heuristic. In order for the designed approach to be an effective meta-heuristic it must be tested on other multi-objective problems. Since the true goals of SCP are not multi-objective, a true multi-objective problem is needed for the validation of the effectiveness of the proposed algorithm. The multi-objective knapsack problem is an NP-hard combinatorial optimisation problem which is an extension of the knapsack problem using multiple knapsacks. One of the motivations for selecting to study this problem is the extensive work that has been done on this problem using multi-objective evolutionary algorithms (MOEAs). The existing literature can provide both a healthy competition of approaches to solve the problem along with guidance on the design of experiments when performing empirical analysis of the AIS.

1.2.3 Memory in dynamic environments

Real world problems often involve changes over time and previously known optimal solutions are usually not robust enough to find the new solution with relative ease for the changed problem [82]. The immune system is a dynamic and continuously adapting system which learns and maintains a memory of previously encountered diseases which makes memory and learning an indispensable feature of the immune system. Dynamic optimisation particularly dealing with a memory component is a suitable domain which closely resembles immune processes. Germinal centres are particularly suited for this application as they are responsible for the production of memory cells in the body during the course of the germinal centre reaction. This motivates the idea to incorporate memory in the proposed GC-AIS and understand the behaviour of the new approach.

As an extension to the study of SCP, it is only fitting to examine a dynamic version of the problem with the motivation being that there is a practical relevance to studying the dynamic SCP as shown in [28] where a model for dynamic facility location are proposed. Another practical application of dynamic SCP can be seen in crew scheduling for airlines or railways. The dynamics in these situations can arise due to variations in the number of rails/ aircraft or the number of crew available at different times.

Most of the work done on dynamic SCP along with techniques developed for evolutionary dynamic optimisation using memory approaches, focus on solving the problem in the dynamic setting and tracking the changing optimal values. Some work on memory techniques has shown the advantages of memory as being suitable in cases only when the changes recur. By extending the study on static SCP to incorporate a dynamic scenario, the goal is to find cases where using memory is useful and where it is harmful as well as a more fundamental question of why does memory behave in this fashion given these cases. To the best of our knowledge this will is the first time a memory based approach has been used to tackle the dynamic SCP.

1.3 Research questions

The following key questions considered in this thesis clarify the main objectives of this research. These questions are presented again in the following chapters where they are addressed along with a discussion on the answers obtained.

The biological immune system's task of fighting against diseases by creating and remembering of antibodies can be modelled as the set cover problem in combinatorial optimisation. Based on the knowledge of recent research in the understanding of the germinal centre reaction in the biological immune system, since the SCP is the key motivating problem behind the design of a new AIS, a key question arises:

> *Is it possible to model the recent understanding of the* GC *reaction into an immune system that tackle the multi-objective* SCP*?*

The algorithm design process must abstract processes from the GC reaction that are suitable for multi-objective optimisation and must be tested on multi-objective problems to analyse its performance. This question is addressed in chapter 4 where the GC-AIS is presented. An abstract model of the immune processes in the germinal centre reaction is created by carefully studying its components. This abstract model is then converted to an artificial immune system which is presented as the GC-AIS algorithm.

Given that the biological immune system like any natural process is robust and performs well, it will be interesting to answer the following query:

How does the novel artificial immune system perform on the set cover problem?

An experimental analysis on the performance GC-AIS is presented on in chapter 4 on the SCP where it is compared with a simple multi-objective evolutionary algorithm called PGSEMO. Run time evaluation of solution quality versus elapsed time as evaluations as well as communication effort are used as the criteria for comparison.

Moreover, since the designed AIS is a meta-heuristic that can be applied to multi-objective problems, after studying the performance on SCP it is well worth to investigate:

How does the novel algorithm perform on a real multi-objective optimisation problem such as the multi-objective knapsack problem?

In the same chapter, along with the experiments on the set cover problem, GC-AIS is also compared with a popular multi-objective optimisation algorithm (NSGA-II) on several instances of the MOd-KP. For this study 3 prominent metrics from multi-objective optimisation are utilised, namely: hypervolume, diversity and generational distance. Potential strengths and weaknesses of the algorithm are then observed based on the experiments.

The studies on SCP and MOd-KP provide an analysis on the behaviour of GC-AIS and these studies lead to another key question:

What can be learned about the strengths and weaknesses of the GC-AIS *based on these studies?*

The key strength of the algorithm which is observed as a result of the experimental analysis in chapter 4 is that it does not require any parameter tuning if the mutation rate remains fixed. This can be very crucial as parameter tuning is an extremely important process to obtain good performance from evolutionary algorithms. At the same time, population size explosion is identified as the key weakness of the algorithm, which is observed in experiments with the knapsack problem when the dimensionality of the instances increased. This is a weakness as it can lead of wastage of available fitness evaluation in explored areas as well as causing challenges for the decision maker to decide on a final solution to the problem.

This is an important question as it helps in understanding the design decisions behind the algorithms which follows into:

> *Knowing certain weaknesses in* GC-AIS*, how can improvements be made to overcome the shortcomings?*

The weakness of the algorithm are addressed in chapter 5 where an improved variant of the GC-AIS is presented by modifying the domination strategy. The well known concept of ϵ-dominance is incorporated into GC-AIS to handle population explosion, which replaces the original dominance relation. Though originally proposed for promoting diversity and convergence in populations, this technique has been used for maintaining population size in algorithms.

These improvements in GC-AIS leads to the testing, analysis and evaluation cycle on the SCP and MOd-KP to determine if the improvements have been successful.

Dynamic optimisation problems involve finding and tracking the changing optimal solution. Knowing that memory and learning are important components of the immune system a key question to investigate is:

> *When using* GC-AIS *for dynamic optimisation of* SCP*, does using solutions from memory perform better than starting from scratch ?*

Upon closer inspection of this question it can be further broken down into a more fundamental query:

When does using memory perform better than a solution from scratch and when does it fail? why does this behaviour occur?

These questions are addressed in chapter 6, where a memory component is added to the GC-AIS and this variant called the M-GC-AIS is analysed in a dynamic setting of the set cover problem. A model to introduce different kinds of changes in static set cover problem is introduced and a single time step dynamic version of the set cover problem is formulated. Using this problem the GC-AIS is compared with M-GC-AIS in order to discern the usefulness of memory in different versions of the dynamic set cover problem.

1.4 Contributions of the thesis

By providing answers to the research questions asked above the following contributions are made in this thesis:

1. **A novel** AIS **designed for multi-objective optimisation.**

   By condensing ideas from the germinal centre theory that are relevant and enticing for multi-objective optimisation a novel AIS is proposed to tackle the SCP. Several algorithms exists in the field of AIS, each with its own strengths and weaknesses. The GC-AIS, presented in this thesis is a simple yet effective algorithm for solving the SCP, that can be applied for multi-objective optimisation, which has enticing properties such a small number of parameters which makes it an appealing algorithm for researchers in the field of AIS and multi-objective optimisation.

2. **An investigation on the performance of** GC-AIS **on** SCP **and MOd-KP**

   A performance analysis which sheds light on the behaviour of GC-AIS when considering the SCP and MOd-KP is presented. Performance benefits over competitors are elucidated as well as a key weaknesses is identified. This is one of the very few experimental works on the multi-objective setting of SCP. Previous works have been limited to single objective scenarios or theoretical studies.

3. **An improved variant of** GC-AIS

   Weaknesses identified in GC-AIS are explored and tackled. An improved variant of the GC-AIS called ϵ-GC-AIS is proposed. A popular technique from multi-objective optimisation which is mostly used for introducing diversity as well as convergence in MOEAs is incorporated in ϵ-GC-AIS. The advantages of the techniques include improving diversity and convergence as well as overcoming the weaknesses in GC-AIS.

4. **Investigative analysis of ϵ-**GC-AIS **on** SCP **and MOd-KP**

   Analysis on the performance of ϵ-GC-AIS is performed by a comparative study with a state of the art algorithm for multi-objective optimisation on SCP and MOd-KP.

5. **Understanding the effects of memory in a dynamic scenario.**

   Using memory may not always be useful when considering a dynamic setting. This phenomenon is well known in biological immune systems and research exists which describes its causes. As a consequence, scenarios when and where using memory is useful for dynamic SCP and what causes memory to behave in this manner is presented.

1.5 Overview

The first three chapters of the thesis introduce relevant problems and concepts and serve as a background for the remaining thesis. The core of the thesis is contained in Chapters 4 through 6 which follow progressively to answer the research questions. Though each core chapter is somewhat self-contained, the thesis can be best appreciated when read in its entirety.

A background of concepts important to the thesis along with a literature review of the relevant work done in the field is provided in Chapter 2. Artificial immune systems are introduced and some important existing models of the immune system are dis-

cussed. A review of the key work done on the different models of the immune system is provided. Multi-objective optimisation along with dynamic evolutionary optimisation is introduced, covering the formal introduction of the set cover problem and the multi-objective d-dimensional knapsack problem. A literature view of relevant work on the set-cover problem and d-dimensional knapsack problem in the multi-objective context is presented along with a review of memory based techniques for dynamic optimisation followed by relevant work done on the dynamic set cover problem. These problems form the test bed for the empirical experiments in this thesis.

The methodology used for the experimentation and analysis of results are detailed in Chapter 3. The multi-objective algorithms used for comparison with GC-AIS are presented in detail followed by a description of the problem instances which are used for the experimentation. The stopping criteria for the various algorithms is discussed followed by the technique used for reference front generation employed in this thesis. Finally the metrics of comparison employed for the analysis of the results are detailed followed by a brief description of the statistical tests used in this thesis.

Chapter 4 introduces the GC-AIS algorithm. The design of the GC-AIS is explained making abstractions from the theory of the GC reaction. This is followed by an evaluation of GC-AIS by comparative experimental studies on the SCP and MOd-KP.

Chapter 5 tries to remedy the shortcomings of GC-AIS observed in Chapter 4. ϵ-dominance is introduced as a method to solve the weaknesses of GC-AIS and is incorporated in the variant of GC-AIS called ϵ-GC-AIS. The performance of ϵ-GC-AIS is evaluated on SCP and MOd-KP by comparing with some state-of-the-art algorithms.

The effects of learning are investigated in Chapter 6, where a memory component is introduced in GC-AIS, called the M-GC-AIS. Two models for dynamic SCP are introduced in detail along with methodology for instance creation for these models alongside motivations and practical applications of the models. Questions such as when and why is using memory useful in the case of the dynamic SCP are considered.

The thesis is concluded in Chapter 7 with a discussion of the contributions and impli-

cations along with some possible directions of future work.

1.6 Publications associated with this research

A list of the refereed publication arising from the research conducted as part of this this is presented below. In addition, the relationship between the thesis chapters and the work presented in these publications is mentioned along with a statement of my contribution in these publications.

1. Ayush Joshi, Jonathan E. Rowe, and Christine Zarges. An immune-inspired algorithm for the set cover problem. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipic, and Jim Smith, editors, *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, volume 8672 of *Lecture Notes in Computer Science*, pages 243–251. Springer, 2014. The work in this paper is featured in chapter 4. For this paper I was responsible for the majority of the work, which was comprised of experimentation, analysis of the results and writing.

2. Ayush Joshi, Jonathan E. Rowe, and Christine Zarges. Improving the performance of the germinal center artificial immune system using \epsilon -dominance: A multi-objective knapsack problem case study. In Gabriela Ochoa and Francisco Chicano, editors, *Evolutionary Computation in Combinatorial Optimization - 15th European Conference, EvoCOP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, volume 9026 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2015. The work in this paper is featuyred in chapter 4 and chapter 5. For this paper I was responsible for the majority of the work, which was comprised of experimentation, analysis of the results and writing.

3. Ayush Joshi, Jonathan. E. Rowe, and Christine Zarges. On the effects of incorporating memory in GC-AIS for the Set Cover Problem. In *Proceedings of the 11th*

Metaheuristics International Conference, 2015 The work in this paper is featured in chapter 6. For this paper I was responsible for the majority of the work, which was comprised of experimentation, analysis of the results and writing.

1.7 Summary

This chapter sets the stage for the motivations and ideas behind the research presented in this thesis. The key problems addressed in this thesis are identified and a connection between the problems is made with the principles and properties of the immune system which serves as the motivation to design and test a new AIS algorithm for these problem domains. The key research questions addressed in this thesis are discussed along with pointers to Chapters where they are discussed. The contributions made by the research in this thesis are listed in a consolidated form along with an overview of the structure of the thesis. Finally the publications arising from the work done towards this thesis are mentioned along with the contribution of the author towards these publications.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

Solving real-world problems can be challenging and the conventional techniques which employ analytical and exact methods to solve them are unable to scale well and maintain robustness [5]. At the same time processes in nature are capable of performing complicated tasks well and are robust. Understanding these natural systems better and borrowing more detailed ideas from these systems will hopefully help us design better systems [82].

Two characteristics of many real-world problems that can make it more complex are multiple objectives and dynamic behaviour. Multi-objective optimisation (MOO) deals with problems which have several objectives, many of which compete with each other. Due to this trade-off between objectives, a single optimal solution to the problem is not possible. The solutions to these problems are in the form of a set called the Pareto optimal set. The two characteristics any algorithms which deal with MOOPs must have are that it must contain a population of solution that is diverse as well as it must constantly improve the population towards the Pareto front [46]. While classical techniques lack the means to find multiple Pareto optimal solutions efficiently, evolutionary algorithms (EAs) have been applied to these problems for over a decade and have shown good success [54].

Another important aspect present in many real-world problems is a dynamic environment which introduces changes in the problem that occur with time. These changes may

be in the form of modified objectives or constraints or even a completely new problem instance [119]. The techniques developed by evolutionary dynamic optimisation (EDO) try to solve problems with time dependent changes, which are referred to as dynamic optimisation problems (DOPs). These two characteristics can be combined to form what are called dynamic multi-objective optimisation problems (DMOPs) where the dynamic problem contains multiple objectives. In recent years EDO has seen a lot of development and it is an active area of research [118].

The remainder of this chapter is organised as follows: The next section describes the basic principles of the immune system along with introducing the germinal centre reaction in more detail. This is followed by a description of the models abstracted from the immune system which are often used as motivations for AIS. Section 2.4 starts with a brief introduction to multi-objective optimisation followed by a description of the set cover problem (SCP). Literature review on the work done for this problem is provided and the gap in literature is pointed out which will be dressed in this thesis. A description of the multi-objective d-dimensional knapsack problem (MOd-KP) follows next along with a review of work done for this problem. Section 2.5 begins with an introduction to dynamic optimisation and some techniques to solve dynamic optimisation problems. Memory techniques are described in more details along with a review of work done in the field. Some gaps are identified in literature concerning memory based methods and this is mentioned as work to be addressed in this thesis. In the same section a description of the single time-step dynamic SCP is provided and a review of work done on the problem is presented.

2.2 The immune system

The human immune system forms the inspiration of the class of meta-heuristic techniques known as artificial immune systems [46]. The immune system has several desirable properties combined together, which makes it a great inspiration for the design of randomised

search heuristics. These properties include diversity, robustness, and memory, and algorithms inspired by the immune system find application in a large number of domains such as machine learning, security, robotics, and optimisation [46]. The presence of these properties in isolation is not uncommon, in fact diversity can be seen in most of the population based approaches, robustness can be found in robust optimisation while memory can be seen in tabu search just to name a few. The uniqueness of the immune system is the fact that in no other system in the body all these properties exist simultaneously while the immune system possesses the properties mentioned above along with others all at the same time [46].

The immune system is the component of the body which deals with fighting infections caused by pathogens [113]. To do so the system needs to be able to perform a number of tasks effectively. It must be able to distinguish between self and non-self cells, followed by a response of some manner when an attack from a pathogen is detected and finally remember the characteristics of the pathogen for a better response in the future.

The immune system has a multilayer structure which performs these tasks efficiently. There are two kinds of immune responses to an invading pathogen, the innate immune response and the adaptive immune response. The innate response is non-specific and can be viewed as the first line of defence. It is formed by natural barriers like skin, mucous, tears and this response is similar in all individuals as it is static in nature and protects against a plethora of invaders. After an invasion is detected from a pathogen the adaptive immune response is triggered. As the name suggests, the cells of the immune system responsible for adaptive immunity (T and B cells) adapt themselves with the aim to detect and eliminate the pathogen. Another feature of adaptive immunity is to generate a kind of retention for the characteristics of the pathogen to provide a quick response in case of future invasions.

Memory forms a key part of the immune system. Over the course of their lifetime, organisms are infected by pathogens many times. In some cases when an organism has never faced the pathogen before the immune response is called a primary response while

in other cases the organism may face a pathogen which it has encountered before and the immune system's response is called a secondary response. The immune system employs a learning mechanism in order to better protect against future infections of pathogens which the organism has already encountered. If an infection occurs for the first time, the immune cells which prove most successful in fighting against it are saved in a pool of memory cells. It is this collection of memory cells which is used in case the same infection occurs in the future, however under some circumstances having memory can be detrimental [56]. In most cases due to repeated exposure to the same pathogen, the memory cells developed are capable of a very fast response and produce efficient antibodies. An example where having memory is not always useful is when antibodies produced due to the exposure to a strain of pathogen causes suppression of the creation of new antibodies for a different strain of the same pathogen. This phenomenon, known as *original antigenic sin*, can cause an organism to be more susceptible to the different mutated variant of the pathogen [56]. The pool of memory cells in any organism is a representation of all the different infections which have affected it. The number of immune cells in the body is dynamic in nature, as new memory cells get added to the pool after every disease and old memory cells may die due to old age and this process is highly regulated [46].

2.2.1 Theory of germinal centres and the germinal centre reaction

Effective and long lasting protection from antigens is provided by highly specific antibodies (Ab) which are produced during adaptive immunity in a process called the humoral immune response. The bone marrow is constantly responsible for producing a large number of B-cells which are created by random recombination of different gene segments. This leads to a repertoire of B cells which can interact with a plethora of antigens. An antigen is any substance that causes the immune system to produce antibodies. An invasion of an pathogen triggers the adaptive immune response by activating a few B cells from the repertoire which can bind with the antigen present on the surface of the pathogen. Some

of the activated B cells proliferate and produce low affinity early stage antibodies which provide an immediate protection for the organism, by forming plasmablasts and plasma cells. At the same time a small number of activated B cells form what is known as a *Germinal Centre* (Figure 2.1). The aim of these centres is to generate a more specific antibody for the infection in case the early stage antibodies are ineffective in completely eradicating the disease and for the purpose of creating memory cells which can provide an effective response to the same infection in case of a future attack from the antigen.

A GC is primarily composed of about 90% proliferating B cells and the rest being mostly antigen specific T cells. The GC can be essentially seen as a growing ball of actively dividing B cells surrounded by an ocean of resting B cell. These resting B cells form the surface of this ball which is called the mantle which marks the boundary of the GC. The GC grows in size during the course of the immune response, after which it begins to shrink and finally disappear when the pathogen has been eradicated.

Inside the GC, two regions can be distinguished, the light region and the dark region. The antigen is presented to the B cells in the light region, upon which the cells enter the dark region. There is continuous proliferation and mutation of these cells in the dark region. The mutated clones move out to the light region in order to be tested for selection. The ability of these cells to bind with the antigen (Ag) is tested in the light region. Cells which are good enough to have some level of binding move back to the dark region for more rounds of the proliferation and mutation while the ones which cannot bind with the antigen suffer cell death via a process known as apoptosis. This is a kind of cycle where cells mutate and proliferate in the dark region and are evaluated in the light region, and is called the *cyclical re-entry model* [102]. This model repeatedly selects cells which have a better binding with the antigen. This is called affinity maturation and is able to evolve B cells to a stage where they are capable of producing antibodies (Ab) which can provide a much better directed response to the antigen. At this stage by a process of class switching the B cells are converted into antibody producing cells known as plasma cells and memory cells. The plasma cells are responsible for providing a more directed

response to antigens which survive the initial low affinity antibodies while the memory cells enter the immune cell repertoire to provide long term protection against a future attack of the same antigen. The number of GCs is dynamic in nature and as the centres which are unable to create B cells that can bind strongly with the antigen, die.

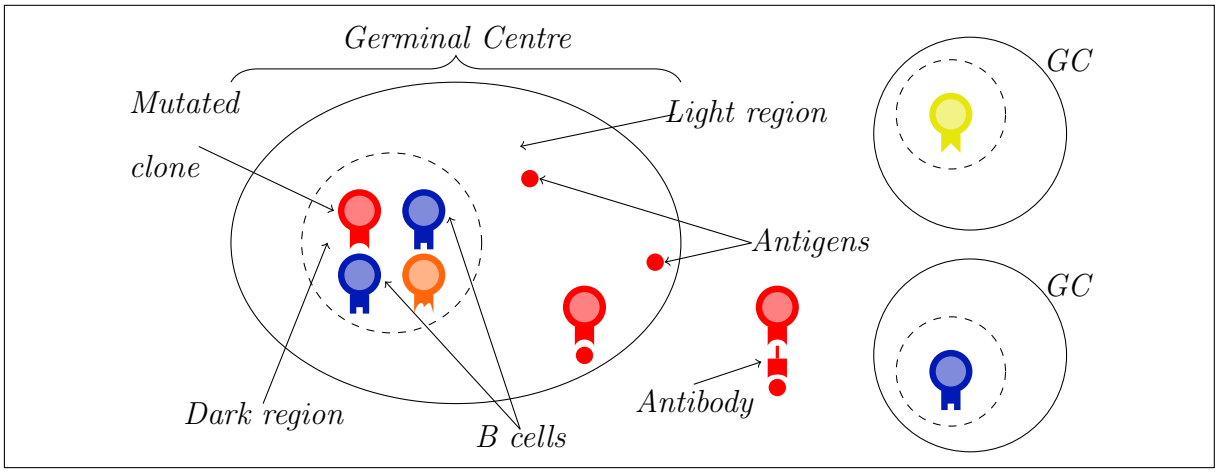


Figure 2.1: An example diagram of the GC reaction showing multiple GCs. A single GC is enlarged to show the light and dark regions. Based on description and figures of GC from [113].

As the whole process of affinity maturation of B cells comprises several iterative steps of proliferation, mutation and selection, it can be seen as an example of Darwinian selection [163], which forms the basis of evolutionary algorithms. Understanding the mechanisms and processes of the GC is an active area of research and the authors of [163] shed light on the selection process inside the GC which was previously not clearly understood. It was previously known that B cell activation depends on the affinity of the cell with the antigen but it was not known how a directional selection pressure is maintained as the affinity of the B cells increases. Another aspect not very well understood is whether GC interact with each other. It was previously known that there is some migration of B cells between GC, but these are mostly naive B cells. The authors put forward an antibody dependant selection process which explains both these unknown phenomenon. In their antibody dependant selection the authors propose that during the iterative steps of the affinity maturation, B cells compete not only with each other but also with antibodies produced from the plasma cells which have left the B cell. These antibodies re-enter the

GC and directly compete with B cells to bind with the antigens. This leads to a kind of systemic threshold which exists between GC as B cells across GC compete with the antibodies and only those B cells which are better than the current antibodies are selected for further rounds of the cyclic re-entry.

The features of the GC reaction that make it a suitable candidate for optimisation in general and multi-objective and dynamic optimisation in particular are the following. As mentioned above the continuous process of improvement of B cells in the GCs is an example of Darwinian selection which has proven itself in the field of optimisation by being the key metaphor for the design of evolutionary algorithms. It also possesses the two key features that any multi-objective optimisation algorithm must posses: a diverse population and continuous improvement towards the Pareto front. The diversity can be seen in the diverse B cells evolving in different B cells in parallel and the continuous iterations of proliferation, mutation and selection in the GCs ensure that the diverse population of B cells move towards an optimal front. The GC reaction is also responsible for the creation of memory cells which are the source for learning in the immune system, a feature very important for dynamic optimisation where learning plays a key role. Finally apart from the key features required for multi-objective and dynamic optimisation the GC reaction also has an added advantage. The whole process is dynamic, automated and self regulated as in the number of B cells in the GC, the number of GCs present are all controlled by the reaction itself. A feature that can be very useful for designing an algorithm which could regulate its own population or islands therefore reducing the number of parameters to be tuned.

An illustrative diagram of the GC is provided in Figure 2.1 . This life cycle of GC which leads to the development *Ab* that fight with the antigens, is called the GC reaction and can be summarised as follows: when an antigen invades the body the number of GC begin to grow. The B cells inside the GC must produce *Ab* which are good enough to eradicate the antigen. This is achieved by repetitive proliferation, mutation and selection of the B cells inside the dark region of the GC. A periodic transfer of *Ab* takes place

between the different GCs and these steps continue till *Ab* are strong enough to bind with the invading antigen and remove them. At these final stages of the reaction a decline in the number of GCs is seen.

2.3 Theoretical AIS models of immunity

The internal working mechanisms of the immune system are not yet completely understood [42], therefore several theories have been developed to explain and model these mechanisms which are then used to design AIS. These theories have been constructed by abstracting key components of the immune system into isolated concepts that form the basis for several artificial immune systems. According to [46], for a system to be classified as an AIS it must incorporate at least one characteristic feature/model from the immune system and must be aimed towards problem solving. Some popular models used for generating AIS are clonal selection theory, negative selection, danger theory, immune network theory [43].

2.3.1 Clonal selection theory

This concept was proposed in [25, 23, 24] which tries to explain the ideas of adaptive immunity when the immune cells are presented with the antigen of the invading pathogen. Due to the diverseness of the immune cells only a small number of them are able to recognise the pathogen. Once a cell recognises the pathogen, it proliferates (i.e. it forms clones of itself) and the clones are mutated. [23, 138] state some key features of the clonal selection principle. Clones that react with the body need to be eliminated while the ones which can interact with the pathogen proliferate. Each cell can bind to a certain kind of pattern and this information is passed to its clones. Finally random changes are introduced in the clones by mutation to introduce diversity.

The way the cloning and mutation takes place is related to the affinity, which is the ability of the B cell to bind with the pathogen. The number of clones produced is

directly proportional to the affinity while the mutation rate is inversely proportional to the affinity [47]. This theory forms the inspiration for clonal selection algorithms due to the dynamic nature of the adaptive immunity.

Algorithms inspired by clonal selection

First published as the Clonal selection algorithm (CSA) [51], this algorithm is one of the most widely known algorithms in the field of AIS. It was used for pattern matching and optimization and was renamed to CLONALG in later publications [47, 49]. A brief description of the algorithm is as follows: antibodies which form the candidate solutions are selected based on their affinity by evaluating the fitness function. These selected antibodies are cloned and the number of clones produced is proportional to the affinity. Mutation is applied to these clones at a rate inversely proportional to the affinity. This forms a clone set which now competes with the parents to enter the next generation, and finally low affinity antibodies are replaced by randomly generated new antibodies. The algorithm has 8 parameters which the user need to define. The outline of the algorithms is shown in 1.

Algorithm 1 Outline of CLONALG based on [51]

while Stopping condition net met **do**
  Generate a set of candidate solutions P comprising of a subset of memory solutions.
  Based on affinity, select n best solutions from P
  Create clones of these solutions. Higher the affinity of the solutions the more clones it makes.
  Perform hypermutation on the clones at a rate proportional to the affinity.
  Select a subset of these clones to be used as memory
  Replace some solutions by random solutions to introduce diversity.
end while

In [49], CLONALG was applied to solve a continuous multi-modal optimization problem with several local and one global optima, using a binary encoding and Hamming shape space. The results obtained showed that most of the optima were reached within 100 generations, though comparison with any other heuristic was not performed. They also tested their approach on the travelling salesman problem using an integer valued

vector encoding. In [47] the authors extended the experiments by including three new continuous multi-modal optimisation problems and compared the immune algorithm with a evolutionary algorithm with fitness sharing. They showed that CLONALG was able to find all the peaks of the functions and used lesser non-peak individuals than the fitness sharing method.

An analysis between evolutionary strategies (ES) and CLONALG was performed in [149] on dynamic function optimization using four functions with different dimensions. It was shown that CLONALG produced better results faster than ES for low dimensions but for higher dimensions ES always performs better if evaluations used by ES for parameter tuning were considered. If equal number of evaluations used by ES for tuning its parameters were allocated for the CLONALG it always outperformed ES. In [69] the author comments that the CLONALG algorithm had reached a standstill state and thus made variations to the CLONALG to make it better and introduced the adaptive clonal selection (ACS). The only variation in this model is that the mutation rate, number of selected antibodies and number of clones produced is tuned automatically by the system and is not a fixed value. CLONALG1 and CLONALG2 were two modified versions proposed in [36] by incorporating elitist mechanisms. In [151] a CSA like algorithm was applied to the power filter design problem from electronics and compared it with a genetic algorithm. This work had a static clone production rate as compared to the dynamic rate in CLONALG.

A parallel version of the CLONALG algorithm was presented in [152] which was tested on the binary character recognition problem. Work in [75, 74] incorporated ideas from evolution by introducing local learning and Baldwinian learning into CLONALG and compared their CSA variants a number of evolutionary algorithms using a test bed of 20 continuous functions. Some different mutation strategies were applied in [103] to create the fast clonal selection where they introduce Gaussian mutation and chaos mutation and tested their approach on 23 function optimisation problems along with the machine loading problem. These mutations are respectively applied to good and not so good

solutions to facilitate both moving to a global optima as well as moving out of local optima. A comparison was performed with CLONALG and simple immune algorithm (SIA) and it was shown that their algorithm called fast clonal algorithm (FSA) could perform better results than SIA and CLONALG with respect to solution quality and convergence speed.

Another popular algorithm inspired by the clonal selection principle is the B-cell algorithm (BCA) which was introduced in [101] for solving continuous function optimization. The paper introduced a new mutation operator called contiguous somatic hypermutation, citing [105, 128] as evidence for this technique. The general outline of the algorithm is as follows: a pool of B cells, where each cell represents a candidate solution is encoded in a binary encoding. Affinity of each solution is calculated based on their fitness. Each solution is then cloned and put in a clonal pool, followed by randomly selecting a clone and mutating it. This is followed by applying the somatic hypermutation to every clone and then evaluating their affinity. A clone replaces its parent if it has higher affinity than the parent. Hypermutation is performed be selecting contiguous regions of solutions and mutating them. The pseudo-code for the algorithm is shown in 2.

Algorithm 2 Outline of BCA based on [101]

```
Create an initial random population of solutions P
while Stopping condition net met do
  Evaluate the affinity of all solutions in P
  for for each solution do
    Clone solution and place it in clonal pool C
    Randomly select a clone from C and replace it with a random solution.
    to each clone in C apply contiguous somatic hypermutation.
    Calculate affinity of clone.
    Replace parent with clone if affinity of clone higher than parent
  end for
end while
```

The authors of [101] compared BCA with a hybrid genetic algorithm using 12 test functions and showed that both algorithms could find the optima for most of the functions but BCA used less evaluations. They claimed that it could be due to mutation operator being particularly suited for this representation of the problems and that they could

not at that stage give a definite answer to this question. In a later comparative study [141] it was shown that success of BCA against a hybrid genetic algorithm and opt-IA, was due to the mutation operator. In [140] a modified variant of the BCA was introduced by including *megamutations* and 4 types of the modified BCA were compared using diophantine equations as the test problem. They showed that with the use of diversity in their method they could achieve better results than the simple BCA.

The opt-IA algorithm which is also known by several names: Simple Immune algorithm (SIA), cloning information gain algorithm or optimization immune algorithm was first proposed in [39, 38] as the immune algorithm. The algorithm consists of B cells which produce clones every generation with inclusion of a new hypermutation operator. A mutation potential is calculated which determines how many points in the string will be mutated. Different versions of such mutations exist such as fitness proportional, inversely fitness proportional, hypermacromutaion and constant mutation rate. Another technique incorporated in this algorithm is the diversity mechanism of ageing which can be static or stochastic in nature. This keeps a track on the number of iterations an individual has been alive. If a solution exceeds a threshold for the ageing then it is replaced by a random new solution.

Algorithm 3 Outline of opt-IA based on [37]

```
Create an initial random population of solutions P
while Stopping condition net met do
  Evaluate the affinity of all solutions in P
  for for each solution do
    Clone solution and place it in clonal pool C
    Mutate the clone using hypermutation according to condition H.
    Evaluate the affinity of the clone.
    Mutate the clone using hypermacromutation according to condition M.
    Calculate affinity of clone.
    Eliminate clones which have survived for several iterations.
    Update P by selecting solutions from clones and parents
  end for
end while
```

In [37] the algorithm was tested on a range of problems including ONE-MAX, TRAP and 23 numerical function optimisation problem along with the protein structure pre-

diction problems. The algorithm was compared with two versions of CLONALG and it was observed that setting mutation potential correctly was most important aspect for the opt-AI. Once this was done opt-IA was shown to perform better than CLONALG for all problems tested.

opt-IA along with its versions has been applied to 2-dimensional HP-bead model (2DHP) protein folding problem [35], multiple sequence DNA alignment problem [34], minimum hitting set problem and the satisfiability problem [40]. A parallel version called Par-IA is also known [41] which is a master-slave version of opt-IA and was applied to numerical functional optimisation. A more detailed survey of applications and algorithms inspired by the clonal selection principle can be found in [22, 144].

2.3.2 Negative selection theory

The negative selection theory describes the process by which the self (body cells) are distinguished from the non-self (everything else). Based on the process of T cell generation in the body, those cells which can bind with the self are eliminated and the rest are allowed to join the pool of immune cells. These T cells are capable of recognizing cells other than the self cells. This can be used for modelling a two class classification problem. The general idea of the system is to construct a set of detectors. Based on negative selection, the detectors are constructed randomly and matched to self, the subset that match are eliminated. The set that remains can be used to monitor changes in a system and this concept was used for intrusion detection.

A diverse set of negative selection algorithms exists and has mostly been applied to areas in anomaly detection, classification and security. An interested reader is directed to [42] for detailed concepts and applications.

2.3.3 Immune network theory

This theory was introduced in [91] and tries to explain how the immune memory works by proposing the existence of an idiotypic network which is composed of antibodies and anti-antibodies that are capable of recognizing each other. The B and T cells interact with each other to either stimulate or suppress the network. Therefore all the key properties of the system such as memory, learning and tolerance are global. The key feature which distinguishes this model from others is the fact that the immune cells are able to recognise each other and there is a form of communication in the network even without the presence of an external pathogen. When a pathogen enters the system, the state of the system changes which causes the system to react to the pathogen. A comparison of different immune network models can be found in [68].

Algorithms inspired by immune network theory

Developed in [50], the aiNet was first proposed for network learning but later a version for multi-modal function optimization was proposed called opt-aiNet [46]. The authors described the algorithm and empirically compared it with CLONALG. The algorithm described was similar to clonal selection methods with addition of some network interaction mechanisms.

The steps followed in the algorithm are as follows: the algorithm starts by random initialization of network cells followed by clonal selection where each parent cell produces a number of clones which are mutated inversely proportional to their fitness. This is continued till the network reaches a stable state, which is checked by calculating the average error of the population. The next phase is the network suppression where affinities of all cells are calculated, and using an elitist strategy cells with low affinities are removed. Random new cells are introduced into the system to compensate for the loss of cells. Therefore a dynamic population is maintained. This model only incorporated network suppression but not network stimulation and the outline of the algorithm is shown in 4. The key features of the algorithm as proposed by the authors are automatic determina-

tion of population size, a combination of exploration and exploitation, a set convergence criterion and the capability to locate and remember the optima. The results published were a comparison of opt-aiNet versus the CLONALG on three multi-modal functions with different number of optima and plateau. The opt-ainet was able to find more peaks in the multi-modal functions but required more fitness evaluations to find the first optima, compared to CLONALG. The justification given was that the dynamic population kept increasing in case of opt-aiNet but this was not the case for CLONALG implying that a dynamic population can lead to utilisation of more fitness evaluations in cases when the population keeps growing.

Algorithm 4 Outline of aiNet based on [50]

while Stopping condition net met **do**
  for each antigen **do**
    Calculate affinity with all antibodies based on some distance measure.
    Select a subset of the high affinity antibodies
    Clone the selected antibodies proportionally to the affinity. Higher the affinity higher the clones for the antibody.
    Mutate the clones with a rate inversely proportional to the affinity of the parent antibody.
    Calculate affinity among all antigens and all mutated clones.
    from the pool of clones, select a percentage of clones to put in the pool for clonal memory.
    Eliminate memory clones whose affinity is greater than a set threshold.
    Calculate the affinity between all the memory clones.
    eliminate the clones whose affinity is less than a set threshold.
    Concatenate memory and antibodies into one pool.
  end for
  Calculate affinity between all antibodies
  Eliminate memory clones whose affinity is greater than a set threshold.
  Concatenate memory and antibodies into one pool.
end while

A multi-objective version of the algorithm was proposed in [65], called vector artificial immune system (VAIS). The algorithm follows the steps of the opt-aiNet identically except for the fitness evaluation part where now along with objective function, pareto dominance was also utilized by combining techniques from SPEA2 and NSGA-II. A simple strength approach is devised where every non-dominated solution is assigned the strength (SPEA2)

if it is non-dominating or rank (NSGA-II) if it is dominated by others. Also a memory is provided which serves to copy the non dominated solutions and affinity based suppression is done there. The algorithm was compared with NSGA II on three continuous functions. The results showed that VAIS obtained similar or better results than NSGA II on 3 continuous multi-objective test functions based on inverse generational distance, spacing and error ratio metrics of evaluation. The authors claimed that AIS inherently have characteristics similar to multi-objective evolutionary algorithms.

2.3.4 Danger theory

Danger theory [111, 2] tries to provide an alternative to the self non-self recognition, as not all non-self antigens are thought to be dangerous. It proposes that the self is everything that is harmless to the body which can include non-self antigens to which the body has developed tolerance. In this model an immune response is triggered by the presence of alarm or danger signals such as cell death or tissue damage. The dendritic cells which are part of the innate immune system play a key role in this theory by mediating between adaptive and innate immunity. Despite having several advantages, the authors of [111] also note certain limitations. They state that the exact nature of the danger signal is unclear. It is unclear how to distinguish this danger and in certain special situations a response to danger is not required (grafts, minor cuts).

An early conceptual work in danger theory was done by [3] where the authors pointed out some analogies of the danger theory to AIS as well as citing anomaly detection and data mining as some relevant application areas. The danger theory algorithm (DTAL) was presented in [123] and applied to robot soccer for the purpose of finding a strategy for the goalkeeper and was shown to perform well.

One of the most important and widely known algorithms inspired from the danger theory is the dendritic cell algorithm (DCA) conceptualised in [77]. DCA was developed as a part of the danger project [4]. A formal description of the algorithm was presented in [78] where the anomaly detection capabilities of the algorithm were explored based on

network intrusion detection. In [80] the authors compared different implementations of DCA with the negative selection algorithm and C4.5 decision tree algorithm on a standard data set for anomaly detection. A deterministic variant of the DCA [76] was proposed to limit the number of stochastic elements in the original implementation. Algorithms inspired by the danger theory have been applied to several problems in anomaly detection, security, classification, data mining. A survey of these applications and others can be found in [44].

2.4 Multi-objective optimisation

Multi-objective optimisation (MOO) deals with problems which have several objectives that often compete against each other. Majority of the real world optimisation problems can be modelled as MOOPs [54]. Instead of having a single optimal solution, these problems have a set of optimal solutions which are known as Pareto-optimal solutions. It is desirable to obtain as many of these solutions as possible since there is trade-off between the objectives of every solution. Having a collection of solutions allows the end user to make an informed decision when selecting which one to use [54]. In contrast to classical techniques which convert the multiple objective into a single objective, EAs have been more successful for MOO as classical techniques lack means to find multiple optimal solutions in an efficient way. The classical techniques often employ methods to scalarise the multiple objectives into a single objective and require several algorithm runs to obtain different Pareto-optimal solutions, while EAs overcome this problem by having a population of solutions. An overview of multi-objective optimisation is provided by [171, 54]. A survey of artificial immune systems used to solve multi-objective optimisation can be found in [26, 64].

The key difference between MOO and single objective problems is that solutions in MOO can no longer be simply compared using a single objective as in the case of single objective problems instead the comparison needs to be done with respect to all objectives.

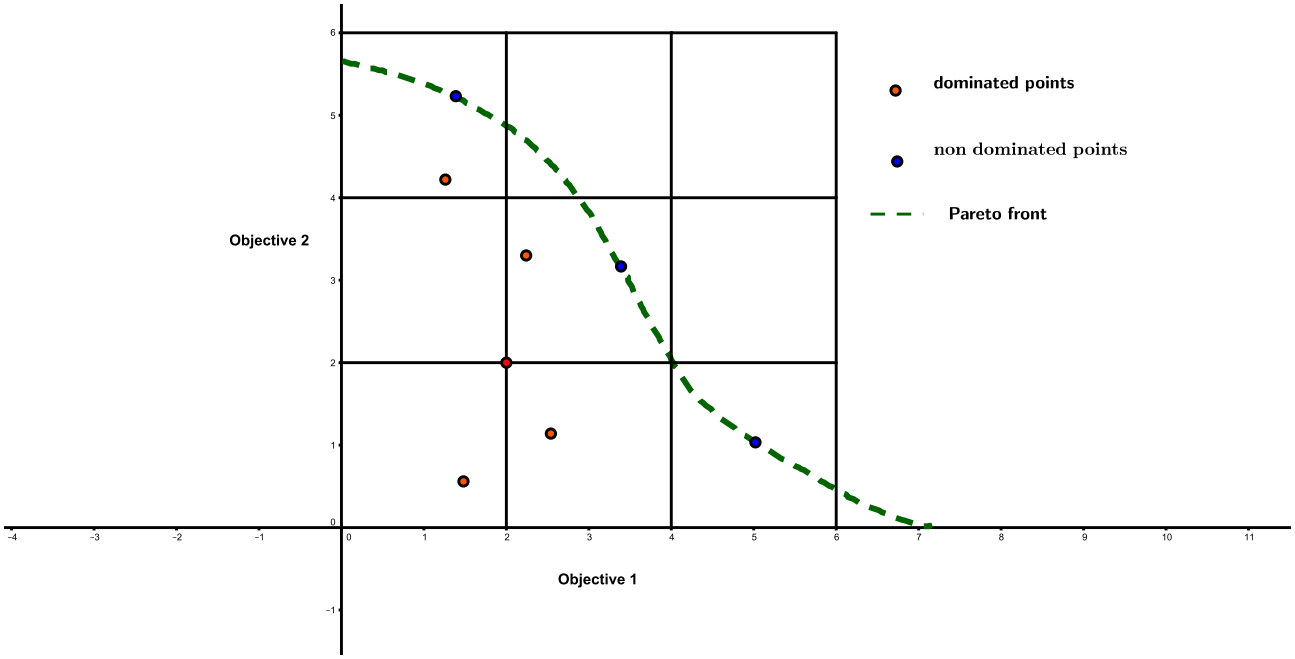


Figure 2.2: Image showing the Pareto front with dominated and non-dominated solutions. Red points show dominated solutions, while blue points show non-dominated solutions. Maximisation of objectives is considered.

The potential solutions to such problems are compared using the dominance relation. Let c and d denote two solutions with m objectives and the fitness of the ith objective be f_i. Assuming maximisation of all objectives, then w.l.o.g. solution c is said to dominate d iff

$$\begin{aligned} &\forall i \in \{1, 2, \cdots, m\} : f_i(c) \geq f_i(d) \quad \wedge \\ &\exists j \in \{1, 2, \cdots, m\} : f_j(c) > f_j(d) \end{aligned} \tag{2.1}$$

In the case of maximisation of objectives, the dominance of solutions by solutions in the Pareto front can be seen in Figure 2.2. Algorithms which deal with MOOPs maintain a set of non-dominating solutions during their run. The best set for a multi-objective problem is a set of non-dominated solutions that contains only Pareto-optimal solutions, called the Pareto-optimal set. A solution is a called Pareto-optimal solution if no other solution dominates it [54].

2.4.1 Set cover problem

The Set Cover Problem (SCP) is an important problem in computer science with many real-world applications like crew and railways scheduling [1]. The SCP is one of the oldest and most studied NP hard problems [135]. It can formally be introduced as:

Definition 1 *Let the set of m items $U := \{u_1, ..., u_m\}$ denote the universe and let $S := \{s_1, ..., s_n\}$ such that $s_i \subseteq U$ for $1 \leq i \leq n$ and $\bigcup_{i=1}^{n} s_i = U$. The uni-cost set cover problem can be defined as finding a selection $I \subseteq \{1, 2, ..., n\}$ such that $\bigcup_{k \in I} s_k = U$ with minimum $|I|$.*

In order to extend this definition for a dynamic setting later, a more widely used standard representation of the SCP, which involves matrices, is introduced. If we view the set U as rows and the set S as the columns in a zero-one matrix then the goal of the static SCP can be stated as to cover the rows of the matrix using the least number of columns. To formalise this notion [10] provides a definition of SCP based on these matrices:

Definition 2 *Let $A = (a_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}$ with $a_{ij} \in \{0, 1\}$ be a matrix with m rows and n columns where a column j is said to cover a row i if $a_{ij} = 1$. Let $X = x_1 \dots x_n \in \{0, 1\}^n$ denote a solution where $x_j = 1$ if column j is in the solution and 0 otherwise. Minimise $\sum_{j=1}^{n} x_j$ subject to $\sum_{j=1}^{n} a_{ij} x_j \geq 1$ for all $i = 1, \dots, m$.*

SCP is a constrained single objective problem which can be converted to a multi-objective problem by using the constraint as the secondary objective [66]. Let a solution be represented as $X = x_1 x_2 \dots x_n$, with $x_i \in \{0, 1\}$ for $1 \leq i \leq n$, where $x_i = 1$ if set s_i is in the solution and 0 otherwise. If the number of subsets selected in X is denoted by N and the number of elements left uncovered in U is denoted by C, then the fitness function for this multi-objective formulation of SCP can be defined as $F = \langle C, N \rangle$. According to [115] this multi-objective formulation can be more efficient than the single objective version in finding the optimal solution, when used by an EA. This is because it can make

the algorithm behave like Chvatal's Greedy algorithm, which is one of the most popular problem specific polynomial time approximation method for SCP [29].

An illustrative example of the SCP is provided in the following paragraph and it will be used for the rest of the thesis. Let the set $U = \{1, 2, 3, 4, 5, 6\}$ and the family of subsets be $S = \{\{4\}, \{6\}, \{1, 2\}, \{3, 4\}, \{5, 6\}, \{1, 2, 3\}, \{4, 5, 6\}, \{1, 2, 4, 5\}\}$. A potential solution to this problem can be a collection of following subsets $x = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$. This however is the not the best solution to the problem as it can be quickly seen that this set contains 3 subsets and a better solution is possible. A different solution $y = \{\{1, 2, 3\}, \{4, 5, 6\}\}$, which uses only 2 subsets, happens to be the best solution to this small toy instance of SCP.

2.4.2 Relevant work on SCP

Several techniques have been developed to tackle the SCP. The most notable methods in literature are based on linear programming and Lagrangian relaxations, heuristic approaches and exact methods [7, 12, 29]. A survey of some dated techniques which have inspired newer methods to solve the SCP can be found in [27]. Some exact methods to solve SCP are presented in [7, 12] but they are unable to scale with larger instances of the problem where approximation and heuristic approaches are more suitable. The Greedy algorithm from [29] is one of the best approximation algorithms for SCP which finds solutions by iteratively selecting sets which covers the most remaining uncovered items. [79] compared a variant of the greedy algorithm with eight other algorithms which included variants of greedy, randomised algorithms and neural networks. A randomised variant of the greedy algorithm was shown to produce the best solutions with respect to cost of the final solution.

A tabu search based approach was developed in [120] and was used to solve a class of SCP problems defined from Steiner triple systems. Ant-colony based approaches to solve the SCP have been used in [108, 124, 126, 125, 33]. These approaches along with GRASP [61] construct new solutions from scratch at every iteration by employing some

random behaviour in the construction process. The solutions obtained at every iteration are feasible but the solutions tend to contain redundant sets and all of these approaches need to rely on dedicated redundancy removal operators after every iteration. Several variants of genetic algorithms have been employed to solve the SCP. These algorithms need to deal with the issue of infeasibility as well as redundancy in their solutions. Two main techniques have been employed to solve the infeasibility issue in genetic algorithms, namely penalty functions and repair. The repair mechanism converts an infeasible solution to a feasible solution by incorporating sets which cover the uncovered elements. [13, 137, 121, 9] employ repair mechanism in their GA implementations and to mitigate the problem of redundancy they use redundancy removal procedures. Penalty based approaches reduce the fitness of infeasible solutions making them less favourable than feasible solutions and driving the search towards feasible regions. [97] reduced the fitness of all infeasible solutions by an equal and high enough penalty ensuring that the fitness of any feasible solution is higher than them. The fitness of the infeasible solutions in this case does not reflect the degree of infeasibility and these solutions can no longer be compared using fitness. The genetic algorithm of [9] uses a penalty proportional to the cost it takes to cover all uncovered elements, while [150] use a penalty proportional to the number of elements not covered. An issue with all of these methods is that after few generations the chances of infeasible solutions surviving in the population are very low and therefore the infeasible region is not searched effectively.

Local search algorithms have also been utilised to solve SCP. Due to the feasibility constraint it is hard to design pure local search algorithms. [114] have proposed a local search method incorporating a tabu search like procedure with a restricted neighbourhood by using a upper bound parameter for the uni-cost version of SCP. Other local search techniques are often enhanced with methods like linear programming and Lagrangian relaxation [63, 155] and sub-gradient optimisation [84]. An AIS used to solve the SCP has been developed by [139] based on the CLONALG. They compared their AIS-SCP with the greedy algorithm and obtained better solutions. However their algorithm involves

several problem specific operators making it hard to qualify as a true meta-heuristic.

All the techniques mentioned above utilise the single objective formulation of SCP and the only known work with the multi-objective formulation is the work done in [110, 130]. Like the theoretical analysis in [115], the work in [110] presents another theoretical analysis where the authors propose two island models with different topology of islands and prove that these models find good approximation for the multi-objective formulation of the SCP. The homogeneous islands model which comprises an archipelago of global simple evolutionary multi-objective optimisers is called the parallel GSEMO (PGSEMO) throughout this thesis. They provide upper bounds on the running time until an H_m approximation for the set cover problem is reached by specifying the number of islands in the archipelago which are proportional to the probability of communication between islands and the size of the problem.

More recently [130] have introduced the constrained multi-objective artificial immune system (MAIS) based on the clonal selection principle, used to handle a real world application of find core collection for germplasms as a strategy for conservation of diversity. The problem of finding core collections is modelled as a SCP with three objectives specific to the problem. They compared their algorithm with NSGA-II on a single instance of their problem of size (642×55) and showed that their MAIS performed better than NSGA-II on this instance based on hypervolume, extent, attainment and spacing metrics of evaluation. Though the authors mention that parameter tuning was performed based on preliminary data, no results about the sensitivity of the algorithm towards these parameters were provided.

Even though the advantages of the multi-objective formulation of SCP are stated in [115] only one theoretical and one empirical work using this formulation is known to us at the time of this research. This presents as an opportunity to study the SCP in a multi-objective scenario as little to no work exist in this area, even when the potential advantages have been shown by theoretical studies. The empirical study by [130] utilises a multi-objective formulation that is extremely specific to the application area of germplasm

selection, which is different from the standard multi-objective formulation. Their work is also limited to a single problem instance which raises the question if the results are applicable to other instances. following their approach and conducting comparative analysis with their work would seriously limit the applicability of the results to the wider problem formulation of SCP and also there is the inherent problem of lack of instances to study. The work in [130] seems the most appropriate to consider for a comparative analysis with our new proposed approach, after identifying the literature gap. This is because, though their analysis is theoretical, it provides performance guarantees for the solution quality that can be achieved by their models. Therefore knowing the guarantee that their model will provide, we can make comparison with implementations of their model. This analysis will add to the scarce literature for the multi-objective formulation of SCP.

2.4.3 Multi-objective d-dimensional knapsack problem

The knapsack problem is a widely studied NP-hard combinatorial optimization problem [170] with real world applications like capital budgeting and resource allocation [132]. The single objective knapsack problem consists of a set of items which have associated weights and profits, and a knapsack which has a fixed capacity. The goal is to find the set of items which can be packed in the knapsack giving the maximum profit without exceeding the capacity of the knapsack. The three main variants of the problem are 0-1, bounded and unbounded knapsack problems. In the 0-1 case an item can either be in or outside the knapsack, while in the bounded case a finite number of copies of the same item exist which can be placed in the knapsack and finally in the unbounded case an infinite supply of the items is available. By introducing multiple knapsacks, this single objective problem can be extended to a multi-objective version. Each knapsack has its own capacity and the items have different profits and weights associated with each knapsack. Another way to look at the multi-objective knapsack problem is to think of the knapsack as having multiple dimensions. For example, let the knapsack have a length and breadth dimension which forms its capacity, therefore it can be seen to have a capacity with respect to the

length dimension and a capacity with respect to breadth. Each item also have a length and breadth and a profit associated with both the length and breadth. The problem now is just a simple extension of the single objective knapsack where the capacity, weights and profits are associated with length and breadth. The goal of the problem remains the same, to pack the items in the knapsack to maximise the profit in both dimensions without exceeding the capacities of the dimensions. A practical example of this problem is packet scheduling for wireless networks with relay nodes [32].

Considering the multi-objective version of the 0-1 knapsack problem, the solutions to the multi-objective d-dimensional knapsack problem (MOd-KP) can be encoded as bit strings of length a, where a is the total number of items available. A 1 indicates the presence of an item while 0 indicates the absence of the item. It should be noted that 1 indicates that an item is present in all the knapsacks and a 0 indicates the absence of the item from all knapsacks. Every possible bit string combination does not represent a valid solution due to the constraint imposed by the knapsack capacity therefore techniques such as penalty or repair must be employed for the the task of constraint handling. A more formal definition of the multi-objective knapsack problem can be stated as:

Definition 3 *Let a denote the number of items and b the number of knapsacks. Let $p_{i,j}$ and $w_{i,j}$ be the profits and weights of item i with respect to knapsack j, respectively, and c_j the capacity of knapsack j. Let a solution be represented as $x = (x_1, x_2, \cdots, x_a)$ with $x_i \in \{0, 1\}$. The objective is to maximize*

$$f(x) = (f_1(x), f_2(x), \cdots, f_b(x)) \text{ where } f_j(x) = \Sigma_{j=1}^{a} p_{i,j} \cdot x_i$$

$$\text{subject to} \qquad \Sigma_{j=1}^{b} w_{i,j} \cdot x_j < c_j; \forall i \in \{1, 2, 3, ...b\}.$$

Repair procedures for MOd-KP

MOd-KP is a constrained multi-objective problem where the constraint is the capacity of the knapsacks. Not all possible bit string combinations represent feasible solutions and

some constraint handling mechanism must be used. Repair techniques can be applied in order to transform infeasible solutions to feasible ones or penalty approaches can be used to reduce fitness of infeasible solutions. In [170], the authors used the maximum profit by weight ratio (greedy repair) method to repair invalid solutions. For an item i, the maximum profit by weight ratio is given by:

$$q_i = \max(p_{i,j}/w_{i,j}), \quad \forall j \in (1, 2, ..., n) \tag{2.2}$$

An item with the lowest maximum profit by weight ratio is removed first, and they are removed iteratively in an increasing order of the ratio, until a feasible solution is obtained.

Another repair heuristic was introduced in [90] where the weighted profit by weight ratio (weighted scalar repair), is used to find the order of removal of items. In this approach the items are sorted based on:

$$q_i = \Big(\sum_{j=1}^{n} \lambda_i p_{i,j}\Big) / \Big(\sum_{j=1}^{n} w_{i,j}\Big), \quad \forall j \in (1, 2, ..., n) \tag{2.3}$$

where the λ_i are the scalar coefficients of the linear utility function used for scalarizing the multi-objective fitness vector in the Multi-objective genetic local search (MOGLS) [90] algorithm. These coefficients are generated randomly at each generation of MOGLS and are used for selection as well as repair. The generation procedure for normalized weight vectors is given in Algorithm 5.

Algorithm 5 Algorithm for generation of normalized weight vectors. rand returns a random number between 0 and 1. [90]

$\lambda_1 = 1 - \sqrt[n-1]{rand()}$
...
$\lambda_j = (1 - \sum_{i=1}^{j-1} \lambda_i) \cdot (1 - \sqrt[j-1]{rand()})$
...
$\lambda_n = 1 - \sum_{i=1}^{n-1} \lambda_i$

2.4.4 Relevant work on MOd-KP

Fully polynomial time approximation schemes (FPTAs) have been proposed for the multi-objective 1-dimensional knapsack problem by [60, 8]. [60] have also developed a polynomial time approximation scheme for the MOd-KP but these approximation schemes are limited with the number of objectives and the size of the problem instances which can be solved by them [58, 59].

Due to the limitations of the approximation techniques, meta-heuristic approaches have been increasingly used to solve this problem. One of the earliest work to solve MOd-KP by utilising EAs was proposed in [173] where the strength Pareto evolutionary algorithm (SPEA) algorithm was introduced and compared against 4 other multi-objective evolutionary algorithms. SPEA was shown to perform better than its competitors with respect to the spread of the achieved solutions as well as the coverage metric. This experiment was extended by [104] to incorporate the memetic algorithm called M-PAES which hybridised local search with recombination schemes. It was conjectured that the memetic algorithm was successful in solving the problem due to the convexity in the search space. SPEA2 was introduced in [169] where the authors improved the SPEA algorithm by incorporating changes in the fitness assignment, archive update and density estimation and showed that it could perform better than the other popular MOEAs with respect to the distribution of solutions. It was also shown that SPEA2 obtained better hypervolume that the other algorithms on the problem with 2 objectives, while as the objectives increase NSGA-II catches up and beats SPEA2. The multi-objective genetic local search (MOGLS) was developed in [90] and was shown to generate better results than other MOEAs of the time using the coverage and R performance metrics. The greedy adaptive search procedure (GRASP) was introduced in [148] which was shown to outperform [90] as well as SPEA2 with respect to the contribution to Pareto front as well as generational distance metric. The comparison performed between these algorithms was not completely unbiased as GRASP and MOGLS both utilize a better repair heuristic proposed in [90] while SPEA2 used a different repair procedure therefore it is hard to say

if GRASP will truly outperform MOGLS or SPEA2 if the same repair technique was utilised in all of them.

The MOGLS and M-PAES algorithms were compared in [87] and it was shown that the diversity of solutions obtained by MOGLS was better than M-PAES. In the paper [85] the authors compared the effect of random repair, greedy repair and weighted scalar repair approaches on the performance of four multi-objective evolutionary algorithms and showed that the choice of repair procedure greatly affects the outcome of the algorithms. In [86] the authors proposed a memetic algorithm and compare it with popular MOEAs of the time and show that their algorithm is efficient than others in terms of CPU time but does not always meet the solution quality of the others. The authors of [88] compare Baldwinian and Lamarckian implementations for the greedy repair strategies on the MOd-KP using NSGA-II. Their work was limited to a single instance with 2 objectives for which the Pareto front was known and they showed the superiority of Baldwinian over Lamarckian repair though they could not explain the reason behind the result.

In their work [99, 100, 98] the authors have proposed a hybrid meta-heuristic HEMH and its improved variant HEMH2 and compared it with a number of state of the art MOEAs. They showed that HEMH and HEMH2 could outperform several state of the art MOEAs on a number of metrics including generational distance, hypervolume, coverage and spread. A key limitation of their work is that the total number of evaluations allocated to each algorithm run has been kept below $2 * 10^5$ for all the instances of the problem. Finally, one of the most recent work on MOd-KP has been done in [132] where the authors have compared three MOEAs namely SPEA2, ϵ-NSGA-II and ϵ-hBoa on uncorrelated and correlated MOd-KP problems. Their work links earlier findings in literature by closely following the settings and parameters from the literature. As opposed to most previous work, their study allocated 10^6 fitness evaluation for every run of the algorithms for each instance of the problem, thereby making their claims more robust than previous studies. They show that the ϵ-NSGA-II and ϵ-hBoa outperform SPEA2 and show that the best solutions are achieved by ϵ-hBoa with respect to ϵ-indicator and hypervolume metrics.

2.5 Evolutionary dynamic optimisation

Evolutionary dynamic optimisation (EDO) techniques are methodologies that employ evolutionary algorithms to solve dynamic optimisation problems (DOPs). Many real world problems involve time dependent changes and these can be modelled as DOPs. Due to the importance of DOPs in real world applications the last two decades have seen a growing interest to employ EAs to solve these problems [160].

DOPs can be formed of static problems linked together with respect to time or they might contain a time dependent component for objectives or constraints in the problem expression [160]. Due to the changes introduced in the problem over time, the optima of the problem also changes with time. Both static and dynamic problems require the optimal value to be located but only in the dynamic scenario, a tracking of the changing optima is needed. This is an important aspect which distinguishes dynamic problems from static ones. It poses a challenge to conventional EAs which once converged, are unable to track the changing optima well. Therefore the goal of EDO is to develop and enhance EAs to overcome this challenge.

Over the years, many EDO techniques have been proposed to tackle these issues, like diversity approaches [142], memory methods [156], prediction based techniques [133], multi-populations [71] and self-adaptive methods [145]. A brief description of these techniques is provided below. A detailed survey of applications can be found in [119].

- Diversity approaches employ methods to either introduce diversity in the solutions when a change is detected or maintain diversity throughout the search process. Diversity can be introduced by incorporating hyper-mutation [116] or varying mutation step size [154] or introducing random individuals into the population [72]. Maintaining diversity is performed by fitness sharing [116], regular introduction of new random individuals [161] or explicitly keeping individuals from coming close to each other [16]. Methods which introduce diversity tend to be good at solving problems with problems that are changing continuously at a small or medium pace.

These techniques behave like local search and if the optimum does not move too far then it can be tracked [147]. However these methods are reliant on the detection of changes [112], and it might not be be possible to detect the right amount of diversity needed [92]. Finally they are not effective when the changes are random or large [117]. Methods which maintain diversity are good at solving problems with large changes [116], however the impact of maintaining diversity slows down the optimisation process [92] and they are less effective for dealing with small changes [30].

- Memory methods, as the name suggests, relies on maintaining a memory component which is useful in cases when the changes introduced in the problem are periodic or recurrent in nature. The memory component can be used when an optima is revisited or for the purpose of maintaining diversity [117]. The main strengths of these approaches is in periodic environments [159] and they can also be helpful to promote diversity [19]. At the same time they rely on the fact that optima will reappear otherwise the information in the memory is obsolete and may affect performance of the algorithms [117].

- Prediction based techniques rely on learning patterns of changes during the search process which could reoccur in future and predict the changes in the future. Prediction can be done on the movement of the optima using forecasting methods [129], on the location of individuals to introduce after a change [165] or on the time when a change will occur [133]. These approaches are hinged on the accuracy of prediction and are prone to errors associated with training. These techniques are hinged upon the accuracy of prediction, in which case they can be successful to rack the optima [83]. The main limitations of of these methods being the training errors due to misleading data, unpredictable problems or stochastic changes [117].

- Self-adaptive approaches rely on the self-adaptation properties of EAs such as evolving the various EA parameters as the algorithm evolves. Self-adaptive approaches have been shown to perform well on simple dynamic problems but are difficult to

apply in complex dynamic scenarios [145].

- Multi-population methods maintain sub-populations simultaneously and can be viewed as a combination of memory, diversity and adaptation techniques. The idea is that each sub-population handles a different area of the search space and some of them may handle different tasks [16]. Some populations may be tasked with search for the optima while others may detect changes. These methods have several advantages such as maintaining diversity [20], recollection of information [18], tracking multiple optima [20] and they can deal with multi modal problems [20]. The main disadvantage is that having many populations can slow down the search [15]. Also calculating some metrics between populations can affect the performance [117].

2.5.1 EDO techniques with memory

One particular EDO method used to track the optima in a dynamic optimisation problem (DOP), is to use some form of memory. This technique revolves around storing solutions and utilising them in the future. It has been particularly useful when the DOPs involve recurrent or periodic changes and there is a high probability for the revisiting of an old optimal value [156, 119]. The methods of storing memory fall into two categories: implicit and explicit memory [119]. Explicit memory, as the name suggests involves storing some information explicitly as memory by utilising some form of storage. Implicit memory on the other hand is maintained within the solutions by representing solutions as multiploid, as opposed to haploid representation which is common for EAs used for static problems. In the case of explicit memory the information stored can be to store the solutions themselves in which case it is called direct memory or it can be associative information about the solutions where it is referred to as indirect memory. This information needs to be updated periodically by replacing old memory with new information. [119] provide a review of memory based techniques where their advantages are cited as maintaining diversity and being particularly useful in periodic scenarios. They can however be disadvantageous

in cases when previous optima are not revisited and the redundant coding of information might not be useful if the fluctuations in the problem are too high.

Considering implicit storage, [73] compared a haploid and a diploid genetic algorithm on the oscillating time knapsack problem and showed that the diploid genetic algorithm adapted more quickly to the changing problem environment. [109] have shown on the same problem, that the simple diploid scheme was not enough to handle the dynamic scenario and some form of dominance scheme was needed. The cardinality of genotypic representation and the uncertainty of the genotype to phenotype mapping was studied in [158] using diploid representation on a test bed of dynamic problems. [162] incorporated a memory technique in the population based incremental learning algorithms, that stored the best solution as well as environment information in the store. Using a new approach called aligned transformation work in [157] led to improved immune inspired genetic algorithms for certain dynamic test problems. The AMIGA (Individual memory aided genetic algorithm) was introduced in [143] where the oldest solution in memory was replaced by new ones. [57] introduced a case based memory in a genetic algorithm setting, where the least contributing memory solution was replaced by newer solutions. [18] has employed an explicit memory method in a finite storage pool with replacement to address several dynamic problems. The dynamic constrained multi-objective artificial immune system (DCMOAIS) was introduced by [164] which consists of three modules inspired by T-cells, B-cells and memory cells.

Based on this literature review it can be seen most of the work done with memory approaches targets dynamic optimisation and tracking the changing optimum. The strengths of the technique are in cases when the changes are periodic in nature along with the fact that maintaining memory can help maintain some diversity in the population. At the same time some disadvantages have also been seen, namely in obsolete memory information when optima are not revisited and in instances when fluctuations are too high. None of these studies address the reason behind these characteristic properties of memory. In other words, they do not address the cause of the effect of memory: why does

memory not work if fluctuations are too high or even a more basic question of is it really worth using memory at all in place of a random solution? How much does the problem need to change in order for the memory to become obsolete? These are fundamental issues in understanding the behaviour of memory and are identified as gaps in previous work, which will be addressed in this thesis.

2.5.2 Single time step dynamic set cover problem

In order to transform the SCP into a dynamic problem, we need to define how the problem changes over time. These changes must be time dependent and can be performed to the set U, or the set S or to both of them simultaneously. Definition 2 can now be extended to incorporate time dependent changes. Two models of changes are presented in this thesis namely modifying set S and modifying set U.

In the first model changes to S are considered and three different types of modifications (addition, removal, editing) are performed. A formal definition of this model can be described as:

Definition 4 *Let $A = (a_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}$ with $a_{ij} \in \{0, 1\}$ be a matrix with m rows and $n = 2^m$ columns where a column j is said to cover a row i if $a_{ij} = 1$. As before we have $|U| = m$ and each column j corresponds to a different subset of U. At time t, we have a particular selection of subsets $s(t)$. The goal is to find, for each t, a binary string $X^t = x_1^t \cdots x_n^t$ which minimises $\sum_{j=1}^{n} x_j^t$ subject to $\sum_{j \in s(t)} a_{i,j} x_j^t \geq 1$ for all $i = 1 \cdots m$.*

Note, that in an implementation one would usually consider bit strings of length $|s(t)|$, i. e, bit strings of varying length over time, instead of n and introduce a mapping to the actual subsets to decrease time and memory requirements.

For the second model changes to U are considered. In this case the size of set U is changed in the sense that only some predefined elements of U are required to be covered. In order to do this the *original* instances are converted to *starting* instances U^* by randomly

selecting some items from U. Following this, the same three kinds of modifications are done to the set U^*. Formally a definition for this model can be stated as:

Definition 5 *Let $A = (a_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}$ with $a_{ij} \in \{0, 1\}$ be a matrix with m rows and n columns representing U and S as before. At each time t, we have a subset of the universe $u(t)$ to be covered. The goal is to find, for each t, a binary string $X^t = x_1^t \cdots x_n^t$ which minimises $\sum_{j=1}^{n} x_j^t$ subject to $\sum_{j=1}^{n} a_{i,j} x_j^t \geq 1$ for all $i \in u(t)$.*

These modifications create new instances which are referred to as novel instances and they form the time dependent changed problem. Linking the original instance in case of Model 1 and starting instance for Model 2 with the novel instances forms a new dynamic instance of SCP.

2.5.3 Relevant work on dynamic SCP

One of the earliest work on dynamic set cover problem was conducted in [28] for the dynamic facility location problem. Given a locations of facilities of emergency services, the problem considered covering demands for emergency services in an area. It was proposed that the need for emergency services along with the location of the available services does not remain constant but changes from time to time. Different types of changes to the cost associated with the facilities, changes to the number of available locations as well as the number of demands were considered. A linear programming technique was compared with an approximation algorithm and it was shown that the approximation algorithm could solve the problems in less time than the linear programming technique. The work in [28] is dated and uses problem specific approaches of linear programming and approximation algorithms which are known to have scaling problems when the problem size becomes large. This presents an opportunity to test a meta-heuristic approach for this kind of problem that can scale well with larger problems.

Several versions of facility location problems including static, probabilistic as well as dynamic models have been proposed in [21, 136] which provides a survey of technique

developed to solve them. Work presented in [134] considers the dynamic multiple fault diagnosis (DMFD) which has applications in medical diagnosis as well as speech recognition, error correction and networks. Four different formulations of the DMFD are proposed in the study and a particular formulation named 'perfectly-observed' can be viewed as a dynamic SCP. A dynamic programming technique was employed in the study, along with Lagrangian multipliers to solve this problem. To the best of our knowledge these applications of dynamic SCP form all the work which has been done on this problem.

2.6 Summary

The background information needed in order to follow the domain specific content in this thesis is presented in this Chapter. Key concepts of the immune system were introduced along with major theories that have been developed to design AIS from immune concepts along with a review of work done on them. Core concepts of multi-objective optimisation and dynamic optimisation are introduced along with the problems addressed in this thesis and a review of work done in these field. The literature gap for the various problems is identified which will be addressed in this thesis.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This thesis adopts an empirical approach in order to study the features and behaviour of the GC-AIS and in its comparison with other multi-objective methods which comprise both simple and state-of-the-art algorithms. A consistent methodology of experimentation and analysis of results is used throughout this thesis and this chapter deals with explaining the key elements of the methodology.

A breakdown of the chapter is follows: first, the multi-objective evolutionary algorithms used for the comparative study are discussed in section 3.2. This is followed by a description of the problem instances of the SCP, MOd-KP and the dynamic SCP used for the various experiments conducted in the remainder of the thesis. The stopping criteria for the MOEAs is discussed in section 3.4 followed by a description of the technique employed for the generation of reference fronts for the problem instances. The different metrics of comparison employed in order to analyse the results of the experiments are detailed in section 3.6 and finally a brief description of the statistical tests used in the thesis are described.

3.2 MOEAs for comparative study

Two multi-objective evolutionary algorithms are used to compare the experimental performance of the *germinal centre artificial immune system* (GC-AIS) proposed in this thesis. The first one is a simple multi-objective optimiser called the parallel global simple evolutionary multi-objective optimiser (PGSEMO) which acts as a baseline algorithm for the very first experiments with GC-AIS. The non-dominated sorting genetic algorithm II (NSGA-II) is the second one, which is selected as it is a popular and well established MOEA and according to work done in [132] NSGA-II still performs as the state of the art algorithm for the MOd-KP, which was the latest work done on MOd-KP at the time of this research.

It must be stated here though that NSGA-II is now dated [55] and many new MOEAs have been proposed since, they have not yet been tested on the MOd-KP. Therefore it makes more sense to select an established algorithm whose performance on the problem is shown to be good, rather than selecting a new MOEA for which there is no work done on the problem being considered here. Another point that must be addressed is the fact that in the choice of algorithms for comparison, no AIS were considered as there were not any known AIS which were the state of the art algorithms for the problems considered here. Being an AIS it is a valid avenue for research to compare GC-AIS with other immune based algorithms and can form future work but that comparative analysis is not performed here as at this stage we are interested in learning about the behaviour and capabilities of GC-AIS and understanding the reasons behind its strengths and weaknesses.

Description of PGSEMO

The *global simple evolutionary multi-objective optimiser* (GSEMO) introduced in [70] is a generalisation of the $(1 + 1)$ EA for multi-objective optimisation. GSEMO is a variant of the simple evolutionary multi-objective optimiser (SEMO) [67], where the mutation operator differs between the two. SEMO uses local mutation that is one random bit

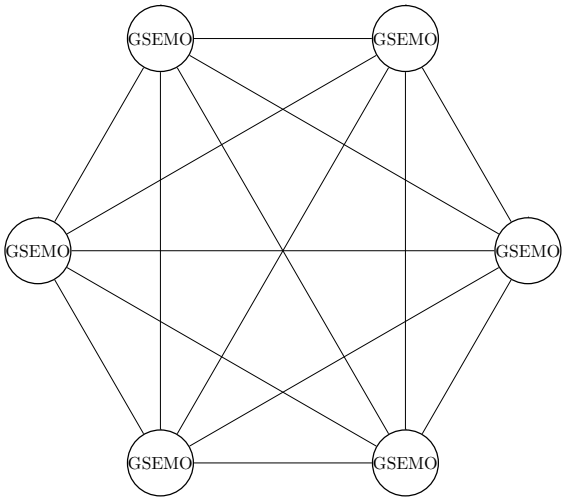


Figure 3.1: A schematic diagram of the archipelago structure of PGSEMO.

flipped which is replaced in GSEMO by standard bit mutation where each bit is flipped with a probability. It maintains a set of non-dominated solutions in every iteration. A new solution is generated by randomly selecting a solution from the current population and mutating it using standard bit mutation. The solution is added to the population and all dominated solutions are then removed.

The parallel variant of GSEMO called the homogeneous island model GSEMO based on [110] can be described as a collection or an archipelago of μ islands which are fully connected to each other where each island runs an independent instance of the GSEMO algorithm. By a fully connected topology, it is implied that each island is a neighbour of all the other islands, therefore any communication will consist between all neighbours. Each island consists of a population of non-dominating solutions and for the purpose of migration a copy of the entire population is sent to all other islands where the solutions are incorporated into their population and dominated solutions are removed. In this thesis this model is referred to as parallel GSEMO (PGSEMO) throughout. The algorithm was proposed by The algorithm is presented in [110], where the authors proposed this model along with other variants with different topologies in order to perform a theoretical analysis on the running time and communication effort needed by th model for the SCP. Other than this one study no other practical applications of this model for any other problems are known to us at this time. Known theoretical analysis on the running time

Algorithm 6 Parallel GSEMO based on homogeneous island model [110]

Let P_i^t denote the population in each island i at generation t, μ denote number of islands, and p denote probability of communication.
Initialise $P^0 = \{P_1^0, \ldots, P_\mu^0\}$ where $P_i^0 = \{0^n\}$ for $1 \leq i \leq \mu$. Let $t := 0$.
loop
  for each island i in parallel **do**
    Select an individual x from P_i^t uniformly at random.
    Create offspring x' by mutating x with standard bit mutation, i. e., flip each bit with probability $1/n$.
    if any individual in P_i^t dominates x' **then**
      Leave P_i^t unchanged.
    else
      Remove all individuals dominated by x' from P_i^t and add x' to P_i^t.
    end if
    With probability p send copy of population P_i^t to all $\mu - 1$ neighbours.
    Combine P_i^t with copies of populations received from neighbours.
    Remove all dominated solutions from P_i^t and let $P_i^{t+1} = P_i^t$.
  end for
  Let $t = t + 1$.
end loop

and communication effort along with a simple model are the only advantages of the model. Algorithm 6 and a schematic diagram is provided in Figure 3.1.

Description of NSGA-II

NSGA-II [55] is one of the most popular elitist multi-objective evolutionary algorithm, where elitism refers to the fact that in every iteration the best solutions are carried forward. This algorithm is similar to the $(\mu + \mu)$ evolutionary algorithm in terms of generation update. After initialisation of a fixed number of λ solutions, the algorithm uses non-dominated sorting to assign a rank or fitness to each solution according to levels of dominated fronts. This is done by pairwise comparison of each solution with others and placing it in fronts according to the number of solutions it dominates. The fronts obtained are ranked according to their level of non-dominations. Considering maximisation, rank 1 is given to solutions which are not dominated by any other solutions, rank 2 to solutions which are dominated by only ranked 1 solutions, and so on. Crowding distance, a measure to evaluate how close the solutions are to each other, is assigned to each solution in every

fitness level. Individuals are selected by binary tournament selection with replacement using the ranks and crowding distance and are then subject to crossover and mutation to create λ offspring. These offspring are combined with the parent population and the best λ solutions are carried to the next generation. In this way, in each generation a fixed population size λ is maintained. A generic outline of the algorithm is provided in Algorithm 7.

Algorithm 7 Outline of NSGA-II

Let P_t denote the population at generation t, N denote the fixed population size.
Initialise (P_0) randomly
Separate (P_0) into fronts $\{\mathcal{F}_1, \mathcal{F}_2, \cdots\}$ by non-dominated sorting.
for each $\mathcal{F}_i$ in P_i **do**
  Calculate crowding distance of each front $(\mathcal{F}_i)$.
end for
Set $t = 0$
while Termination criteria not met **do**
  create child population Q_t from P_t
  Merge child population Q_t with parents P_t to create R_t
  Separate R_t into fronts $\{\mathcal{F}_1, \mathcal{F}_2, \cdots\}$ by non-dominated sorting.
  Set $P_{t+1} = \emptyset$ and $i = 0$
  while Union of P_{t+1} and front $\mathcal{F}_i$ is less than population size N. **do**
    Calculate crowding distance of front $(\mathcal{F}_i)$.
    Add front $(\mathcal{F}_i)$ to P_{t+1}
    i = i+1
  end while
  Calculate crowding distance of front $(\mathcal{F}_i)$.
  add enough individuals from front $(\mathcal{F}_i)$ to P_{t+1} to reach population size N.
  $t = t + 1$
end while
return Non-dominated front $(\mathcal{F}_1)$

The NSGA-II has been applied to a host of applications which can be seen in over eighteen thousand citations of the original paper on Scholar. The limitations of NSGA-II along with other EMO techniques as state in [52] become apparent when dealing with many objective optimisation, i.e. when the number of objective is larger than 3. In such cases the computation of crowding distance becomes expensive and the search process is slowed down due to the large search space causing the non-dominated solutions to occupy most of the population.

Problem	$m \times n$	density	Known [114]
41	200×1000	2%	(0,38)
52	200×2000	2%	(0,34)
63	200×1000	5%	(0,21)
a5	300×3000	2%	(0,38)
b4	300×3000	5%	(0,22)
c3	400×4000	5%	(0,43)
d2	400×4000	5%	(0,25)
e1	50×500	20%	(0,5)
re1	500×5000	10%	(0,17)
rf2	500×5000	10%	(0,10)
rg3	1000×10000	2%	(0,62)
rh4	1000×10000	5%	(0,35)

Table 3.1: Best known solutions for the original problem instances of SCP. Density represents the the percentage of 1s in the $(m \times n)$ matrix. 'Known' contains solutions presented in [114]

3.3 Problem instances

To perform experimental analysis on the problems chosen for this thesis, namely the Set cover problem (SCP) and the multi-objective knapsack problem (MOd-KP), instances for these problems must be selected. Popular and widely studied instances from literature have been selected for this purpose.

Instances for the set cover problem have been taken from the OR-library [11]. The instances in the OR-library are divided into classes labelled from 4-6, A-E and rE-rH. The problems in the class 4-6 are originally taken from [7], which were generated randomly with the density of the matrix as 2% for classes 4 and 5 and 5% for class 6, with the requirement that every column is covered by at least one row and each row at least covers two columns. Using a similar scheme [10] generated instances which have been labelled from A-E. Larger instances in the classes rE-rH are originally provided by [14].

One instance from each class is selected as the class representative to be studied in this thesis. In order to avoid any selection bias favouring any algorithms, the instances were selected using a simple iterative scheme. Since each class is ordered alphabetically, the instances were selected in increments of 1 from either the beginning or the end of

the class groups. For example from class 4, the first instance (scp41) was selected, from class 5 the second (scp52) and so on till class 6 (scp63). From classes A-E the same scheme was used in descending order, so for class A the last instance (scpa5), class B the second to lass (scpb4) and so on till class e where the first instance (scpe1) was chosen. Finally for classes rE-rH the scheme was used again in ascending order. The size of these representative instances are shown in Table 3.1 where the size refers to number of sets in the universe set times the number of subsets in the set of subsets. The density of the instances indicates the number of sets in the universe which are covered by each subset in the set of subsets. This implies that in instances with low density the subsets cover less elements from the universe set, while for high density instances the subsets cover higher elements of the universe set. Roughly problems can be grouped into the instance size and the density, for each size there are 2 instances (except scpe1 and scp52), one with high density and one with low density.

Problem instances for MOd-KP provided by [167] have been widely studied in the MOEA literature. These instances have 2, 3 and 4 objectives corresponding to the number of knapsacks each with 100, 250, 500 and 750 items making a total of 12 instances. The instances have been generated randomly with no correlation between the profits and weights and the capacity of each knapsack are equal to half of the total weight of the items. These problem instances are selected for experimental study in this thesis as they help to connect the previous work in the literature with the research in this thesis.

3.4 Stopping criteria

Setting an appropriate criteria for stopping multi-objective evolutionary algorithms is a difficult problem, in fact it is a multi-objective problem in itself where the goal is to maximise the Pareto front approximation by utilising minimum number of generations [81]. Authors of [81] provide five suitable conditions under which an MOEA could be stopped, namely:

1. The amount of computation performed is sufficiently large enough.
2. The obtained solution is satisfactory.
3. A better solution is unlikely to be produced, even if the current one is not satisfactory.
4. The algorithm is unable to converge to a solution.
5. Any more computation spent is unlikely to provide any improvement.

Finding the right condition to stop the algorithm, or even detecting the presence of these conditions is hard and may depend on several factors such as prior information or experience with the problem, knowledge of the true Pareto front. These factors may not always be easily determined and [81] provide a review of techniques used in literature to deal with this scenario.

Specifically for this thesis, which deals with SCP and MOd-KP suitable stopping conditions need to be identified and selected for the various experiments. Criteria which satisfy condition 1, 2 or 5 are used for different cases of experiments in this thesis, along with a justification for the selection of the chosen criteria. Condition 1 and 5 are used for initial experiments with GC-AIS and PGSEMO on SCP in Chapter 4 when no past information is known about the algorithms. Sufficient time is allotted to the algorithms so that they can achieve some desired solution quality. Following these initial experiments and obtaining experience in the behaviour of the algorithms condition 5 is used in the experiments conducted in Chapter 5 for the same problem. Setting the stopping criteria for the experiments for MOd-KP however is slightly easier due to the presence of existing work which provides previous experience with the problem. In this case criteria 1 is selected based on the reliability of several previous experiments conducted on the problem in the literature. Finally for the single time-step dynamic SCP previous information from chapters 4 and 5 provide experience in setting the stopping criteria based on condition 1 and 5.

3.5 Reference front generation

When dealing with real world problems, true Pareto fronts are often unknown [89] and in these circumstances, in order to utilise performance metrics which rely on the knowledge of the Pareto front, reference fronts must be generated. A popular method used within the MOEA literature, when trying to generate reference fronts is to pool the non-dominated sets from the algorithms being investigated [89, 132]. The authors of [170] have provided the Pareto fronts for the smaller instances of the MOd-KP (2 sacks with 100, 250 and 500 items, 3 sacks with 100 items). For the remaining instances, using the method proposed in [132] reference fronts are generated for the following MOd-KP instances, 2 sacks with 750 items, 3 sacks with 250, 500 and 750 items, and all instances with 4 sacks.

The method to generate the reference fronts involves pooling the obtained non-dominated fronts from the algorithms being considered, preferably using multiple runs and finally removing all dominated solutions, resulting in a reference Pareto front. Following this method, the *germinal centre artificial immune system* (GC-AIS) or an improved variant, and its competitors are allowed to run, each for 30 runs, until a pre-specified stopping criterion. The non-dominated sets obtained after each run are recorded and they are all merged together. Finally after removing the dominated solutions from the merged set, an approximated Pareto front is generated.

3.6 Metrics for comparison of MOEAs

Several metrics for analysing the performance of MOEA have been proposed in literature, and it is known that unary performance measures which report a single value, cannot capture all the desired measurements of the MOEA performance. According to [172] a MOEA should be characterised by having the following 3 important properties:

- The distance between the Pareto front and the non-dominated solution set must be minimised.

- The solutions in the non-dominated set should be well spread.
- The extent of the non-dominated solution set should be maximised. In other terms, the solutions obtained should cover a wide range of values for each of the objective considered.

In order to cover all these criteria, three metrics commonly used in evolutionary multi-objective optimization are employed in order to compare the performance of the two algorithms. These metrics are the hypervolume metric [168], generational distance [146] and the generalized diversity metric [166]. Given the Pareto front solution set P^* and a set of N solutions with p objectives in the obtained non-dominated front Q, the three metrics are defined as follows:

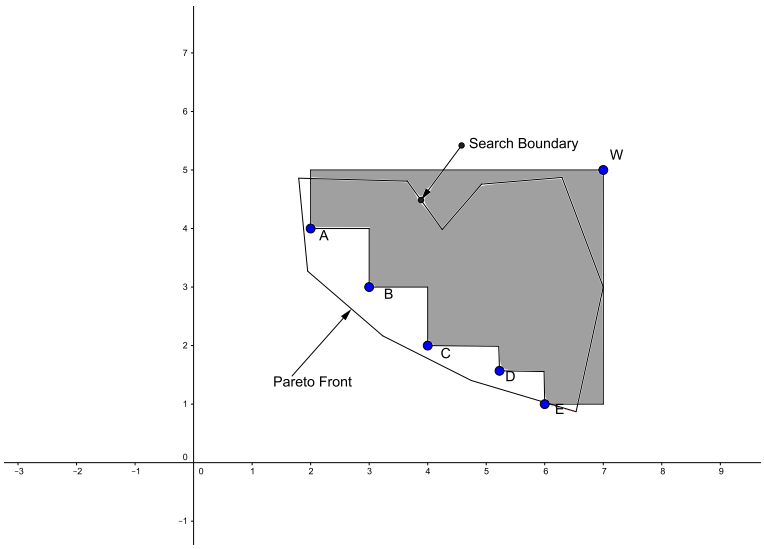
1. **Hypervolume** H: First introduced in [168] the hypervolume metric measures the volume dominated by the obtained non-dominated set. For each solution $s_i \in Q$ a hypervolume h_i is calculated by using a reference point P and the solution s_i as the diagonal corners of the hypercube. Considering minimisation the point W can be calculated as the least value of each objective. The total hypervolume can now be calculated as the sum of the hypervolumes covered by each point in the non-dominated set:

$$H = volume\left(\bigcup_{i=1}^{|Q|} h_i\right)$$

and higher values of this metric are preferred.

A diagram showing the hypervolume enclosed by the set of points in the non-dominated front, as the dotted region is shown in Figure 3.2. To calculate the hypervolume, consider the points in the non-dominated front, namely $A = (2, 3.9)$, $B = (3, 3)$, $C = (4, 2)$, $D = (5.3, 1.5)$ and $E = (5.9, 1)$, and the point $W = 7, 5$ as the reference point. The metric H is calculated as:

Figure 3.2: An example showing the hypervolume enclosed by non-dominated solutions in a 2 objective setting as the shaded region. Points A, B, C, D and E depict the solutions obtained and point W is the reference point.



$$
\begin{aligned}
H &= (7 - 5.9) \times (5 - 1) + (5.9 - 5.3) \times (5 - 1.5) \\
&\quad + (5.3 - 4) \times (5 - 2) + (4 - 3) \times (5 - 3) \\
&\quad + (3 - 2) \times (5 - 3.9) \\
&\quad = 11.7
\end{aligned}
$$

2. **Diversity** Δ: The diversity metric measures the distribution of solutions in the non-dominated front. This measure originally proposed for problems with two objectives in [55], was later generalized for any number of objectives in the work by [166]. This metric measures the spread between the obtained solutions.

$$
\Delta(Q, P^*) = \frac{\sum_{i=1}^{z} d(e_i, Q) + \sum\limits_{X \in P^*} \left| d(X, Q) - \overline{d} \right|}{\sum_{i=1}^{z} d(e_i, Q) + |P^*| \, \overline{d}},
$$

where $\{e_1, e_2 \ldots e_z\}$ are the z extreme solutions in the Pareto front and d is a distance measure between neighbouring solutions. Then the average of these distances

becomes:

$$\overline{d} = \frac{1}{|P^*|} \sum_{X \in P^*} d(X, Q).$$

The equation for Δ

The design of the metric is such that better spread is depicted by a lower value of this metric.

Figure 3.3 depicts the distances between the solutions in the non-dominated front and the 2 extreme solutions as dotted lines. Using the same co-ordinates of points from Figure 3.2 the distances between them become:

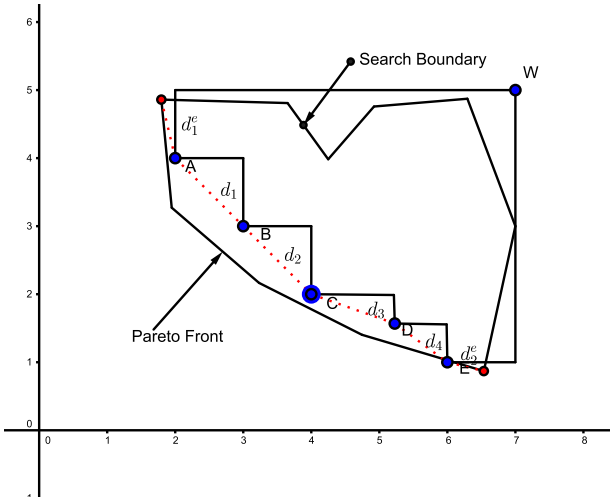
$$d_1 = 1.34, \quad d_2 = 1.41$$
$$d_3 = 1.39, \quad d_4 = 0.78$$
$$d_1^e = 1.01, \quad d_2^e = 0.42$$

the average distance $\overline{d}$ becomes 1.23. The spread metric now evaluates to:

$$\Delta = \frac{1.01 + 0.42 + |1.34 - 1.23| + |1.41 - 1.23| + |1.39 - 1.23| + |0.78 - 1.23|}{1.01 + 0.42 + 4 \times 1.23}$$
$$= 0.36$$

3. **Generational distance (GD)**: The generational distance estimates the distance between the solutions in the non-dominated set and the Pareto optimal set. It is defined as:

Figure 3.3: An example showing the diversity of 5 non-dominated solutions in a 2 objective. Points A, B, C, D and E depict the solutions obtained and point W is the reference point. Dotted line shows the distance between the non-dominated points.

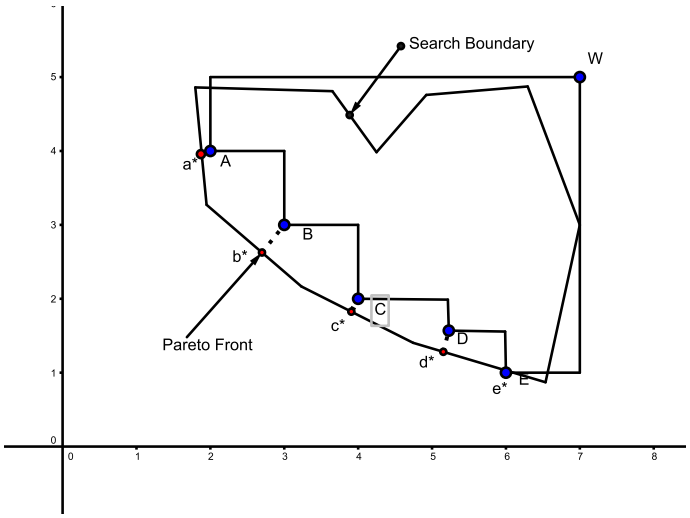


$$GD = \frac{\sqrt{\sum_{i=1}^{|Q|} d_i^p}}{|Q|},$$

where d_i is the Euclidean distance between each solution in the non-dominated set and its nearest neighbour in the Pareto set. The lower the value for this metric the closer the obtained solutions are to the Pareto front.

As an example, consider Figure 3.4 where the distance between the solutions in the Pareto set and the obtained non-dominated set are shown by a dotted line. The co-ordinates of the solutions on the Pareto front are $a^* = (1.8, 3.8)$, $b^* = (2.7, 2.6)$, $c^* = (4, 2)$, $d^* = (5.2, 1.2)$ and $e^* = (5.9, 1)$. The corresponding nearest solutions in the obtained fronts are $A = (2, 3.9)$, $B = (3, 3)$, $C = (4, 2)$, $D = (5.3, 1.5)$ and $E = (5.9, 1)$. Based on these points in the objective space, the Euclidean distances between them and finally the generational distance becomes:

Figure 3.4: An example diagram showing the distance between solutions in the non-dominated front (shown in capitals letters) and the Pareto front (shown in small starred letters) used for calculating the generational distance .



$$d_{Aa^*} = \sqrt{(2-1.8)^2 + (3.9-3.8)^2} = 0.22$$

$$d_{Bb^*} = \sqrt{(3-2.7)^2 + (3-2.6)^2} = 0.46$$

$$d_{Cc^*} = \sqrt{(4-3.8)^2 + (2-1.8)^2} = 0.28$$

$$d_{Dd^*} = \sqrt{(5.3-5.2)^2 + (1.5-1.2)^2} = 0.26$$

$$d_{Ee^*} = \sqrt{(5.9-5.9)^2 + (1-1)^2} = 0$$

$$\sum_{i=1}^{|Q|} d_i^p = 1.22$$

$$GD = \frac{\sqrt{\sum_{i=1}^{5} d_i^2}}{5} = 0.22$$

3.7 Statistical tests

The following statistical tests have been used to ascertain if there is any significant difference between the performance of the different algorithms tested. For each experiment 30 runs are performed and the data after the runs are used for the calculation of these

test metrics.

1. **Wilcoxon rank-sum test**: is a non-parametric test of the null hypothesis, i. e. there is no relationship between two methods. It is a replacement for the student's t-test, when the population of samples cannot be assumed to be drawn from the normal distribution. The test was first proposed by [153]. For the purpose of experiments conducted in this thesis, the `ranksum` procedure from MATLAB was used. A significance level of 0.05 for the p value is used throughout this thesis in order to determine if algorithms being compared are statistically significantly different.

2. **Kruskal-Wallis test**: is another non-parametric test which is used to determine if two or more population of samples originate from the same distribution. For the purpose of experiments conducted in this thesis, the `kruskalwallis` procedure from MATLAB was used when results from the runs of more than 2 algorithms were being compared, at a significance level of 0.05.

3.8 Summary

This chapter serves as the description of the algorithms, tools and techniques used for the empirical analysis throughout this thesis. The algorithms being used for comparison in this research are introduced along with the description of the problem instances used for the testing. The criteria employed for setting the stopping time for the algorithm is described along with the technique used for reference front generation. The metrics used for comparison of algorithms in a multi-objective setting used in this thesis are described along with the statistical tests used. This forms as a framework of the consistent set of experimental practices which have been employed in this thesis.

CHAPTER 4

DESIGNING THE GC-AIS

4.1 Introduction

This chapter describes the formulation of a novel immune inspired general randomised search heuristic called the *germinal centre artificial immune system* (GC-AIS) following new advances in immunology and proceeds by investigating the behaviour of GC-AIS on some multi-objective combinatorial optimisation problems. Artificial immune systems are relatively newer classes of meta-heuristics when compared with other established nature inspired techniques and novel research in immunology often finds translations into development of these algorithms. One such recent immunological advancement is the light that has been shed on the understanding of the germinal centre reaction (GC reaction) of the adaptive immune system.

An important aspect to consider while designing immune algorithms is to tackle specific problems. Recent research in the area is focussed towards finding the right application domain which is particularly suited for immune systems [82], by associating the problem characteristics with AIS mechanisms. One such problem which follows directly from the assertion that eradication of pathogens by the immune system can be modelled as set covering, is the set cover problem (Section 1.2.1).

The key research questions which are considered in this chapter are:

Is it possible to model the recent understanding of the GC *reaction into an*

immune system that tackle the multi-objective SCP*?*

Given that the biological immune system like any natural process is robust and performs well, it will be interesting to answer:

How does the novel artificial immune system perform on the set cover problem?

Being a general randomised search heuristic the novel immune algorithm should not only be limited to solving the SCP but should also be applicable to other problems. Exploring multi-objective combinatorial optimisation is the focus of this thesis therefore, this follows directly into:

Does the novel algorithm perform well on a real multi-objective optimisation problem such as the multi-objective knapsack problem?

Finally the last question concerning the performance which is considered in this chapter is:

What can be learned about the strengths and weaknesses of the novel artificial immune system based on the comparative analysis?

The structure of the chapter is as follows: Section 4.2 introduces the design of the GC-AIS. It is here that the model of the GC-AIS is first presented by abstracting key components from the GC reaction. In order to test the performance of the GC-AIS a comparative study is performed in Section 4.3 and 4.4. The performance of the algorithm is compared against a simple baseline MOEA on the SCP as well as a state of the art randomised search heuristic on the MOd-KP. The chapter is concluded by a discussion of the results and an analysis of the potential of the algorithm along with its shortcomings followed by a brief summary.[1]

[1]Part of work presented in this chapter has been published in [94] and [95].

4.2 The GC-AIS Algorithm Description

The GC-AIS is a new AIS used for multi-objective optimisation which is inspired by new advancements in the field of immunology namely the working principles of the germinal centre reaction as described in Section. The key advantages offered by the germinal centre theory that can be useful for multi-objective optimisation are related to the features required by any algorithm to perform multi-objective optimisation which are maintaining a diverse population of solutions which converge towards the Pareto frontier. As explained in the GC reaction, by the process of maintaining a dynamic number of germinal centres each having the stages of continuous selection, cloning and mutation, and communication, a diverse set of solutions is obtained which continuously improves towards the optimal front. Along with this, the added advantage of the theory is that the starting from a single GC, a pool of GC is created and throughout the process of evolution the number of these GC is dynamic is nature. The implication of this dynamic nature results in not having to specify any fixed parameter for population size or island number to be hard coded into the algorithm.

A detailed description of the steps of the GC-AIS (Algorithm 8) is provided next. At the start of the algorithm a single GC is created which encodes a potential solution to the problem. The GC creates a clone which is followed by the process of standard bit mutation where each bit of the cloned GC is flipped with a probability $1/n$, where n is the length of the bit string. At this stage communication between GC is performed by migration of fitness values of the offspring to all GC. All dominated solutions are deleted which represents natural cell death of B cells which are unable to compete in the selection part of the GC reaction and new GC are created from all the surviving offspring. Following these steps the GC-AIS always maintains a set of non-dominated solutions and the population size of GC is dynamic in nature.

From the pseudocode of the algorithm in 8, it can be seen that the process of clone creation can be performed in a parallel manner. This is seen in the bold text in 8 where for each GC, a clone needs to be produced for the B cell by standard bit mutation. This

Algorithm 8 The GC-AIS

Let G^t denote the pool of GCs at generation t and g_i^t the i-th GC in G^t.
Create GC pool $G^0 = \{g^0\}$ and initialise g^0. Let $t := 0$.
loop
  for each GC g_i^t in pool G^t in **parallel do**
    Create clone y_i of B-cell in g_i^t by standard bit mutation.
  end for
  Add all y_i to G^t, remove all dominated solutions from G^t and let $G^{t+1} = G^t$.
  Let $t = t + 1$.
end loop

means that the GC-AIS can also be seen as a parallel algorithm, where each GC is an island of population of containing 1 B cell. The communication between the GCs takes place by the transfer of fitness between the islands and the GCs which are dominated are killed off. Thus resulting in a dynamic island topology in a parallel setting. Following this description a flow diagram of the algorithm is shown in Figure 4.1. In order to facilitate the understanding of the AIS specific terminology, a table showing related terms from EA literature is provided in Table 4.1.

This abstraction of the immune processes does not adhere to each and every immune concept stringently. New GCs are formed by the surviving non-dominated GC while due to the nature of biological GC, new clones of B cells remain inside GCs. The design choices in GC-AIS have been made to tackle the problem that adhering to immune principles stringently increases the complexity of the algorithm which in turn makes it difficult to tune and analyse. Keeping the algorithm simple at the beginning provides for a better environment which can be analysed more thoroughly and all of its parameters can be tuned. This approach has provided significant insights which have led to subsequent improvements of the general algorithmic idea, detailed in Chapter 5 and 6.

The GC-AIS is a novel and unique artificial immune system that can be clearly distinguished from other existing approaches. There are several existing artificial immune algorithms and some are more similar to GC-AIS than others. A few differences between GC-AIS and the popular artificial immune systems CLONALG [49], BCA [101] are, that GC-AIS is proposed for multi-objective optimisation while CLONALG and

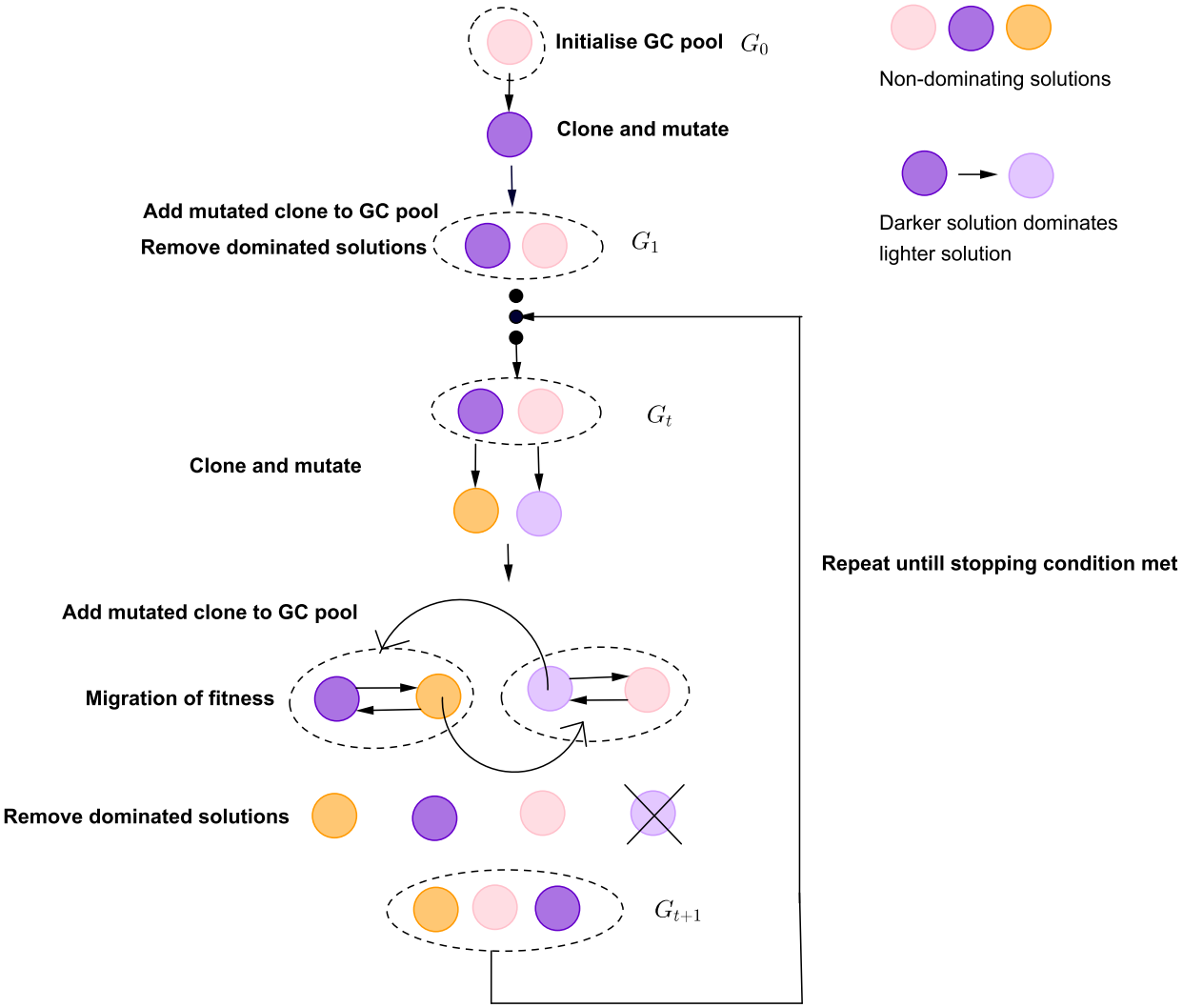


Figure 4.1: An illustrative diagram of a random instance of a GC-AIS run. GCs are depicted as coloured circles and each GC in this figure contains 1 B-cell (potential solution). Different coloured GCs imply the solutions within them are non-dominating while same coloured ones imply dominating solutions. Lighter solutions are dominated by darked solutions.

IS terminology	EA terminology
B cell	Individual
Clone	Offspring
B cell pool	Population
GC	Island
GC pool	Archipelago
Antibody	Fitness
Migration of Antibody	Communication of Individual

Table 4.1: Key terminology used in IS with its corresponding terms from EAs

BCA were originally proposed for single objective optimisation (see Section 2.3.1). Both CLONALG and the BCA use specific mutation operators inspired by the immune system namely inversely fitness proportionate mutations and somatic contiguous hypermutations. The population size in CLONALG is fixed and each iteration includes an introduction of new random solutions which replace poor quality solutions. The BCA has its own unique feature of comprising a clonal pool for every B cell and applying the contiguous hypermutation as well as standard bit mutation to one random clone.

When considering multi-objective artificial immune systems, VAIS [65] and MISA [31] are considered (see Section 2.3.1) which are well known multi-objective AIS. Both MISA and VAIS employ two populations of solutions, one for the evolution and one for storing elite solutions. They both have a fixed population size and create a variable number of clones of each solution and employ different kinds of non-uniform mutations to the clones. In MISA, the clones are mutated in a hierarchical manner where non-dominated solutions are mutated on less bits than solutions which are dominated, and a similar strategy is employed for infeasible solutions with an even higher number of bits being mutated. In VAIS, local mutation is used which is inversely fitness proportional to the fitness of the parent. The GC-AIS differs from all these algorithms as the mutation employed in GC-AIS is the standard bit-mutation. Each B cell produces only one clone and the clonal pool is merged with the parents, following which all dominated solutions are removed.

4.3 Testing the performance of GC-AIS on SCP

This section presents the experimental analysis performed in order to determine the performance of GC-AIS on the set cover problem (SCP). For this experimental study GC-AIS is compared with the *parallel global simple evolutionary algorithm* (PGSEMO) [110] on the SCP. As the first set of experiments on GC-AIS, it is a reasonable starting point to compare it with a simple EA rather than a state of the art algorithm for this reason

PGSEMO was selected as the competitor for GC-AIS. The decision also makes sense since [110] provides conditions for the best performance guarantee of PGSEMO in the form of theoretical analysis based on the different parameters of PGSEMO. In light of PGSEMO being a parallel algorithm, it must be noted that GC-AIS can also be viewed as a parallel algorithm if each GC is considered as an island with the population of each island limited to size 1. In this case the communication between island happens by transmission of fitness between GCs. Selecting a simple EA is a useful starting point because at this stage it can provide feedback on the current state of GC-AIS. If it can compete with a simple EA then further studies can be performed with state of the art algorithms. On the other hand if it can not even compete with a simple EA then modifications and changes can be introduced at an early stage.

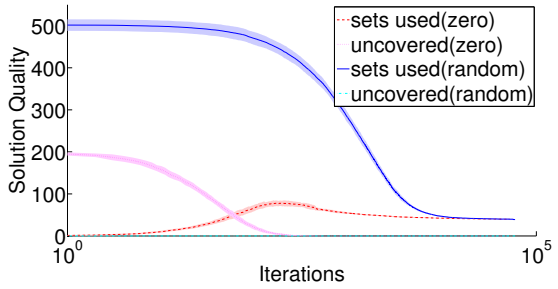
Since PGSEMO is a parallel EA, apart from the standard fitness evaluation based running time, two metrics pertaining to parallel models have been used for comparison: parallel running time and communication effort [110]. The parallel running time is the number of iterations the algorithm requires to reach an optimal solution while the communication effort measures the amount of communication between islands. The total communication effort is described as the total number of individuals migrated per iteration of the algorithm.

4.3.1 Testing initialisation conditions for GC-AIS

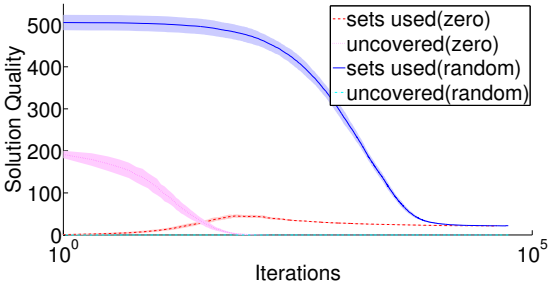
In order to perform a fair comparison between the two algorithms, some parameter tuning must be performed to ensure that the algorithms configurations can provide the best results on the problem. It can be seen from the Algorithm 8 that the only variable parameter in GC-AIS is the mutation. Since the mutation rate is fixed to $1/n$ for both PGSEMO and GC-AIS, the initialisation of the solution in GC-AIS is a feature which needs to be explored. The authors of [110] set the initial solution in the islands of PGSEMO as the all 0s bit strings citing the theoretical work in [67] as motivation. To distinguish between the effects of different seeds for GC-AIS, experiments are conducted

to test the performance on SCP by comparing the performance of the algorithm when setting the seed as a random bit string versus a fixed all 0s bit string. The algorithm is allowed to run until the solution does not improve for $en \log n$ iterations and the solutions are recorded. The reason to set this value is ensure that with high probability a one bit flip for each bit of the solution has been observed. The results are recorded and the plots can be seen in Figure 4.2 and 4.3

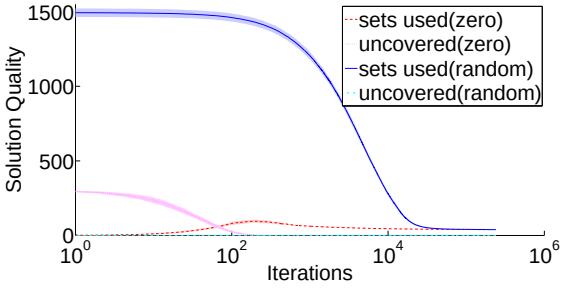
Figure 4.2: Run time plots of solution quality for GC-AIS when starting from random bit-string versus all 0 bit string. Solid blue lines show sets used and the dot-dashed cyan line shows the uncovered elements, when starting from the random string. The dotted magenta line shows the uncovered elements and the dashed red line shows sets used when starting from all 0s string.



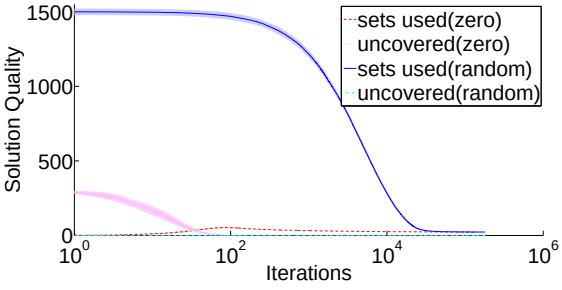
(a) Instance: 41



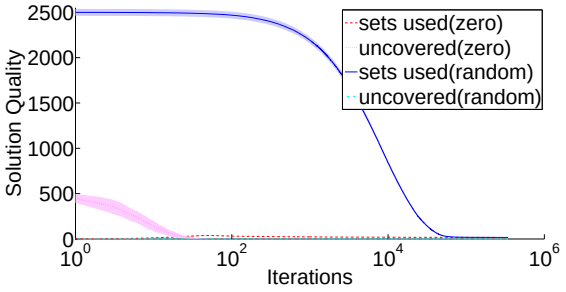
(b) Instance: 63



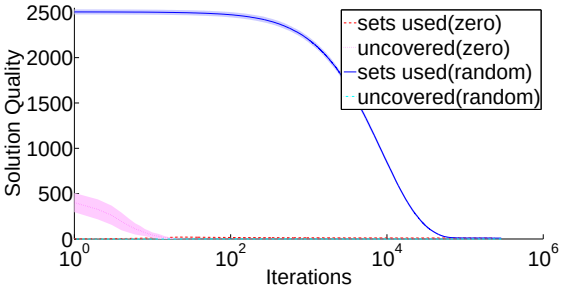
(c) Instance: a5



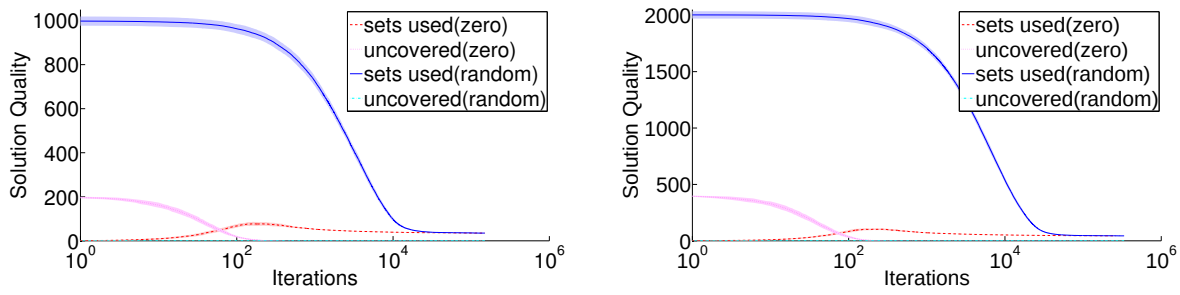
(d) Instance: b4



(e) Instance: re1

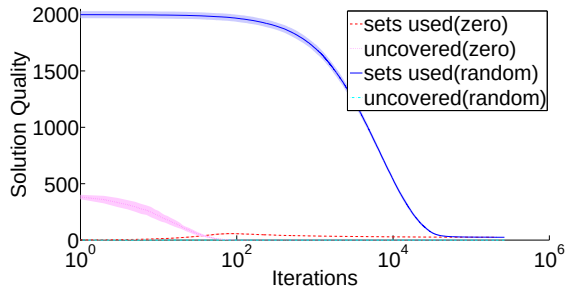


(f) Instance: rf2

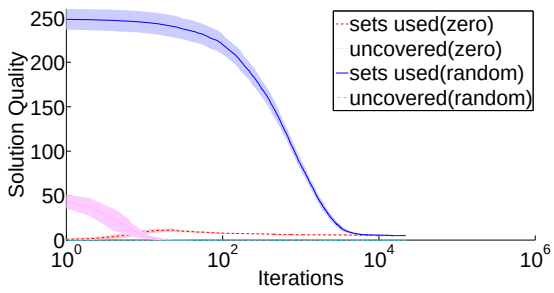


(a) Instance: 52

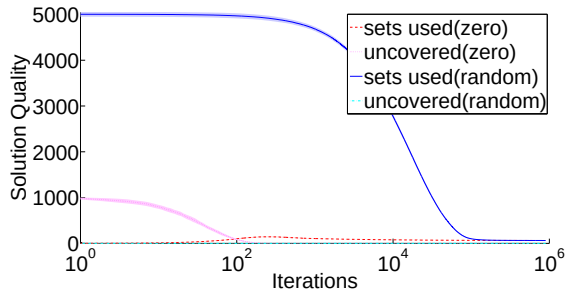
(b) Instance: c3



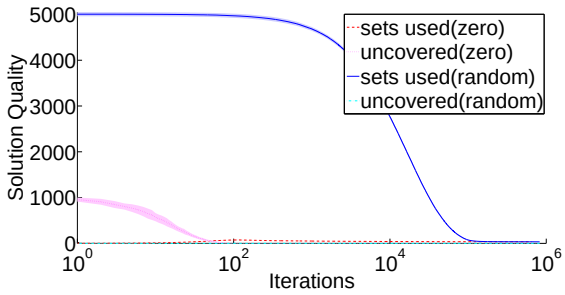
(c) Instance: d2



(d) Instance: e1



(e) Instance: rg3



(f) Instance: rh4

Figure 4.3: Run time plots of solution quality for GC-AIS when starting from random bit-string versus all 0 bit string. Solid blue lines show sets used and the dot-dashed cyan line shows the uncovered elements, when starting from the random string. The dotted magenta line shows the uncovered elements and the dashed red line shows sets used when starting from all 0s string. X-axis is plotted on a log scale and standard deviation is shown as shaded error-bars.

Analysis of initialisation conditions

It can be distinguished that setting the initial solution to the all 0s bit string results in GC-AIS reaching an optimal solution faster than starting from a random bit string. The dotted red line in Figure 4.2 shows the sets used when starting from the all 0s string, in the best solution per iteration, whereas the solid blue line is the sets used when solution is initialised as a random string. Clearly the red line is seen to flatten out much quicker

after 100 iterations, meaning that the solution obtained is very close to the optimal and it quickly reaches the optimal solution when the line flattens. Meanwhile the blue line shows that the best solutions when starting from a random string contains a lot more sets than the red line and it takes a much longer time to reach the optimal solution which can be seen towards the end of each plot. The variation in the limits of the x and y axes of the plots for different instances can be attributed to the problem sizes of the instances and the time taken to reach the stopping condition. The larger the instance, the larger the limit of the y-axis, since the randomly initialised string will correspond to roughly half of the sets used, as on average half of the bits will be set to 1 upon initialisation. The x-axis limits depend on the stopping condition for the instances, which is dependent on the combination of instance size and the percentage density of the instances.

The reasoning behind this can be explained as follows, starting from a random string leads to approximately half of the bit string initialised with 1s. Thus half of the total sets are used in the starting solution with high probability and this leads to a solution which covers all items and several of these items are covered multiple times due to their presence in several sets. This is due to the nature of the instances where every subset covers several rows. Therefore from the beginning, the solutions are all feasible as there is no uncovered element but they are very far away from the optimal solution. For a large portion of the iterations only a single solution is observed in this case, as the solutions generated by mutations which add any more sets are discarded and ones which remove sets replace the current solution. It is only towards the later stages that the solution contains items which are only covered once, and a mutation which may lead to them being uncovered can create a non-dominating solution. For the case of staring from all 0s the initial solution covers nothing and is an infeasible solution. From the very first mutation onwards sets are added to the solution which in turn leads to covering certain items. Iterative addition of sets leads to quick covering of items which moves the solution to the feasible region. Also it can be seen that mutation in different bit positions may lead to a set of non-dominating solutions being produced when the solutions are in the

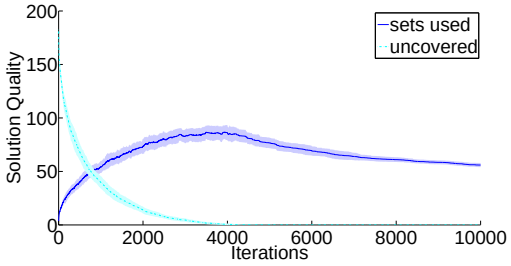
infeasible region. This set represents a population of solutions which improves the chances of a better solution being produced, as compared to a single solution when starting from a random string. Based on these results, the initial solution for GC-AIS is fixed to the all 0s bit string.

4.3.2 Parameter settings for PGSEMO

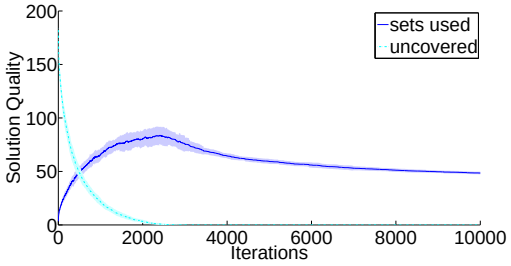
According to Algorithm 6 it can be seen that PGSEMO requires two parameters other than the mutation rate, to be manually tuned in order to get the best performance on a problem. These are the probability of communication p and the number of islands μ. For a fair comparison between the two algorithms in a parallel setting the number of islands should be equal or similar. The islands in a real parallel setting would represent the number of available nodes of a parallel computer and it is only fair that both algorithms are subject to same availability of resources. Since the population in GC-AIS is dynamic in nature, averaging over 30 runs, the maximum population reached in any iteration by GC-AIS is used as the limit for the number of islands for PGSEMO. This is calculated from the experiments in the previous subsection with the GC-AIS starting from all 0s string.

The probability of communication in the PGSEMO was initially based on the equation $p = \mu/(mn)$ from [110] where m is the number of rows and n is the number of columns. Experiments were performed to compare the performance of PGSEMO versus GC-AIS using this value for probability of communication. It can be seen that this setting results in poor performance of PGSEMO and more testing for a better value of p is needed. Different parametrised values of p are selected for this analysis, $p = 1/n$, $p = 1/m$ and $p = 1/\mu$ which are higher than $p = \mu/mn$. The results of the any-time behaviour of PGSEMO using these settings for the probability of communication can be seen in Figures 4.4, 4.5 and Figures 4.6-4.14 (see Appendix A).

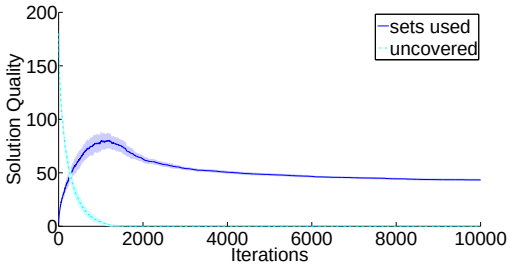
Figure 4.4: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance 41 shows setting with best performance.



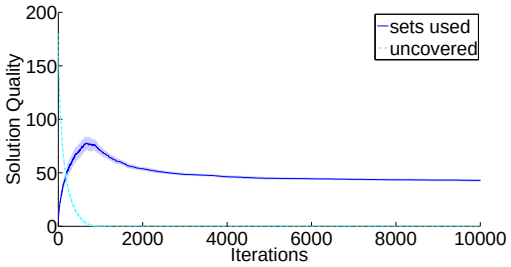
(a) $p = \mu/mn$, instance: 41



(b) $p = 1/m$, instance: 41

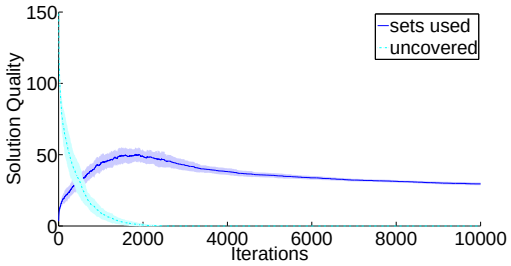


(c) $p = 1/n$, instance: 41

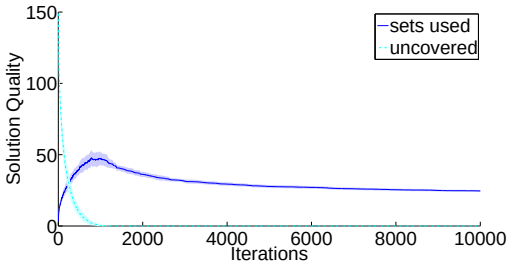


(d) $p = 1/\mu$, instance: 41

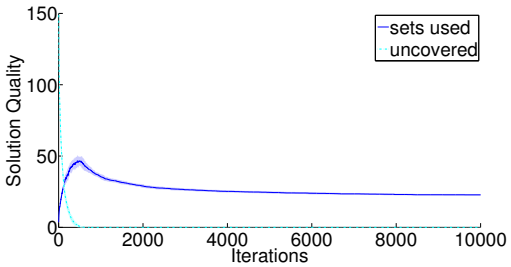
Figure 4.5: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance 63 shows setting with best performance.



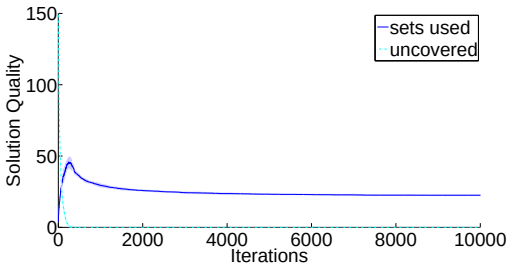
(a) $p = \mu/mn$, instance: 63



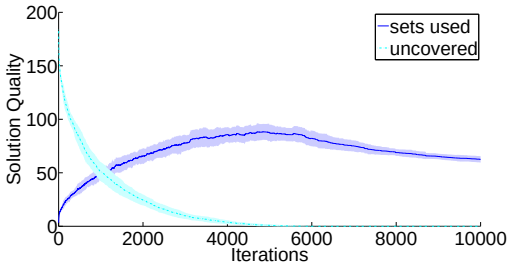
(b) $p = 1/m$, instance: 63



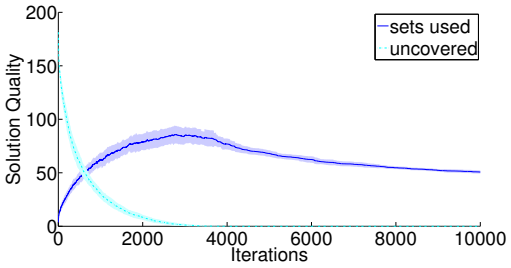
(c) $p = 1/n$, instance: 63



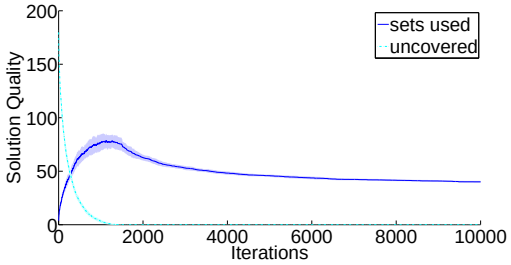
(d) $p = 1/\mu$, instance: 63



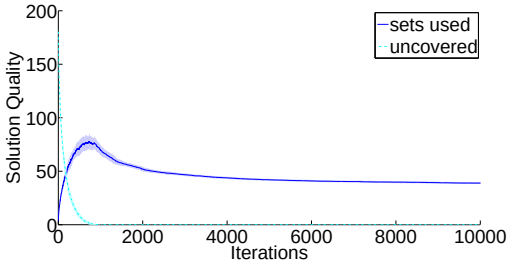
(a) $p = \mu/mn$, instance: 52



(b) $p = 1/m$, instance: 52

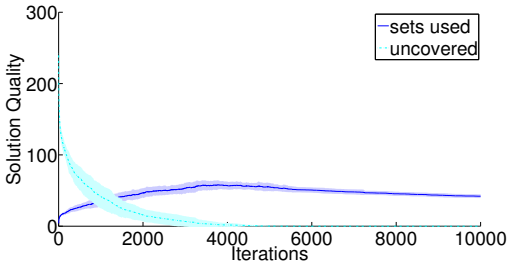


(c) $p = 1/n$, instance: 52

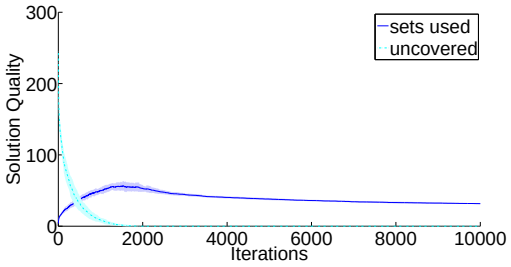


(d) $p = 1/\mu$, instance: 52

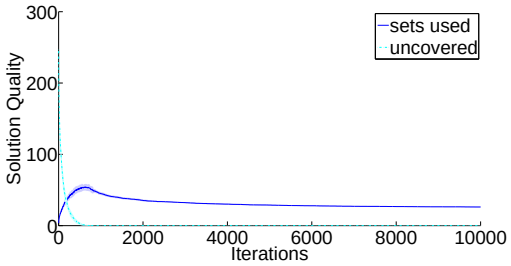
Figure 4.6: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance 52 shows setting with best performance. Standard deviation shown as shaded error-bars.



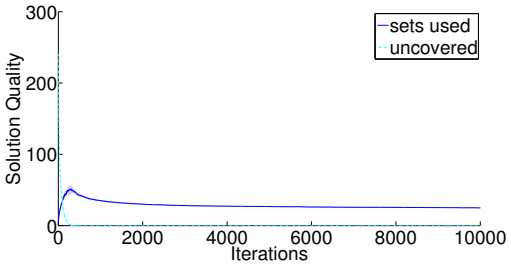
(a) $p = \mu/mn$, instance: b4



(b) $p = 1/m$, instance: b4

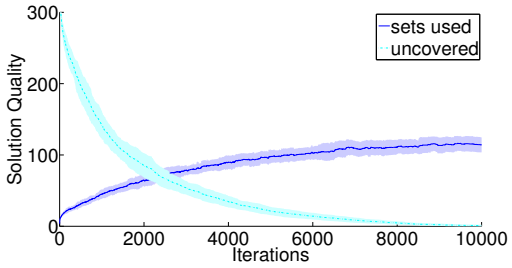


(c) $p = 1/n$, instance: b4

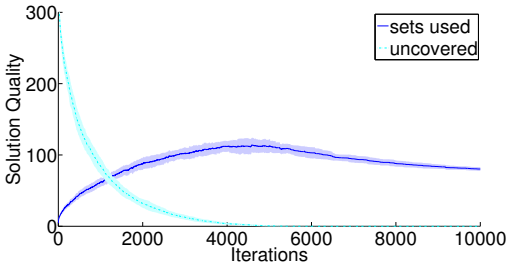


(d) $p = 1/\mu$, instance: b4

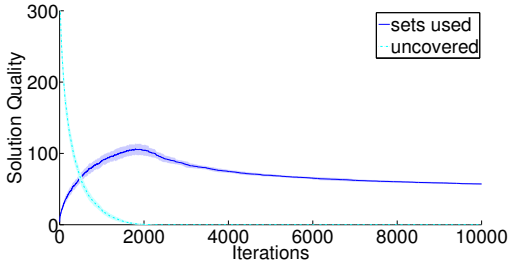
Figure 4.7: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance b4 shows setting with best performance. Standard deviation shown as shaded error-bars.



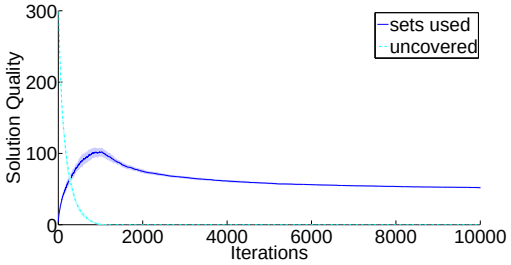
(a) $p = \mu/mn$, instance: c3



(b) $p = 1/m$, instance: c3

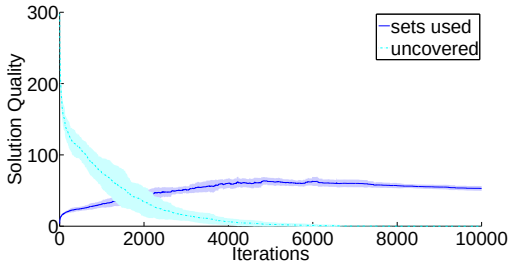


(c) $p = 1/n$, instance: c3

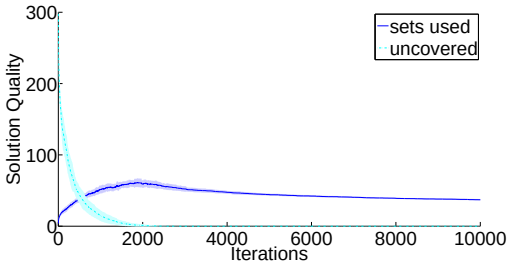


(d) $p = 1/\mu$, instance: c3

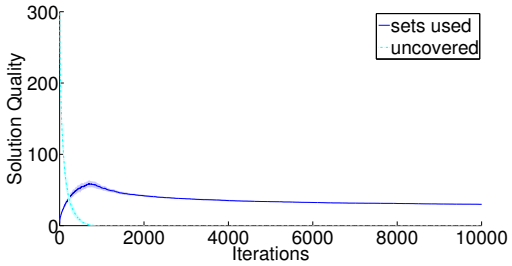
Figure 4.8: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance c3 shows setting with best performance. Standard deviation shown as shaded error-bars.



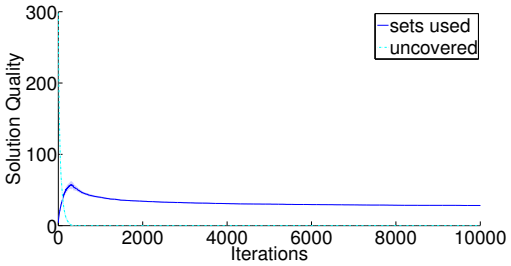
(a) $p = \mu/mn$, instance: d2



(b) $p = 1/m$, instance: d2

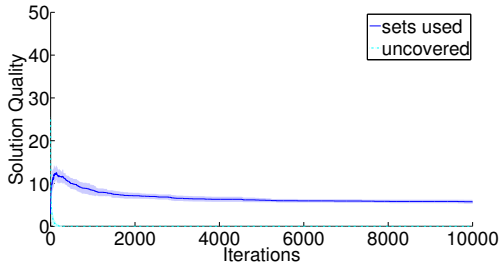


(c) $p = 1/n$, instance: d2

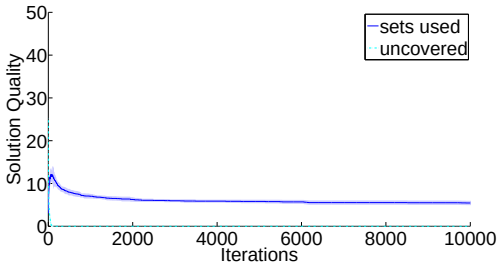


(d) $p = 1/\mu$, instance: d2

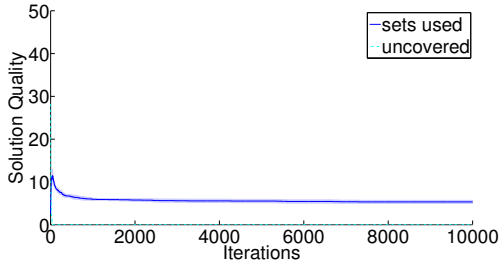
Figure 4.9: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance d2 shows setting with best performance. Standard deviation shown as shaded error-bars.



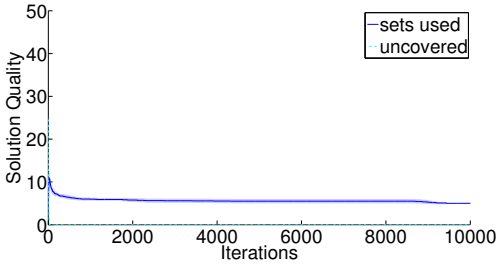
(a) $p = \mu/mn$, instance: e1



(b) $p = 1/m$, instance: e1

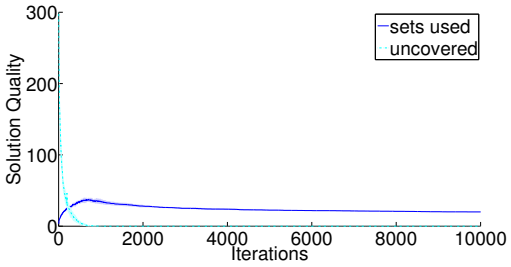


(c) $p = 1/n$, instance: e1

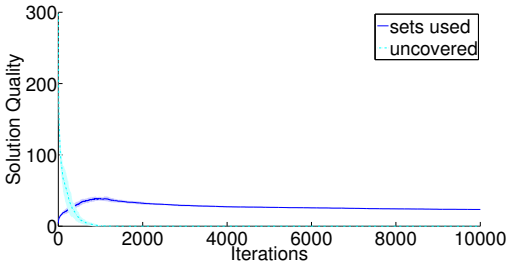


(d) $p = 1/\mu$, instance: e1

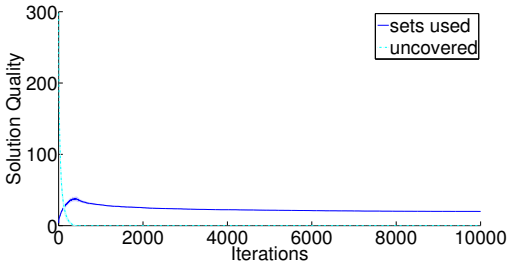
Figure 4.10: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance e1 shows setting with best performance. Standard deviation shown as shaded error-bars



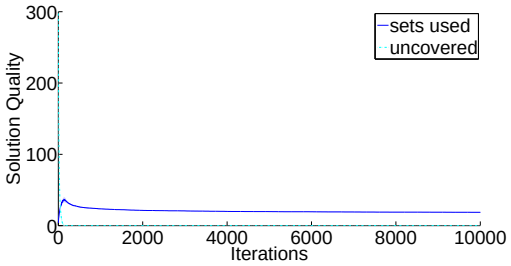
(a) $p = \mu/mn$, instance: re1



(b) $p = 1/m$, instance: re1

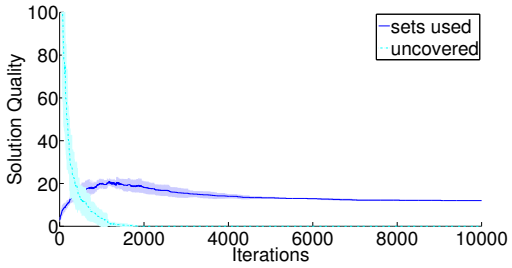


(c) $p = 1/n$, instance: re1

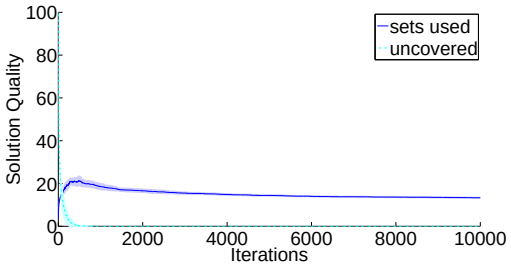


(d) $p = 1/\mu$, instance: re1

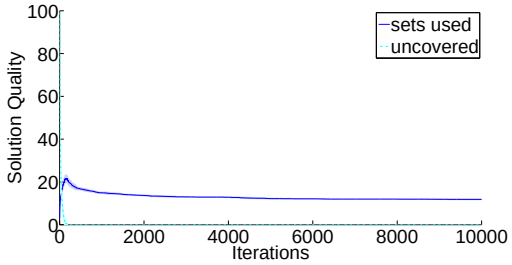
Figure 4.11: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance re1 shows setting with best performance. Standard deviation shown as shaded error-bars.



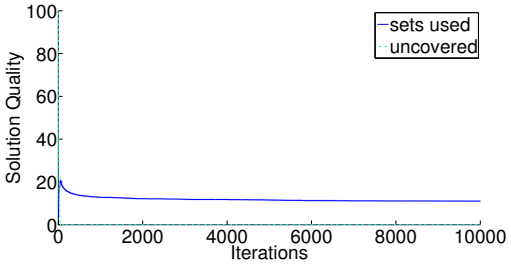
(a) $p = \mu/mn$, instance: rf2



(b) $p = 1/m$, instance: rf2

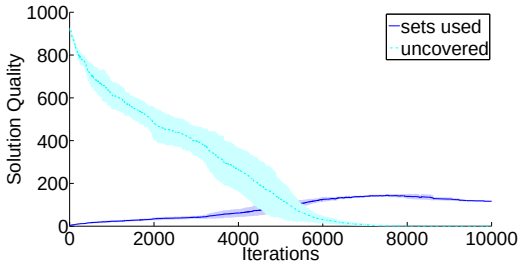


(c) $p = 1/n$, instance: rf2

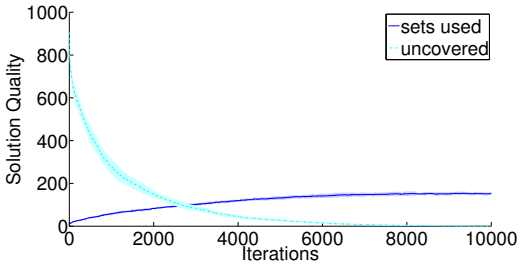


(d) $p = 1/\mu$, instance: rf2

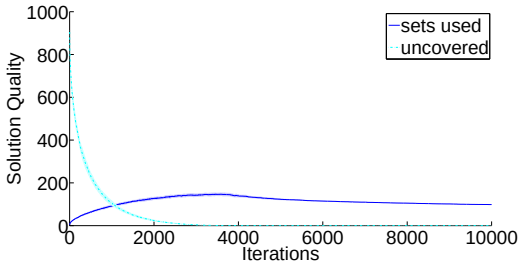
Figure 4.12: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance rf2 shows setting with best performance.Standard deviation shown as shaded error-bars.



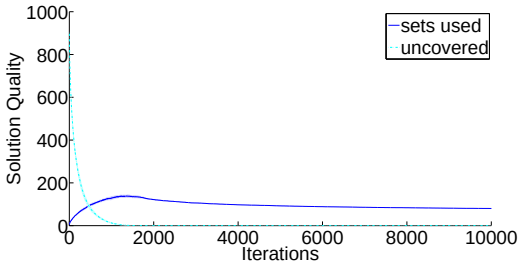
(a) $p = \mu/mn$, instance: rg3



(b) $p = 1/m$, instance: rg3

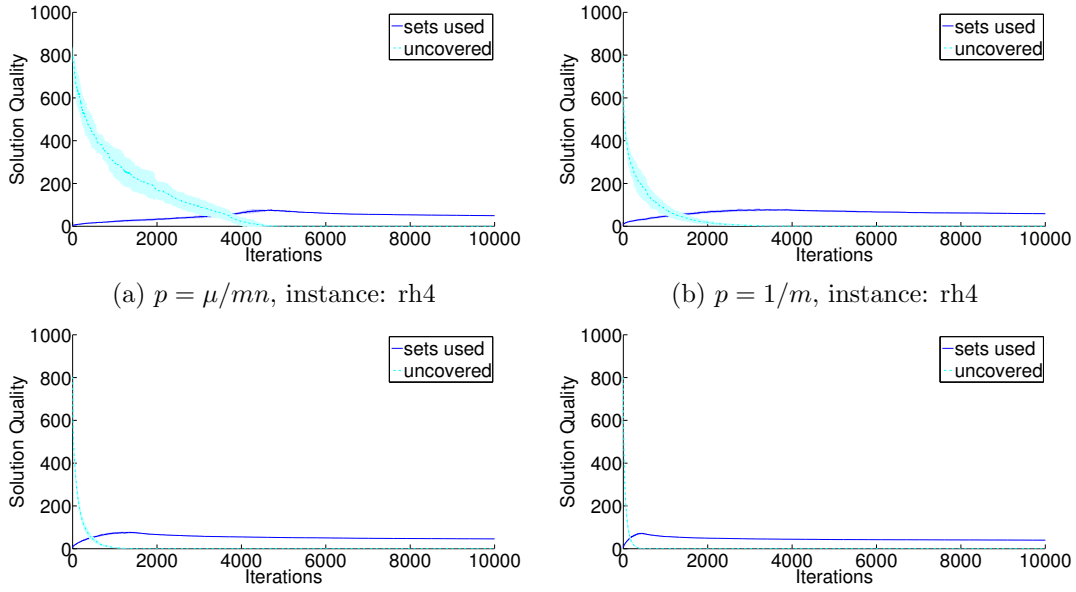


(c) $p = 1/n$, instance: rg3



(d) $p = 1/\mu$, instance: rg3

Figure 4.13: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance rg3 shows setting with best performance. Standard deviation shown as shaded error-bars.



(a) $p = \mu/mn$, instance: rh4

(b) $p = 1/m$, instance: rh4

(c) $p = 1/n$, instance: rh4

(d) $p = 1/\mu$, instance: rh4

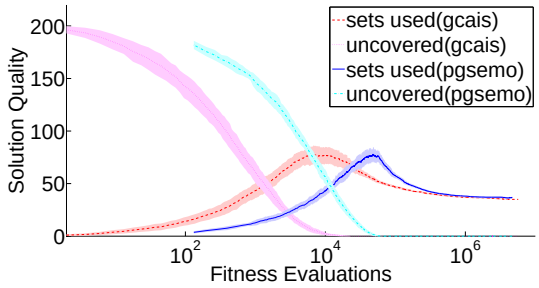
Figure 4.14: Plots of PGSEMO with various settings for the probability of communication p. 4 plots are shown per instance with $p = 1/m$, $p = 1/n$, $p = 1/\mu$ and $p = \mu/mn$. Solid blue line shows the sets used and the dot-dashed cyan line shows the uncovered elements. Bottom right plot for instance rh4 shows setting with best performance.

It can be seen from these representative plots that the probability $p = 1/\mu$, where on average one island communicates per iteration results in the best performance seen in the bottom-right sub-plots in the two sub-figures for each instance. Based on these experiments the probability of communication for PGSEMO is set to $1/\mu$ for experiments comparing GC-AIS and PGSEMO.

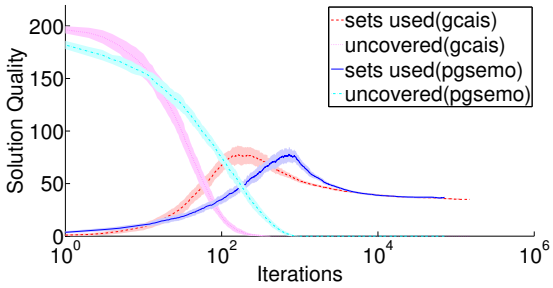
4.3.3 Experimental results

Finally experiments are performed to compare the performance of GC-AIS and PGSEMO using the parameter settings found in the previous two subsections. The first set of experiments analyses the solution quality that can be achieved by both the algorithms. Both algorithms are allowed to run until no improvement in the best solution is seen for at least $en \log n$ iterations. Each algorithms is given 30 runs per problem instance and the mean of the best solution quality per iteration as well as per fitness evaluation averaged over the runs is plotted. These results can be seen in Figures 4.15 and Figures 4.16 and 4.17.

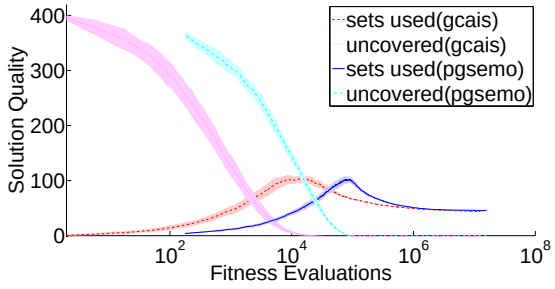
Figure 4.15: Run time plots for GC-AIS versus PGSEMO. Solution quality plotted as fitness evaluation are expended on left column of plots and as iteration complete on right column of plots for instances 52, c3, d2 and rg3. Solid blue lines show sets used and the dot-dashed cyan line shows the uncovered elements used by PGSEMO. The dotted magenta line shows the uncovered elements and the dashed red line shows sets used by GC-AIS. X-axes are plotted on a log scale and the standard deviation is shown as shaded error bars.



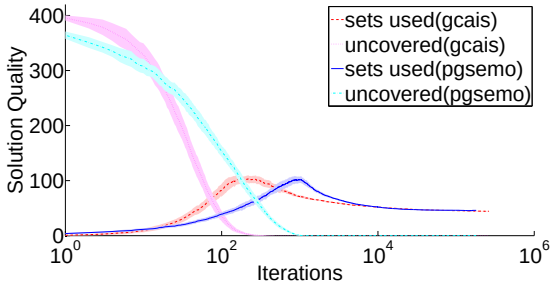
(a) instance: 52



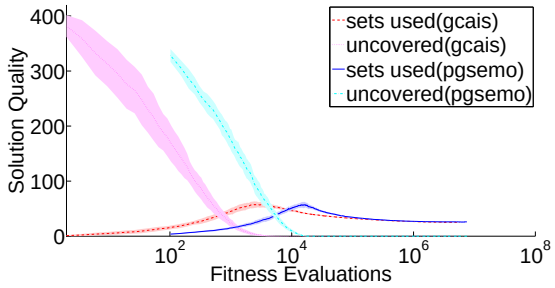
(b) instance: 52



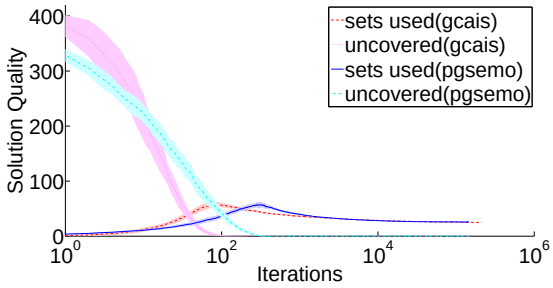
(c) instance: c3



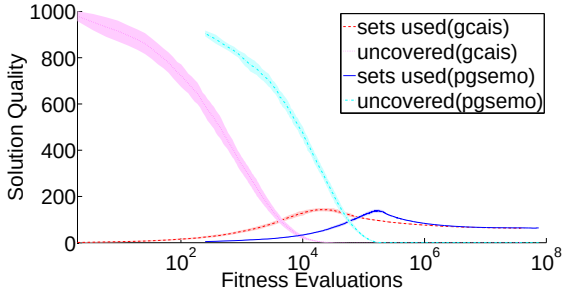
(d) instance: c3



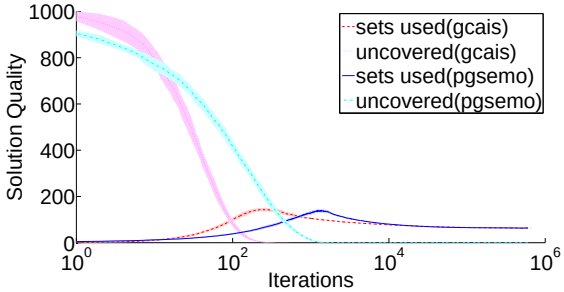
(e) instance: d2



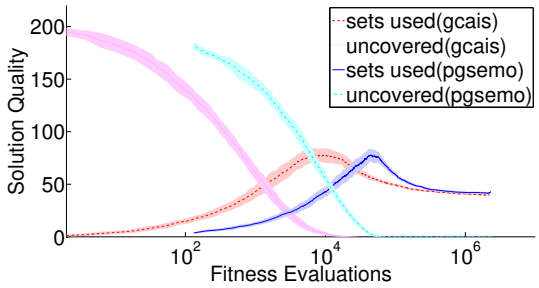
(f) instance: d2



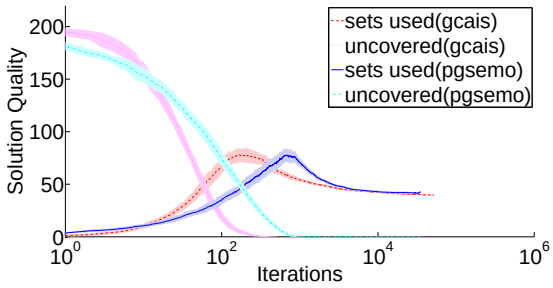
(g) instance: rg3



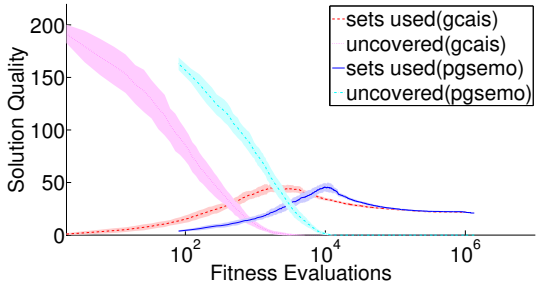
(h) instance: rg3



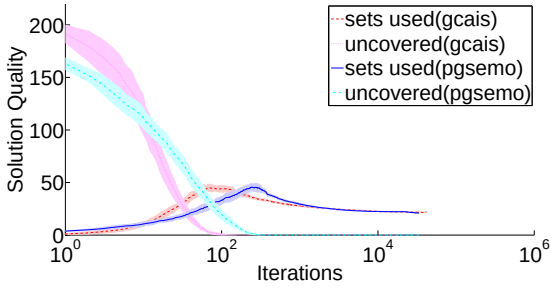
(a) instance: scp41



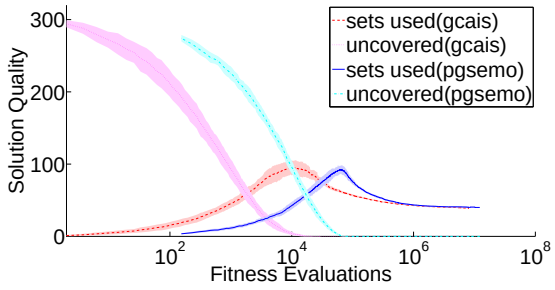
(b) instance: scp41



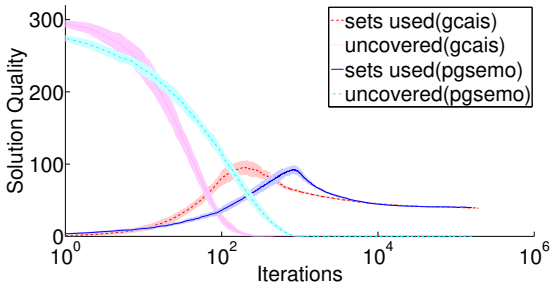
(c) instance: scp63



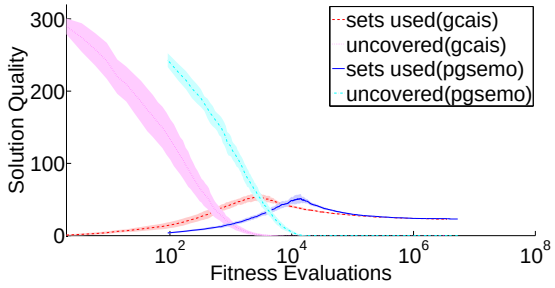
(d) instance: scp63



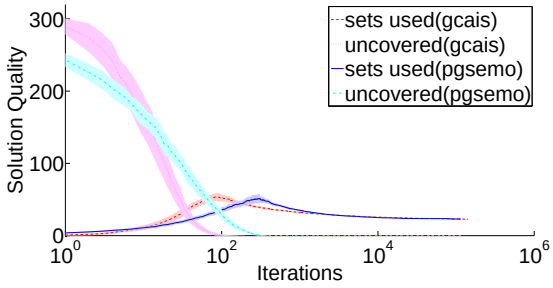
(e) instance: scpa5



(f) instance: scpa5

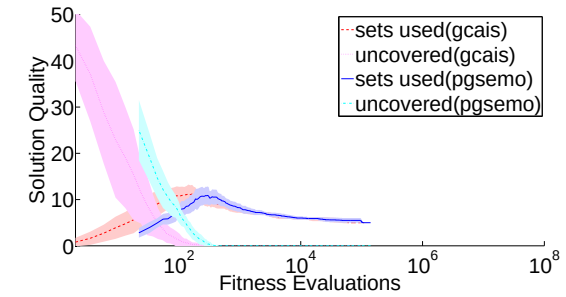


(g) instance: scpb4

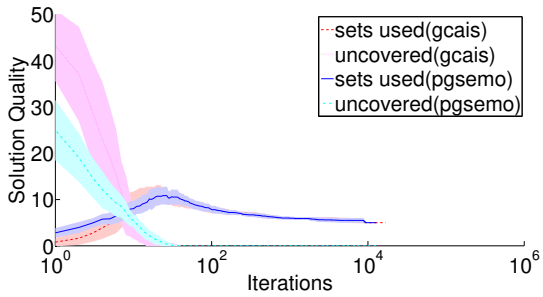


(h) instance: scpb4

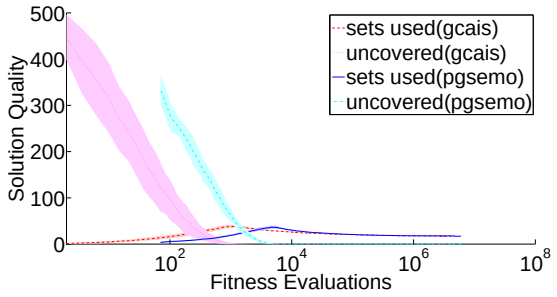
Figure 4.16: Run time plots for GC-AIS versus PGSEMO. Solution quality plotted per fitness evaluation on left column and per iteration on right column for instance 41, 63, a5 and b4. Solid blue lines show sets used and the dot-dashed cyan line shows the uncovered elements used by PGSEMO. The dotted magenta line shows the uncovered elements and the dashed red line shows sets used by GC-AIS. X-axes are plotted on a log scale and the standard deviation is shown as shaded error bars.



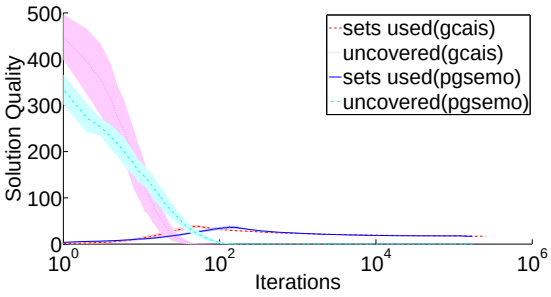
(a) instance: scpe1



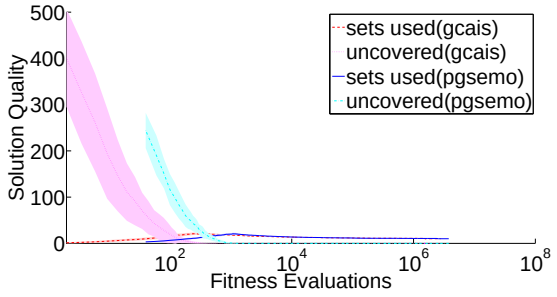
(b) instance: scpe1



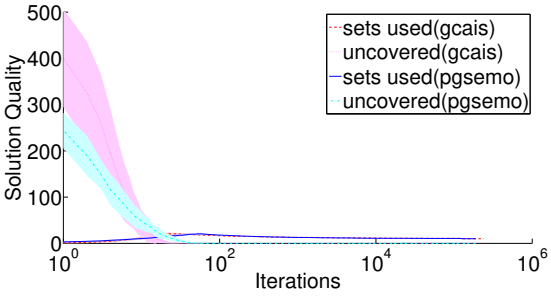
(c) instance: scpre1



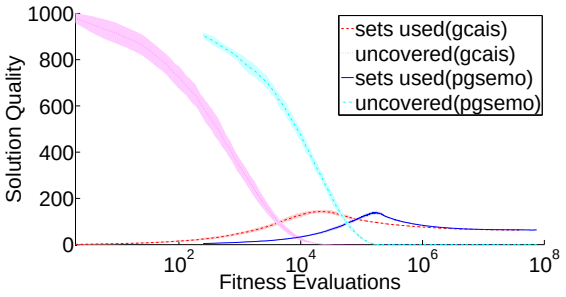
(d) instance: scpre1



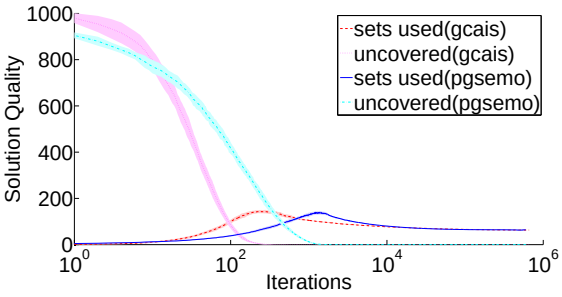
(e) instance: scprf2



(f) instance: scprf2



(g) instance: scpnrg3



(h) instance: scpnrg3

Figure 4.17: Run time plots for GC-AIS versus PGSEMO. Solution quality plotted per fitness evaluation on left column and per iteration on right column for instance e1, re1, rf2 and rg3. Solid blue lines show sets used and the dot-dashed cyan line shows the uncovered elements used by PGSEMO. The dotted magenta line shows the uncovered elements and the dashed red line shows sets used by GC-AIS. X-axes are plotted on a log scale and the standard deviation is shown as shaded error bars.

The next set of experiments are geared towards finding the iterations required to reach

a fitness value close to the best solution reported in literature. In most cases a solution either equal or very close to the best reported was observed, but sometimes it was not present in all the 30 runs per instance. This could happen for either of the algorithms, therefore the solution which was closest to the best reported solution, and which appeared in every run of both the algorithms was used for this test. The average number of iterations to reach this value are computed along with the Wilcoxon rank-sum test. These results can be seen in Table 4.2. The end of run performance of both the algorithms is also compared by plotting the iterations and evaluations used by the algorithms at the end along with the solution quality achieved at the end. Both the iterations and the fitness evaluations are recorded as error-bars along with the solution quality achieved. These results can be seen in Figures 4.18.

scp	$m \times n$	Fitness	AIS	PGSEMO	STest	g^{AIS}	g^{PGSEMO}	STest
41	200 × 1000	(0,45)	3605.7	4723.7	3.591e-11	**317.1**	816.1	3.01e-11
52	200 × 2000	(0,39)	9223.2	8818.2	0.85	**304.9**	809.3	3.01e-11
63	200 × 1000	(0,24)	3609.3	3097.3	0.04	**100.4**	268.9	2.98e-11
a5	300 × 3000	(0,42)	2.366e4	2.038e4	0.02	**283.2**	893.8	3.01e-11
b4	300 × 3000	(0,24)	2.236e4	1.504e4	1.193e-6	**106.7**	307.8	2.99e-11
c3	400 × 4000	(0,48)	2.269e4	1.885e4	3.56e-4	**295.6**	993.9	3.01e-11
d2	400 × 4000	(0,27)	1.841e4	1.490e4	0.003	**113.1**	321.7	2.99e-11
e1	50 × 500	(0,6)	1432.3	1416.7	0.51	**19.6**	23.6	0.026
re1	500 × 5000	(0,18)	2.341e4	1.277e4	1.15e-7	**51.3**	130.9	2.99e-11
rf2	500 × 5000	(0,11)	9379.2	5499.4	6.35e-5	**24.7**	47.4	3.58e-10
rg3	1000 × 10000	(0,65)	1.220e4	9.576e4	2.68e-4	**322**	1483.6	3.01e-11
rh4	1000 × 10000	(0,36)	6.475e4	4.719e4	2.67e-6	**125.7**	454.3	3.00e-11

Table 4.2: Comparison between GC-AIS and PGSEMO at the beginning of the feasible region and at the end of runs. Column entries in 'AIS' and 'PGSEMO' show the average number of generations required to reach the fitness values of the column 'Fitness'. Columns 'g^{AIS}' and 'g^{PGSEMO}' represent the generations required to reach the feasible region. 'STest' shows the p values obtained from the Wilcoxon rank-sum tests.

Finally the last set of experiment in this study pertains to the communication effort

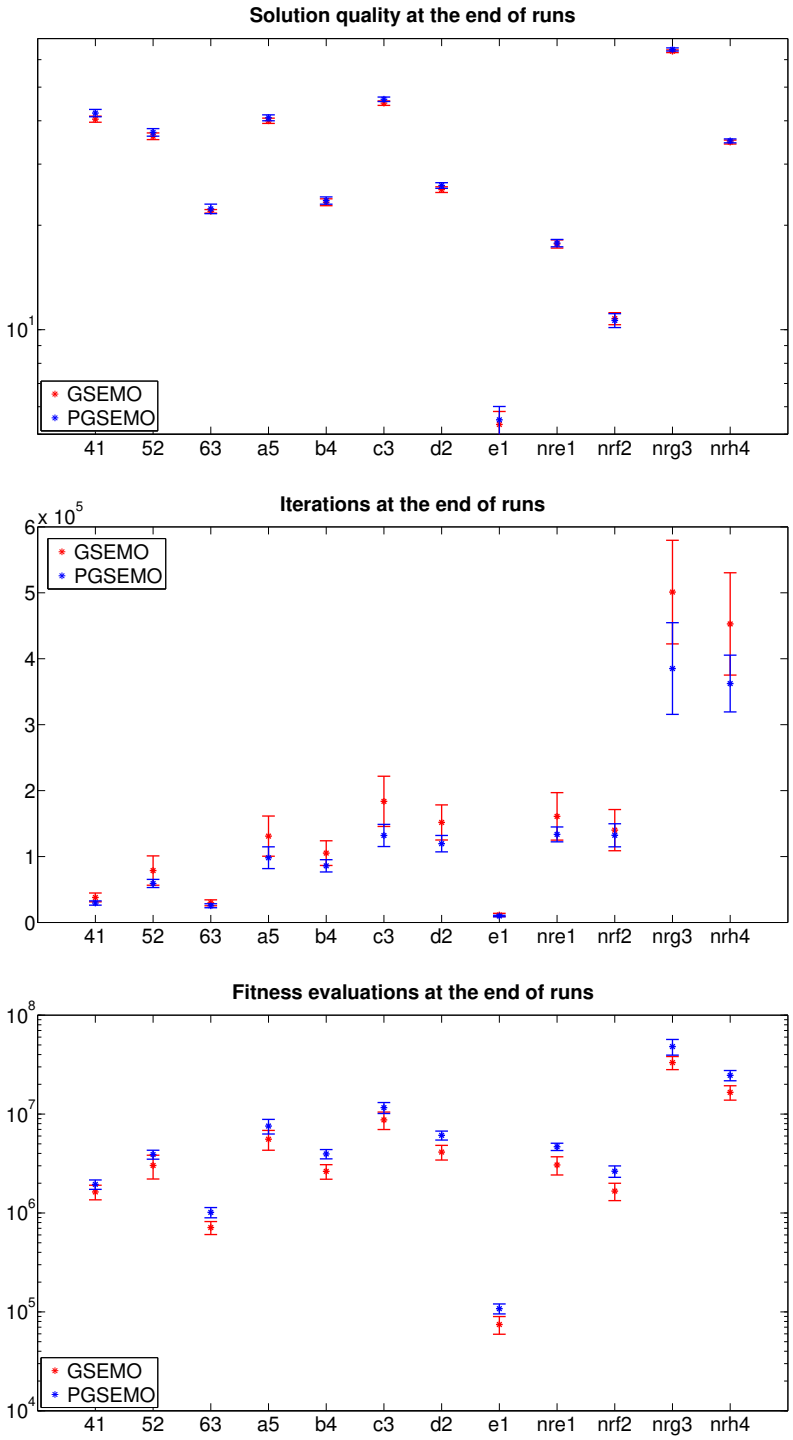
required by the two algorithms. To estimate the communication effort of the two algorithms, the number of individuals which are exchanged between islands per iteration are counted. The plots for the total number of communications until each generation t are shown in Figure 4.19 and 4.20.

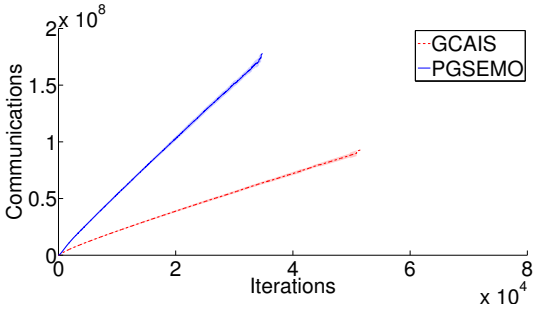
Discussion of Results

It can be seen from the plots in Figures 4.15 and Figures 4.16 and 4.17 that GC-AIS reaches feasible region much faster than PGSEMO which can be seen by observing that the magenta line touches the x axis before the cyan line. Also upon careful observation it can be seen that the solid red line towards the end of the runs for all plots has a value lower by a small fraction than the solid blue line. This means that the average solution quality obtained by GC-AIS is better than the solution quality obtained by PGSEMO. The impact of the stopping criteria can be seen on the size of the y-axis where it is observed that the instances with higher density and smaller size tend to stop earlier than the instances with lower density and large size. The x-axes limit depends on the problem instance size, as for the all 0s initialisation, the number of uncovered elements will be equal to the size of universe set for the instance, which reflects in the plots of uncovered elements on the y axis for both the algorithm. Another point of note is that in the sub figures comparing solution quality versus fitness evaluations, the plots for PGSEMO tend to be shifted to the right. This artefact can be explained by the fact that for PGSEMO μ islands are initialised, therefore increasing the fitness evaluations at the start to μ used at the time of initialisation. This trend is not seen in GC-AIS as only 1 GC is initialised there.

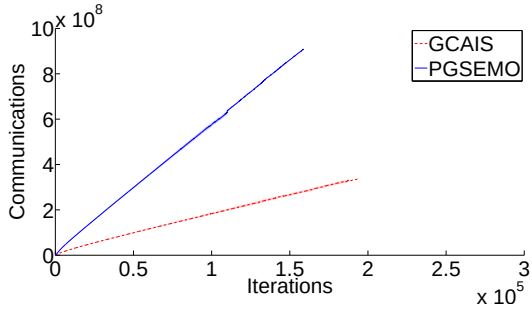
In the case of the sub figures comparing the solution quality versus iterations, it is seen that at the start PGSEMO has fewer uncovered elements than GC-AIS. This is explained due to the presence of μ initial solutions in PGSEMO which result in a overall faster improvement than GC-AIS which only has 1 GC in the initial stages of the runs. However this early advantage of PGSEMO is quickly lost, which can be seen

Figure 4.18: Plots of error-bars for end of run comparison between GC-AIS and PGSEMO. Topmost plot shows average solution quality in terms of sets used, middle and bottommost plots show the iterations elapsed and fitness evaluations expended by the algorithms at the end of runs.

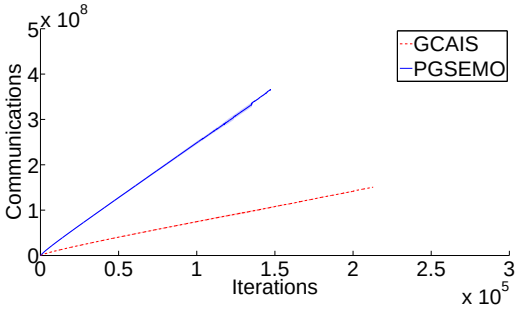




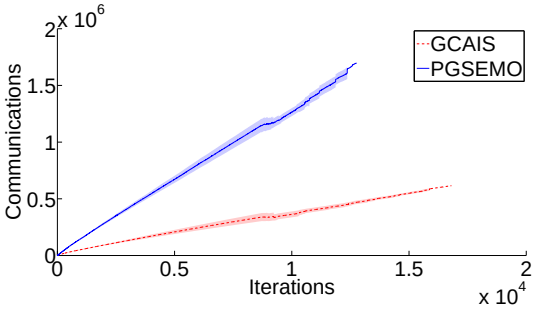
(a) instance: 41



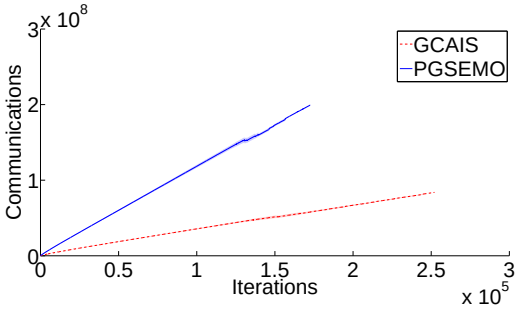
(b) instance: a5



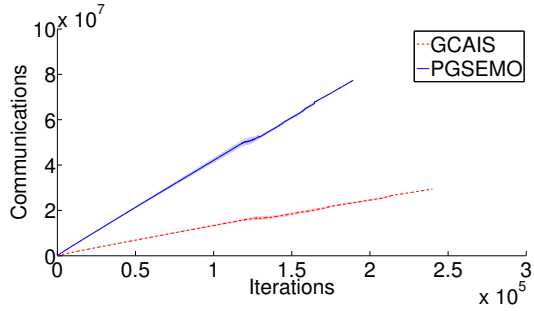
(c) instance: d2



(d) instance: e1

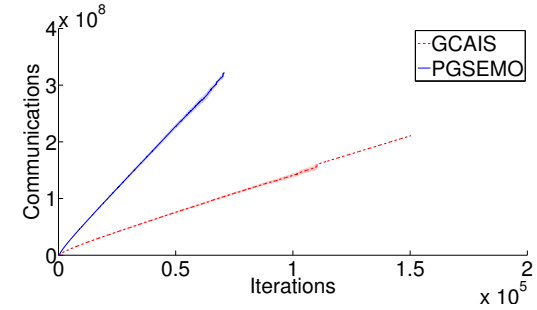


(e) instance: re1

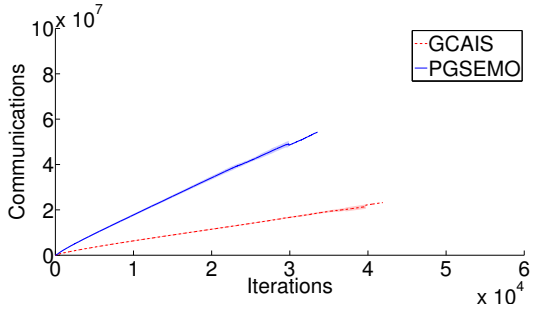


(f) instance: rf2

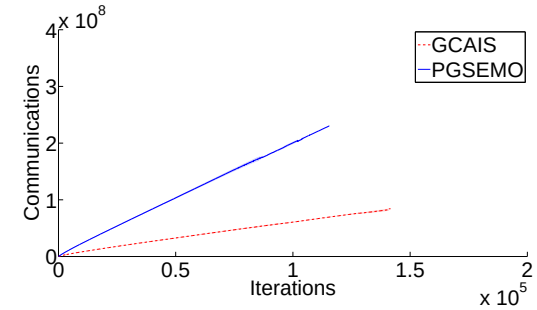
Figure 4.19: Plots of communication effort of GC-AIS versus PGSEMO. Solid blue line shows the average number of communications for PGSEMO while dashed red line shows the average number of communications for GC-AIS versus iterations of the algorithms. X-axes are plotted on a log scale and shaded error-bars depict the standard deviation.



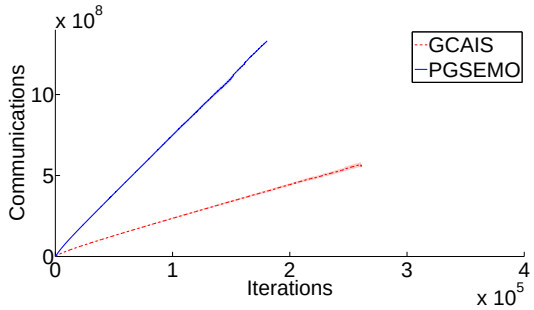
(a) instance: 52



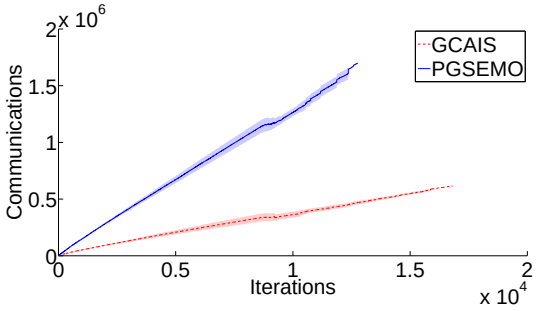
(b) instance: 63



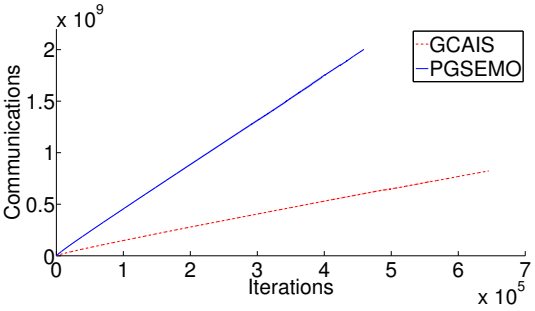
(c) instance: b4



(d) instance: c3



(e) instance: e1



(f) instance: rh4

Figure 4.20: Plots of communication effort of GC-AIS versus PGSEMO. Solid blue line shows the average number of communications for PGSEMO while dashed red line shows the average number of communications for GC-AIS versus iterations of the algorithms. X-axes are plotted on a log scale and shaded error bars depict standard deviation.

in the plots at the point where the cyan line intersects the magenta line. It is by these many iterations that GC-AIS is able to acquire a non-dominated population of solutions, due to the dynamic population size of the algorithms, that starts performing better than PGSEMO.

It can be seen from Table 4.2 that GC-AIS is faster than PGSEMO to reach the feasible region for all instance based on iterations. This is the same feature seen from the run-time plots. In all these instances the algorithms are statistically different according to the p values obtained from the rank-sum test at a confidence level of 0.05. On the other hand for majority of the instances it is seen that GC-AIS requires more iterations to improve towards the end. This is seen as more iterations to reach the near best known solutions, by GC-AIS. These results are complemented with end of run performance of both the algorithms, which are shown in Figure 4.18. Once again it can be seen that when considering iterations GC-AIS uses more time than PGSEMO to reach the same or very similar solution qualities. It is seen that instances 'rG' and 'rH' use an unusually higher number if iterations than the other instances. This is due to the large size and low density of these instances making them harder to solve, as for these instances each subset covers fewer elements of the universe set. On the other hand it is seen that when considering fitness evaluations GC-AIS is better than PGSEMO in all instances.

These seemingly contradictory results can be explained by considering the dynamic nature of GC-AIS. The number of islands in PGSEMO is fixed to the maximum number of GCs in GC-AIS. After reaching the feasible region the population of GCs starts to slowly decrease and gets lower than that of PGSEMO. This results in an uneven number of fitness evaluations per iteration. PGSEMO using more evaluations per iteration finds solutions faster when compared to GC-AIS in a parallel setting. Another factor which influences the performance of PGSEMO is the presence of a population of solutions in each island, whereas GC-AIS contains only one set of non-dominated solutions. Towards the end, due to communication and non-dominated removal, the solutions in each population tend to be very similar in PGSEMO.

It can be seen from the plots that GC-AIS requires less communication effort than PGSEMO on all the instances. This is due to the fact that PGSEMO transfers the entire population of solutions to its neighbours when a communication occurs while the communication in GC-AIS is only limited to transferring the newly created offspring to other GCs.

4.4 Testing the performance of GC-AIS on MOd-KP

After the analysis on SCP the next problem considered for examining with GC-AIS is the MOd-KP. As introduced in Section 2.4.3 the goal of the MOd-KP is to fill multiple knapsacks with items in such a way as to maximise profits without exceeding the capacity of the sacks. This problem has been studied extensively and is considered as a suitable test bed to compare GC-AIS with NSGA-II, one of the state of the art and extremely popular multi-objective evolutionary algorithms used to solve this problem [132]. The reason to select a state of the art algorithm for this problem as shown in literature, and not PGSEMO is due to the promising results seen for the SCP. As mentioned before, a simple algorithm was needed to test in the initial state to see if GC-AIS can compete with it. After verifying the fact that GC-AIS can indeed compete with a simple algorithm, in the next set of experiments it makes sense to see how it fares against a state of the art algorithm. Also, another factor which prohibits the selection of PGSEMO for MOd-KP is the lack of any theoretical analysis about the performance guarantee of PGSEMO on MOd-KP.

4.4.1 Set-up for MOd-KP experiments

In this section the experimental set-up used for this study is described. GC-AIS and the NSGA-II are compared by testing them on benchmark instances of the multi-objective knapsack problem taken from literature. A total of 12 instances are provided in [170] and are grouped into 3 classes based on the number of knapsacks. Each class consists of 4

Algorithm parameters	NSGA-II	ϵ-GC-AIS
Initial Population	Refer Table-4.4	1
Mutation Probability	1	1
Mutation rate	5/n	1/n
Crossover Probability	0.8	-

Table 4.3: Parameter settings for GC-AIS and NSGA-II to be used for MOd-KP

instances based on the number of total items. The different numbers of knapsack in these instances are 2, 3 and 4 and the number of items are 100, 250, 500 and 750 (see Section 3.3). 30 independent runs each with one million fitness evaluations are performed for each algorithm and statistical results are recorded. The number of fitness evaluations is set to one million as these are the maximum evaluations used in the previous work by [132] done on this problem and therefore it helps to link this study with previous results from literature.

It has shown previously in [85] that the weighted scalar repair approach provides better results than greedy repair, due to increasing the diversity of the solutions and also helps in increasing converge speed. Authors of [132] have shown that the performance of algorithms used to solve MOd-KP greatly depends on the choice of the repair approach. It is for this reason that the weighted scalar approach for repair is used in this study and a random weight vector is generated for each solution that needs repair.

In order to set the parameters for the algorithms, previous studies on the same problem using NSGA-II are considered. Three parameters must be set for the NSGA-II while GC-AIS only requires one, to get the possible best settings for the algorithms. NSGA-II requires the population size, probability of crossover and rates of mutation to be set while GC-AIS only needs mutation rate to be set. The settings for NSGA-II have been kept as close to the work in [170, 168] which have also been utilised by [132]. The rate of mutation for NSGA-II has been set to $5/n$ based on the guidelines in [107] while [132] use a fixed mutation rate of 0.006 which is not optimal. These settings are shown in Table 4.3. The population size for NSGA-II must be set separately for each instance of the problem according to the size and dimension of the instances. These settings are taken from [132]

	Knapsacks		
Items	2	3	4
100	150	200	250
250	150	200	250
500	200	250	300
750	250	300	400

Table 4.4: Population size used in NSGA-II for different instances of MOd-KP

and are shown in table 4.4. The choice of these parameters is heavily based on settings from previous literature on the problem, which helps compare the current experimental results with those in the literature.

In order to evaluate the metrics for the comparison of the algorithms, reference fronts must be generated as true Pareto fronts for all the problem instances are not available [170]. By following the procedure of creating reference fronts explained in Section 3.5, both the NSGA-II and GC-AIS are run 30 times for one million fitness evaluations each and the non-dominated solutions are recorded. The final non-dominated fronts from each run are combined and all dominated solutions are removed. The resulting fronts are used as reference Pareto fronts for the generation of performance measures.

4.4.2 Experimental Results

The contribution of the two algorithms to the generation of the Pareto fronts are first calculated. Per run contribution refers to the contribution in percentage of each run of the algorithm towards the final Pareto front. Overall contribution refers to the combined contribution in percentage of all the runs of the algorithm towards the final Pareto front. It must be noted that since the different runs of the two algorithms can contain identical solutions the overall contribution of both algorithms can sum to more than 100%. These results can be seen in Tables 4.5, 4.6 and 4.5

		GC-AIS		NSGA-II	
Sacks	items	Per run	Final	Per run	Final
2	100	$\mathbf{40.409_{(3.742)}}$	**96.491**	$2.398_{(2.656)}$	48.246
	250	$\mathbf{3.474_{(2.536)}}$	**99.614**	$0.013_{(0.070)}$	0.3861
	500	$\mathbf{3.345_{(5.791)}}$	**100**	$0_{(0)}$	0
	750	$\mathbf{3.350_{(4.870)}}$	**100**	$0_{(0)}$	0

Table 4.5: Percentage contribution of NSGA-II and GC-AIS to the generation of reference fronts for 2 sacks. Column 'Per Instance' shows average contribution of the algorithms per run, column 'Final' shows average contribution of the algorithms by combining solutions from all the runs. Standard deviation is shown as the subscript for column 'Per Instance'. Values in bold show higher percentage contribution.

		GC-AIS		NSGA-II	
Sacks	items	Per run	Final	Per run	Final
3	100	$\mathbf{3.763_{(1.159)}}$	**97.401**	$0.087_{(0.888)}$	2.5987
	250	$\mathbf{3.231_{(3.092)}}$	**96.493**	$0.102_{(0.070)}$	3.0571
	500	$\mathbf{3.096_{(8.167)}}$	**92.886**	$0.237_{(0.321)}$	7.1141
	750	$\mathbf{3.013_{(4.693)}}$	**90.409**	$0.319_{(0.348)}$	9.5914

Table 4.6: Percentage contribution of NSGA-II and GC-AIS to the generation of reference fronts for 3 sacks. Column 'Per Instance' shows average contribution of the algorithms per run, column 'Final' shows average contribution of the algorithms by combining solutions from all the runs. Standard deviation is shown as the subscript for column 'Per Instance'. Values in bold show higher percentage contribution.

		GC-AIS		NSGA-II	
Sacks	items	Per run	Final	Per run	Final
4	100	$\mathbf{3.272}_{(\mathbf{1.807})}$	**97.991**	$0.066_{(0.040)}$	2.0093
	250	$\mathbf{3.194}_{(\mathbf{3.619})}$	**95.825**	$0.139_{(0.089)}$	4.1754
	500	$\mathbf{3.044}_{(\mathbf{4.784})}$	**91.333**	$0.288_{(0.163)}$	8.6671
	750	$\mathbf{2.931}_{(\mathbf{4.468})}$	**87.953**	$0.401_{(0.214)}$	12.047

Table 4.7: Percentage contribution of NSGA-II and GC-AIS to the generation of reference fronts for 4 sacks. Column 'Per Instance' shows average contribution of the algorithms per run, column 'Final' shows average contribution of the algorithms by combining solutions from all the runs. Standard deviation is shown as the subscript for column 'Per Instance'. Values in bold show higher percentage contribution.

In order to compare the end-of-run performance of the two algorithms, the average values of the performance metrics are calculated over all runs. Along with the standard deviation, statistical testing using the Wilcoxon rank-sum test is performed for each instance of the problem. The rank-sum test is used to measures the level of significant difference between the algorithms at a threshold of 0.05. These metric values along with the statistical results are shown in Tables 4.8, 4.9, and 4.10 for generational distance, Tables 4.11 , 4.12 and 4.13 for hypervolume and finally Tables 4.14, 4.15 and 4.16 for diversity.

		GC-AIS	NSGA-II	
Sacks	items	average	average	rank-sum
2	100	$\mathbf{0.0007}_{(\mathbf{0.0001})}$	$0.003_{(0.0006)}$	3.019e-11
	250	$\mathbf{0.001}_{(\mathbf{0.0001})}$	$0.007_{(0.0005)}$	3.019e-11
	500	$\mathbf{0.001}_{(\mathbf{0.0001})}$	$0.013_{(0.0007)}$	3.019e-11
	750	$\mathbf{0.0006}_{(\mathbf{0.0001})}$	$0.023_{(0.001)}$	3.019e-11

Table 4.8: Values for the generational distance metric obtained for NSGA-II and GC-AIS averaged over 30 runs of the algorithms for 2 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the p values obtained for the Wilcoxon rank-sum test. Values in bold show winner based on lower generational distance.

		GC-AIS	NSGA-II	
Sacks	items	average	average	rank-sum
3	100	$\mathbf{0.001}_{(\mathbf{8.98e-5})}$	$0.007_{(0.0004)}$	3.019e-11
	250	$\mathbf{0.0008}_{(\mathbf{0.0001})}$	$0.013_{(0.0013)}$	3.019e-11
	500	$\mathbf{0.0009}_{(\mathbf{0.0002})}$	$0.014_{(0.0014)}$	3.019e-11
	750	$\mathbf{0.0008}_{(\mathbf{0.0002})}$	$0.013_{(0.0010)}$	3.019e-11

Table 4.9: Values for the generational distance metric obtained for NSGA-II and GC-AIS averaged over 30 runs of the algorithms for 3 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the p values obtained for the Wilcoxon rank-sum test. Values in bold show winner based on lower generational distance.

		GC-AIS	NSGA-II	
Sacks	items	average	average	rank-sum
4	100	$\mathbf{0.0005}_{(\mathbf{8.37e-5})}$	$0.012_{(0.0007)}$	3.019e-11
	250	$\mathbf{0.0005}_{(\mathbf{0.0001})}$	$0.015_{(0.0016)}$	3.019e-11
	500	$\mathbf{0.0005}_{(\mathbf{0.0001})}$	$0.011_{(0.009)}$	3.019e-11
	750	$\mathbf{0.0007}_{(\mathbf{0.0002})}$	$0.005_{(0.0006)}$	3.019e-11

Table 4.10: Values for the generational distance metric obtained for NSGA-II and GC-AIS averaged over 30 runs of the algorithms for 4 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the p values obtained for the Wilcoxon rank-sum test. Values in bold show winner based on lower generational distance.

		GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
2	100	$\mathbf{1.69e+07_{(2.12e4)}}$	$1.68e+07_{(3.86+e04)}$	2.37e-07
	250	$\mathbf{9.60e+07_{(4.08+e05)}}$	$9.52e+07_{(3.68+e05)}$	2.57e-07
	500	$\mathbf{3.93e+08_{(1.8e+06)}}$	$3.84e+08_{(1.37e+06)}$	7.38e-11
	750	$\mathbf{8.45e+08_{(3.68e+06)}}$	$8.13e+08_{(2.89e+06)}$	3.01e-11

Table 4.11: Values for the hypervolume metric obtained for NSGA-II and GC-AIS averaged over 30 runs of the algorithms for 2 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the p values obtained for the Wilcoxon rank-sum test. Values in bold show winner based on higher hypervolume.

		GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
3	100	$\mathbf{6.12e+10_{(3.99e+08)}}$	$6.08e+10_{(2.97e+08)}$	0.15
	250	$8.17e+11_{(6.76e+09)}$	$\mathbf{8.27e+11_{(6.82e+09)}}$	0.14
	500	$6.33e+12_{(3.80e+10)}$	$\mathbf{6.40e+12_{(4.98e+10)}}$	4.11e-06
	750	$2.14e+13_{(1.81e+11)}$	$2.14e+13_{(1.43e+11)}$	0.079

Table 4.12: Values for the hypervolume metric obtained for NSGA-II and GC-AIS averaged over 30 runs of the algorithms for 3 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the p values obtained for the Wilcoxon rank-sum test. Values in bold show winner based on higher hypervolume.

		GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
4	100	$1.62e+14_{(1.96e+12)}$	$1.62e+14_{(1.33e+12)}$	0.569
	250	$6.19e+15_{(8.08e+13)}$	$\mathbf{6.41e+15_{(7.18e+13)}}$	3.01e-11
	500	$8.93e+16_{(1.03e+15)}$	$\mathbf{9.72e+16_{(9.22e+14)}}$	3.01e-11
	750	$4.21e+17_{(4.9e+15)}$	$\mathbf{4.79e+17_{(5.88e+15)}}$	3.01e-11

Table 4.13: Values for the hypervolume metric obtained for NSGA-II and GC-AIS averaged over 30 runs of the algorithms for 4 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the p values obtained for the Wilcoxon rank-sum test. Values in bold show winner based on higher hypervolume.

		GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
2	100	$0.58_{(0.039)}$	$\mathbf{0.55_{(0.055)}}$	0.0251
	250	$0.75_{(0.026)}$	$\mathbf{0.68_{(0.037)}}$	4.182e-9
	500	$0.77_{(0.025)}$	$\mathbf{0.71_{(0.039)}}$	2.782e-7
	750	$\mathbf{0.60_{(0.039)}}$	$0.67_{(0.036)}$	3.645e-8

Table 4.14: Values for the diversity metric obtained for NSGA-II and GC-AIS averaged over 30 runs of the algorithms with 2 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the p values obtained for the Wilcoxon rank-sum test. Values in bold show winner based on lower diversity.

		GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
3	100	$0.44_{(0.016)}$	$\mathbf{0.41_{(0.025)}}$	1.429e-5
	250	$0.43_{(0.012)}$	$\mathbf{0.42_{(0.025)}}$	0.0270
	500	$0.45_{(0.010)}$	$0.45_{(0.026)}$	0.818
	750	$\mathbf{0.47_{(0.010)}}$	$0.49_{(0.024)}$	0.0021

Table 4.15: Values for the diversity metric obtained for NSGA-II and GC-AIS averaged over 30 runs of the algorithms for 3 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the p values obtained for the Wilcoxon rank-sum test. Values in bold show winner based on lower diversity.

		GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
4	100	$\mathbf{0.33_{(0.007)}}$	$0.36_{(0.020)}$	2.601e-8
	250	$\mathbf{0.35_{(0.006)}}$	$0.41_{(0.020)}$	3.019e-11
	500	$\mathbf{0.37_{(0.006)}}$	$0.42_{(0.016)}$	3.019e-11
	750	$\mathbf{0.37_{(0.007)}}$	$0.42_{(0.020)}$	9.918e-11

Table 4.16: Values for the diversity metric obtained for NSGA-II and GC-AIS averaged over 30 runs of the algorithms for 4 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the p values obtained for the Wilcoxon rank-sum test. Values in bold show winner based on lower diversity.

Finally, to better understand the behaviour of the algorithms, the run-time plots of the population sizes of both NSGA-II and GC-AIS have been plotted and Figure 4.21 and Figures 4.22, 4.23 and 4.24show the population sizes during run-time of the algorithms.

Discussion of results

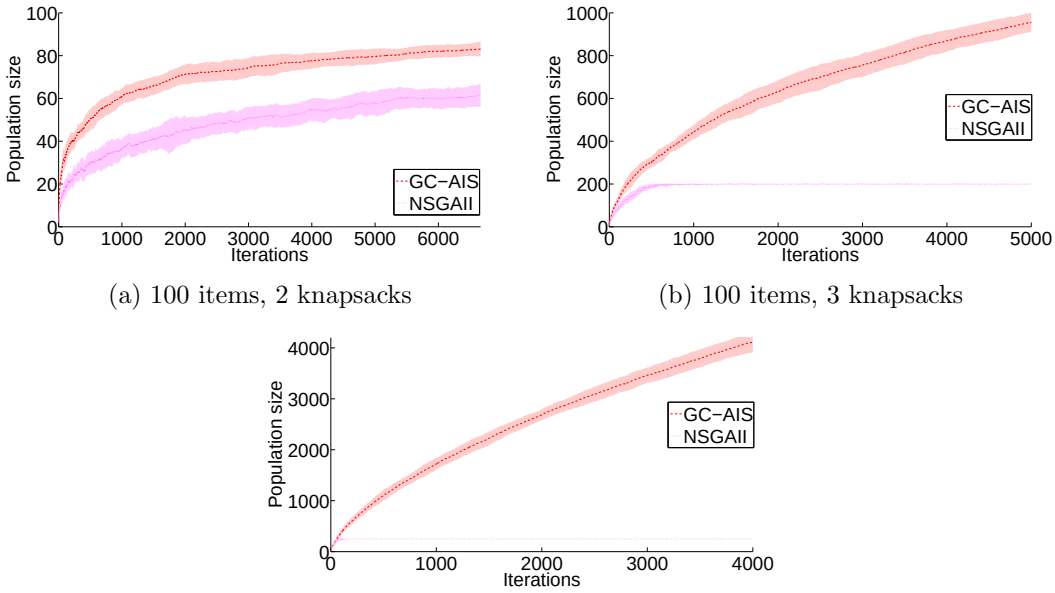
It can be seen from Table 4.5, 4.6 and 4.5 that for all the 12 instances of the MOd-KP, GC-AIS contributes the most towards the generation of the Pareto front and the contribution of NSGA-II is is very low. This is shown by the bold text in the tables.

It can be seen from the statistical test for the generational distance that the algorithms are statistically different on all instances. The values for generational distance show that on all instances, the fronts achieved by GC-AIS are closer to the reference fronts than the fronts achieved from NSGA-II. The table for hypervolume on the other hand shows mixed results. For instances with small dimensions the hypervolume covered by GC-AIS is more than NSGA-II. But as the potential solutions in the instances increase due to increased size, the hypervolume of NSGA-II becomes better.

On 11 out of the 12 instances the rank-sum test shows significant difference between the algorithms with respect to the diversity measure. In 6 of these instances the diversity of solutions in GC-AIS is better than NSGA-II. It is hard to clearly define a winner in terms of diversity as for most of the instances with two or three dimensions NSGA-II has better diversity but for instances with 4 dimensions solutions from GC-AIS are better in terms of diversity.

A drastic increase of the population size in GC-AIS can be seen from the Figure 4.21, 4.22, 4.23, 4.24. For 2 knapsacks the population of both the algorithms is similar while as the number of knapsacks increase GC-AIS starts accumulating considerably more solutions than NSGA-II. This is due to the dynamic population size of GC-AIS, and as the search space increases due to the increase in dimension of the problem the possible number of non-dominated solutions increases. This can be used to explain the behaviour of the diversity metric. This large population of solutions in the higher dimensional cases, is due to the increase in the search space. This leads to the potential non-dominated solutions to increase dramatically too. GC-AIS does not restrict the population of solutions and thus accumulated many non-dominated solutions which may not be very far from each other. This is avoided in NSGA-II by 2 methods, the crowding distance and explicit limit on

the population size. Therefore NSGA-II contains a fixed number of solutions which are more spread out, while GC-AIS contains a very large number of solutions which may have areas that are more dense than others along the front.

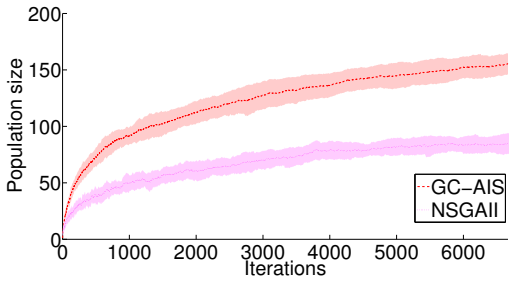


(a) 100 items, 2 knapsacks

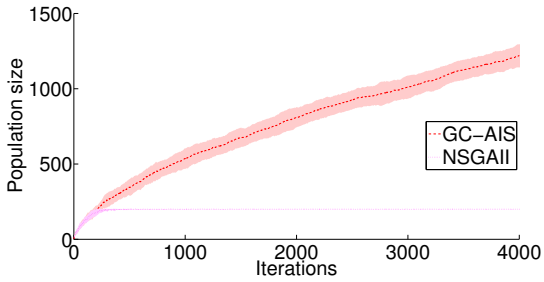
(b) 100 items, 3 knapsacks

(c) 100 items, 4 knapsacks

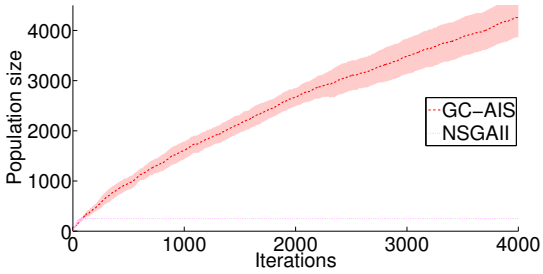
Figure 4.21: Runtime plots for population for NSGA-II and GC-AIS showing population explosion for problem size 100 with 2, 3 and 4 objectives. Solid magenta line shows the average population of NSGA-II while dashed red line shows average population of GC-AIS plotted versus the algorithm iterations. Shaded error bars depict the standard deviation.



(a) 250 items, 2 knapsacks

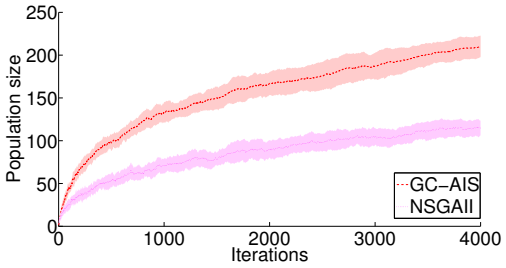


(b) 250 items, 3 knapsacks

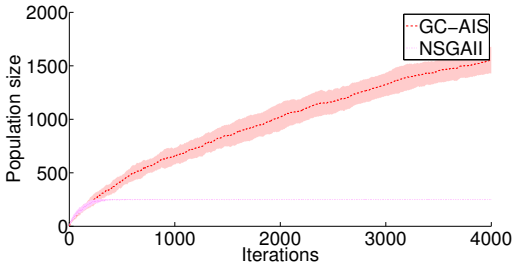


(c) 250 items, 4 knapsacks

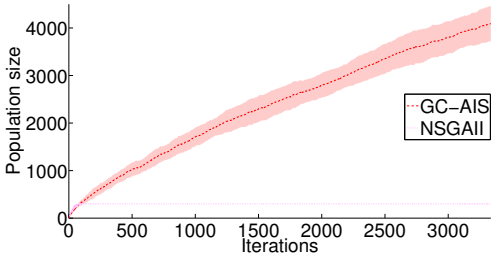
Figure 4.22: Runtime plots for population for NSGA-II and GC-AIS showing population explosion for problem size 100 with 2, 3 and 4 objectives. Solid magenta line shows the average population of NSGA-II while dashed red line shows average population of GC-AIS plotted versus the algorithm iterations. Shaded error bars depict the standard deviation.



(a) 500 items, 2 knapsacks

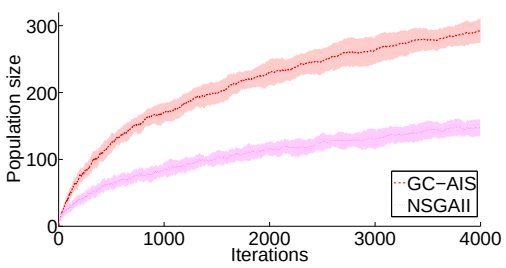


(b) 500 items, 3 knapsacks

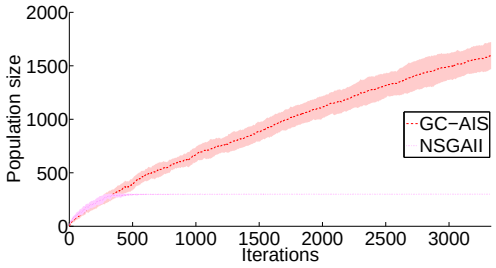


(c) 500 items, 4 knapsacks

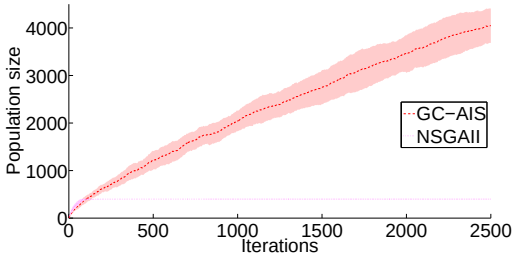
Figure 4.23: Runtime plots for population for NSGA-II and GC-AIS showing population explosion for problem size 100 with 2, 3 and 4 objectives. Solid magenta line shows the average population of NSGA-II while dashed red line shows average population of GC-AIS plotted versus the algorithm iterations. Shaded error bars depict the standard deviation.



(a) 750 items, 2 knapsacks



(b) 750 items, 3 knapsacks



(c) 750 items, 4 knapsacks

Figure 4.24: Runtime plots for population for NSGA-II and GC-AIS showing population explosion for problem size 100 with 2, 3 and 4 objectives. Solid magenta line shows the average population of NSGA-II while dashed red line shows average population of GC-AIS plotted versus the algorithm iterations. Shaded error bars depict the standard deviation.

4.5 Discussion and Conclusion

The GC-AIS is compared with PGSEMO on the static version of SCP and some promising results are observed. GC-AIS has some useful features over PGSEMO, such as finding the feasible region faster and utilising less communication effort. Considering a parallel setting, during the early stages of the runs, GC-AIS has more uncovered elements than PGSEMO, which is explained by the initial islands of PGSEMO that result in more overall individuals than GC-AIS. This advantage is lost rather quickly and GC-AIS finds the feasible region faster, due to the build up of non-dominated individuals in GC-AIS. The end-of-run performance of PGSEMO was better than GC-AIS but this was at the cost of high communication effort and fitness evaluations. This is due to the fact that during later stages of the run, islands of PGSEMO converge such that they all have similar non-dominated solution sets. This increases the chances of an improving move, as compared to GC-AIS which only has one non-dominated set. Another enticing feature of GC-AIS is that no parameters require tuning considering a fixed mutation rate,

while it is an important part for PGSEMO. The fact that parameter tuning is not required in GC-AIS is important as this has the potential to save a lot of initial time which is spent in setting up of experiments by carefully tuning the parameters. This makes it ideal for practitioners who either lack the knowledge of parameter tuning or simply do not have the time to spend on many trials in parameter tuning, to use GC-AIS. Though at this stage, it cannot be said that GC-AIS should be the first choice for solving the SCP since a comparison was only performed versus a simple MOEA, yet it can be a given a consideration where time is more important than other resources such as processing powers, as no set up time on parameter tuning needs to be considered.

After the first set of experiments on GC-AIS against a simple MOEA, it was then compared versus a state of the art MOEA for a more realistic multi-objective problem, MOd-KP. When studying the performance on MOd-KP, GC-AIS had mixed results. GC-AIS contributes more than NSGA-II towards the generation of reference fronts which is evident by the fact that the generational distance metric showed that GC-AIS managed to achieved fronts closer than NSGA-II to the reference fronts. But the diversity of the solutions and the hypervolume covered by NSGA-II was better in majority of the instances. The main cause for this issue seems to be the population explosion seen in GC-AIS when the objectives of the problem increases, causing the solutions in the search space to increase dramatically. Clearly an uninhibited population is not desired as it can lead to fitness evaluations being wasted for solutions which are too close to each other. This is the reason for NSGA-II to achieve better diversity and hypervolume, as due to the controlled size of population the steps of iterations resulted in a more diverse spread of solutions, while in GC-AIS the steps of the iterations were wasted in generation of an uncontrolled number of solutions too close to each other resulting in lower diversity and hypervolume. Another disadvantage of a very large number of solutions in the population is hindrance for the decision maker as it makes the decision to select desired solutions from the given Pareto front all the more difficult.

In conclusion, GC-AIS has shown potential and some encouraging results have been

observed on the SCP and MOd-KP. At the same time a key issue, namely population explosion has been identified which leads to poor performance when considering problems with higher number of objectives. This issue of population size explosion causes the number of solutions in the non-dominated front too increase at a very fast pace during the course of evolution. A possible drawback of this is that having a very large number of solutions can lead to utilising the fitness evaluation quota inefficiently. More research is required to tackle the issue with population explosion and this is addressed in Chapter 5.

4.6 Summary

A new artificial immune algorithm called the GC-AIS is presented which is motivated by new advancements in immunology. The performance of GC-AIS on the set cover problem along with multi-objective d-dimensional knapsack problem has been investigated. On the SCP, GC-AIS has shown encouraging results when compared with a simple MOEA, namely the PGSEMO. Along with performing better than the PGSEMO in terms of time and solution quality, GC-AIS also has the added advantage of requiring no parameter tuning. These features make it a strong contended to try GC-AIS for the SCP, in cases where set up time for parameter tuning is not available. The results for MOd-KP however were not so impressive, when GC-AIS was compared with a state of the art algorithm for the problem. On contribution towards the front generation and generational distance GC-AIS performed better than NSGA-II while on diversity and hypervolume metric NSGA-II beat GC-AIS. The issue behind the weak performance of GC-AIS was identified as the phenomenon of population size explosion which needs to be addressed.

CHAPTER 5

IMPROVING THE PERFORMANCE OF GC-AIS

5.1 Introduction

The GC-AIS has shown some promising results and possesses some interesting qualities like a few parameters than the algorithms it is compared with and a dynamic population size. Despite the advantages a key weaknesses has also been reported namely that of population explosion. The research question addressed here are

> *Knowing certain weaknesses in* GC-AIS*, how can improvements be made to overcome the shortcomings?*

which can be divided into two aspects, namely overcoming population explosion as well as re-evaluation of performance on SCP and MOd-KP to determine if the changes were successful.

The chapter is outlined as follows: Section 5.2 describes the population explosion problem of GC-AIS. ϵ-dominance, the mechanism we used to tackle the problem, is introduced in Section 5.3 which is followed by a description of the ϵ-GC-AIS. The performance of ϵ-GC-AIS is evaluated in the Sections 5.5 and 5.6 which is followed by discussion and conclusion and finally a summary of the Chapter is provided at the end. [2].

[2]Part of work presented in this chapter has been published in [95]

5.2 Understanding population explosion in GC-AIS

In Chapter 4 the population of GC-AIS was shown to explode in problems with higher dimensions. The cause of this is the dynamic and unbounded population size of GC-AIS along with the high dimensionality of the problem. As problem size increases along with the number of objectives of the problem the search space of the problem becomes larger. For these problems the non-dominated solutions begin to accumulate in the pool of solutions for GC-AIS. Since each generation can potentially double the population size, if all mutated clones are non-dominating with the parents this can cause a very large increase in the population size. The chances, that this extreme case happens are rare, what is observed is that some solutions from the final pool are discarded, these could be clones or parents which are dominated by others in the pool. Yet overall a gain in the population size is seen in every generation. Generating too many solutions for large problems with many objectives can be problematic. The decision maker for multi-objective optimisation needs to be presented with several solutions on the non-dominated front in order to guide his/her decision, but this can become useless if the number of solutions is beyond reasonable bounds [170]. Another potential problem of a large population of solutions is that it can lead to 'wasteful' fitness evaluations. A cluster of solutions which are close to each other may all lead to a single or very similar better solution by mutation and since each of the solutions in the cluster uses a fitness evaluation, these could be saved if only one solution was mutated. Add to that, a cluster of solutions is undesirable as it will not be well spread along the Pareto front.

It is clear from this discussion that some methods for controlling the population size of solutions must be incorporated into the GC-AIS to control the population size. Most multi-objective evolutionary algorithms employ a static population size amalgamated by operators which help to promote diversity and convergence. The NSGA-II maintains a fixed size population along with using crowding distance during its runs. A natural approach to select a desired solution from a group of solutions is clustering. Clustering as an approach for controlling the size of the Pareto set has been used in [170] and [127].

The method is invoked when the size of the non-dominated front exceeds the externally stored archive and relies on calculation of distance between all possible clusters in the non-dominated front and selecting a representative solution from each cluster. In addition to clustering, fitness assignment based on a measure called *strength* is employed in [170] and [169], and both these procedures together are needed to maintain diversity of solutions and control population in the non-dominated archive solution.

ϵ-dominance is a technique often employed in multi-objective evolutionary algorithm to maintain diversity and promote convergence to the Pareto optimal set. It was proposed in [106] and the procedure has a practical advantage that is the ϵ-approximate front obtained contains fewer solutions than the obtained non-dominated front using ordinary dominance relation while still aiming to converge to the Pareto front and promoting good spread. This method was used as stated to aid the diversity and convergence of the solution set, in algorithms with fixed size populations and archives. This method is discussed in detail in the next section.

5.3 The concept of ϵ-dominance

ϵ-dominance is a generalization of the dominance relation, which can be conceptually visualized for the two dimensional case as follows: the objective space is divided into a grid of rectangles of dimensions ϵ and solutions are mapped onto this grid. Assuming maximization without the loss of generality, if two solutions are in different boxes then they are compared using the standard dominance relation, but if they are in the same box then they are compared based on their euclidean distance from the origin, and the solution with the larger distance is kept. Figure 5.1 gives a diagram of the ϵ-dominated front, where the red circles represent the dominated points and the blue circles represent the point on the ϵ-dominated front.

More formally the concept of ϵ dominance as described in [106] can be defined as

Definition 6 *Let $f, g \in \mathbb{R}^m_+$ where m denotes the dimension, f and g are vectors along*

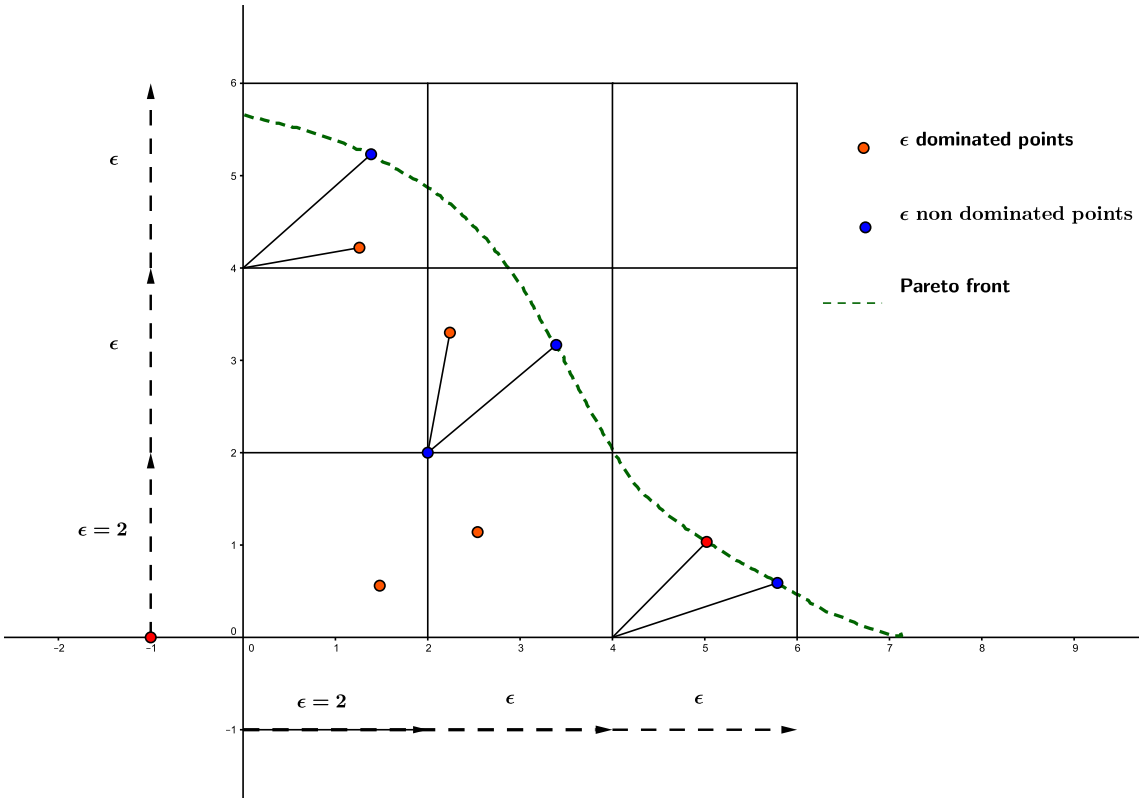


Figure 5.1: An illustrative diagram of ϵ non-dominated front with dominated points. Blue points on the dashed curve show the ϵ non-dominated solutions while red points show the ϵ dominated solutions, where the dashed curve is the pareto frontier. An ϵ value of 2 is is used which divides the search space in a grid of squares shown in solid lines.

positive real numbers $\mathbb{R}_+^m$. *f is said to dominated g for some* $\epsilon_i > 0$, *depicted as* $f \succ_\epsilon g$, *iff for all* $i \in \{1, \cdots, m\}$

$$\epsilon_i + f_i \geq g_i \tag{5.1}$$

Based on Definition 6 the ϵ approximate set can be defined as

Definition 7 *Given a set of vectors* $F \subseteq \mathbb{R}_+^m$ *and* $\epsilon > 0$, *then the set* F_ϵ *is an* ϵ*-approximate Pareto set of F, if any vector* $g \in F$ *is* ϵ*-dominated by at least one* $f \in F_\epsilon$.

$$\forall g \in F : \exists f \in F_\epsilon \text{ such that } f \succ_\epsilon g \tag{5.2}$$

Since ϵ-dominance permits only 1 solution inside each hypercube of side ϵ, it helps promote diversity as accepted solutions lie within difference hypercubes this in turn promotes convergence as only non-dominated solutions are permitted. Being a generalisation of regular dominance, and having both the properties of maintaining convergence and diversity it is a great choice to incorporate ϵ-dominance into GC-AIS. Since the choice of utilising other techniques, such as clustering and crowding distance would add additional components to the algorithm, while the dominance relation can easily be extended to use ϵ-dominance.

5.4 The ϵ-GC-AIS

The ϵ-GC-AIS algorithm replaces the dominance operator in GC-AIS with the ϵ dominance as explained in the previous section. The resulting algorithm is quite similar to GC-AIS and is introduced in Algorithm 9 with the key difference that the removal of dominated solutions is now performed by ϵ dominance.

A brief description of ϵ-GC-AIS is as follows. The ϵ-GC-AIS starts with one GC which contains one individual that represents a B cell. Offspring are created by standard bit mutation of B cells in GCs. Standard bit mutation means that each bit can be

Algorithm 9 The ϵ-GC-AIS

Let G^t denote the population of GCs at generation t and g_i^t the i-th GC in G^t.
Create GC pool $G^0 = \{g_1^0\}$ and initialise g_1^0. Let $t := 0$.
loop
  for each GC g_i^t in pool G^t in parallel **do**
    Create clone y_i of individual g_i^t by standard bit mutation.
  end for
  Add all y_i to G^t, remove all **ϵ-dominated** solutions from G^t and let $G^{t+1} = G^t$.
  Let $t = t + 1$.
end loop

flipped with the probability $1/n$. In every generation there is a migration of fitness values of the offspring between GCs, which corresponds to the migration of Ab between GCs. After migration, ϵ-dominated solutions are deleted which correspond to the eradication of GCs. The surviving offspring correspond to new GCs. This leads to a model where the number of GCs is dynamic in nature. The difference in this modified GC-AIS from the original model is the incorporation of ϵ-dominance which replaces the previous standard dominance relation. The ϵ-GC-AIS always maintains a set of ϵ-non-dominated solutions in every generation and the population size of GCs is dynamic in nature. This leads to a potential smaller overall population size per each iteration of the algorithm than if regular dominance criteria was used, which in turn mitigates the population size explosion. This also helps the decision maker as the number of final solutions in the non-dominated front are limited in this case therefore making the choice of a final solution easier.

5.5 Testing the performance of ϵ-GC-AIS on SCP

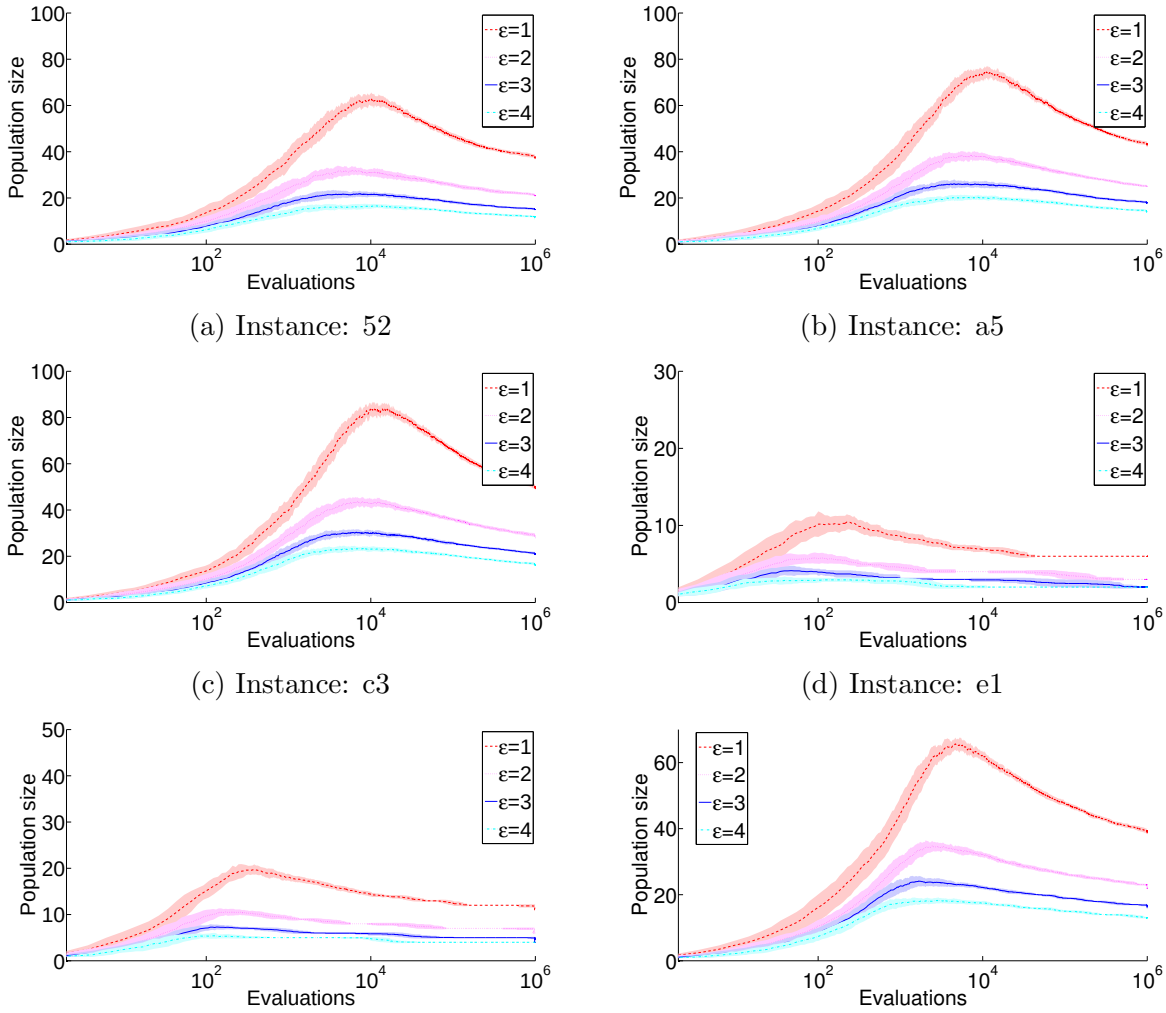
This section presents the experimental analysis performed in order to compare the performance of ϵ-GC-AIS on the SCP with a state of the art MOEA for the problem, namely NSGA-II [130]. Following the comparative study from the previous chapter, between GC-AIS and PGSEMO and obtaining promising results, it now makes sense to compare it with a state of the art algorithm, after incorporating ϵ-dominance.

5.5.1 Testing the effect of introducing ϵ-dominance

In the previous chapter parameter tuning for GC-AIS was not an issue as there were no other parameters to be optimised if mutation rate is kept fixed. After the introduction of ϵ-dominance in ϵ-GC-AIS a suitable value of ϵ must be found. In chapter 4 it was seen that population explosion was a big issue for problems with more than 1 objectives while the SCP only has 2 dimensions. Introducing a large ϵ can remove a lot of solutions from the non-dominated front in this case, and therefore smaller values of $\epsilon = 1, 2, 3, 4$ are tested. A value of $\epsilon = 1$ essentially reduces the ϵ dominance to behave as regular dominance for this problem. This is due to the nature of the instances which only allow whole number values of the objectives, therefore the smallest possible ϵ is 1 which eliminates the case of any two solutions to occupy the same box, making it behave like normal dominance.

These settings of ϵ-GC-AIS are tested to see the effect on the population size. Each ϵ setting of the algorithm is allowed to run for 10^6 fitness evaluations for 30 independent runs and the average population size per iteration of the algorithm is recorded. The number of fitness evaluations is set based on the experiments in the previous chapter where in Figures 4.15 and Figures 4.17, 4.16 (see Appendix A) it can be seen that by 10^6 GC-AIS has converged.

Run-time plot of the population size shown in Figure 5.2 and 5.3 (see Appendix **??**) show how the population size varies as the algorithm progresses. It can easily be distinguished that a small value of ϵ is able to reduce the population size, negating any effects of population explosion. The average final population at the end of the run is shown as a histogram in Figure 5.4 where it is seen that the final population size of ϵ-GC-AIS for all 12 instances is less than 100 and therefore, the value of $\epsilon = 2$ is set for further experiments. Population explosion was particularly obvious for the MOd-KP problem seen in Chapter 4, these experiments are performed here for the sake of completeness and to ascertain if it is really an issue for SCP.



(a) Instance: 52

(b) Instance: a5

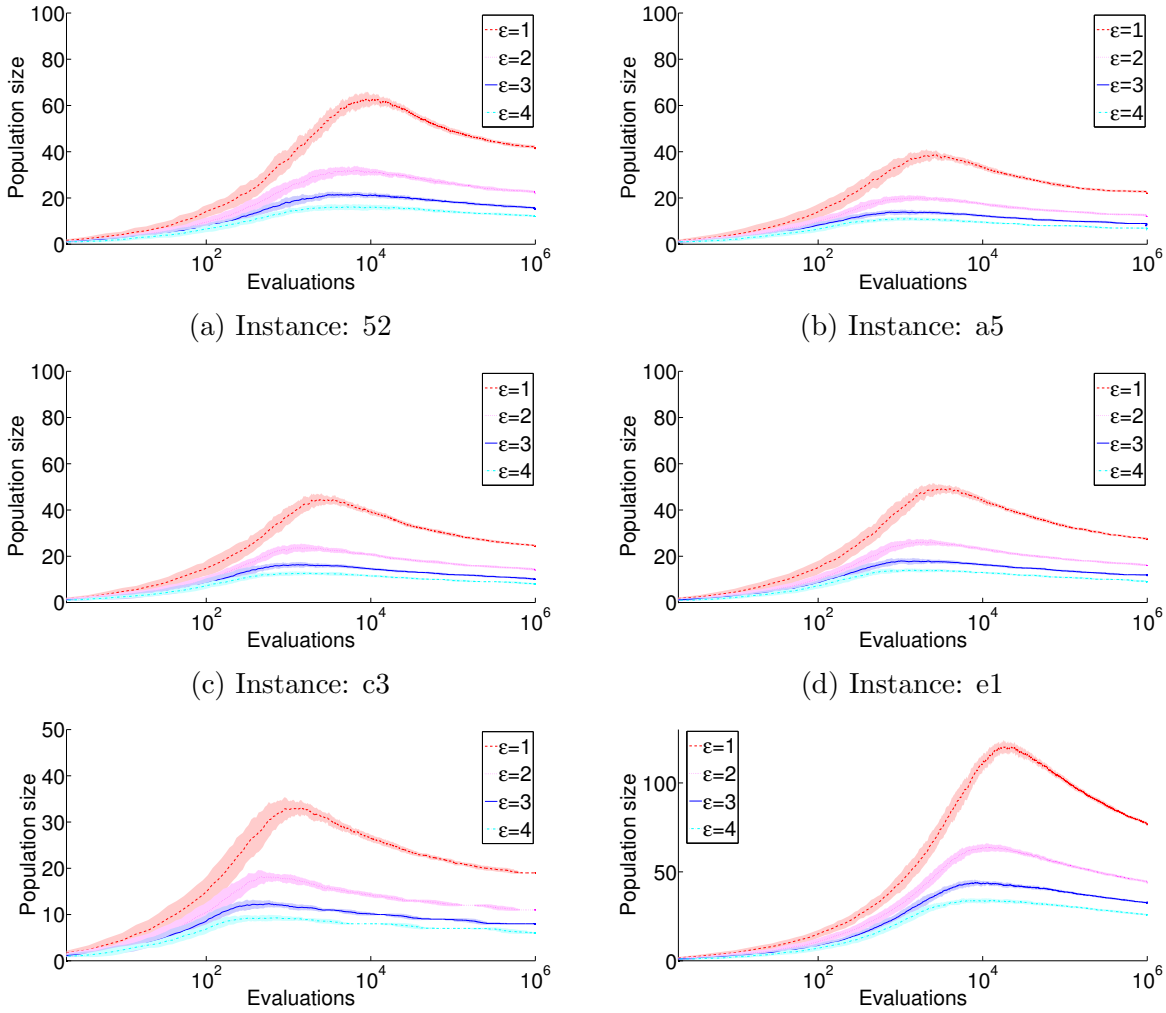
(c) Instance: c3

(d) Instance: e1

(e) Instance: rf2

(f) Instance: rh4

Figure 5.2: Run time plots of population size for ϵ-GC-AIS with $\epsilon = 1, 2, 3, 4$. Average population is plotted versus fitness evaluations expended by the algorithm, where the dashed red line, dotted magenta line, solid blue line and dot-dashed cyan lines correspond to ϵ-GC-AIS $= 1, 2, 3, 4$ respectively. x-axes plotted on a log scale with standard deviations shown as shaded error bars.



(a) Instance: 52

(b) Instance: a5

(c) Instance: c3

(d) Instance: e1

(e) Instance: rf2

(f) Instance: rh4

Figure 5.3: Run time plots of population size for ϵ-GC-AIS with $\epsilon = 1, 2, 3, 4$. Average population is plotted versus fitness evaluations expended by the algorithm, where the dashed red line, dotted magenta line, solid blue line and dot-dashed cyan lines correspond to ϵ-GC-AIS $= 1, 2, 3, 4$ respectively. x-axes plotted on a log scale with standard deviations shown as shaded error bars.

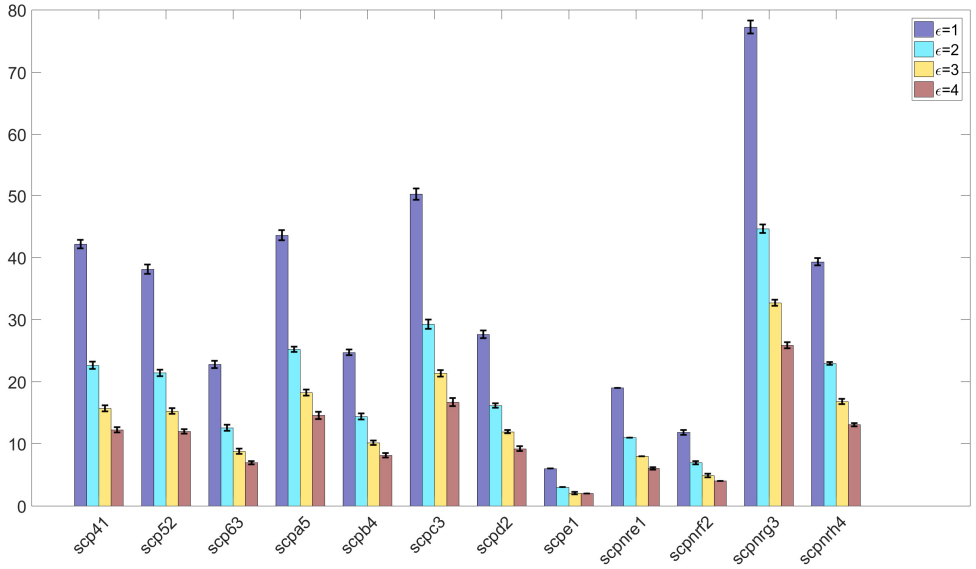
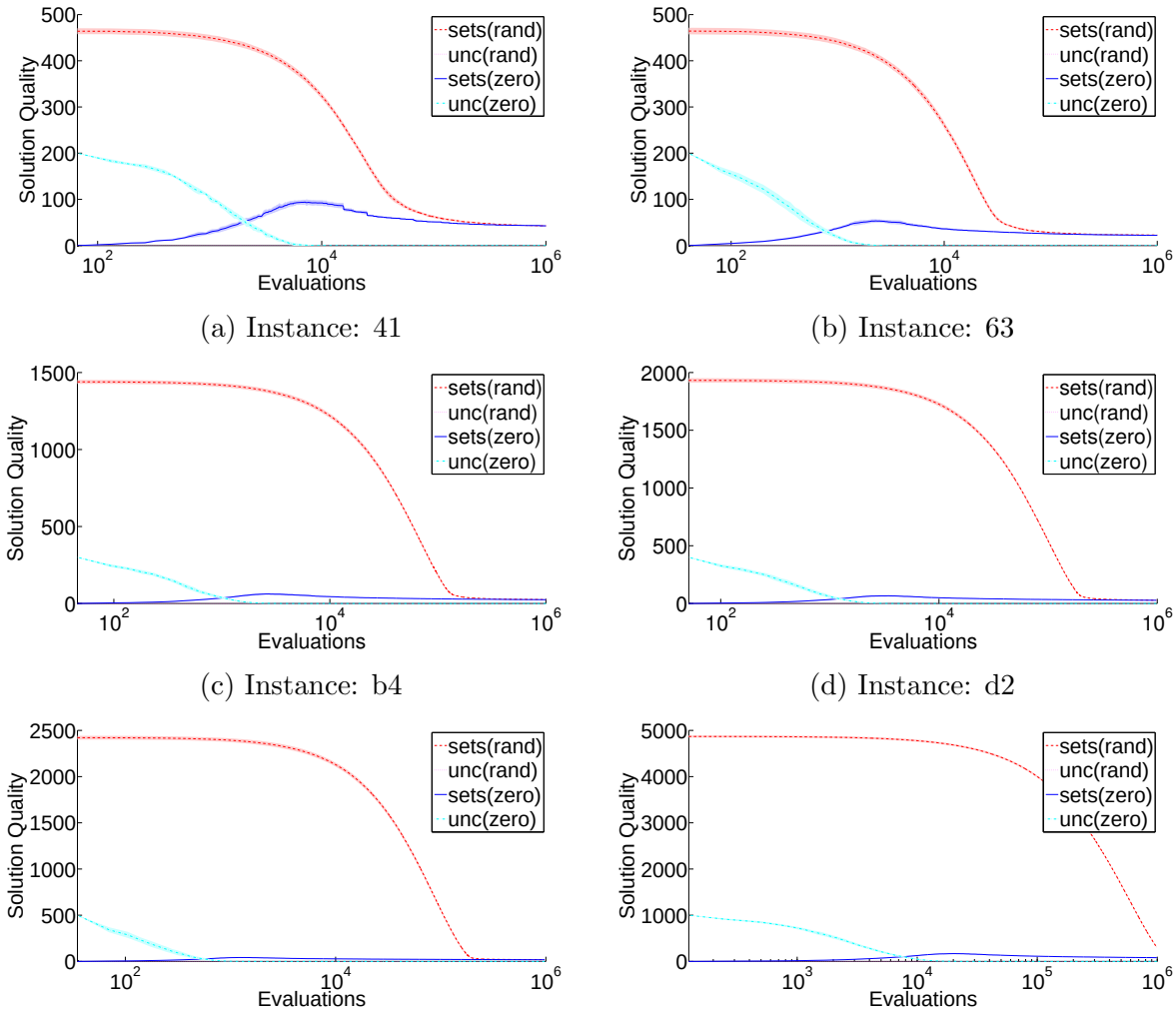


Figure 5.4: Average population size obtained at the end of runs for GC-AIS and ϵ-GC-AIS for the twelve problem instances listed as labels on the x-axes. The error-bars show the standard deviation observed in the population size.

5.5.2 Testing initialisation conditions for NSGA-II

As performed in chapter 4, the effects of different seeds are tested here for NSGA-II to ensure the best results of the algorithm configuration. Following the suggestion in [67] the all 0s bit string and the more commonly used random initialisation are compared. The performance of NSGA-II, when initialised from all 0s bit string and the random bit string is compared. As in the previous section, the algorithm with the 2 settings is run 30 independent times on all the 12 instances of the SCP for 10^6 fitness evaluations. The solution quality is recorded for each run and the run-time performance of the two settings is plotted in Figure 5.5 and 5.6.



(a) Instance: 41

(b) Instance: 63

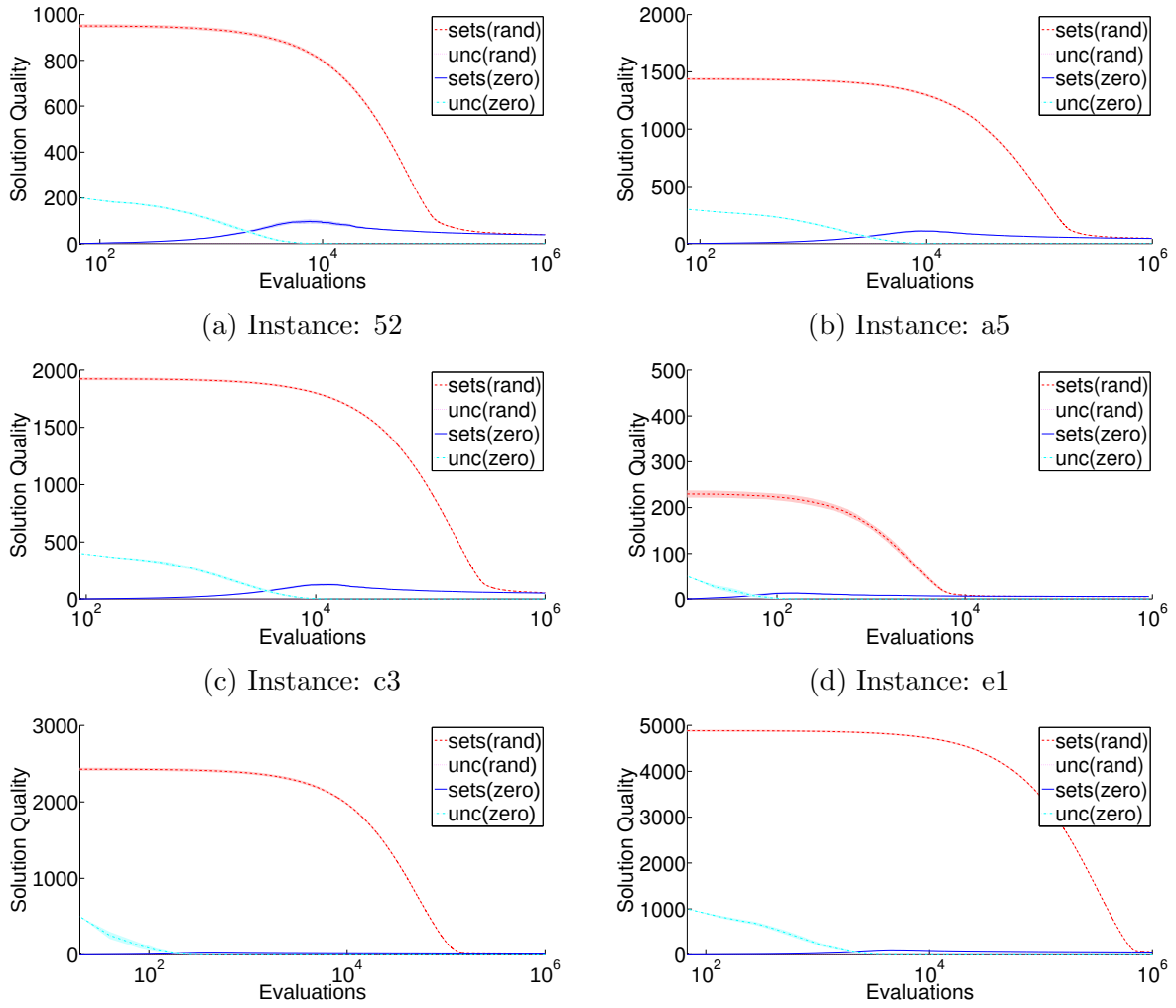
(c) Instance: b4

(d) Instance: d2

(e) Instance: re1

(f) Instance: rg3

Figure 5.5: Run time plots of solution quality for NSGA-II when starting from random bit-string versus all 0 bit string. Red and magenta lines show the average sets used and average uncovered elements when starting from random bit string while blue and cyan lines show the average sets used and average uncovered elements when starting from all 0s bit string respectively. X-axes plotted on a log scale and standard deviation is shown as shaded error bars.



(a) Instance: 52

(b) Instance: a5

(c) Instance: c3

(d) Instance: e1

(e) Instance: rf2

(f) Instance: rh4

Figure 5.6: Run time plots of solution quality for NSGA-II when starting from random bit-string versus all 0 bit string. Red and magenta lines show the average sets used and average uncovered elements when starting from random bit string while blue and cyan lines show the average sets used and average uncovered elements when starting from all 0s bit string respectively. X-axis is plotted in a log scale and standard deviation is shown as shaded error bars.

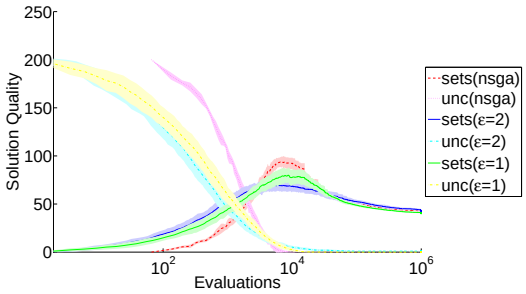
Discussion of results

It can be seen from the plots that starting from all 0s bit string causes NSGA-II to converge faster than starting from a random bit string. The dotted red line depicts the sets used when starting from the all 0s string while the solid blue line shows the sets used when solution is initialised as a random string. The solid cyan line shows the number of uncovered elements when the all 0s bit string initialisation is used while the dotted

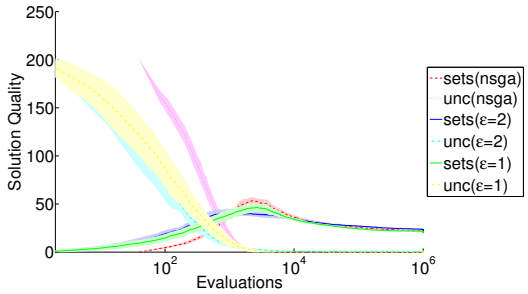
magenta line depicts the uncovered elements when solutions are initialised as random bit strings. It can be seen for all the instances that the blue line flattens much quicker than the red line, which means that starting from random string takes much longer to reach a solution which might be close to some optimal solution than starting from the all 0s bit string. This is the same behaviour seen in section 4.3.1 where starting from a random string, though leads to a feasible solution from the beginning, yet it is far away from the optimal while a solution starting from all 0s creates an infeasible solution which improves quickly by adding sets. Based on these findings the initialisation of the ϵ-GC-AIS is set to be the all 0s bit string for further experiments.

5.5.3 Experimental Results

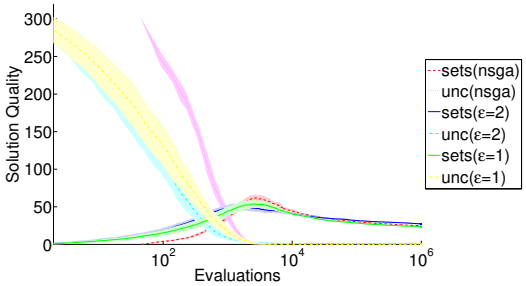
After setting the appropriate parameters and tuning the algorithms, they are now compared with each other. Both algorithms are allowed to run for 30 independent runs each for 10^6 fitness evaluations and average solution quality is recorded per iteration. The population size of NSGA-II is set to the maximum number of solutions seen during the runs of ϵ-GC-AIS which can be seen in Figure 5.2 and 5.3. The solution qualities obtained by the algorithms are plotted as run-time plots and results can be seen in the Figures 5.7 and 5.8. Table 5.1 shows the average solution qualities obtained by the two algorithms reported at the end of the allotted fitness evaluations.



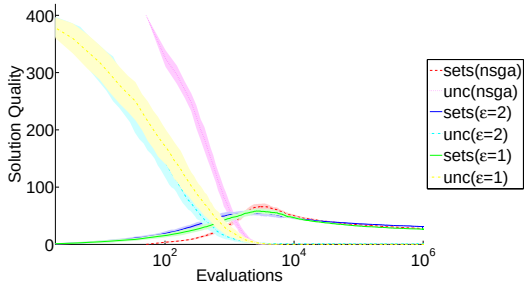
(a) Instance: 41



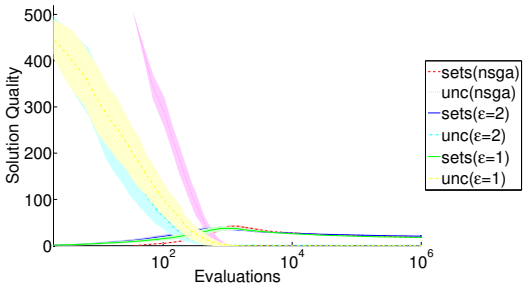
(b) Instance: 63



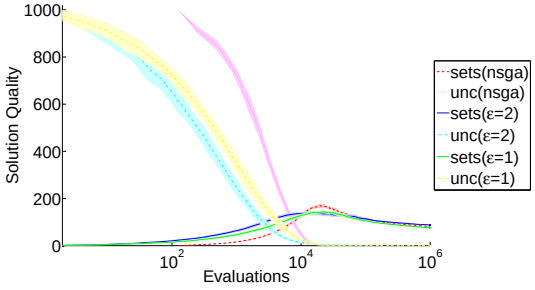
(c) Instance: b4



(d) Instance: d2

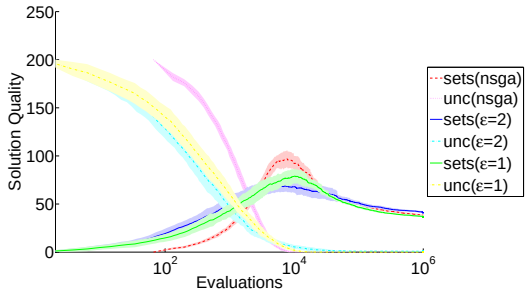


(e) Instance: re1

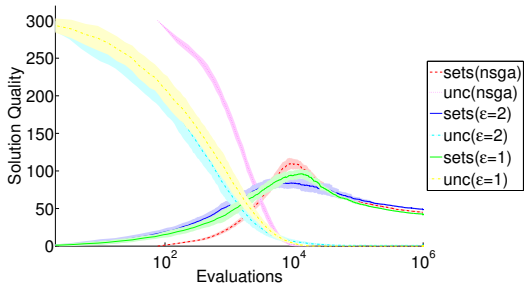


(f) Instance: rg3

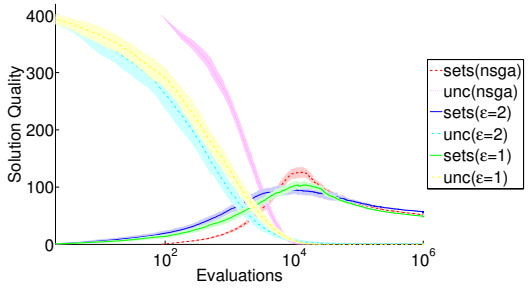
Figure 5.7: Run time plots showing solution quality per fitness evaluation for ϵ-GC-AIS with $\epsilon = 1$ and $\epsilon = 2$ versus NSGA-II. Yellow and cyan lines show average uncovered sets while green blue and lines show average sets use by ϵ-GC-AIS with $\epsilon = 1$ and $\epsilon = 2$ respectively. Magenta line shows average uncovered sets and red line shows average sets use by NSGA-II. X-axes plotted on log scale and standard deviation shown as shaded error-bars.



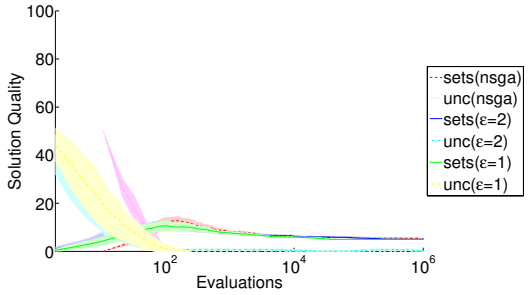
(a) Instance: 41



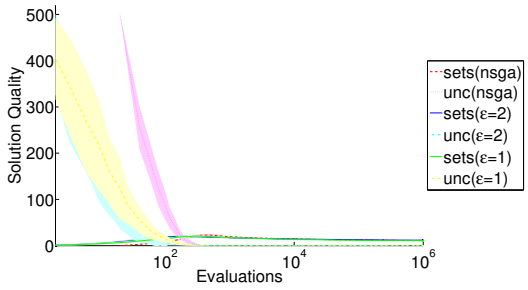
(b) Instance: 63



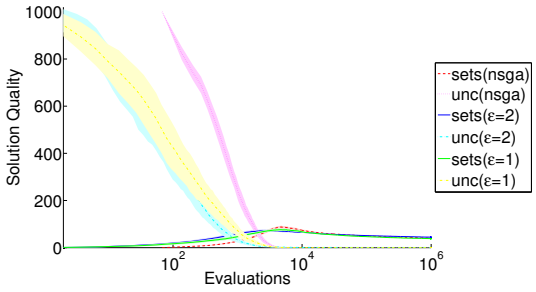
(c) Instance: b4



(d) Instance: d2



(e) Instance: re1



(f) Instance: rg3

Figure 5.8: Run time plots showing solution quality per fitness evaluation for ϵ-GC-AIS with $\epsilon = 1$ and $\epsilon = 2$ versus NSGA-II. Yellow and Cyan lines show uncovered sets while green blue and lines show sets use by ϵ-GC-AIS with $\epsilon = 1$ and $\epsilon = 2$ respectively. Magenta line shows uncovered sets and red line shows sets use by NSGA-II. X-axes plotted on log scale and standard deviation shown as shaded error-bars.

Discussion of results

It can be seen from the Figures 5.7 and 5.8 that both the algorithms reach the feasible region at around a similar time. The magenta line shows the uncovered elements for NSGA-II while the yellow and cyan lines depict the uncovered elements for ϵ-GC-AIS with $\epsilon = 1$ and $\epsilon = 2$ respectively. Looking closely it can be seen that NSGA-II is the first to reach the feasible region followed by ϵ-GC-AIS with $\epsilon = 1$ while the ϵ-GC-AIS

with $\epsilon = 2$ takes a bit more evaluations than the others. This can be explained as due to the effects of crossover and selection in the early stages of NSGA-II which leads to accumulation of more sets in the offspring solutions. Both settings of ϵ-GC-AIS are similar in this behaviour as they accumulate sets more gradually than NSGA-II due to the lack of crossover operator, which is present in NSGA-II and is more disruptive. After reaching the feasible region it can be seen that the solution quality of NSGA-II is worse than both settings of ϵ-GC-AIS. This can be seen from the red line, which shows that the sets used by NSGA-II in the figure are more than the green and the blue lines which depict the sets used by ϵ-GC-AIS with $\epsilon = 1$ and ϵ-GC-AIS with $\epsilon = 2$ respectively. Towards the end of the runs it can be seen that the green line is lower than all the others, meaning that the solution quality obtained by ϵ-GC-AIS with $\epsilon = 1$ is better than the other two algorithms.

The different behaviour seen in the figures can be explained by the nature of the problem instances especially the percentage density and size of the problem. The instances with higher percentage diversity converge more quickly as they are in some sense easier to solve. This is due to the fact that these instances cover more elements of the universe set per subset and every addition of a subset covers more elements in the universe when compared with the low percentage density instances. The effect of problem size can be seen on the size of the y axis, as the scale of the y axis is directly proportional to the size of the problem. Since the algorithms start from the all 0s string, the plots for uncovered sets begin from the y axis at a height equal to the problem size and the plots of sets used begin at 0. This is because at the beginning the solutions cover nothing and therefore the uncovered sets equal the problem size and the sets used equal to 0. This phenomenon makes the plots appear scaled and translated differently for different problems, while the inherent nature of the curves is the same in all of them.

These findings are further confirmed by Table 5.1 where the average solution qualities obtained by the two algorithms are reported at the end of the allotted fitness evaluations.

The reason why ϵ-GC-AIS with $\epsilon = 1$ performs better than ϵ-GC-AIS with $\epsilon = 2$ on

Instance	$\epsilon = 1$	$\epsilon = 2$	NSGA-II	KW test
scp41	(0,41)	(0,44)	(0,42)	2.78e-14
scp52	(0,37)	(0,41)	(0,38)	1.60e-16
scp63	(0,21)	(0,24)	(0,22)	3.87e-13
scpa5	(0,42)	(0,49)	(0,44)	1.56e-17
scpb4	(0,24)	(0,27)	(0,24)	3.29e-17
scpc3	(0,49)	(0,57)	(0,51)	8.77e-18
scpd2	(0,27)	(0,31)	(0,27)	3.78e-17
scpe1	(0,5)	(0,5)	(0,5)	1.33e-06
scpnre1	(0,18)	(0,20)	(0,18)	4.50e-17
scpnrf2	(0,10)	(0,12)	(0,11)	8.96e-17
scpnrg3	(0,76)	(0,88)	(0,80)	5.02e-18
scpnrh4	(0,38)	(0,44)	(0,40)	3.82e-18

Table 5.1: Final solution qualities obtained by ϵ-GC-AIS with $\epsilon = 1$, $\epsilon = 2$ and NSGA-II at the end of the runs. Column KW test shows the p values obtained from the Kruskal-Wallis test.

this problem is perhaps due to the small size of the problem. Since it can be seen from the histogram that the solutions in the non-dominated fronts obtained by ϵ-GC-AIS with $\epsilon = 1$ are less than 100 for all instances there is no need to reduce population. Increasing the ϵ value performs its intended task of controlling population and in this case makes the algorithm slower by limiting the number of solutions during the runs. Therefore using a higher ϵ dominance is not a good idea in this case where there is no real population explosion.

5.6 Testing the performance of ϵ-GC-AIS on MOd-KP

Following the study on SCP, experimental performance of ϵ-GC-AIS on the MOd-KP is investigated. These experiments can be seen as an extension of the work from Chapter 4 where GC-AIS was compared with NSGA-II on MOd-KP. The goal is that the introduction of ϵ-dominance will get rid of the population explosion and can also perhaps obtain better results than before.

5.6.1 Finding an appropriate value of ϵ

In this section experiments are performed to obtain an appropriate value of ϵ. ϵ-GC-AIS is run for 30 independent runs on the 12 instances of the knapsack problem each for 10^6 fitness evaluations with different values of ϵ. This number of fitness evaluations are set based on work in [132] so that the results obtained can be linked with the past literature and compared with them. In their work [132] the authors have used the same value of ϵ=10 for their external archive population for every instance of the MOd-KP. Setting an identical value of ϵ for the problem instances with different dimensions is not ideal and therefore different values of ϵ are tested and the average population after the runs are recorded. The different values of ϵ which are used for testing can be seen in Table 5.2. Histograms showing the population for each setting of ϵ for all the 4 problem sizes for the 3 dimensions is shown in Figure 5.9.

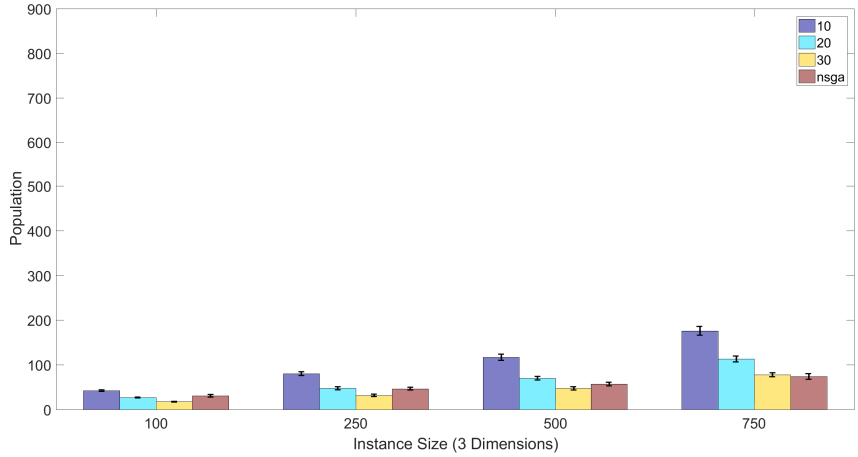
Values of ϵ				
Sacks	items	ϵ_1	ϵ_2	ϵ_3
2	100	10	20	30
	250	10	20	30
	500	10	20	30
	750	10	20	30
3	100	30	40	50
	250	30	40	50
	500	30	40	50
	750	30	40	50
4	100	60	80	100
	250	60	80	100
	500	60	80	100
	750	60	80	100

Table 5.2: The different values of ϵ tested with ϵ-GC-AIS on the different instances of MOd-KP.

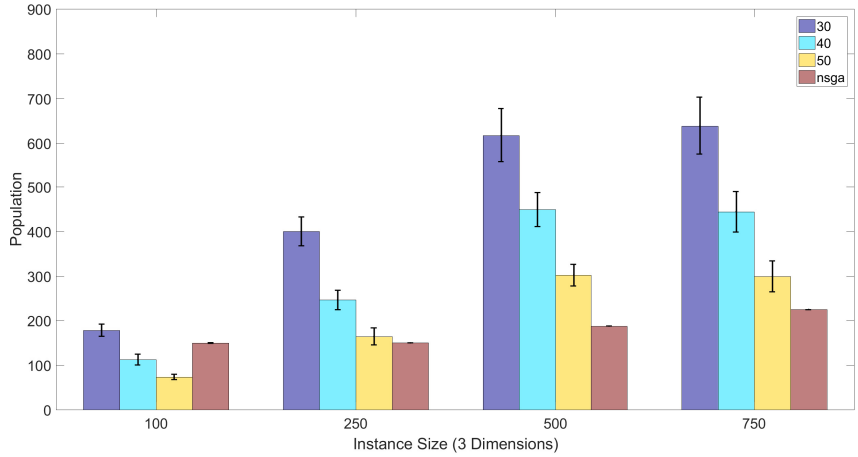
It can be seen from the histograms that ϵ can have a huge impact on the population size of the algorithm. In all the three histograms the higher the ϵ value used in ϵ-GC-AIS, the lower the final population in the non-dominated set obtained. It can be clearly seen from the three sub figures that population explosion is much larger in the instances with higher dimensions.

5.6.2 Set-up for MOd-KP experiments

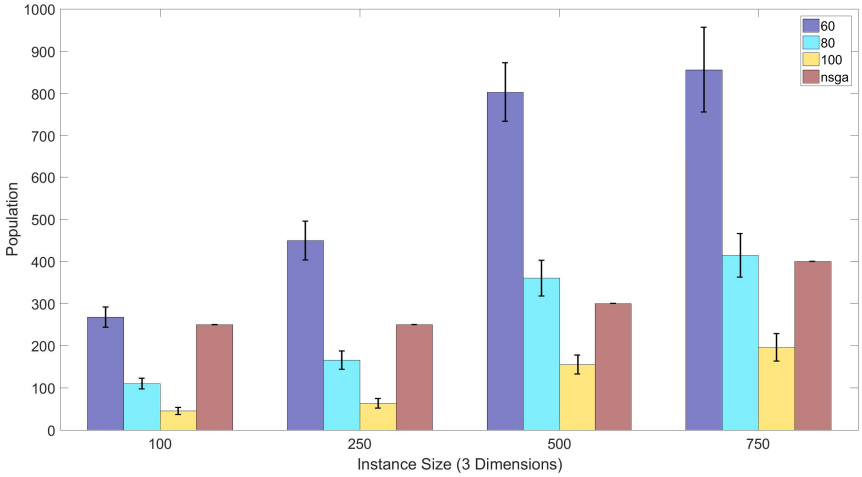
Similar to the set-up in Section 4.4.1 the settings used for the comparison between ϵ-GC-AIS and NSGA-II are described here. The same 12 problem instances of MOd-KP taken from[170] are used which are described in Section 3.3. 30 independent runs each with one million fitness evaluations are performed for each algorithm and statistical re-



(a) Instance: 41



(b) Instance: 63



(c) Instance: b4

Figure 5.9: Final population size at the end of runs averaged over 30 runs for ϵ-GC-AIS with different values of ϵ and NSGA-II. Error-bars show the standard deviation in population size.

		ϵ_1		ϵ_2		ϵ_3		NSGA-II	
Sacks	items	Per Instance	Overall	Per Instance	Overall	Per Instance	Overall	Per Instance	Overall
2	100	$26.321_{(2.78)}$	**76.41**	$19.245_{(1.59)}$	54.71	$10.40_{(1.53)}$	46.22	$2.67_{(2.65)}$	55.66
	250	$2.06_{(1.98)}$	**60.185**	$1.00_{(1.06)}$	30.09	$0.47_{(0.71)}$	13.88	0	0
	500	$1.45_{(2.17)}$	**43.125**	$1.39_{(2.04)}$	41.7	$0.48_{(0.81)}$	14.57	0	0
	750	$1.49_{(2.51)}$	**44.551**	$1.13_{(1.67)}$	33.97	$0.73_{(1.38)}$	22.11	0	0
3	100	$2.06_{(0.63)}$	**54.174**	$1.33_{(0.38)}$	34.31	$0.85_{(0.31)}$	22.62	$0.05_{(0.07)}$	1.78
	250	$1.45_{(1.39)}$	**43.764**	$0.96_{(0.90)}$	29.09	$0.79_{(0.59)}$	23.84	$0.10_{(0.18)}$	3.29
	500	$0.83_{(1.33)}$	24.987	$1.05_{(1.20)}$	31.56	$1.21_{(1.66)}$	**36.45**	$0.23_{(0.35)}$	6.99
	750	$0.39_{(0.83)}$	11.761	$0.07_{(1.51)}$	21.26	$1.62_{(2.68)}$	**48.87**	$0.60_{(0.71)}$	18.09
4	100	$2.03_{(0.51)}$	**61.158**	$0.83_{(0.22)}$	25.12	$0.34_{(0.11)}$	10.30	$0.11_{(0.09)}$	3.44
	250	$2.01_{(0.77)}$	**60.247**	$0.76_{(0.46)}$	23.03	$0.33_{(0.12)}$	10.17	$0.21_{(0.14)}$	6.54
	500	$0.32_{(0.56)}$	9.749	$0.92_{(1.45)}$	27.68	$1.56_{(1.19)}$	**46.88**	$0.52_{(0.40)}$	15.68
	750	$0.05_{(0.13)}$	1.6465	$0.33_{(0.43)}$	10.16	$1.55_{(1.86)}$	**46.63**	$1.38_{(0.90)}$	41.55

Table 5.3: Percentage contribution of NSGA-II and ϵ-GC-AIS with the ϵ values from Table 5.4 to the generation of reference fronts. Column 'Per Instance' shows average contribution of the algorithms per run, column 'Final' shows average contribution of the algorithms by combining solutions from all the runs. Standard deviation is shown as the subscript for column 'Per Instance'.

sults are recorded. The number of fitness evaluations is set to one million as per the previous experiments to link these experiments with previous results from literature. The parameter settings of the algorithms and the population size for NSGA-II have been kept the same as in Section 4.4.1 (see Tables 4.4 and 4.3). The only new parameter value to be selected is the value for ϵ.

In order to select an appropriate value of ϵ, the contribution of ϵ-GC-AIS with the different setting of ϵ along with NSGA-II, towards the generation of reference fronts is calculated. As in Section 4.4.1 the per run contribution refers to the contribution in percentage of each run of the algorithm towards the final Pareto front. Overall contribution refers to the combined contribution in percentage of all the runs of the algorithm towards the final Pareto front. The overall contribution of both algorithms can sum to more than 100% as the runs of the two algorithms can contain identical solutions. It can be seen from Table 5.3 that for all the 12 instances of the MOd-KP, ϵ-GC-AIS contributes the most towards the generation of the reference front and the contribution of NSGA-II is is very low. For 8 out of the 12 instances the ϵ-GC-AIS with ϵ_1 contributed the most towards the reference fronts, while for the remaining 4 instances ϵ-GC-AIS with ϵ_3 contributed the most. Based on these findings the value of ϵ which results the best contribution towards

Final value of ϵ				
Objectives	100	250	500	750
2	10	10	10	10
3	30	30	50	50
4	60	60	100	100

Table 5.4: Final value of ϵ selected for ϵ-GC-AIS to be used for the different instances of MOd-KP based on contribution to the reference fronts.

the reference front and in obtaining a population size closest to the population size of NSGA-II (see Figure 5.9), is selected. These values are shown in Table 5.4.

5.6.3 Experimental results

Finally experiments are conducted to test the performance of ϵ-GC-AIS with NSGA-II on the MOd-KP. Using the parameter settings found in the previous 2 subsections the algorithms are run for 30 independent runs each with 10^6 fitness evaluations on the 12 instances of the MOd-KP. The end of run performance is evaluated by recording the average values of the performance metrics at the end of runs. Standard deviation in the findings along with statistical tests to find significance difference are recorded. The metric values are shown in Tables 5.5, 5.6, 5.7, 5.8, 5.9, 5.10 and 5.11, 5.12, **??**.

		ϵ-GC-AIS	NSGA-II	
Sacks	items	average	average	rank-sum
2	100	$\mathbf{0.001}_{(\mathbf{0.0002})}$	$0.004_{(0.0006)}$	3.017e-11
	250	$\mathbf{0.001}_{(\mathbf{0.0002})}$	$0.007_{(0.0005)}$	3.019e-11
	500	$\mathbf{0.003}_{(\mathbf{0.0003})}$	$0.013_{(0.0007)}$	3.019e-11
	750	$\mathbf{0.002}_{(\mathbf{0.0002})}$	$0.018_{(0.001)}$	3.019e-11

Table 5.5: Average values for the generational distance metric obtained for NSGA-II and ϵ-GC-AIS for 30 runs of the algorithms for 2 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the results for the Wilcoxon rank-sum test.

		ϵ-GC-AIS	NSGA-II	
Sacks	items	average	average	rank-sum
3	100	$\mathbf{0.002}_{(\mathbf{0.0001})}$	$0.007_{(0.0004)}$	3.019e-11
	250	$\mathbf{0.002}_{(\mathbf{0.0002})}$	$0.014_{(0.0013)}$	3.019e-11
	500	$\mathbf{0.002}_{(\mathbf{0.0005})}$	$0.017_{(0.0014)}$	3.019e-11
	750	$\mathbf{0.004}_{(\mathbf{0.0008})}$	$0.012_{(0.0010)}$	3.019e-11

Table 5.6: Average values for the generational distance metric obtained for NSGA-II and ϵ-GC-AIS for 30 runs of the algorithms for 3 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the results for the Wilcoxon rank-sum test.

		ϵ-GC-AIS	NSGA-II	
Sacks	items	average	average	rank-sum
4	100	$\mathbf{0.003}_{(\mathbf{0.0002})}$	$0.014_{(0.0008)}$	3.019e-11
	250	$\mathbf{0.002}_{(\mathbf{0.0001})}$	$0.019_{(0.0016)}$	3.019e-11
	500	$\mathbf{0.001}_{(\mathbf{0.0004})}$	$0.018_{(0.0011)}$	3.019e-11
	750	$\mathbf{0.002}_{(\mathbf{0.0005})}$	$0.014_{(0.0009)}$	3.019e-11

Table 5.7: Average values for the generational distance metric obtained for NSGA-II and ϵ-GC-AIS for 30 runs of the algorithms for 4 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the results for the Wilcoxon rank-sum test.

		ϵ-GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
2	100	$\mathbf{1.69e+07}_{(\mathbf{6.7692+e04})}$	$1.68e+07_{(3.86+e04)}$	2.37e-07
	250	$\mathbf{9.54e+07}_{(\mathbf{4.08+e05})}$	$9.52e+07_{(3.68+e05)}$	2.57e-07
	500	$\mathbf{3.89e+08}_{(\mathbf{1.8e+06})}$	$3.84e+08_{(1.37e+06)}$	7.38e-11
	750	$\mathbf{8.45e+08}_{(\mathbf{3.68e+06})}$	$8.14e+08_{(2.89e+06)}$	3.01e-11

Table 5.8: Average values for the hypervolume metric obtained for NSGA-II and ϵ-GC-AIS for 30 runs of the algorithms for 2 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the results for the Wilcoxon rank-sum test.

		ϵ-GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
3	100	$6.07e+10_{(3.99e+08)}$	$6.08e+10_{(2.97e+08)}$	0.15
	250	$8.24e+11_{(6.76e+09)}$	$8.27e+11_{(6.82e+09)}$	0.14
	500	$6.34e+12_{(3.80e+10)}$	$\mathbf{6.40e+12_{(4.98e+10)}}$	4.11e-06
	750	$\mathbf{2.17e+13_{(1.81e+11)}}$	$2.14e+13_{(1.43e+11)}$	2.67e-06

Table 5.9: Average values for the hypervolume metric obtained for NSGA-II and ϵ-GC-AIS for 30 runs of the algorithms for 3 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the results for the Wilcoxon rank-sum test.

		ϵ-GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
4	100	$1.59e+14_{(1.96e+12)}$	$\mathbf{1.62e+14_{(1.33e+12)}}$	2.38e-08
	250	$6.24e+15_{(8.08e+13)}$	$\mathbf{6.41e+15_{(7.18e+13)}}$	3.01e-11
	500	$9.29e+16_{(1.03e+15)}$	$\mathbf{9.72e+16_{(9.22e+14)}}$	3.01e-11
	750	$4.55e+17_{(4.9e+15)}$	$\mathbf{4.79e+17_{(5.88e+15)}}$	3.01e-11

Table 5.10: Average values for the hypervolume metric obtained for NSGA-II and ϵ-GC-AIS for 30 runs of the algorithms for 4 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the results for the Wilcoxon rank-sum test.

		ϵ-GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
2	100	$\mathbf{0.49_{(0.032)}}$	$0.55_{(0.055)}$	1.16711e-05
	250	$0.68_{(0.034)}$	$0.68_{(0.037)}$	0.318304
	500	$\mathbf{0.50_{(0.051)}}$	$0.57_{(0.048)}$	9.51e-06
	750	$\mathbf{0.58_{(0.038)}}$	$0.65_{(0.038)}$	1.35943e-07
3	100	$\mathbf{0.34_{(0.021)}}$	$0.41_{(0.025)}$	8.99341e-11

Table 5.11: Average values for the diversity metric obtained for NSGA-II and ϵ-GC-AIS for 30 runs of the algorithms for 2 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the results for the Wilcoxon rank-sum test.

		ϵ-GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
3	100	$\mathbf{0.34_{(0.021)}}$	$0.41_{(0.025)}$	8.99341e-11
	250	$\mathbf{0.39_{(0.020)}}$	$0.42_{(0.025)}$	4.11271e-07
	500	$\mathbf{0.38_{(0.020)}}$	$0.45_{(0.027)}$	6.06576e-11
	750	$\mathbf{0.46_{(0.025)}}$	$0.49_{(0.023)}$	0.0002

Table 5.12: Average values for the diversity metric obtained for NSGA-II and ϵ-GC-AIS for 30 runs of the algorithms for 3 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the results for the Wilcoxon rank-sum test.

		ϵ-GC-AIS	NSGA-II	
Sacks	items	average	average	Wilcoxon test
4	100	$\mathbf{0.27}_{(\mathbf{0.015})}$	$0.36_{(0.020)}$	3.33839e-11
	250	$\mathbf{0.33}_{(\mathbf{0.027})}$	$0.41_{(0.020)}$	7.38908e-11
	500	$\mathbf{0.42}_{(\mathbf{0.020})}$	$0.44_{(0.016)}$	0.000110577
	750	$0.46_{(0.020)}$	$0.44_{(0.021)}$	0.074827

Table 5.13: Average values for the diversity metric obtained for NSGA-II and ϵ-GC-AIS for 30 runs of the algorithms for 4 sacks. Standard deviations are shown in the subscript. Column rank-sum shows the results for the Wilcoxon rank-sum test.

Discussion of results

Just like in chapter 4, the solutions obtained by ϵ-GC-AIS are closer to the Pareto front than the solutions obtained by NSGA-II, which can be seen by the lower values of generational distance metric for the two algorithms. In all the 12 instances ϵ-GC-AIS performs better when considering this metric and the Wilcoxon rank-sum test shows the two algorithms are statistically different at a significance level of 0.05.

The table for hypervolume shows mixed results, as seen by the larger hypervolume obtained by ϵ-GC-AIS for smaller dimensional instances while for larger dimensions the value is smaller than NSGA-II. Some improvements are seen in values of hypervolume for ϵ-GC-AIS than GC-AIS from the previous chapter, but these are still not enough for the larger instances.

Since the algorithmic settings of the experiments have been kept similar to those in [132], ϵ-GC-AIS can be compared to the 3 algorithms presented in [132] namely ϵ-hBOA, ϵ-NSGA-II and SPEA2, with respect to the hypervolume metric. Comparing Table 5.14, 5.15, 5.16 with Tables 5.8, 5.9, 5.10 it can be seen ϵ-GC-AIS obtained higher hypervolume on 11 out of the 12 instances then all the three algorithms in [132]. For the instance with 4 sacks and 100 items, a satisfactory comparison could not be made as the

findings in [132] are unclear.

		ϵ-hBOA	ϵ-NSGA-II	SPEA2
Sacks	items	average	average	average
2	100	$4.32e+4_{(0.60+e4)}$	$2.03e+4_{(1.21+e04)}$	$16.24e+4_{(0)}$
	250	$5.72e+04_{(1.16+e4)}$	$6.74e+4_{(3.25+e4)}$	$29.76e+4_{(0)}$
	500	$4.31e+05_{(1.54e+5)}$	$11.11e+5_{(2.34e+5)}$	$29.36e+5_{(0)}$
	750	$9.15e+06_{(3.15e+6)}$	$85.30e+6_{(13.54e+6)}$	$75.34e+6_{(0.03)}$

Table 5.14: Average values for the hypervolume metric for some recent MOEAs, taken from [132] for 2 sacks. Standard deviations are shown in the subscript.

		ϵ-hBOA	ϵ-NSGA-II	SPEA2
Sacks	items	average	average	average
3	100	$3.37e+8_{(0.63e+8)}$	$2.98e+8_{(0.72e+8)}$	$11.43e+8_{(0.03+e8)}$
	250	$3.08e+11_{(0.07e+11)}$	$3.15e+11_{(0.16e+11)}$	$3.17e+11_{(0.07+e11)}$
	500	$1.22e+12_{(0.08e+12)}$	$2.44e+12_{(0.21e+12)}$	$2.87e+12_{(0.06+e12)}$
	750	$2.77e+12_{(0.03e+12)}$	$4.08e+12_{(0.10e+12)}$	$14.38e+12_{(0.05+e12)}$

Table 5.15: Average values for the hypervolume metric for some recent MOEAs, taken from [132] for 3 sacks. Standard deviations are shown in the subscript.

		ϵ-hBOA	ϵ-NSGA-II	SPEA2
Sacks	items	average	average	average
4	100	$3.00e+14_{(0.84e+14)}$	$11.40e+14_{(0.49e+12)}$	-
	250	$3.98e+15_{(0.01e+15)}$	$4.17e+15_{(0.03e+15)}$	$5.07e+14_{(0.01+e15)}$
	500	$2.16e+15_{(0.25e+15)}$	$11.24e+15_{(0.94e+15)}$	$12.80e+15_{(2.10+e15)}$
	750	$1.94e+16_{(18.57e+16)}$	$9.87e+16_{(0.53e+16)}$	$10.59e+16_{(0.15+e16)}$

Table 5.16: Average values for the hypervolume metric for some recent MOEAs, taken from [132] for 4 sacks. Standard deviations are shown in the subscript.

In 10 out of the 12 instances, the value of the diversity metric shows that solutions

obtained by ϵ-GC-AIS have better diversity than solutions obtained by NSGA-II. On these 10 instances, the Wilcoxon rank-sum test shows significant difference between the algorithms at a significance level of 0.05.

Based on the tables of metrics it can be seen that ϵ-GC-AIS performs better than NSGA-II on majority of the instances of MOd-KP. The generational distance defines the clear winner to be ϵ-GC-AIS in all the instances. Considering diversity and hypervolume ϵ-GC-AIS is a clear winner in the smaller dimension instances and mixed results are obtained for higher dimensions. Based on the hypervolume metric it can also be seen that ϵ-GC-AIS performs better than the algorithms presented in [132]. A comparison of other metrics is not performed as a fair comparison would required identical reference fronts to be used used to evaluate the metrics.

5.7 Discussion and Conclusion

On the static SCP the comparison between ϵ-GC-AIS and NSGA-II shows that the best performance of ϵ-GC-AIS is achieved with a very small value of ϵ. This can be attributed to the small size of the problem as the number of objectives for the SCP are only 2. This means that the search space of the problem is comparatively small. It is seen that in this case no actual population explosion occurs and using a higher value of ϵ-dominance operator can slow down the performance of the algorithm due to limiting the number of potential solutions in the population. The solution quality obtained by ϵ-GC-AIS is equal to or better than the solution quality obtained by NSGA-II.

On the other hand, a clear advantage of using ϵ-GC-AIS can be seen for MOd-KP, where the population explosion was seen in Chapter 4 and incorporating the ϵ operator can help reduce the number of solutions to a more manageable level for the decision maker. When considering the generational distance ϵ-GC-AIS performs better than NSGA-II on all the instances while when considering hypervolume and diversity, NSGA-II is better for most of the instances with higher number of objectives while ϵ-GC-AIS performs

better on smaller instances. When compared with other popular algorithms in literature proposed for this problem in [132] based on the hypervolume metric, ϵ-GC-AIS obtained better hypervolume on 11 out of the 12 instances tested. These algorithms from literature included popular MOEAs such as SPEA-2, NSGA-2 as well as ϵ-hBOA proposed in [132]. Another aspect of ϵ-GC-AIS which is considered is the sensitivity to the value of ϵ. Setting different values of the ϵ operator results in dramatic difference in the population size obtained at the end of the runs by ϵ-GC-AIS. This is due to the nature of the ϵ-dominance technique and results in different performance when different ϵ value is set which is seen in the percentage contribution of ϵ-GC-AIS towards the generation of the reference fronts when different values of ϵ are used. It is seen that a small values of ϵ contributes more to the refernce fronts for smaller problem sizes while for larger problems a large ϵ values must be set. This result is intuitive as for large problem size with many objectives, the need to control population explosion becomes important which is achieved by a large ϵ value, while for smaller problems with fewer objectives a low value of ϵ is needed as a higher ϵ can hinder proper exploration and search due to too few solutions.

Clear advantage of utilising the ϵ operator can be seen for the MOd-KP, yet experiments for SCP show that using ϵ-GC-AIS in a situation where population explosion does not occur can lead to slow performance. This means that before deciding between GC-AIS and ϵ-GC-AIS the problem must be studied carefully. If the problem is big with a larger number objectives and will potentially cause population the ϵ-GC-AIS should be preferred, while if the problem is smaller with a small number of objectives then it can be suitably solved by GC-AIS without the need to use ϵ dominance. Another important factor to note is that the ϵ-GC-AIS is sensitive to the value of ϵ. Setting too large a value for this operator can result in the population size to be reduced drastically which can slow down the search process, while setting its value too low may not solve a population explosion issue of the problem. Currently these values for ϵ are tested by estimation and guidance from literature. A potential improvement could be to eliminated the need to manually set this value by proposing a dynamic ϵ resizing mechanism.

In conclusion, it can be said that ϵ-GC-AIS performs better than NSGA-II on majority of the instances of the SCP and MOd-KP problems tested. There are clear advantages of incorporating ϵ to control population size as well as promote convergence and diversity when the search size of the problem increases, which happens dramatically when adding more dimensions. These benefits can be seen clearly for MOd-KP but not for SCP where population explosion does not occur. Finally, tuning the ϵ parameter is crucial in order to control the population size and hence the convergence and diversity in the algorithm.

5.8 Summary

Improvements to deal with population explosion in GC-AIS are presented in the form of ϵ-GC-AIS which incorporates the ϵ dominance relation. Extending the study in the previous chapter, the ϵ-GC-AIS algorithm is tested on the SCP and the MOd-KP. ϵ-GC-AIS is shown to perform better than NSGA-II on most instances of SCP when using a small ϵ value where population explosion is not a big issue. When dealing with MOd-KP ϵ dominance plays a key role to mitigate population explosion and ϵ-GC-AIS performs better than NSGA-II on majority of the instances based on the metrics of comparison. The advantage of incorporating the ϵ operator to thwart population explosion are shown by careful setting of the value of the ϵ parameter.

CHAPTER 6

INTRODUCING MEMORY IN GC-AIS

6.1 Introduction

Learning and memory are important features of the immune system. The focus in the last two chapters has been to investigate the optimisation capabilities of GC-AIS on some static multi-objective problems, therefore a memory component is not usually required to solve them. The main focus of this chapter is to understand the behaviour of memory on the performance of GC-AIS in optimising dynamic problems where memory techniques are a popular approach to tackle them. The benefit of incorporating memory in the GC-AIS algorithm is examined and the key research questions addressed here are:

> *When using* GC-AIS *for dynamic optimisation of* SCP*, does using solutions from memory perform better than starting from scratch?*

The term scratch here refers to solutions generated initially for an evolutionary algorithm for the problem being considered. In the case of SCP a scratch solution is the all 0s string, but for other problems it could be a randomly generated solution. Answering this question can provide insight as to whether using memory approach is ever better than just using standard initialisation upon the detection of a change in a dynamic scenario.

Also, further questioning the behaviour of the algorithm which contains memory on different problem settings, by answering

When does using memory perform better than a solution from scratch and when does it fail? why does this behaviour occur ?

The outline of the chapter is as follows: In section 6.2 a possible real world setting for the dynamic SCP is introduced. This is followed by introducing two models for the dynamic SCP. The GC-AIS with a memory component is introduced in section 6.4 followed by a comparative analysis between GC-AIS and GC-AIS with memory on the two models of dynamic SCP instances in section 6.4. Experimental results are presented in Section 6.6 followed by discussion and conclusion. Finally a brief summary of the chapter is provided in section 6.7[1]

6.2 A real world scenario for dynamic SCP

The static set cover problem has been applied in a variety of applications including crew and airline scheduling, tool selection in manufacturing systems, facility location, just to name a few [45]. In order to consider a dynamic version of the problem, potential real world scope and application of such a scenario must be considered. Indeed there is a practical relevance to studying dynamic SCP, an example of which is demonstrated in [28] where a model for dynamic facility location are proposed. In a similar vein, here a practical application of dynamic SCP is considered in the setting of airline crew scheduling.

Consider first the static crew scheduling problem for airlines [45], where a set of flight legs i are given. In addition to the flight legs, crew schedules j, in the form of tours of duty are provided which consist of sequence of flight legs. In such a scenario

$$sc_{ij} = \begin{cases} 1 & \text{if flight leg } i \text{ is part of tour j} \\ 0 & \text{otherwise} \end{cases}$$

and

[1]Part of work presented in this chapter has been published in [96].

Table 6.1: An example of flight legs for a given airline with 3 planes. BHX, LHR, MAN and EDI represent the cities Birmingham, London, Manchester and Edinburgh respectively. Times for each legs are provided in 24 hour format.

Plane 1		Plane 2		Plane 3	
A:BHX-LHR	9:00-10:00	G:BHX-MAN	10:00-12:00	K:BHX-MAN	9:30-11:30
B:LHR-BXH	10:30-11:30	H:MAN-BHX	12:30-14:30	L:MAN-EDI	12:00-13:00
C:BHX-LHR	12:00-13:00	I:BHX-MAN	15:00-17:00	M:EDI-BHX	13:15-14:30
D:LHR-BHX	13:30-14:30	J:MAN-BHX	17:30-19:30	N:BHX-MAN	15:00-17:00
E:BHX-LHR	15:00-16:00	-	-	O:MAN-EDI	17:30-18:30
F:LHR-BHX	16:30-17:30	-	-	P:EDI-BHX	18:45-20:00

$$X_j = \begin{cases} 1 & \text{if tour } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

The problem of staffing all flight legs utilising the least number of tours of duty can be essentially modelled as the uni-cost SCP. Formally, the goal is to minimise the number of schedules

$$\text{Minimise} \quad \sum_j X_j$$

subject to the constraint

$$\sum_j sc_{ij} X_j \geq 1 \quad \forall i$$

A simplified example of crew scheduling is now provided. Suppose an airline has 3 planes in Birmingham which fly between Birmingham and London, Birmingham and Manchester and Birmingham and Edinburgh via Manchester respectively. If we label the cities as BHX, LHR, EDI and MAN respectively, the flight legs of these planes are as shown in Table 6.1. A pairing of flight legs correspond to a combination of flights which begin and return to Birmingham served by a single flight crew. Example pairings could be AB, GH and KLM. Since each flight leg must be covered by the crews, a schedule of the crew corresponds to a set of pairings which covers all the legs.Two examples of possible schedules could be AB, CD, EF, GH, IJ, KLM, NOP which utilise 7 crews and ABCDEF, GHIJ, KLMNOP which utilise 3. It is easy to see that having hundreds of flight legs per

day and potentially thousands of crew schedules it becomes extremely hard to find the best schedules easily.

Now, consider the day to day working for an airline company. Two variables which may change during the course of operation are the number of flights, which directly affects the flight legs and secondly the availability of crew, which affects the schedules. The number of flights is affected by user demand which fluctuates both on a daily as well as monthly or seasonally basis. For example, the number of flights to a location are usually not the same for weekdays and weekends depending upon the type of location. Similarly flights may be increased or decreased to locations based on seasonal factors such as best time to travel, festival season to name a few. This would cause the flight legs in the Table 6.1 to change over time.

On the other hand, crew availability which affects the tours of journey change according to factors such as personal preference, leave or absence, cost, logistics etc. According to this dynamic behaviour in the industry, a good solution to the problem of utilising the least crew members to cover all flight legs at any given day varies according to the dynamic variables. This would cause changes or potentially new crew schedules to be generated. Therefore it is worth asking, if a good solution to the problem is known for some day with a fixed setting of the dynamic variables, how useful is this on a different set of variables. Does it really make sense to use the previously known solution or is it better to find a good solution from scratch?

6.3 Models for the single time-step dynamic SCP

In order to formalise the different ways in which the variables of SCP can change, two models of changes are presented here. Since the immune system solves the SCP (for defintion see Section 2.4.1), biological motivations for the dynamic SCP are provided for each model. Considering the crew scheduling example, these models depict the different ways in which the problem can change. The changes in model 1 correspond to variations

of the schedules while model 2 deals with variations which correspond to the number of flight legs.

6.3.1 Model 1: Changes to S (columns)

This model involves making modifications to the set S of the original instances to create the novel instances. The biological motivation of this kind of modifications comes from the principles of dynamic memory pool regulation. The memory pool of the immune system changes with time and is regulated[46]. Upon arrival of a pathogen, which the organism has previously encountered, the memory pool could be potentially different than during a previous encounter of the pathogen. This situation can arise due to the dynamic nature of the immune system since the memory pool both reflects the organisms encountered over time as well as potentially loses some memory of encounters from distant past. [SOURCE] This translates to the changes in S which in an abstract way represents the information that is available to use at a particular time.

The modifications proposed in this model correspond to Definition 4 (see 2.5.2). The three kinds of modifications are performed to the original static instances to create the dynamic instances: addition of columns, removal of columns and editing of columns. Example novel instances are provided with each modification based on the SCP example in Section 2.4.1. The set $U = \{1, 2, 3, 4, 5, 6\}$ remains the same and the different changes into the family of subsets $S = \{\{4\}, \{6\}, \{1, 2\}, \{3, 4\}, \{5, 6\}, \{1, 2, 3\}, \{4, 5, 6\}, \{1, 2, 4, 5\}\}$ are shown below.

In order to create novel instances for this model prior information about the current solution of the original static instances is required in the case of removal and editing. These solutions are converted into memory by saving the subsets from S which have corresponding $1s$ in the solution bit string. It is important to save the individual subsets and not just the bit string as the changes to S in the newly created instances can lead to change in the index of subsets in the new bit-string.

- **Addition**: Addition is performed by adding columns to the original $(m \times n)$ matrix for each original problem instance. This can be viewed as adding subsets to set S while keeping in mind that the original density of the $(m \times n)$ matrix is maintained. This is done by ensuring that the density of the columns added is the same as the density of the original problem. Let $k \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ be the number of columns to be added to the original instance. For each original problem instance 30 novel instances are created for every value of k. It should be noted that by performing this addition, the size of the novel instance now becomes $(m \times (n + k))$. A subset can be added (marked in bold) to S such that the new $S' = \{\mathbf{\{1\}}, \{4\}, \{6\}, \{1, 2\}, \{3, 4\}, \{5, 6\}, \{1, 2, 3\}, \{4, 5, 6\}, \{1, 2, 4, 5\}\}$.

- **Removal**: Removal of columns from the $(m \times n)$ matrix is performed using the prior knowledge of the best solution. The goal here is that the original best solution is not valid any longer for the novel instance created and this is performed by removing only the subsets which correspond to 1s in the original solution. Let $k \in \{0.1c, 0.2c, \cdots, 0.9c\}$ be the number of columns removed from the original instance, where c is the original solution quality (sets used by the current known solution). For every original instance 30 novel instances are created for every value of k. This size of the novel instance now becomes $(m \times (n - k))$. Subsets can be removed from S by removing only the sets which occur in the best solution found, i. e., $y = \{\{1, 2, 3\}, \{4, 5, 6\}\}$. Therefore a possible new S can be formed by removing set **{4, 5, 6}** so that $S' = \{\{4\}, \{6\}, \{1, 2\}, \{3, 4\}, \{5, 6\}, \{1, 2, 3\}, \{1, 2, 4, 5\}\}$.

- **Editing**: Editing is performed by moving items from one column of the $(m \times n)$ matrix to others. As in the case of removal only the columns which have corresponding 1s in the best solution are used. Items are removed from these columns randomly and moved to other randomly selected columns which are not present in the best solution and which do not contain these particular items. Let $k \in \{0.1d, 0.2d, \cdots, 0.9d\}$ be the number of items to be moved, where d is the total

number of items in the best solution (sets used by the current known solution). For every value of k, 30 novel instances are created for each original instance. It should be noted that in this case of modification the size of the novel instance remains the same as in the original instance. In order to make this change, again the best solution $y = \{\{1, 2, 3\}, \{4, 5, 6\}\}$ is needed. By moving item 5 from the set **{4, 5, 6}** present in the solution to set **{3, 4}** of S a possible novel instance is $S' = \{\{4\}, \{6\}, \{1, 2\}, \{3, 4, \mathbf{5}\}, \{5, 6\}, \{1, 2, 3\}, \{4, 6\}, \{1, 2, 4, 5\}\}$.

6.3.2 Model 2: Changes to U (rows)

In this model, the original SCP problem is modified such that only certain specific elements of U are required to be covered. The biological motivation of this model can be seen as understanding the response of the memory pool of the immune system to the different pathogen strains which have varying degrees of similarity [56]. An organism could potentially be attacked by different strains of the same pathogen over time and the memory pool may have previous information about previously encountered strains. This model abstracts this idea as the changes in U can be seen as slight modifications to a previously encountered strain of pathogens and the system must use the memory resources which can deal well with the previously encountered strain.

The modifications in this model are performed by making changes to the set U from the original instance. This is composed of two steps. First the original instances are modified to create new starting instances. For each original problem instance, new starting instances U^* are created by randomly selecting $m/2$ items from U. In other words, the goal now is to cover the randomly selected $m/2$ items from the original instances given the same set S. This makes the size of the starting instance $(m^* \times n)$ where $m^* = m/2$. Following the simple example from the previous section, out of the six elements in the original $U = \{1, 2, 3, 4, 5, 6\}$, the starting instance U^* can be formed as $U^* = \{2, 3, 4\}$. Like the previous model, novel instances for this model are created from the starting instances by introducing three kinds of modifications: Addition, removal and swapping.

The dynamic changes in this model are related to Definition 5 (see Chapter 2.5.2).

- **Addition**: Addition is performed by introducing more rows in the matrix $(m^* \times n)$ of the starting instance. k rows are added to the starting instance where $k \in \{0.1f, 0.2f, \cdots, 0.9f\}$ and $f = m/2$ is the number of rows in the starting instance. These new k rows are selected from the rows of the original instance which are not present in the starting instance. For each starting instance 30 novel instances are created for each value of k. The size of the problem for the novel instances now becomes $((m^* + k) \times n)$. This kind of change corresponds to more rows to be covered in the starting instance which could now be $U^{'} = \{1, 2, 3, 4\}$.

- **Removing**: Novel instances are created by removing k randomly selected rows from matrix $(m^* \times n)$ where $k \in \{0.1f, 0.2f, \cdots, 0.9f\}$ and $f = m/2$ is the number of rows in starting instance. The size of the novel instances in the case of removing now becomes $((m^* - k) \times n)$. For each original problem instance 30 novel instances are created for each value of k. Removing of rows from the $U^* = \{2, 3, 4\}$ could be done as $U^{'} = \{2, 3\}$.

- **Swapping**: Swapping is performed by selecting random rows from the matrix $(m^* \times n)$ of the starting instance and replacing them with a different row from the original instance matrix $(m \times n)$. k rows are swapped between the starting instance and the original instance where $k \in \{0.1f, 0.2f, \cdots, 0.9f\}$ and $f = m/2$ is the number of rows in $(m^* \times n)$. The size of the novel instance remains the same as the starting instance for this kind of modification. For every starting instance 30 novel instances are created for each value of k. Swapping of rows between U and U^* can be done as $U^{'} = \{2, 3, 6\}$.

It should be noted that for all these changes, the subsets are allowed to remain the same, and we do not care if a particular subset covers a row that is now not required to be covered.

6.4 Introducing Memory in GC-AIS

In order to integrate a memory component into the GC-AIS, a very simple explicit memory mechanism is proposed. This memory solution is a finite store, known as explicit memory [119], which saves the best solution of the problem found by the algorithm to be used at a later stage. The stored memory is then used to initialise the GC at the beginning of the algorithm run. In other words the initialisation of the starting solution in M-GC-AIS is performed by utilising a stored solution from memory in the past. This simple extension to GC-AIS which we refer to as M-GC-AIS for the rest of the thesis can now be used for the proposed dynamic SCP. The differences between GC-AIS and M-GC-AIS are in the initialisation phase at the beginning of the algorithm and the storage of memory at the end of the algorithm, which are marked as bold text in Algorithm 10. It should be made clear that this memory scheme is different than the concept of elitism in evolutionary computation, which refers to letting the best solution be carried forward in every generation of the algorithm. The idea of memory here pertains to a known optimal solution to a problem which is saved as memory and used later when the problem changes in the dynamic scenario. This is visible in Algorithm 10 as the memory saving is only done at the very end of the algorithm after the generation loop, while elitism would reflect propagating an intermediate best solution within the loop of the algorithm.

Algorithm 10 The GC-AIS WITH MEMORY (M-GC-AIS)

Let G^t denote the population of GC at generation t and g_i^t the i-th GC in G^t.
Create GC pool $G^0 = \{g_1^0\}$ and initialise g_1^0 **from memory**. Let $t := 0$.
loop
  for each GC g_i^t in pool G^t in parallel **do**
    Create offspring y_i of individual g_i^t by standard bit mutation.
  end for
  Add all y_i to G^t, remove all dominated solutions from G^t and let $G^{t+1} = G^t$.
  Let $t = t + 1$.
end loop
Save best solution information as memory

6.5 Experimental Evaluation of m-GC-AIS on single time step Dynamic SCP

Twelve original SCP instances from the OR-library [11] are used to create the dynamic SCP instances. Table 3.1 on page 53 lists the twelve problems along with their size and density. The density refers to the percentage of 1s in the $(m \times n)$ matrix. The GC-AIS and M-GC-AIS are both run on the dynamic SCP proposed to evaluate their performance.

In this particular case the dynamic SCP is composed of two distinct time steps. In the first step the memory solution is generated and in the second step a comparison between GC-AIS and M-GC-AIS is made. The size of memory has been limited to only one in these experiments which is generated in the first step. The dynamic problem does not change frequently when compared with the speed of the algorithm and the trade-off between using a memory solution and starting from scratch is observed. As there are just two time steps, a single memory solution is stored, to be used in the second time step when the problem change occurs and it is compared with a solution generated from scratch. At time step $t = 0$ the GC-AIS is run 30 times on all the instances until no improvement is observed for $en \log n$ generations and the best solution is recorded to be used as memory. The obtained results are compared with the best known results which are shown in Table 6.2 and it is seen that values in the best known results were obtained in time step 0.

At step $t = 1$, the novel instance is presented to both the algorithms and the results are recorded. Based on the experiments performed in Chapter 4 for the any-time behaviour of GC-AIS on the original instances, a large quota of 10^5 generations is allotted to both M-GC-AIS and GC-AIS, as it can be seen that by this time no more improvement is seen in the solution quality obtained by GC-AIS. 30 independent runs of both algorithms are performed on each of the novel instances created and the mean of the sets used and uncovered sets at each generation are recorded.

Three measures used to compare and assess the algorithm performance are introduced

Problem	$m \times n$	Obtained	Known [114]
41	200×1000	(0,38)	(0,38)
52	200×2000	(0,34)	(0,34)
63	200×1000	(0,21)	(0,21)
a5	300×3000	(0,38)	(0,38)
b4	300×3000	(0,22)	(0,22)
c3	400×4000	(0,43)	(0,43)
d2	400×4000	(0,25)	(0,25)
e1	50×500	(0,5)	(0,5)
re1	500×5000	(0,17)	(0,17)
rf2	500×5000	(0,10)	(0,10)
rg3	1000×10000	(0,62)	(0,62)
rh4	1000×10000	(0,35)	(0,35)

Table 6.2: Best known and obtained solutions for the original problem instances of SCP. Column 'Known' contains solutions presented in [114] while 'Obtained' contains solutions found by GC-AIS.

in here. The difference between the average sets used at the end of the stopping criteria (D_s), the difference between the average number of generations at the feasible region (D_g), and finally the difference between the mean fitness evaluations used (D_e). If $N^{\text{GC-AIS}}$ denotes the sets used by GC-AIS and $N^{\text{M-GC-AIS}}$ denotes the sets used by M-GC-AIS at the end of the run then D_s is defined as $N^{\text{GC-AIS}} - N^{\text{M-GC-AIS}}$. Similarly, let $T^{\text{GC-AIS}}$ denote the time in generations, taken by GC-AIS to reach the feasible solution region and $T^{\text{M-GC-AIS}}$ denote the time in generations, taken by M-GC-AIS to reach the feasible region then D_g is defined as $T^{\text{GC-AIS}} - T^{\text{M-GC-AIS}}$. Finally, let $E^{\text{GC-AIS}}$ denote the fitness evaluations, used up by GC-AIS and $E^{\text{M-GC-AIS}}$ denote the fitness evaluations, used up by M-GC-AIS to reach the feasible region then D_e is defined as $E^{\text{GC-AIS}} - E^{\text{M-GC-AIS}}$. This measure can be calculated at both the end of runs as well as the time when the algorithms obtain the first feasible solution. To test the statistical significance of the results, the Wilcoxon rank-sum test is used for each value of modification performed for both the algorithms. Tables of p values are included in the appendix for completeness.

6.6 Experimental Results

In this section the results obtained by running the GC-AIS and M-GC-AIS at time $t = 1$ of the dynamic SCP are presented for both the models of the problem. Since the instance 'e1' (size (50×500)) is quite small it can be seen for all the experiments that both the algorithms reach approximately the same solution quality for every kind of modification in both the models. It can be seen from Table 3.1 that the density of this problem is high (20%) and the known best solution only contains 5 sets. Even by introducing any of the proposed modifications, the problem size remains small and both algorithms quickly find the feasible regions (can be seen in D_f plots) and by the end of the run, both are able to reach the same or extremely close final solutions. Therefore for the instance 'e1', this results in a flat line around 0 for all the plots of D_s in every type of modification.

6.6.1 Results for Model 1

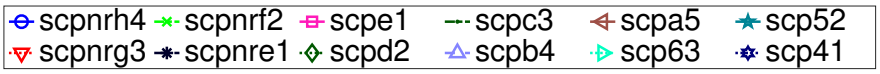
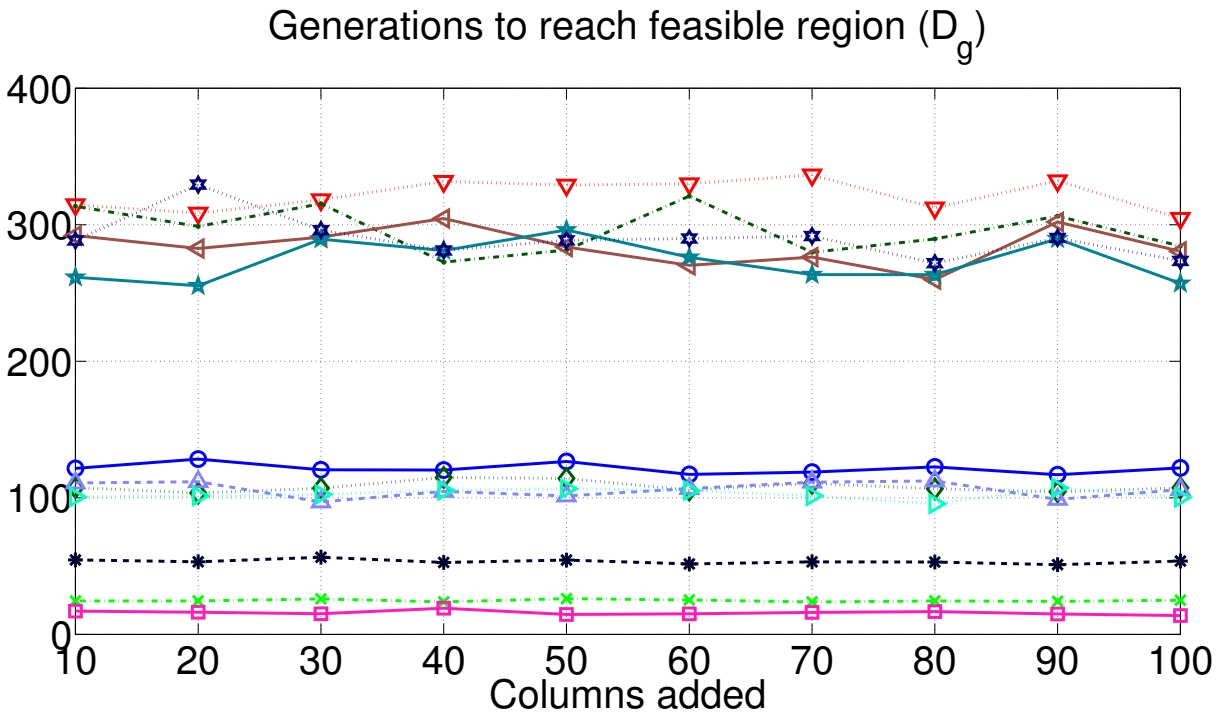
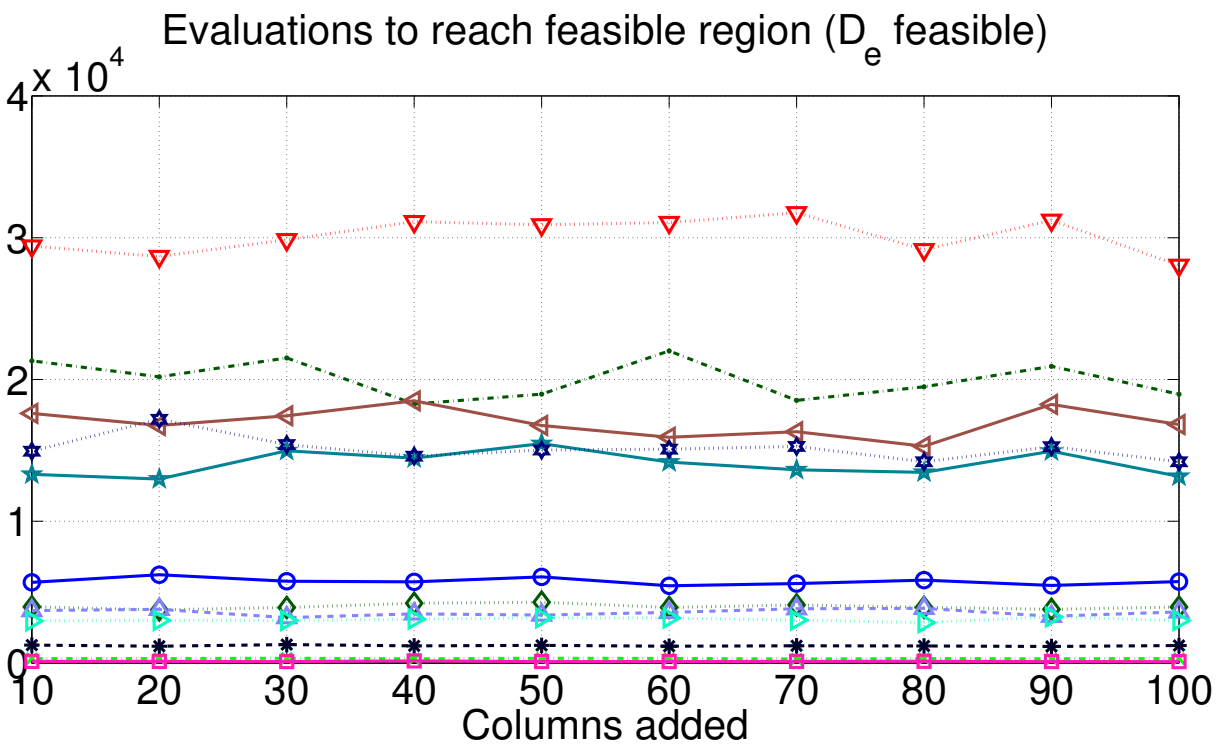


Figure 6.1: Legend of the problem instances for addition.


Generations to reach feasible region (D_g)
400
300
200
100
0
10
20
30
40
50
60
70
80
90
100
Columns added


Evaluations to reach feasible region (D_e feasible)
x 10^4
4
3
2
1
0
10
20
30
40
50
60
70
80
90
100
Columns added

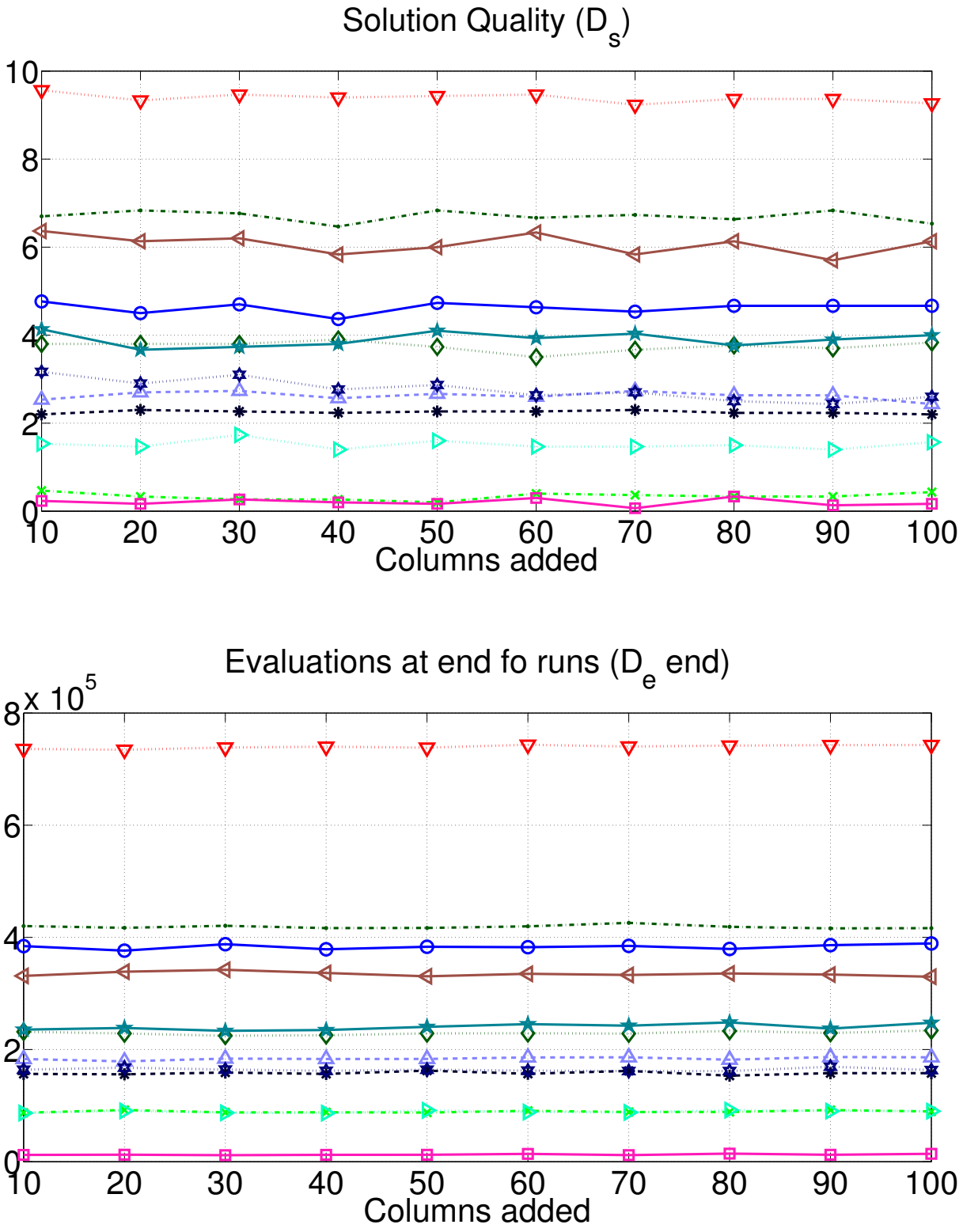


Figure 6.5: Plots of results obtained on novel instances with columns added. Plots for solution quality shown as D_s, time to reach feasible solution shown as D_g and fitness evaluations used shown as D_e averaged over 30 novel instances for each value of k.

Addition

At the start of the run at $t = 1$, despite the change in the problem size due to addition of columns, the solution in memory still remains feasible. However, by adding columns in S we might introduce new optima with a smaller number of sets as before. Therefore this case is the easiest among the three modifications for Model 1. At the end of runs, it can be seen from Figure 6.5 that the points for D_s are always positive, which implies that the number of sets used by GC-AIS is more than M-GC-AIS. This makes sense as M-GC-AIS starts with a valid solution and then continues to improve it for the duration of the whole run, while GC-AIS starts building a solution from scratch and is unable to reach the solution quality of M-GC-AIS. The plots for D_g show values $\gg 1$, which is easy to understand as M-GC-AIS has a feasible solution since generation 1. Therefore the plotted values are time taken by GC-AIS to reach the feasible region minus 1. The statistical test for D_g shows that the two algorithms are significantly different. Values of D_e for both the end of runs and the time the algorithms reach feasible region show that GC-AIS uses more evaluations than M-GC-AIS. Hence, for the addition of columns it is always better to start with a memory solution since only the size of the problem has changed and the memory solution is still feasible.

Removal

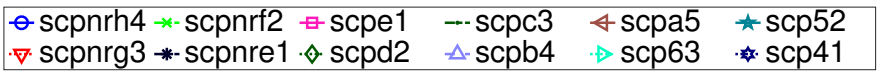
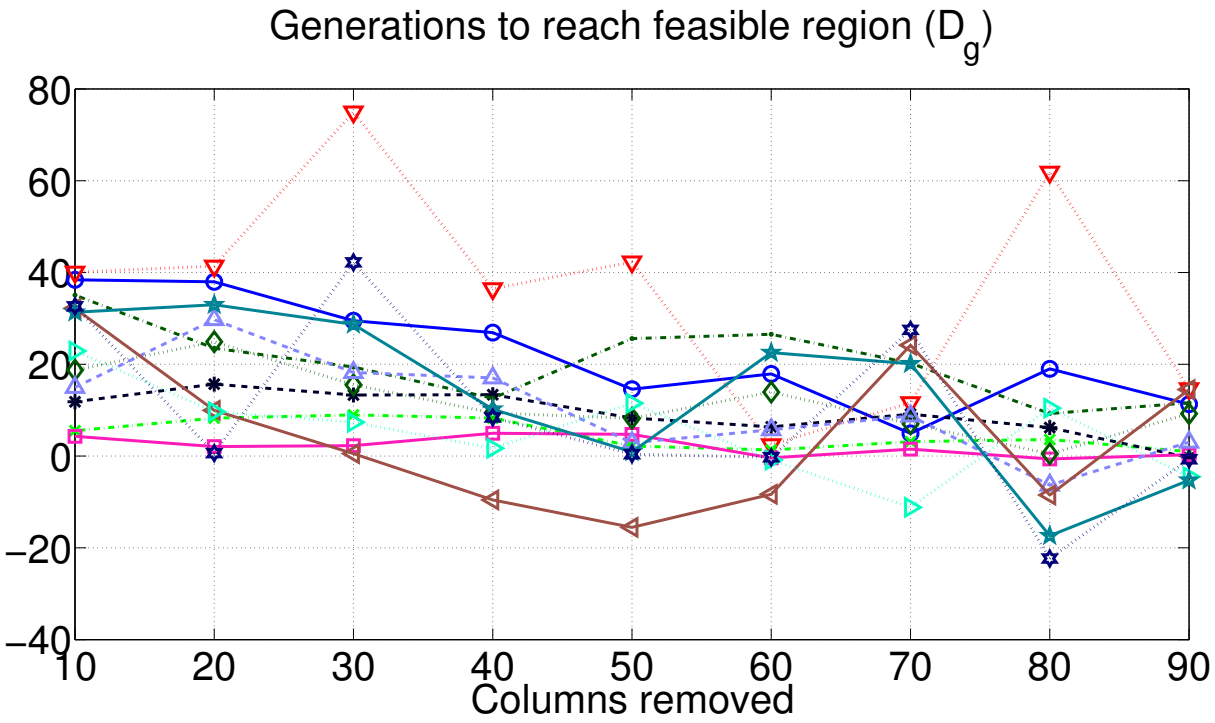
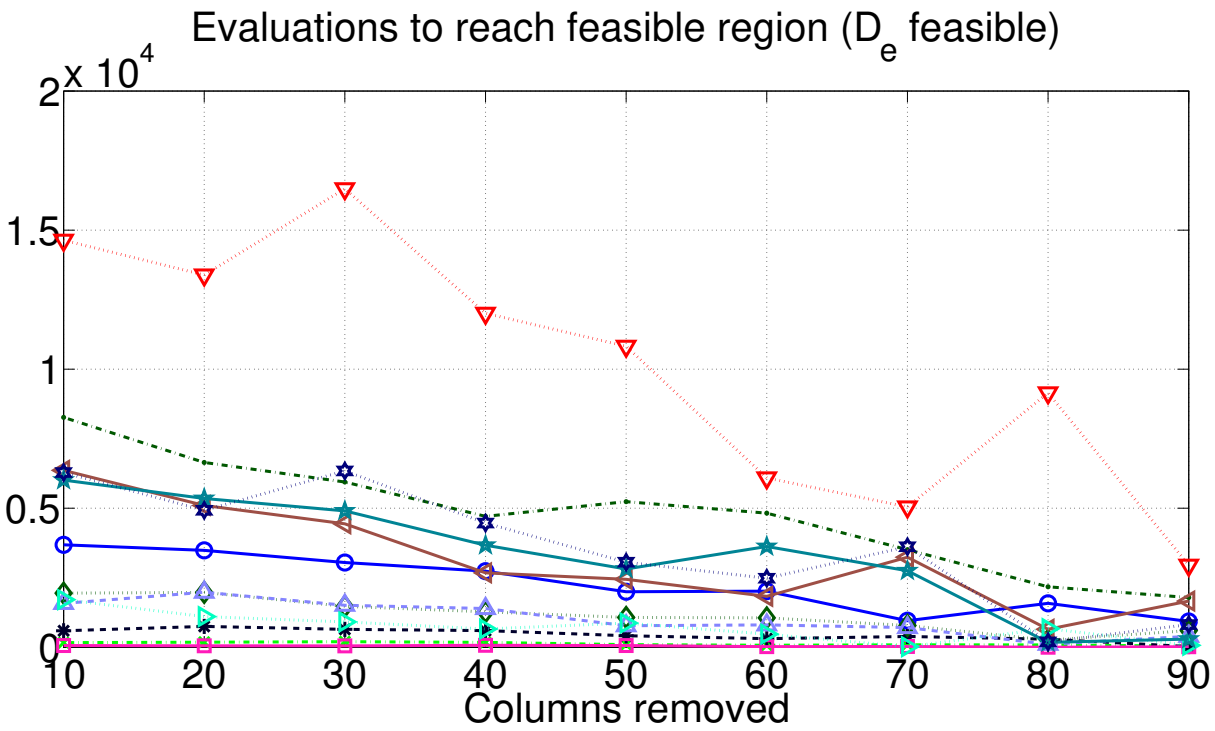
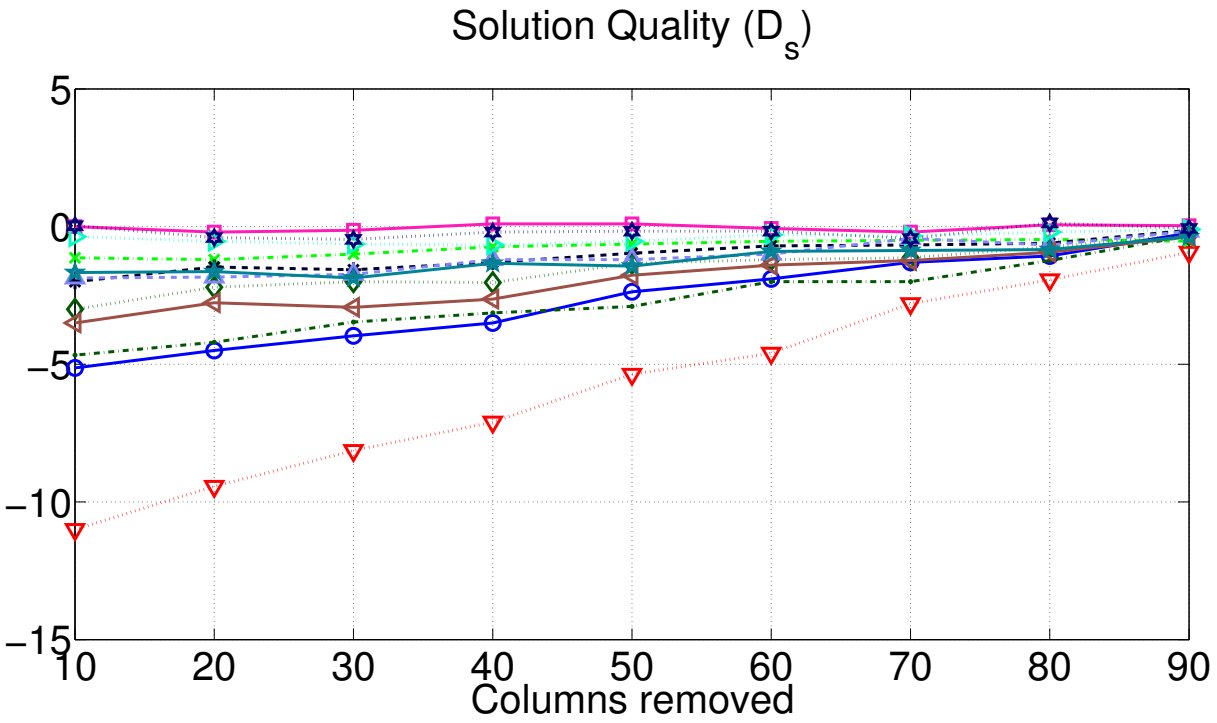


Figure 6.6: Legend of the problem instances for removal.


Generations to reach feasible region (D_g)
80
60
40
20
0
−20
−40
10
20
30
40
50
60
70
80
90
Columns removed


Evaluations to reach feasible region (D_e feasible)
x 10^4
2
1.5
1
0.5
0
10
20
30
40
50
60
70
80
90
Columns removed



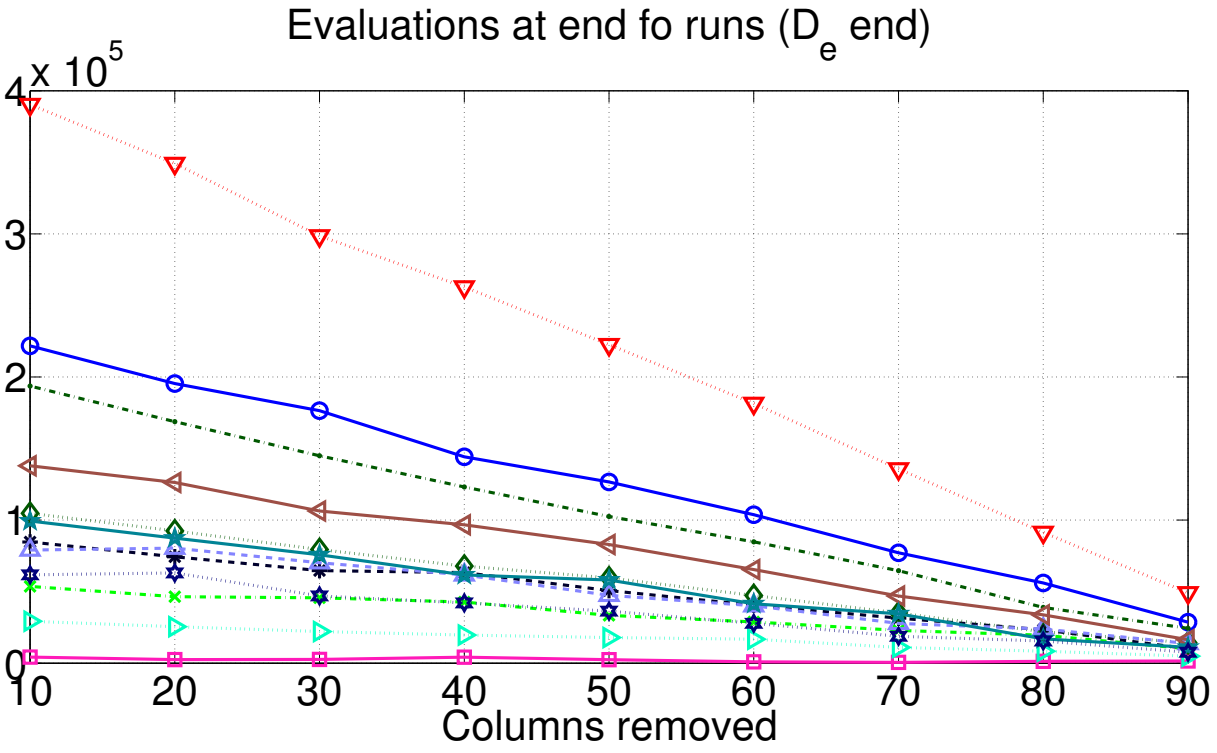


Figure 6.10: Plots of results obtained on novel instances with columns removed as percentage of c on x-axis. Plots for solution quality shown as D_s, time to reach feasible solution shown as D_g and fitness evaluations used shown as D_e averaged over 30 novel instances for each value of k.

It can be seen from the plots in Figure 6.10 that a clear trend is visible for D_s. As the number of sets removed is increased the values of D_s approach zero. This implies that for small values of removal, GC-AIS uses less sets than M-GC-AIS and as the values of k approach $0.9c$ the two algorithms use almost the same number of sets. Except for two instances, the statistical tests for D_s show that both algorithms are significantly different up to $k = 0.7c$. The interpretation of this result can be explained as follows: due to removal of sets the memory solution is no longer feasible and it behaves as a random string, while starting with all $0s$ string for GC-AIS gives better results. This is similar to the observation in experiments performed in Section 4.3.1 where starting from the all $0s$ string was better than starting with a randomly generated string. Values of D_g do not seem to follow any pattern and the statistical tests did not show any significant difference. Based on plots of D_e, a pattern can be observed which starts weakly in the sub-figure for D_e at feasible region (second from top sub-figure on left column of Figure 6.10) and becomes much more stronger in sub-figure for D_e at the end of runs. It can be seen that the difference between the fitness evaluations used by GC-AIS and M-GC-AIS decreases as the number of columns removed increases. Therefore it is better to use GC-AIS when only a few items are removed and as the number of of items removed increases both algorithms tend to behave in a similar manner.

Editing

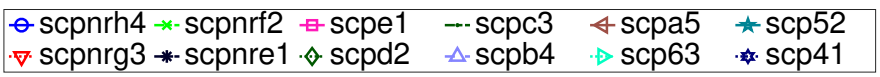
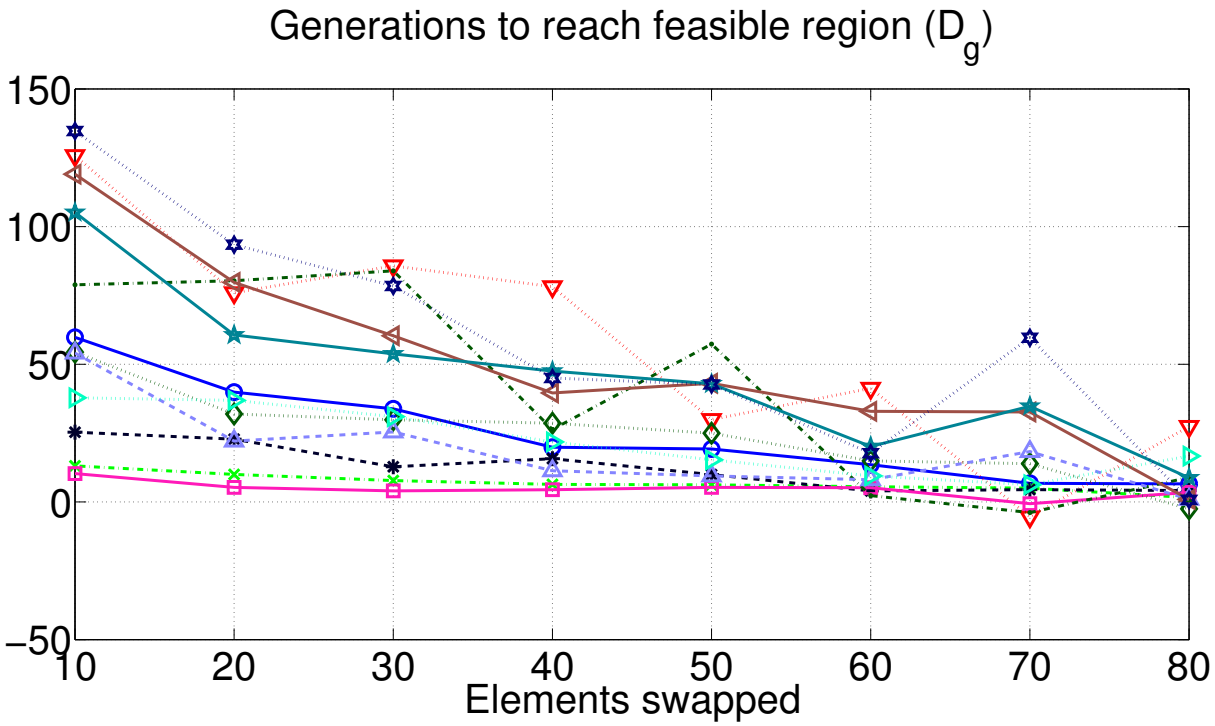
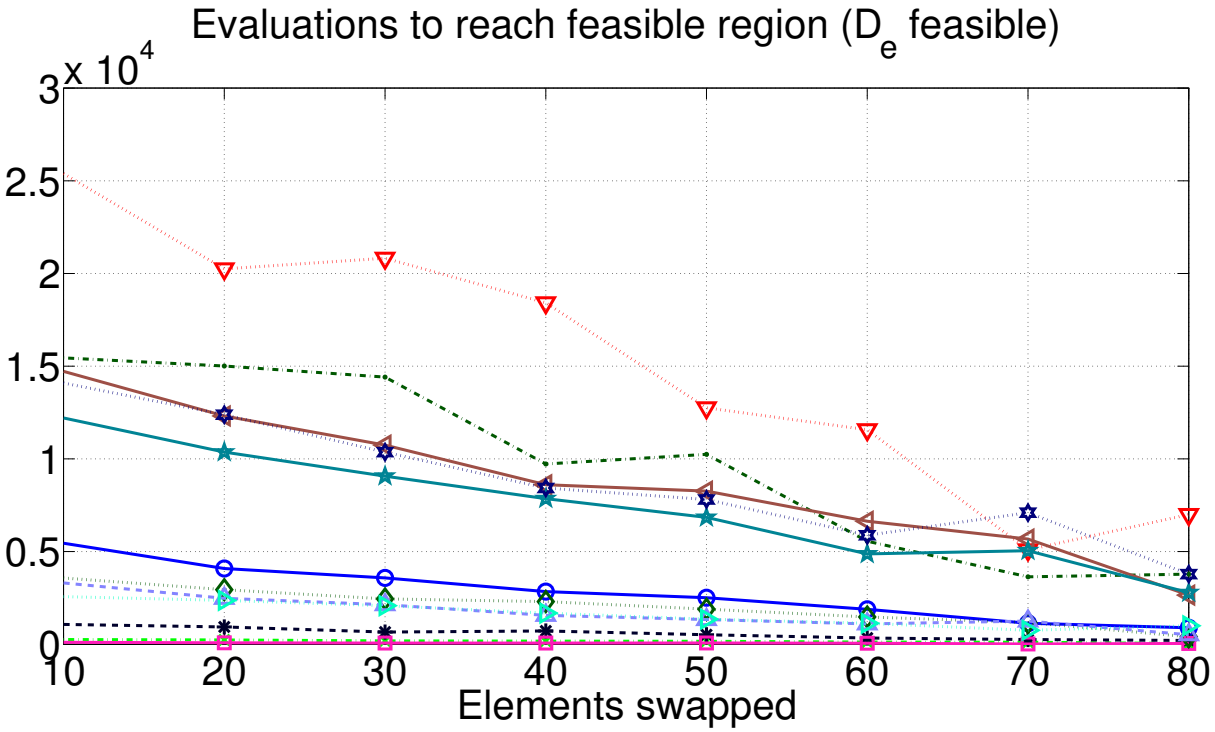


Figure 6.11: Legend of the problem instances for editing.


Generations to reach feasible region (D_g)
150
100
50
0
−50
10
20
30
40
50
60
70
80
Elements swapped


Evaluations to reach feasible region (D_e feasible)
x 10^4
3
2.5
2
1.5
1
0.5
0
10
20
30
40
50
60
70
80
Elements swapped

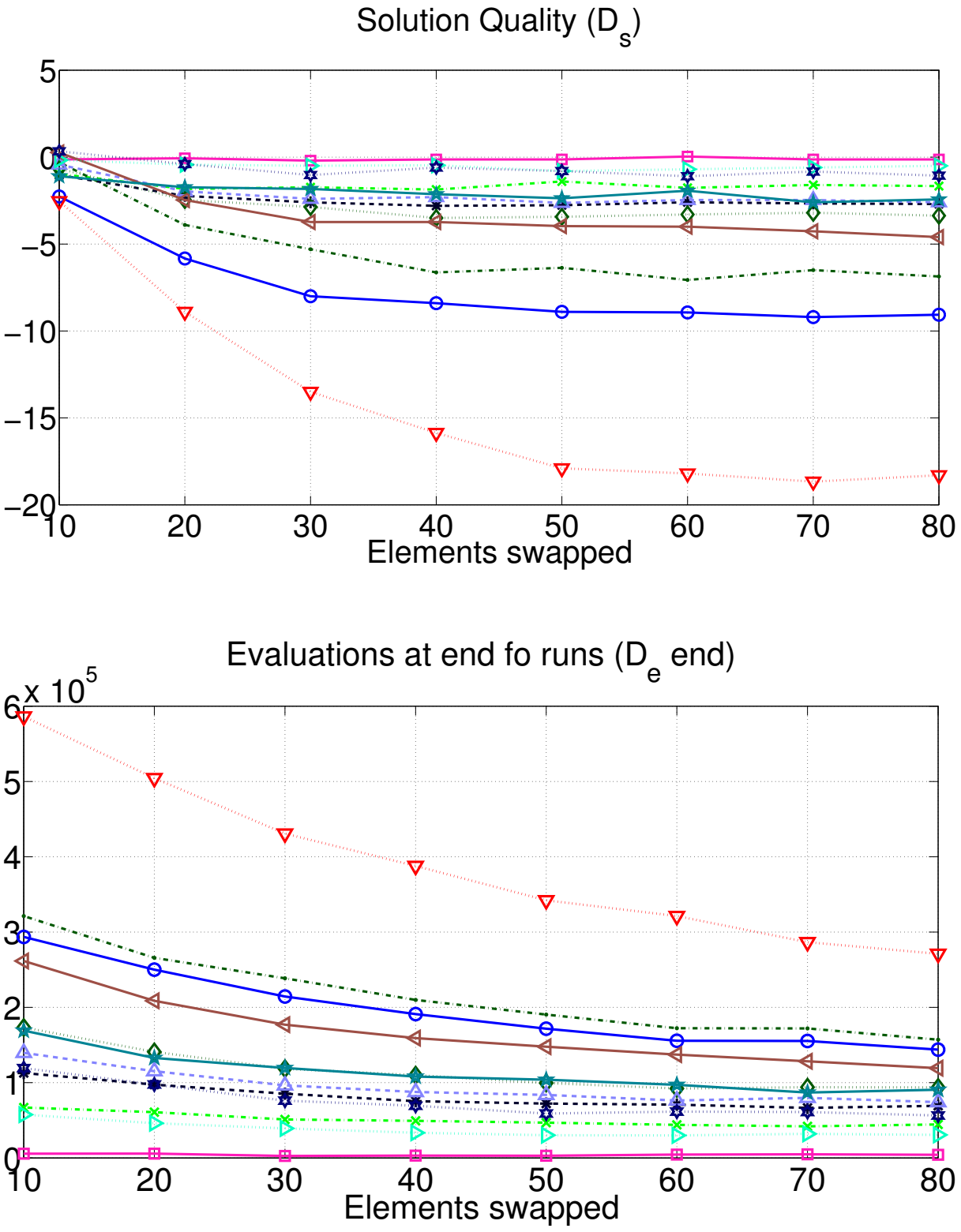


Figure 6.15: Plots of results obtained on novel instances with columns edited as percentage of d on x-axis. Plots for solution quality shown as D_s, time to reach feasible solution shown as D_g and fitness evaluations used shown as D_e averaged over 30 novel instances for each value of k.

A much more clear pattern can be observed for the case of editing which is depicted in Figure 6.15. For all values of d, the data points plotted for D_s decrease and tend to get increasingly negative. This means that for smaller editing up to $k = 0.1d$, GC-AIS uses almost the same number of sets as M-GC-AIS at the end of the runs. Therefore, using memory is preferred for very small edits ($kupto0.1d$) while for larger edits ($k > 0.1d$) the difference becomes negative and GC-AIS becomes better than M-GC-AIS. The statistical test performed for D_s shows that the two algorithms are significantly different except for the small values for four instances where the test was unable to determine any significant difference. The difference between the fitness evaluations declines, which can be seen in the plots of the 2 D_e values.

An explanation of this behaviour can be made as follows: for smaller editing the solution from memory might require either very few or no new sets to make it feasible. Few sets might be required in the case when few elements, which are no longer covered due to the editing, are not present in any other sets in the solution. No sets might be required if the removed item did not affect the solution as that item is present in another set of the solution. For larger changes the memory solution starts behaving like a randomly generated solution and hence is worse than starting from scratch for GC-AIS. Therefore for very low values of editing using memory is preferred while for larger editing starting from scratch might be a better option at the expense of slightly more fitness evaluations. The plots for D_g show that GC-AIS takes more time to find the feasible solution than M-GC-AIS but this difference decreases slowly to almost 0 as the amount of editing increases. This is evident from the statistical results where it can be seen that for lower values of editing up to $k = 0.4d$ the algorithms are statistically different. Similar behaviour is observed in the plots of D_e where the difference between the evaluations required by GC-AIS and M-GC-AIS decreases as the percentage of editing increases.

6.6.2 Results for Model 2

Addition

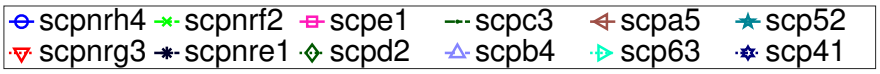
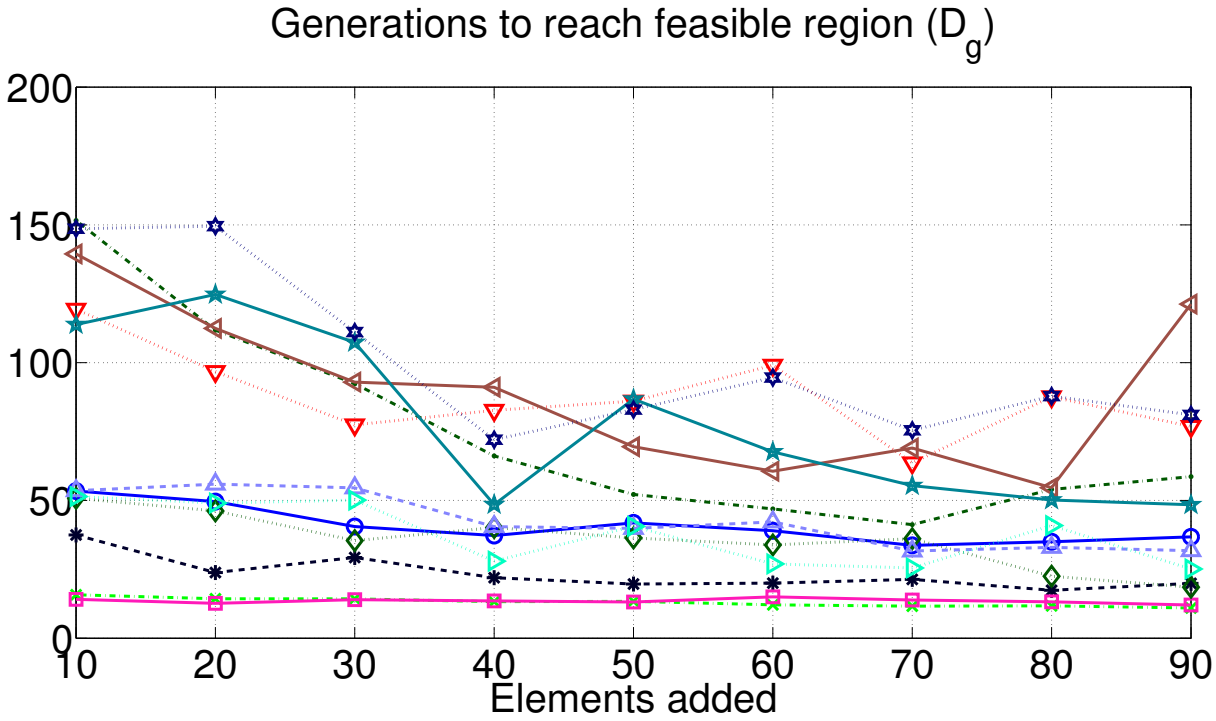
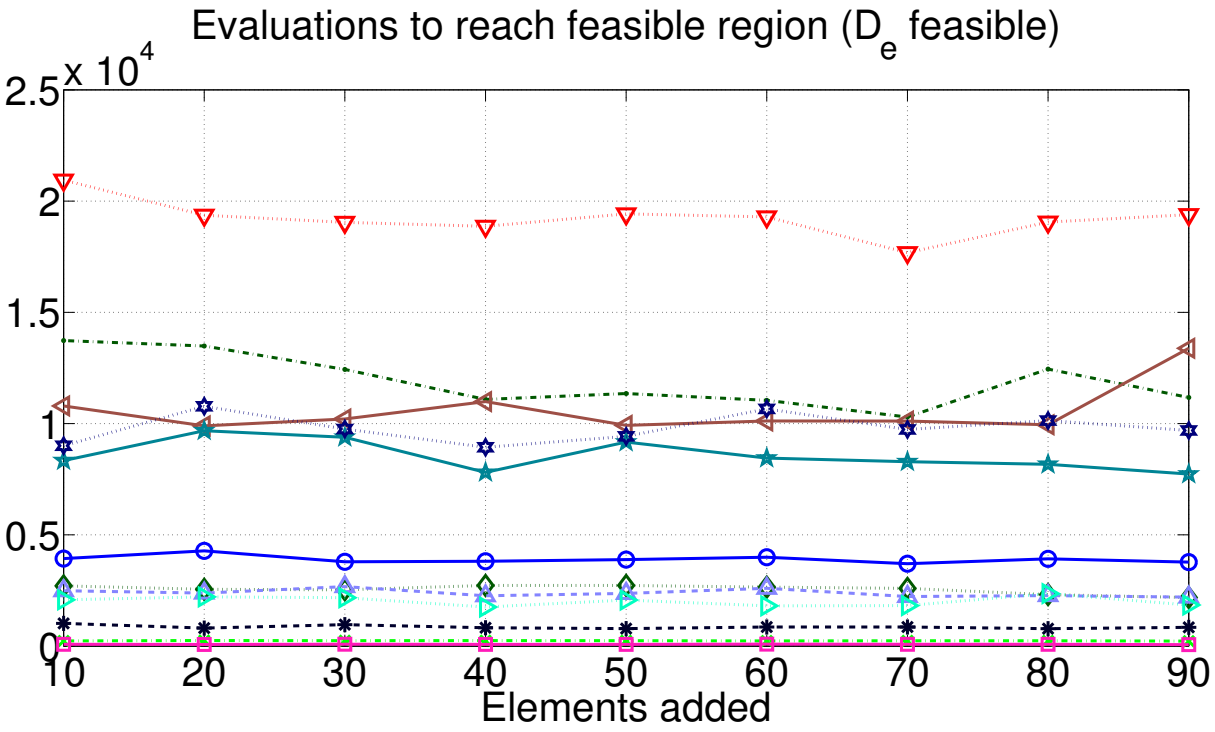
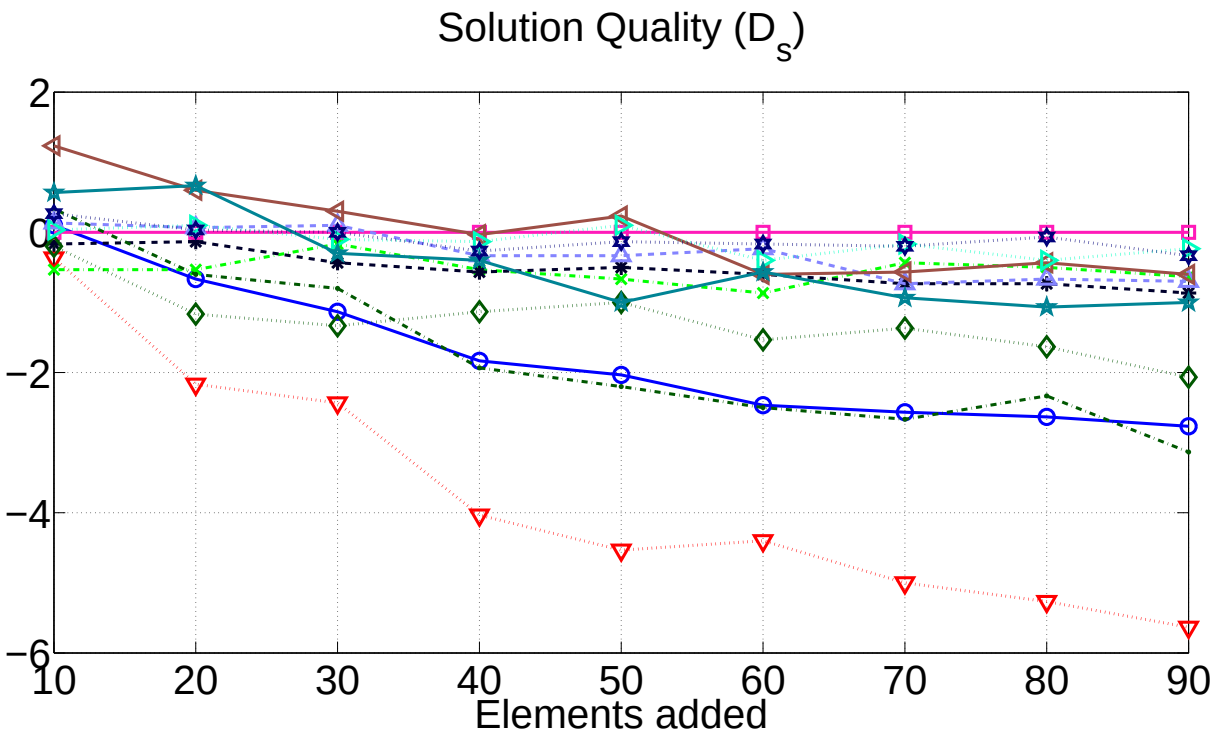


Figure 6.16: Legend of the problem instances




Evaluations to reach feasible region (D_e feasible)
x 10^4
2.5
2
1.5
1
0.5
0
10
20
30
40
50
60
70
80
90
Elements added


Solution Quality (D_s)
2
0
−2
−4
−6
10
20
30
40
50
60
70
80
90
Elements added

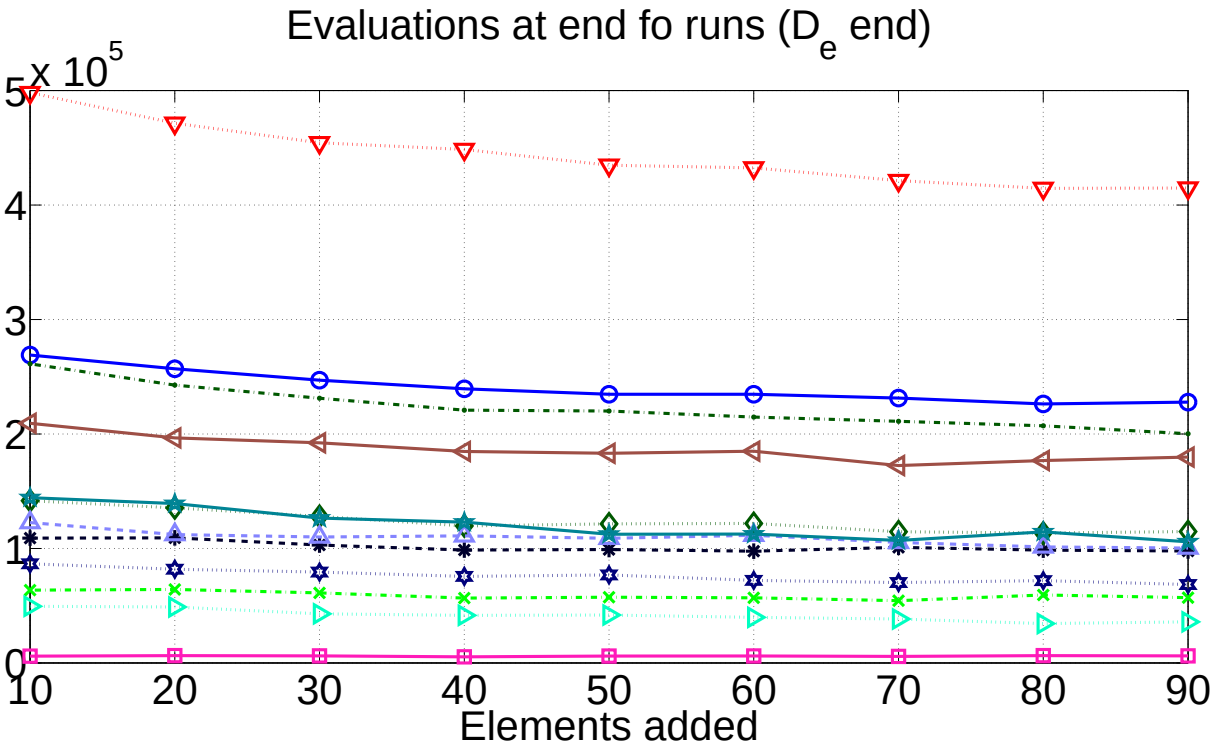


Figure 6.20: Plots of results obtained on novel instances with rows added as percentage of f on x-axis. Plots for solution quality shown as D_s, time to reach feasible solution shown as D_g and fitness evaluations used shown as D_e averaged over 30 novel instances for each value of k.

For addition of elements to the set U it can be seen from the Plots of D_s in Figure 6.20 that for small addition of up to $k = 0.2f$ the difference is positive or zero for most instances, which implies that the sets used by GC-AIS are more than the sets used by M-GC-AIS. Therefore, using memory is desirable in this case. When more elements are added then the difference becomes negative showing that it becomes better to use GC-AIS in this case. This can be explained as in the case of small addition of elements (k up to 0.2f), the sets in the memory solution might already cover the new element therefore making the memory solution desirable. However when the added elements are large ($k >$ 0.2f) then the memory solution starts behaving as a random string. The plots for D_g and D_e show positive values for all values of k which means that GC-AIS takes more iterations and evaluations to reach the feasible region as well as more evaluations at the end of runs in this case. It can be said that for the case of adding rows, using memory is beneficial for small additions, while for larger additions starting from scratch is better in terms of

solution quality obtained at the cost of more evaluations.

Removal

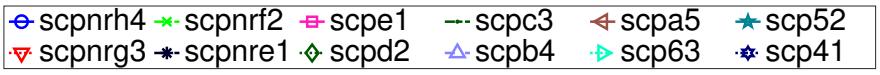
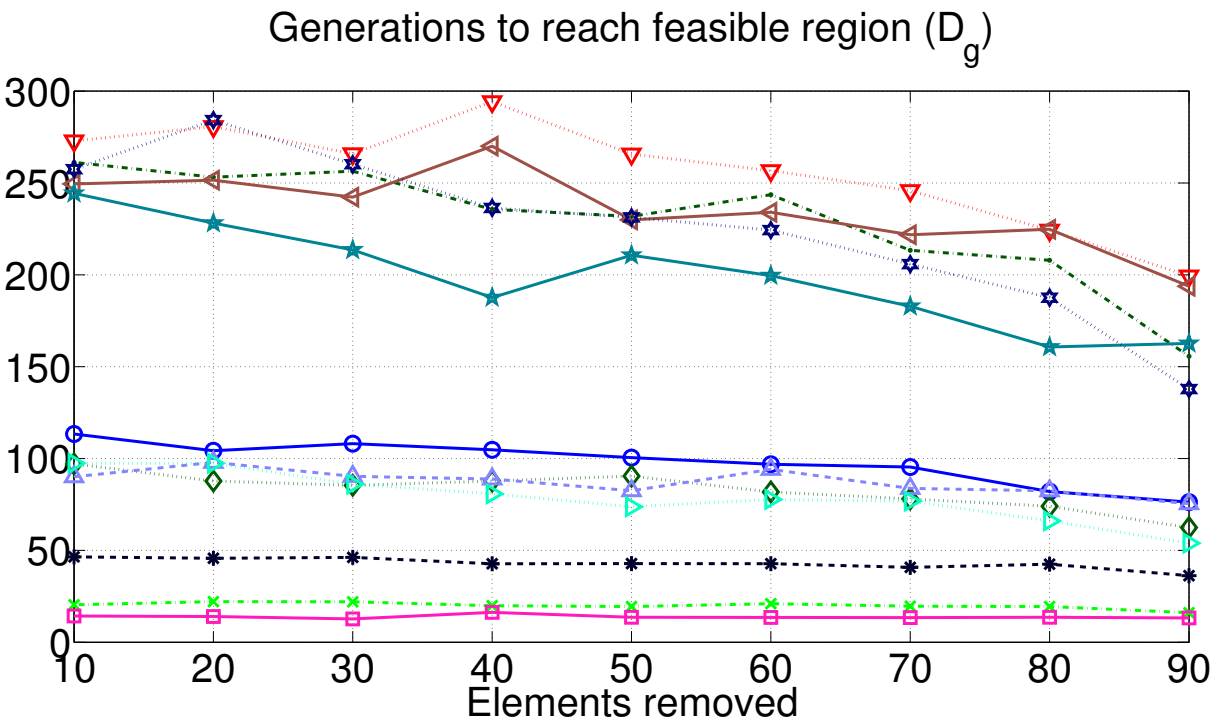
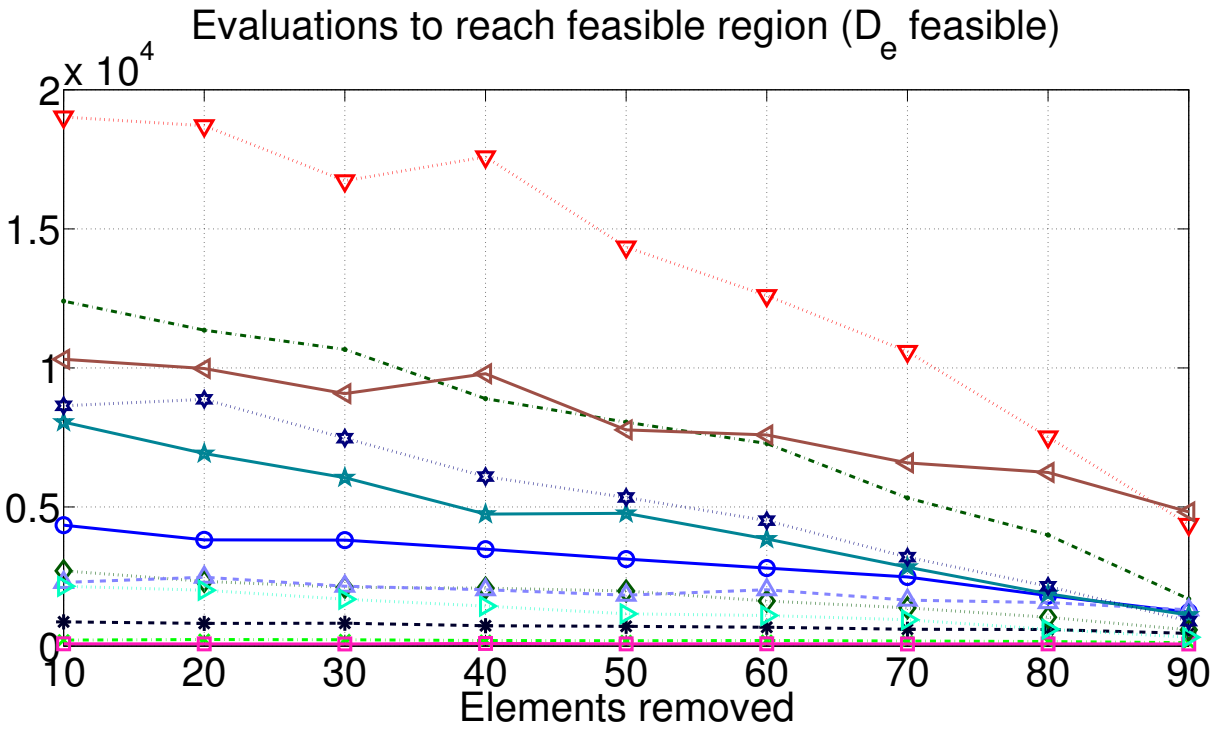
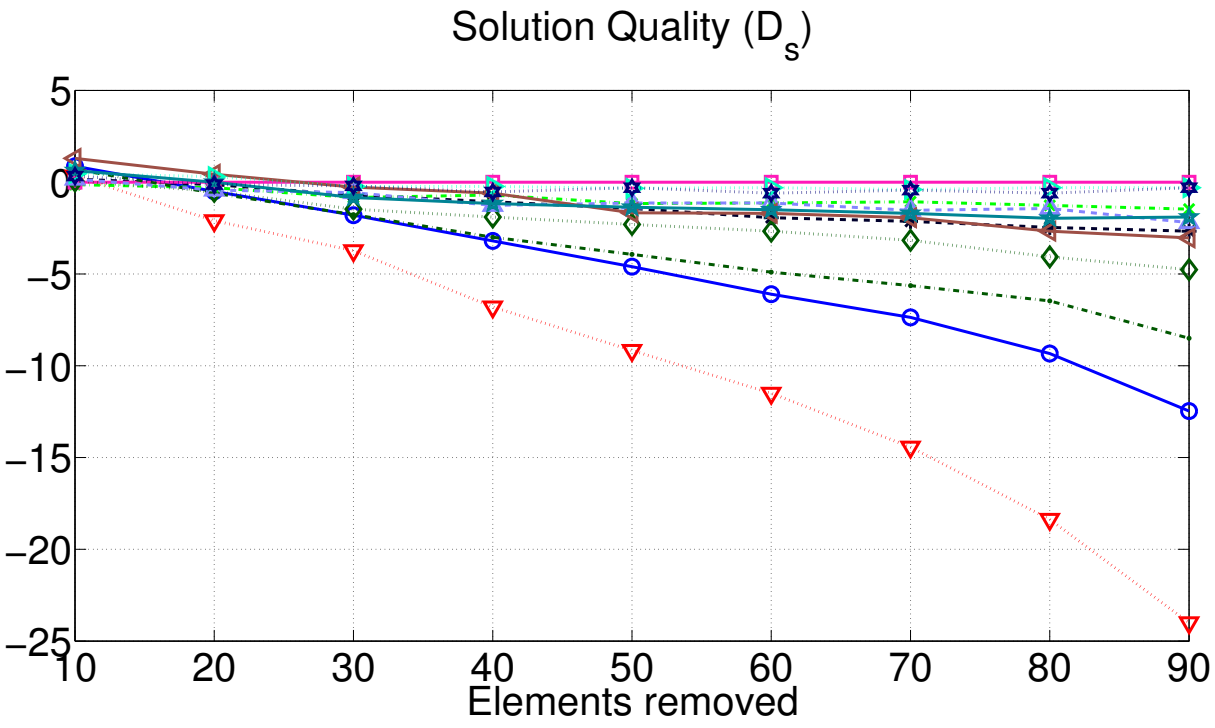


Figure 6.21: Legend of the problem instances




Evaluations to reach feasible region (D_e feasible)
x 10^4
2
1.5
1
0.5
0
10
20
30
40
50
60
70
80
90
Elements removed


Solution Quality (D_s)
5
0
−5
−10
−15
−20
−25
10
20
30
40
50
60
70
80
90
Elements removed

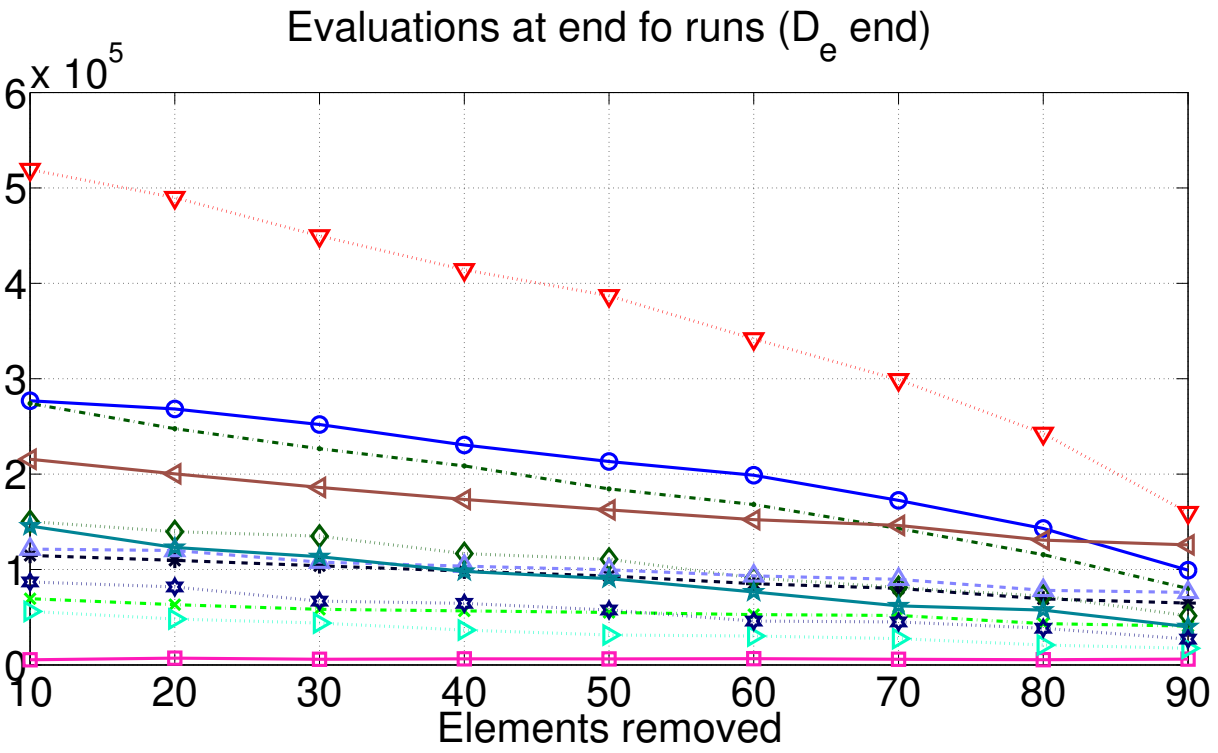


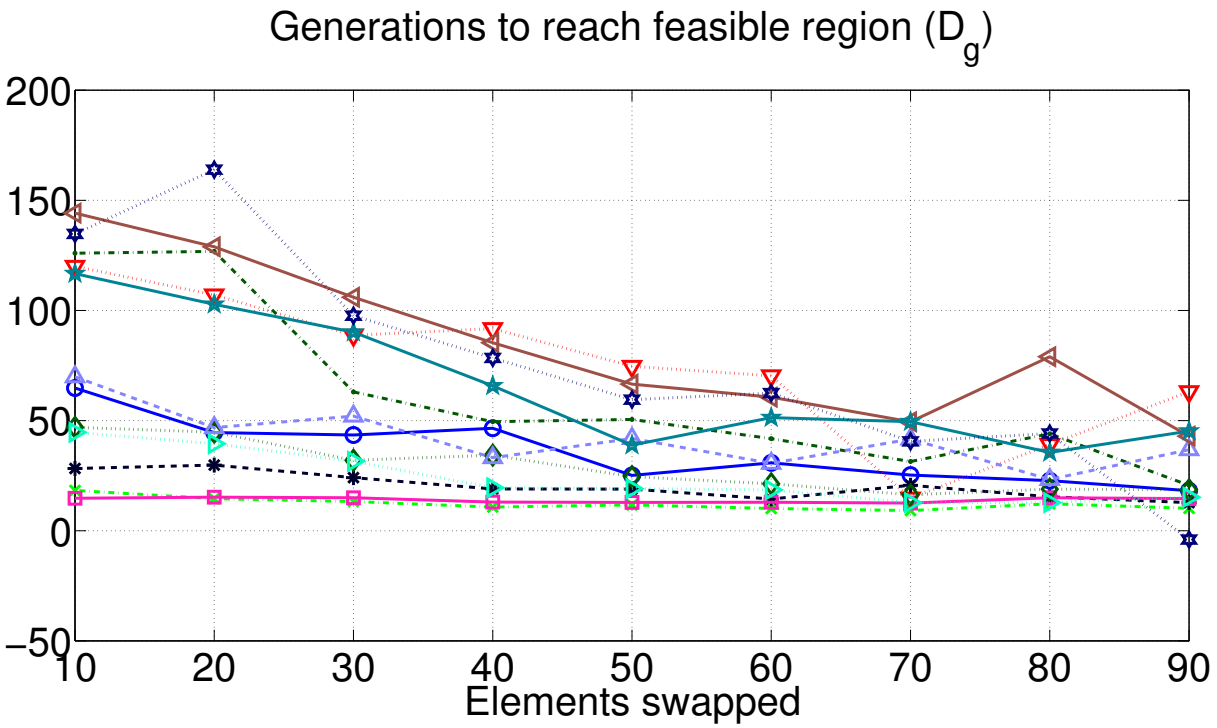
Figure 6.25: Plots of results obtained on novel instances with rows removed as percentage of f on x-axis. Plots for solution quality shown as D_s, time to reach feasible solution shown as D_g and fitness evaluations used shown as D_e averaged over 30 novel instances for each value of k.

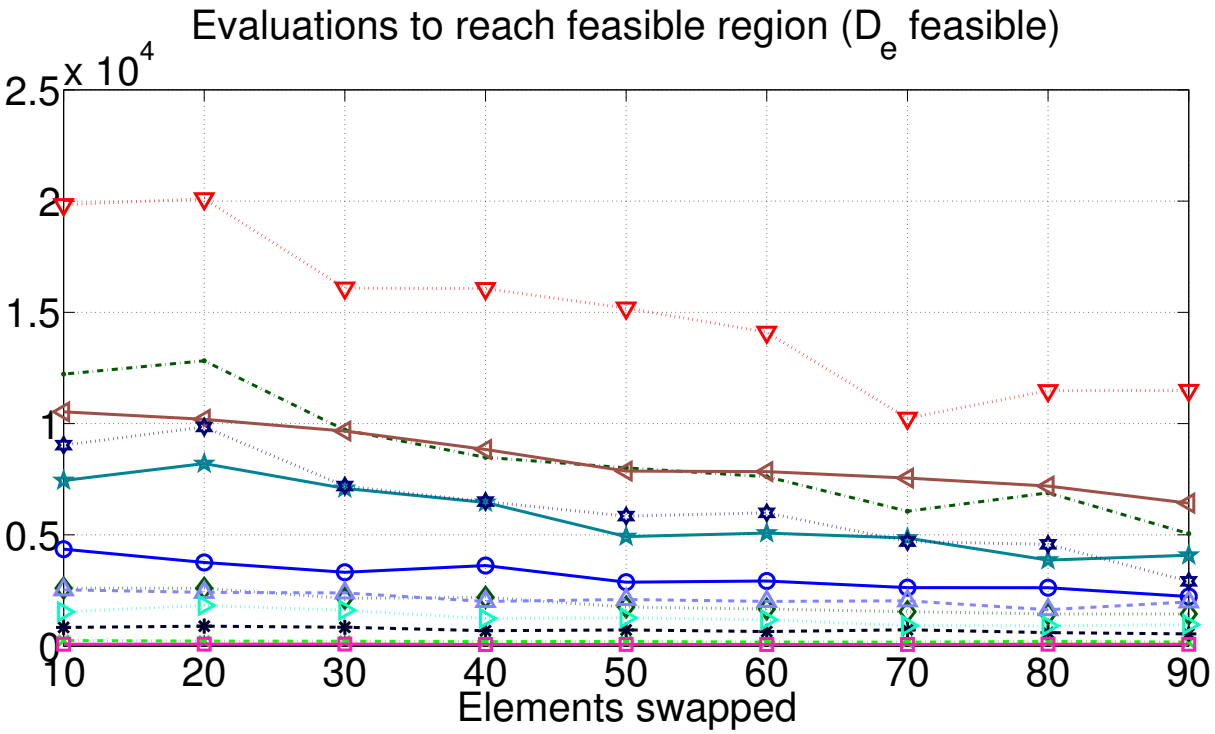
The plots for removing rows depict a distinct pattern for values of D_s in Figure 6.25. For values of k up to $0.2f - 0.3f$, the plots are either zero or very slightly positive while for higher values of k the plots become negative. This means that for low to medium changes (k up to 0.1f - 0.3f) using memory is better as GC-AIS uses either more or equal sets in this case, while for higher removal (k $>$ 0.4f) of rows starting from scratch may be better in terms of solution quality. The statistical test shows significant difference for the two algorithms except when D_s values are ≈ 0. To understand this result remember that in this case the memory solution is always feasible from the start. Since removal of elements requires less rows to be covered and the memory solution already covered these rows, the memory solution remains feasible. The problem now becomes about solution quality, for low modifications the memory solution is still good enough and with slight changes during the course of the algorithm run it continues to improve. For larger modifications on the other hand, even though the memory solution might be valid, it covers too many

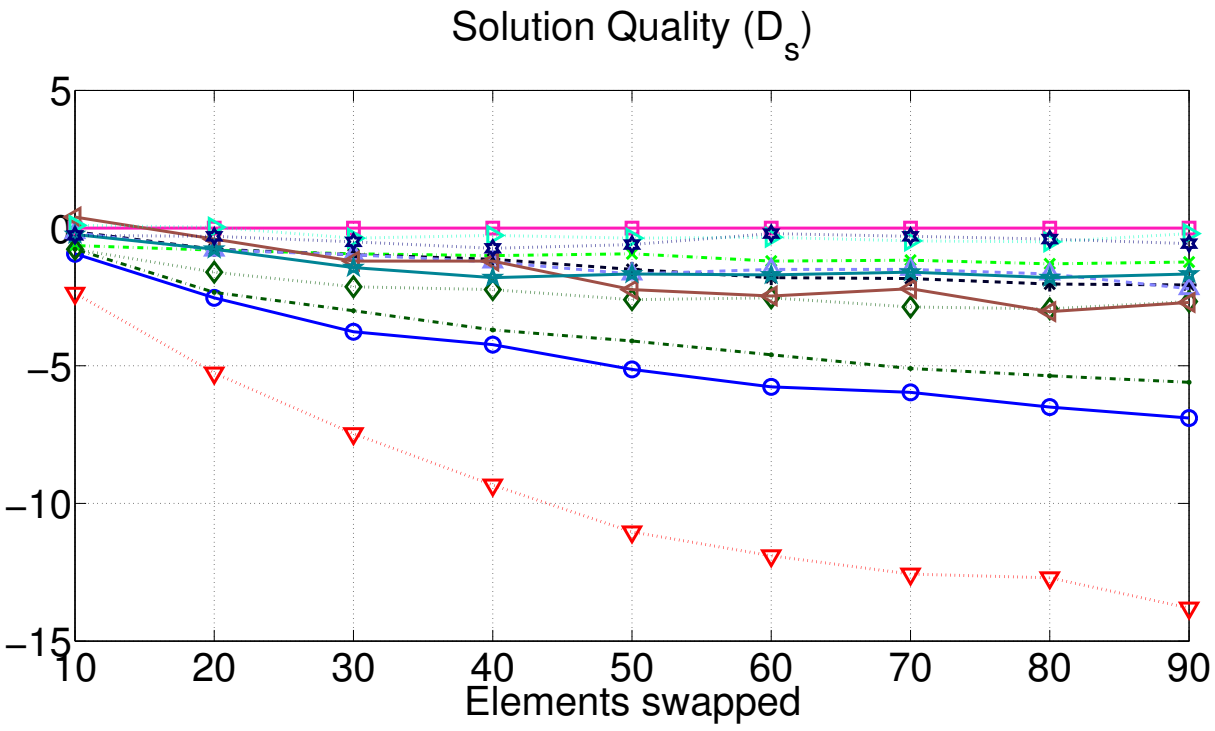
unneeded rows. In this case it turns out that starting from the all 0s string is a better idea. This behaviour is reflected in the plots for D_g where values are always positive meaning GC-AIS takes more time to reach the feasible region than M-GC-AIS. Plots for D_e shows that GC-AIS uses more evaluations than M-GC-AIS for all levels of change though the difference decreases as the number of elements removed increases. The Wilcoxon rank sum test shows that the two algorithms are significantly different for D_f.

Swapping

Figure 6.26: Legend of the problem instances




Evaluations to reach feasible region (D_e feasible)
x 10^4
2.5
2
1.5
1
0.5
0
10
20
30
40
50
60
70
80
90
Elements swapped


Solution Quality (D_s)
5
0
−5
−10
−15
10
20
30
40
50
60
70
80
90
Elements swapped

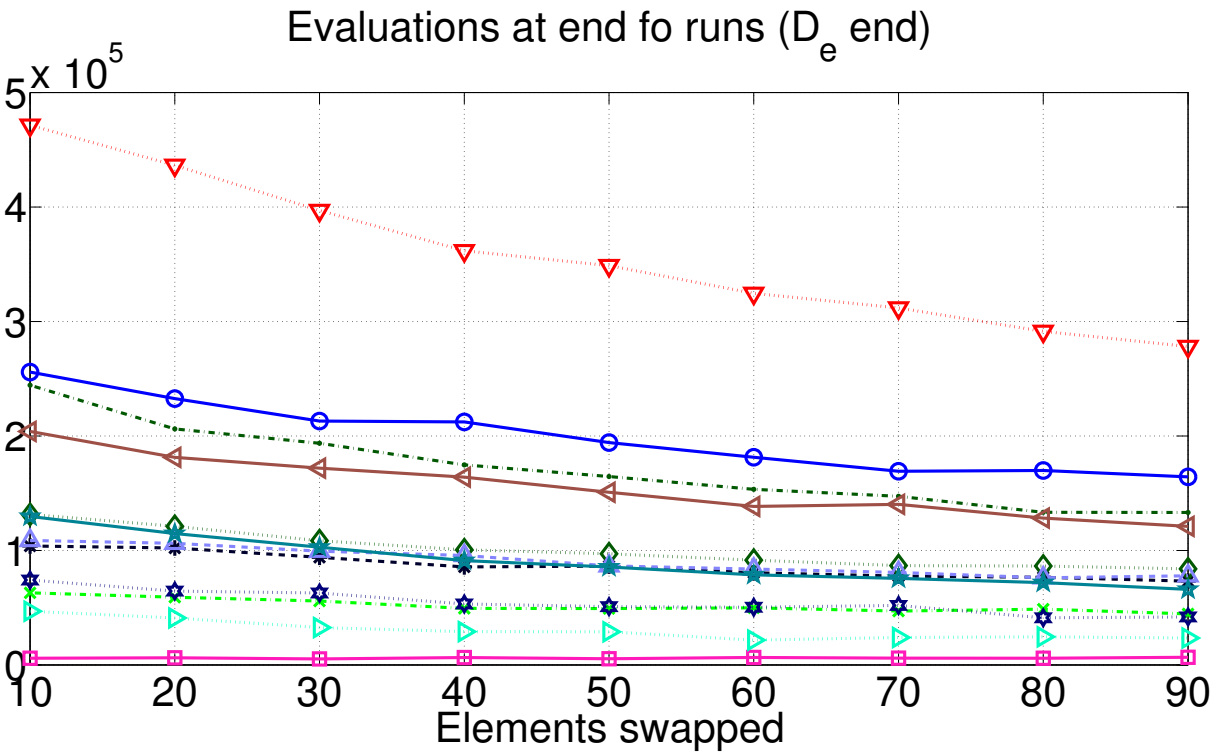


Figure 6.30: Plots of results obtained on novel instances with rows swapped as percentage of f on x-axis. Plots for solution quality shown as D_s, time to reach feasible solution shown as D_g and fitness evaluations used shown as D_e averaged over 30 novel instances for each value of k.

The plots for swapping in Figure 6.30 show a drastic decrease in the value of D_s. For a low level of swapping rows $k \leq 0.1f$, it is observed that using memory is better as the plot values are greater or close to zero. For values of $k \geq 0.2f$, it can be seen that the plots become negative implying that the GC-AIS is better. The Wilcoxon rank-sum test is unable to show statistical significance for low levels of swapping where the values of $D_s \approx 0$. Plots of D_e show that GC-AIS uses more evaluations than M-GC-AIS in all cases, but the difference tends to decrease as the numbers of swaps increases. For low levels of swapping rows, the sets in the memory solution may require very few or no additions as the swapped rows may already be covered or are covered early in the run of the algorithm with small variations of the solution. On the other hand for larger swapping, the memory solution is always of very poor quality as most of the rows it covers are no longer needed to be covered. Hence the GC-AIS becomes better when the amount of swapping is high. The plots for D_g are all positive which means that the GC-AIS takes

more time to reach the feasible region and the statistical test shows significant difference up to $k = 0.6f$. Therefore for swapping rows, it is useful to use memory for very low modifications, while for higher level of changes it is better to start from scratch at the cost of some more fitness evaluations.

6.7 Discussion and Conclusion

Model 1 introduces changes to the family of subsets S. In this model when subsets are added to S it is shown that using memory is always beneficial. In this kind of modification the memory solution always remains feasible for the created novel instances and the algorithm exploits this memory from the start of the run. When removing subsets from S is performed, it is seen that memory is always worse than starting from the all 0s string. When editing is performed inside subsets of S, it is observed that for very low levels of editing memory is useful while for higher degree of changes memory should be avoided.

Model 2 consists of changes to the universe set U of the problem. Changes are first made to the original instances such that only certain elements are required to be covered in the universe and then three kinds of modifications are introduced. Instances produced by addition of elements in the universe are better solved by using memory for small additions. For larger additions using memory is detrimental. For small additions, this is the case as solutions in memory may already cover some additional required elements, while for larger additions the memory solution starts behaving like a random string. For the case of removing elements from U it is seen that for low modification, memory is useful while for higher degree of changes it is better not to use memory. This occurs as if the removal percentage is high then the memory solution is of low quality as it tends to utilise more unnecessary sets and starting from all 0s performs better. Finally for the case of swapping elements, using memory is advised for very low degree of swaps and for higher number of swaps solutions must be found from scratch.

Understanding the behaviour of memory under different dynamic circumstances can be useful in the design of meta-heuristics used for dynamic optimisation. Strategies on the storage and use of memory can be devised based on the behaviour learnt‘ from the results in this chapter. For example, if the magnitude of change in a dynamic problem environment is known then based on the insights obtained, a strategy can be devised which can store and use memory or scratch solutions at the time of change. Furthermore this can be combined with prediction based techniques which can predict the magnitude of the change and employ a suitable strategy to use memory or scratch solution. Besides, design of new meta-heuristics, the insights gained here can help understand the behaviour of other memory based dynamic optimisation algorithms in literature and shed light on the reasons for their successes and failures.

To conclude based on these results, it can be said that memory solutions behave differently when different types of changes are made to a problem. Therefore understanding the types of changes to the problem should be a key consideration when employing memory techniques in dynamic scenarios so that the right choice of utilising memory or randomly generated solutions is made. By Knowing the behaviour of memory solutions under different dynamic problem scenarios, clever memory management schemes can be devised which can lead to better performance of meta-heuristics which employ memory techniques to solve dynamic problems.

6.8 Summary

The effect of incorporating memory in GC-AIS are studied in this chapter. Two models of dynamic SCP are introduced each with three kinds of modifications. Using these models to create instances of dynamic SCP, the performance of GC-AIS with and without a memory component is tested. It was observed that with different kinds of modifications to the problem, memory behaves differently. This insight is useful to learn about the behaviour of memory under different circumstances, which can be used in the design as

well as understanding the behaviour of meta-heuristics which employ memory techniques for solving dynamic optimisation.

CHAPTER 7

CONCLUSION

7.1 Introduction

This thesis presented a novel artificial immune system called GC-AIS which is inspired by advancements in the understanding of the immune system. Important characteristics of the immune system are identified, namely multi-objective and dynamic behaviour and they are chosen as problem domains to examine the performance of the GC-AIS. The algorithm is first developed to tackle multi-objective optimisation problems and extended with a memory component to handle dynamic scenarios. Adopting a problem focused approach, the immune system's goals are observed to be analogous to the goals of the set cover problem. Significant insights into the working and behaviour of the algorithm have been gained in the context of optimisation, particularly multi-objective and dynamic optimisation.

The structure of this section is as follows: in section 7.2 the answers to the research questions asked in Chapter 1 are provided. This is followed by a section which details the knowledge gained in this thesis, which is followed by a summary of contributions in section 7.4. Finally the future direction of research are provided in section 7.5.

7.2 Answers to research questions

At the beginning of the this thesis, several key questions pertaining to what the research in this thesis presents, have been stated. Summarised answers to these questions are provided below.

Is it possible to model the recent understanding of the GC *reaction into an immune system that tackle the multi-objective* SCP*?* This question is addressed in Chapter 4 where the GC-AIS is presented. The germinal centre reaction is studied and an abstract model of an artificial immune is obtained. The model of the system is then converted to an artificial immune system and presented as the GC-AIS algorithm. The proposed algorithm has some promising features, namely, an easy to understand model and relatively few parameters to optimise. Due to the nature of AIS, certain immune specific keywords are used in the design of the model, yet at the same time clear correspondence with other well known features from evolutionary computation is provided to help the reader understand the model. A quantitative and qualitative measure of the performance of the algorithm is covered in the next few questions.

How does the novel artificial immune system perform on the set cover problem? In Chapter 4 an analysis on the performance of GC-AIS is presented on the SCP. The GC-AIS is compared with a simple multi-objective evolutionary algorithm called PGSEMO and it is shown that GC-AIS performs better than PGSEMO. GC-AIS is faster than PGSEMO to reach the feasible region as well as uses lower fitness evaluations to reach the best solution. In a parallel setting however, PGSEMO is faster than GC-AIS at the cost of much higher communication effort.

Does the novel algorithm perform well on a real multi-objective optimisation problem such as the multi-objective knapsack problem? In the same chapter, GC-AIS is compared with a popular multi-objective optimisation algorithm (NSGA-II) on several instances of the MOd-KP. Using 3 prominent metrics from multi-objective optimisation, the two algorithms performed differently for each metric and no clear winners was found over all three metrics. GC-AIS produced fronts closer to the Pareto front, but the diversity

and hypervolume of the fronts obtained by NSGA-II were better for majority of the instances. These findings show the potential of the proposed novel algorithm against a popular MOEA.

What can be learned about the strengths and weaknesses of the GC-AIS *based on these studies?* Based on the findings on the performance and behaviour of GC-AIS, the key strength of the algorithm is that it does not require any parameter tuning if the mutation rate remained fixed. Parameter tuning is extremely important to obtain good performance from evolutionary algorithms, the fact that GC-AIS does not require it, saves a lot of time in tuning the algorithm. The key weakness identified from the behaviour of GC-AIS is the population explosion in problems with multiple objectives. It was observed for the MOd-KP problem that as the number of knapsacks increased, the population size of GC-AIS increased rapidly. This can lead to wasted resources in the form of fitness evaluations as well as presents a problem in identifying important solutions for the decision maker.

Knowing certain weaknesses in GC-AIS*, how can improvements be made to overcome the shortcomings?* Based on the issue of population explosion identified in Chapter 4, the GC-AIS is improved by modification to the strategy of identifying dominated solutions. An important method from MOEA literature called ϵ-dominance is incorporated into GC-AIS to handle population explosion. This method, which was originally proposed to improve diversity and convergence of MOEAs, has been utilised for maintaining population size of the algorithm. With the incorporation of this metric, improvements are observed in ϵ-GC-AIS which are seen in the form of a controlled population size and high contribution to the reference fronts. With respect to the performance metrics ϵ-GC-AIS performs better than NSGA-II on all the instances when considering generational distance, while when hypervolume is considered NSGA-II performs better on most of the larger instances and ϵ-GC-AIS performs better on majority of the smaller instances. Results for diversity metric shows that ϵ-GC-AIS performs better than NSGA-II on majority of the instances. The performance of the algorithm is sensitive to ϵ and param-

eter tuning for ϵ is crucial for obtaining good results and desired number of solutions in the non-dominated front from the algorithm.

When using GC-AIS *for dynamic optimisation of* SCP*, does using solutions from memory perform better than starting from scratch ? When does using memory work and when does it fail? and why is this behaviour seen ?* These questions are addressed in Chapter 6, where a memory component is added to the GC-AIS and its behaviour analysed in a dynamic setting of the set cover problem. When considering alterations to the number of subsets in the collection, using memory is preferable when addition is performed, while when removal is performed one is better off starting from a scratch solution. In the case of editing, small edits can be handled by using memory while larger ones should be handled with a solution from scratch. On the other hand when considering alterations to the number of items in the universe set, when adding items, memory is useful for small cases while scratch solution for larger ones Similar result is obtained for removal of items and swapping of items.

7.3 Knowledge gained

The work done in this thesis makes contributions to both the fields of artificial immune systems as well as optimisation. The results obtained on the performance of the GC-AIS show that AIS in general and the germinal centre model in particular are suitable and promising avenues of research in the field of multi-objective optimisation. The contribution made in this thesis towards the field of AIS is a new algorithms which has interesting properties like a low number of parameters, inherent parallel nature, dynamic population resizing and an analysis on its behaviour on the set cover and knapsack problems. The design benefits of GC-AIS which has few parameters as modelled after the GC reaction are one of the advantages which helps reduce set-up time utilised for parameter tuning. The other key advantage being the performance of the algorithm for the problems tested in this thesis when compared with the known state-of-the-art algorithms for these prob-

lems from literature. The knowledge gained on the performance and behaviour of memory can provide valuable insight on the design of memory based techniques for dynamic optimisation. The behaviour of memory was investigated in a predicted dynamic scenario in order to understand how the memory solution will behave. This seemingly trivial scenario can provide deep insight into memory based techniques. A key questions which can be addressed by this knowledge is: when faced in a dynamically changing environment when should memory be used and when should a random initialisation be used? This can have implications on the design of memory saving mechanisms in the design of dynamic optimisation algorithms. Along with the design of memory storing mechanisms, coupling of memory techniques could be performed with predictive techniques in dynamic optimisation that predict the direction of change in the optima, in order to select the best solution of memory which will propel the algorithm in the right direction.

7.4 Summary of contributions

The main contributions of this thesis are:

1. **A novel** AIS **designed for multi-objective optimisation.**

   The novel artificial immune system is proposed in this thesis which has a small number of parameters. The general model is versatile and can be applied in sequential as well as parallel systems.

2. **An investigation of the performance of** GC-AIS **on** SCP **and MOd-KP**

   The empirical comparison of GC-AIS with a simple MOEA and a state of the art MOEA is performed and the potential usefulness of GC-AIS is observed. GC-AIS is shown to be better than PGSEMO on SCP and worse on 2 out of 3 performance metrics than NSGA-II, on MOd-KP. A potential issue of population size explosion when dealing with large problems with multiple objectives is found which was later addressed.

3. **An improved variant of** GC-AIS

   Population explosion as a weakness of GC-AIS is addressed by incorporating the ϵ operator. It is shown that, indeed including this operator can remedy the shortcomings of GC-AIS.

4. **Investigative analysis of ϵ-**GC-AIS **on** SCP **and MOd-KP**

   After introducing the ϵ operator to overcome the population size explosion problem of GC-AIS, an empirical investigation on the performance of ϵ-GC-AIS is performed on SCP and MOd-KP. The importance of setting a suitable value for the ϵ operator is demonstrated.

5. **Understanding the effects of memory in a dynamic scenario.**

   The usefulness of the concept of memory in different dynamic scenarios was examined. The different scenarios were constructed by introducing different changes in the problem instances. Investigation into the usefulness of memory provides insight as to why does incorporating memory behave in such as way.

7.5 Future directions

This thesis introduces a new algorithm in the field of artificial immune systems, thus there are several directions of potential future work that are identified. The abstract model of the GC-AIS presented can be extended to resemble the immune system more realistically by removing the limit of B-cells per germinal centre. Dynamics of GC growth, with respect to their size and number, as well as communication between them, is an area which is worth exploring as this simple model is capable of producing good results. Another feature which has not been explored in this thesis is the different mutations specific to immune systems. The effects of the incorporation of different mutation operators on the behaviour of the GC-AIS can possibly lead to improvements in the overall performance, as well as shed light on the interaction of the operator with current GC-AIS characteristics.

The problem of population explosion identified in GC-AIS is tackled with the inclusion of the ϵ-dominance operator which seems a perfect candidate to solve the issue. Though overall improvement in the performance of the algorithm are seen, it does however add a new parameter to the algorithm which needs to be tuned. A possible future extension of the work can be done to devise a clever dynamic ϵ resizing mechanism which sets the parameter based on the objectives and size of the problem. Currently, the ϵ parameter values are set by running the algorithm with several tentative values and selecting the best value to be used by the algorithm. Also, the values are chosen to be the same in every dimension which might not be the best option for every kind of problem. The dynamic resizing could be used to adjust the value of the parameter while the algorithm runs, where these adjustments is based on learning the changes in the non-dominated fronts obtained during the runs and setting the appropriate values for every dimension.

In the context of learning and dynamic settings, the current memory model is very simplistic in nature. This is chosen to better understand the effect of using memory in different scenarios, and based on the insights obtained more complex models of memory can be designed and tested. Memory models which contain more than a single solution can be studied to perhaps improve currently obtained results. Methods for populating such memory pools will have to be constructed and tested.

Finally, as GC-AIS is still in its infancy, a lot of work needs to be done in order to make it well known within the AIS community. This involves finding more applications suited for AIS to be solved by GC-AIS, and understanding how it works on them. Variations in the model which may be suited for different domains will have to be found and tested with GC-AIS on different problems.

7.6 Summary

The Chapter marks the conclusion of this thesis by listing the answers to all the research questions that were asked at the beginning of Chapter 1. Knowledge gained as a result of

this research is presented which makes contributions both to the field of AIS as well as optimisation, more specifically multi-objective and dynamic optimisation. The contributions made in this thesis are presented in a consolidated way and finally some directions of future research are presented which include ideas for further experiments and design changes to GC-AIS.

LIST OF REFERENCES

[1] Uwe Aickelin. An indirect genetic algorithm for set covering problems. *Journal of the Operational Research Society*, 53(10):1118–1126, 2002.

[2] Uwe Aickelin and Steve Cayzer. The danger theory and its application to artificial immune systems. *arXiv preprint arXiv:0801.3549*, 2008.

[3] Uwe Aickelin and Steve Cayzer. The danger theory and its application to artificial immune systems. *CoRR*, abs/0801.3549, 2008.

[4] Uwe Aickelin and Julie Greensmith. Sensing danger: Innate immunology for intrusion detection. *Information Security Technical Report*, 12(4):218–227, 2007.

[5] Bahriye Akay and Dervis Karaboga. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, 23(4):1001–1014, 2012.

[6] Thomas Bäck, DB Fogel, and Z Michalewicz. Handbook of evolutionary computation. *Release*, 97(1):B1, 1997.

[7] Egon Balas and Andrew Ho. Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study. In M.W. Padberg, editor, *Combinatorial Optimization*, volume 12 of *Mathematical Programming Studies*, pages 37–60. Springer, 1980.

[8] Cristina Bazgan, Hadrien Hugot, and Daniel Vanderpooten. An efficient implementation for the 0-1 multi-objective knapsack problem. In Camil Demetrescu, editor,

Experimental Algorithms, 6th International Workshop, WEA 2007, Rome, Italy, June 6-8, 2007, Proceedings, volume 4525 of *Lecture Notes in Computer Science*, pages 406–419. Springer, 2007.

[9] Thomas Bck, Martin Schtz, and Sami Khuri. A comparative study of a penalty function, a repair heuristic, and stochastic operators with the set-covering problem. In Jean-Marc Alliot, Evelyne Lutton, Edmund Ronald, Marc Schoenauer, and Dominique Snyers, editors, *Artificial Evolution*, volume 1063 of *Lecture Notes in Computer Science*, pages 320–332. Springer, 1996.

[10] J. E. Beasley. An algorithm for set covering problem. *European Journal of Operational Research*, 31(1):85–93, 1987.

[11] J. E. Beasley. OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, Nov 1990.

[12] J.E. Beasley. An algorithm for set covering problem. *European Journal of Operational Research*, 31(1):85 – 93, 1987.

[13] J.E Beasley and P.C Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392 – 404, 1996.

[14] John E Beasley. A lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*, 37(1):151–164, 1990.

[15] T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, Aug 2006.

[16] Tim Blackwell. Particle swarm optimization in dynamic environments. In Shengxiang Yang, Yew-Soon Ong, and Yaochu Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 29–49. Springer, 2007.

[17] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford university press, 1999.

[18] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 1875–1882. IEEE, 1999.

[19] J. Branke. Evolutionary approaches to dynamic optimization problems-updated survey. In *GECCO Workshop on evolutionary algorithms for dynamic optimization problems*, pages 27–30, 2001.

[20] Jürgen Branke. *Evolutionary optimization in dynamic environments*, volume 3. Springer, 2012.

[21] Luce Brotcorne, Gilbert Laporte, and Frederic Semet. Ambulance location and relocation models. *European journal of operational research*, 147(3):451–463, 2003.

[22] Jason Brownlee. Clonal selection algorithms. *Complex Intelligent Systems Laboratory, Swinburne University of Technology, Australia*, 2007.

[23] FM Burnet. A modification of jerne's theory of antibody production using the concept of clonal selection. *CA: a cancer journal for clinicians*, 26(2):119–121, 1976.

[24] FM Burnet. Clonal selection and after. *Theoretical Immunology*, 63:85, 1978.

[25] Sir Frank Macfarlane Burnet et al. *The clonal selection theory of acquired immunity*, volume 3. Vanderbilt University Press Nashville, 1959.

[26] Felipe Campelo, Frederico G. Guimarães, and Hajime Igarashi. Overview of artificial immune systems for multi-objective optimization. In *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*, EMO'07, pages 937–951. Springer-Verlag, 2007.

[27] Alberto Caprara, Paolo Toth, and Matteo Fischetti. Algorithms for the set covering problem. *Annals of Operation Research*, 98(1-4):353–371, 2000.

[28] J. W Chrissis, R. P Davis, and D. M Miller. The dynamic set covering problem. *Applied Mathematical Modelling*, 6(1):2–6, 1982.

[29] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[30] Helen G Cobb and John J Grefenstette. Genetic algorithms for tracking changing environments. Technical report, DTIC Document, 1993.

[31] Carlos A. Coello Coello and Nareli Cruz Cortés. Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6(2):163–190, 2005.

[32] Reuven Cohen and Guy Grebla. Multi-dimensional OFDMA scheduling in a wireless network with relay nodes. In *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, pages 2427–2435. IEEE, 2014.

[33] Broderick Crawford and Carlos Castro. Integrating lookahead and post processing procedures with ACO for solving set partitioning and covering problems. In Leszek Rutkowski, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, editors, *Artificial Intelligence and Soft Computing - ICAISC 2006, 8th International Conference, Zakopane, Poland, June 25-29, 2006, Proceedings*, volume 4029 of *Lecture Notes in Computer Science*, pages 1082–1090. Springer, 2006.

[34] Vincenzo Cutello, Doheon Lee, Giuseppe Nicosia, Mario Pavone, and Igor Prizzi. Aligning multiple protein sequences by hybrid clonal selection algorithm with insert-remove-gaps and blockshuffling operators. In *Artificial Immune Systems*, pages 321–334. Springer, 2006.

[35] Vincenzo Cutello, Giuseppe Morelli, Giuseppe Nicosia, and Mario Pavone. Immune algorithms with aging operators for the string folding problem and the protein folding problem. In *Evolutionary Computation in Combinatorial Optimization*, pages 80–90. Springer, 2005.

[36] Vincenzo Cutello, Giuseppe Narzisi, Giuseppe Nicosia, and Mario Pavone. Clonal selection algorithms: a comparative case study using effective mutation potentials. In *Artificial Immune Systems*, pages 13–28. Springer, 2005.

[37] Vincenzo Cutello, Giuseppe Narzisi, Giuseppe Nicosia, and Mario Pavone. Clonal selection algorithms: A comparative case study using effective mutation potentials. In Christian Jacob, Marcin L. Pilat, Peter J. Bentley, and Jonathan Timmis, editors, *Artificial Immune Systems: 4th International Conference, ICARIS 2005, Banff, Alberta, Canada, August 14-17, 2005, Proceedings*, volume 3627 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2005.

[38] Vincenzo Cutello and Giuseppe Nicosia. An immunological approach to combinatorial optimization problems. In *Advances in Artificial Intelligence-IBERAMIA 2002*, pages 361–370. Springer, 2002.

[39] Vincenzo Cutello and Giuseppe Nicosia. Multiple learning using immune algorithms. In *Proceedings of 4th International Conference on Recent Advances in Soft Computing, RASC*, pages 102–107, 2002.

[40] Vincenzo Cutello and Giuseppe Nicosia. A clonal selection algorithm for coloring, hitting set and satisfiability problems. In *Neural Nets*, pages 324–337. Springer, 2006.

[41] Vincenzo Cutello, Giuseppe Nicosia, and Emilio Pavia. A parallel immune algorithm for global optimization. In *Intelligent Information Processing and Web Mining*, pages 467–475. Springer, 2006.

[42] Dipankar Dasgupta and Fernando Nino. *Immunological computation: theory and applications*. CRC Press, 2008.

[43] Dipankar Dasgupta, Senhua Yu, and Fernando Nino. Recent advances in artificial immune systems: Models and applications. *Applied Soft Computing*, 11(2):1574 – 1587, 2011.

[44] Dipankar Dasgupta, Senhua Yu, and Fernando Niño. Recent advances in artificial immune systems: Models and applications. *Appl. Soft Comput.*, 11(2):1574–1587, 2011.

[45] Mark S. Daskin. *Covering Problems*, pages 92–153. John Wiley & Sons, Inc., 1995.

[46] L. Nunes de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[47] Leandro N De Castro and Fernando J Von Zuben. Learning and optimization using the clonal selection principle. *Evolutionary Computation, IEEE Transactions on*, 6(3):239–251, 2002.

[48] Leandro Nunes De Castro. *Fundamentals of natural computing: basic concepts, algorithms, and applications*. CRC Press, 2006.

[49] Leandro Nunes De Castro and Fernando J Von Zuben. The clonal selection algorithm with engineering applications. In *Proceedings of GECCO*, volume 2000, pages 36–39, 2000.

[50] Leandro Nunes de Castro and Fernando J Von Zuben. ainet: an artificial immune network for data analysis. *Data mining: a heuristic approach*, 12:231–259, 2001.

[51] Leandro Nunes de Castro and Fernando José Von Zuben. Artificial immune systems: Part i–basic theory and applications. Technical Report TR-DCA 0199, Universidade Estadual de Campinas, Dezembro de, Tech. Rep, 1999.

[52] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, Aug 2014.

[53] Kalyanmoy Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[54] Kalyanmoy Deb. Multi-objective optimization. In Edmund K. Burke and Graham Kendall, editors, *Search Methodologies*, pages 273–316. Springer, 2005.

[55] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan J. Merelo Guervós, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Paris, France, September 18-20, 2000, Proceedings*, volume 1917 of *Lecture Notes in Computer Science*, pages 849–858. Springer, 2000.

[56] M. W Deem and H. Y Lee. Sequence space localization in the immune system response to vaccination and disease. *Physical review letters*, 91(6):068101, 2003.

[57] Jeroen Eggermont, Tom Lenaerts, Sanna Poyhonen, and Alexandre Termier. Raising the dead: Extending evolutionary algorithms with a case-based memory. In Julian Miller, Marco Tomassini, PierLuca Lanzi, Conor Ryan, AndreaG.B. Tettamanzi, and WilliamB. Langdon, editors, *Genetic Programming*, volume 2038 of *Lecture Notes in Computer Science*, pages 280–290. Springer, 2001.

[58] Matthias Ehrgott and Xavier Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, 2000.

[59] Matthias Ehrgott and Xavier Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *Top*, 12(1):1–63, 2004.

[60] Thomas Erlebach, Hans Kellerer, and Ulrich Pferschy. Approximating multiobjective knapsack problems. *Management Science*, 48(12):1603–1612, 2002.

[61] ThomasA. Feo and MauricioG.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[62] Emile Fiesler and Russell Beale. *Handbook of neural computation*. Oxford University Press, 1996.

[63] Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 50(12-Supplement):1861–1871, 2004.

[64] Fabio Freschi, Carlos A Coello Coello, and Maurizio Repetto. Multiobjective optimization and artificial immune systems: a review. *Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies*, 4:1–21, 2009.

[65] Fabio Freschi and Maurizio Repetto. Multiobjective optimization by a modified artificial immune system algorithm. In *Artificial Immune Systems*, pages 248–261. Springer, 2005.

[66] Tobias Friedrich, Jun He, Nils Hebbinghaus, Frank Neumann, and Carsten Witt. Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633, 2010.

[67] Tobias Friedrich, Jun He, Nils Hebbinghaus, Frank Neumann, and Carsten Witt. Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633, 2010.

[68] Juan Carlos Galeano, Angélica Veloza-Suan, and Fabio A González. A comparative analysis of artificial immune network models. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 361–368. ACM, 2005.

[69] Simon M Garrett. Parameter-free, adaptive clonal selection. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 1052–1058. IEEE, 2004.

[70] Oliver Giel. Expected Runtimes of a Simple Multi-objective Evolutionary Algorithm. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1918–1925. IEEE, 2003.

[71] C. Goh and K. Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *Evolutionary Computation*, 13(1):103–127, 2009.

[72] Chi Keong Goh and Kay Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 13(1):103–127, 2009.

[73] David E. Goldberg and Robert E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In John J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms, Cambridge, MA, USA, July 1987*, pages 59–68. Lawrence Erlbaum Associates, 1987.

[74] Maoguo Gong, Licheng Jiao, Jie Yang, and Fang Liu. Lamarckian learning in clonal selection algorithm for numerical optimization. *International Journal on Artificial Intelligence Tools*, 19(01):19–37, 2010.

[75] Maoguo Gong, Licheng Jiao, and Lining Zhang. Baldwinian learning in clonal selection algorithm for optimization. *Information Sciences*, 180(8):1218–1236, 2010.

[76] Julie Greensmith and Uwe Aickelin. The deterministic dendritic cell algorithm. In *Artificial Immune Systems*, pages 291–302. Springer, 2008.

[77] Julie Greensmith, Uwe Aickelin, and Steve Cayzer. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In Christian Jacob, Marcin L. Pilat, Peter J. Bentley, and Jonathan Timmis, editors, *Artificial Immune*

Systems: 4th International Conference, ICARIS 2005, Banff, Alberta, Canada, August 14-17, 2005, Proceedings, volume 3627 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2005.

[78] Julie Greensmith, Uwe Aickelin, and Jamie Twycross. Articulation and clarification of the dendritic cell algorithm. In Hugues Bersini and Jorge Carneiro, editors, *Artificial Immune Systems, 5th International Conference, ICARIS 2006, Oeiras, Portugal, September 4-6, 2006, Proceedings*, volume 4163 of *Lecture Notes in Computer Science*, pages 404–417. Springer, 2006.

[79] Tal Grossman and Avishai Wool. Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*, 101(1):81 – 92, 1997.

[80] Feng Gu, Julie Greensmith, and Uwe Aickelin. Further exploration of the dendritic cell algorithm: Antigen multiplier and time windows. In Peter J. Bentley, Doheon Lee, and Sungwon Jung, editors, *Artificial Immune Systems, 7th International Conference, ICARIS 2008, Phuket, Thailand, August 10-13, 2008. Proceedings*, volume 5132 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2008.

[81] José Luis Guerrero, Luis Martí, Antonio Berlanga, Jesús García, and José M. Molina López. Introducing a robust and efficient stopping criterion for moeas. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010*, pages 1–8. IEEE, 2010.

[82] Emma Hart and Jon Timmis. Application areas of AIS: the past, the present and the future. *Applied Soft Computing*, 8(1):191–201, 2008.

[83] Iason Hatzakis and David Wallace. Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1201–1208. ACM, 2006.

[84] Michael Held, Philip Wolfe, and Harlan P. Crowder. Validation of subgradient optimization. *Math. Program.*, 6(1):62–88, 1974.

[85] H. Ishibuchi and S. Kaige. Effects of repair procedures on the performance of emo algorithms for multiobjective 0/1 knapsack problems. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 4, pages 2254–2261, Dec 2003.

[86] Hisao Ishibuchi and Shiori Kaige. Implementation of simple multiobjective memetic algorithms and its applications to knapsack problems. *Int. J. Hybrid Intell. Syst.*, 1(1):22–35, 2004.

[87] Hisao Ishibuchi, Shiori Kaige, et al. Comparison of multiobjective memetic algorithms on 0/1 knapsack problems. Technical report, 2003.

[88] Hisao Ishibuchi, Shiori Kaige, and Kaname Narukawa. Comparison between lamarckian and baldwinian repair on multiobjective 0/1 knapsack problems. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, volume 3410 of *Lecture Notes in Computer Science*, pages 370–385. Springer, 2005.

[89] Hisao Ishibuchi, Yuji Sakane, Noritaka Tsukamoto, and Yusuke Nojima. *Evolutionary Multi-Criterion Optimization: 5th International Conference, EMO 2009, Nantes, France, April 7-10, 2009. Proceedings*, chapter Adaptation of Scalarizing Functions in MOEA/D: An Adaptive Scalarizing Function-Based Multiobjective Evolutionary Algorithm, pages 438–452. Springer, 2009.

[90] Andrzej Jaszkiewicz. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *IEEE Trans. Evolutionary Computation*, 6(4):402–412, 2002.

[91] NK Jerne. Towards a network theory of the immune system. In *Annales dimmunologie*, volume 125, pages 373–389, 1974.

[92] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on evolutionary computation*, 9(3):303–317, 2005.

[93] Ayush Joshi, Jonathan E. Rowe, and Christine Zarges. An immune-inspired algorithm for the set cover problem. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipic, and Jim Smith, editors, *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, volume 8672 of *Lecture Notes in Computer Science*, pages 243–251. Springer, 2014.

[94] Ayush Joshi, Jonathan E. Rowe, and Christine Zarges. An immune-inspired algorithm for the set cover problem. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipic, and Jim Smith, editors, *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, volume 8672 of *Lecture Notes in Computer Science*, pages 243–251. Springer, 2014.

[95] Ayush Joshi, Jonathan E. Rowe, and Christine Zarges. Improving the performance of the germinal center artificial immune system using \epsilon -dominance: A multi-objective knapsack problem case study. In Gabriela Ochoa and Francisco Chicano, editors, *Evolutionary Computation in Combinatorial Optimization - 15th European Conference, EvoCOP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, volume 9026 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2015.

[96] Ayush Joshi, Jonathan. E. Rowe, and Christine Zarges. On the effects of incorporating memory in GC-AIS for the Set Cover Problem. In *Proceedings of the 11th Metaheuristics International Conference*, 2015.

[97] J. S. Nizami K. S. Al-Sultan, M. F. Hussain. A genetic algorithm for the set covering problem. *The Journal of the Operational Research Society*, 47(5):702–709, 1996.

[98] Ahmed Kafafy, Ahmed Bounekkar, and Stéphane Bonnevay. A hybrid evolutionary metaheuristics (HEMH) applied on 0/1 multiobjective knapsack problems. In Natalio Krasnogor and Pier Luca Lanzi, editors, *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, July 12-16, 2011*, pages 497–504. ACM, 2011.

[99] Ahmed Kafafy, Ahmed Bounekkar, and Stéphane Bonnevay. HEMH2: an improved hybrid evolutionary metaheuristics for 0/1 multiobjective knapsack problems. In Lam Thu Bui, Yew-Soon Ong, Nguyen Xuan Hoai, Hisao Ishibuchi, and Ponnuthurai Nagaratnam Suganthan, editors, *Simulated Evolution and Learning - 9th International Conference, SEAL 2012, Hanoi, Vietnam, December 16-19, 2012. Proceedings*, volume 7673 of *Lecture Notes in Computer Science*, pages 104–116. Springer, 2012.

[100] Ahmed Kafafy, Ahmed Bounekkar, and Stéphane Bonnevay. Hybrid metaheuristics based on MOEA/D for 0/1 multiobjective knapsack problems: A comparative study. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2012, Brisbane, Australia, June 10-15, 2012*, pages 1–8. IEEE, 2012.

[101] J. Kelsey and J. Timmis. Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation. In E. Cant-Paz et al, editor, *Genetic and Evolutionary Computation Conference - GECCO 2003*, volume 2723 of *Lecture Notes in Computer Science*, Chicago. USA., July 2003. Springer-Verlag.

[102] Thomas B Kepler and Alan S Perelson. Cyclic re-entry of germinal center b cells and the efficiency of affinity maturation. *Immunology Today*, 14(8):412 – 415, 1993.

[103] Nitesh Khilwani, Anoop Prakash, Ravi Shankar, and MK Tiwari. Fast clonal algorithm. *Engineering Applications of Artificial Intelligence*, 21(1):106–128, 2008.

[104] J. D. Knowles and D. W. Corne. M-PAES: A Memetic Algorithm for Multiobjective Optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation*

(CEC00), pages 325–332, Piscataway, NJ, 2000. IEEE Press. Nominated by CEC Program Committe to be extended and submitted to Knowledge and Information Systems (KAIS) Journal.

[105] Hanan Lamlum, Mohammad Ilyas, Andrew Rowan, Susan Clark, Victoria Johnson, Jennie Bell, Ian Frayling, Jason Efstathiou, Kevin Pack, Stewart Payne, et al. The type of somatic mutation at apc in familial adenomatous polyposis is determined by the site of the germline mutation: a new facet to knudson's' two-hit'hypothesis. *Nature medicine*, 5(9):1071–1075, 1999.

[106] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

[107] Marco Laumanns, Eckart Zitzler, and Lothar Thiele. On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *Evolutionary Multi-Criterion Optimization, First International Conference, EMO 2001, Zurich, Switzerland, March 7-9, 2001, Proceedings*, volume 1993 of *Lecture Notes in Computer Science*, pages 181–196. Springer, 2001.

[108] Lucas Lessing, Irina Dumitrescu, and Thomas Stützle. A comparison between ACO algorithms for the set covering problem. In Marco Dorigo, Mauro Birattari, Christian Blum, Luca Maria Gambardella, Francesco Mondada, and Thomas Stützle, editors, *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5 - 8, 2004, Proceedings*, volume 3172 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2004.

[109] Jonathan Lewis, Emma Hart, and Graeme Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In A. E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem*

Solving from Nature - PPSN V, 5th International Conference, Amsterdam, The Netherlands, September 27-30, 1998, Proceedings, volume 1498 of *Lecture Notes in Computer Science*, pages 139–148. Springer, 1998.

[110] Andrea Mambrini, Dirk Sudholt, and Xin Yao. Homogeneous and heterogeneous island models for the set cover problem. In Carlos A. Coello Coello, Vincenzo Cutello, Kalyanmoy Deb, Stephanie Forrest, Giuseppe Nicosia, and Mario Pavone, editors, *Parallel Problem Solving from Nature - PPSN XII - 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I*, volume 7491 of *Lecture Notes in Computer Science*, pages 11–20. Springer, 2012.

[111] Polly Matzinger. Tolerance, danger, and the extended family. *Annual review of immunology*, 12(1):991–1045, 1994.

[112] Ronald W Morrison. *Designing evolutionary algorithms for dynamic environments*. Springer Science & Business Media, 2013.

[113] Kenneth M. Murphy, Paul Travers, and Mark Walport. *Janeway's Immunobiology (Immunobiology: The Immune System (Janeway))*. Garland Science, 2007.

[114] Nysret Musliu. Local search algorithm for unicost set covering problem. In Moonis Ali and Richard Dapoigny, editors, *Advances in Applied Artificial Intelligence*, volume 4031 of *Lecture Notes in Computer Science*, pages 302–311. Springer, 2006.

[115] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Natural Computing Series. Springer, 2010.

[116] Trung Thanh Nguyen. *Continuous dynamic optimisation using evolutionary algorithms*. PhD thesis, University of Birmingham, 2011.

[117] Trung Thanh Nguyen, Shengxiang Yang, Juergen Branke, and Xin Yao. *Evolutionary Dynamic Optimization: Methodologies*. Springer, 2013.

[118] Trung Thanh Nguyen, Shengxiang Yang, and Jürgen Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012.

[119] TrungThanh Nguyen, Shengxiang Yang, Juergen Branke, and Xin Yao. Evolutionary dynamic optimization: Methodologies. In Shengxiang Yang and Xin Yao, editors, *Evolutionary Computation for Dynamic Optimization Problems*, volume 490 of *Studies in Computational Intelligence*, pages 39–64. Springer, 2013.

[120] Koji Nonobe and Toshihide Ibaraki. A tabu search approach to the constraint satisfaction problem as a general problem solver. *European Journal of Operational Research*, 106(23):599 – 623, 1998.

[121] David Orvosh and Lawrence Davis. Using a genetic algorithm to optimize problems with feasability constraints. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 548–553, 1994.

[122] Gheorghe Paun. *Membrane computing: an introduction*. Springer Science & Business Media, 2012.

[123] Camilo Eduardo Prieto, Fernando Niño, and Gerardo Quintana. A goalkeeper strategy in robot soccer based on danger theory. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008, June 1-6, 2008, Hong Kong, China*, pages 3443–3447. IEEE, 2008.

[124] Malek Rahoual, Riad Hadji, and Vincent Bachelet. Parallel ant system for the set covering problem. In Marco Dorigo, Gianni Di Caro, and Michael Sampels, editors, *Ant Algorithms, Third International Workshop, ANTS 2002, Brussels, Belgium, September 12-14, 2002, Proceedings*, volume 2463 of *Lecture Notes in Computer Science*, pages 262–267. Springer, 2002.

[125] Zhi-Gang Ren, Zu-ren Feng, Liang-Jun Ke, and Zhao-Jun Zhang. New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, 58(4):774–784, 2010.

[126] Zhigang Ren, Zuren Feng, Liangjun Ke, and Hong Chang. A fast and efficient ant colony optimization approach for the set covering problem. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008, June 1-6, 2008, Hong Kong, China*, pages 1839–1844. IEEE, 2008.

[127] M. A. ROSENMAN and J. S. GERO. Reducing the pareto optimal set in multicriteria optimization(with applications to pareto optimal dynamic programming). *Engineering Optimization*, 8(3):189–206, 1985.

[128] Rina Rosin-Arbesfeld, Fiona Townsley, and Mariann Bienz. The apc tumour suppressor has a nuclear export function. *Nature*, 406(6799):1009–1012, 2000.

[129] Claudio Rossi, Mohamed Abderrahim, and Julio César Díaz. Tracking moving optima using kalman-based predictions. *Evolutionary Computation*, 16(1):1–30, 2008.

[130] Shana Schlottfeldt, Maria Emilia M. T. Walter, Jon Timmis, André C. P. L. F. Carvalho, Mariana P. C. Telles, and José Alexandre Felizola Diniz-Filho. Using multi-objective artificial immune systems to find core collections based on molecular markers. In Sara Silva and Anna Isabel Esparcia-Alcázar, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*, pages 1271–1278. ACM, 2015.

[131] Lee A. Segel. Diffuse feedback from diffuse information in complex systems. *Complexity*, 5(6):39–46, 2000.

[132] Ruchit Shah and Patrick Reed. Comparative analysis of multiobjective evolutionary algorithms for random and correlated instances of multiobjective d-dimensional knapsack problems. *European Journal of Operational Research*, 211(3):466–479, 2011.

[133] Anabela Simões and Ernesto Costa. Improving prediction in evolutionary algorithms for dynamic environments. In Franz Rothlauf, editor, *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009*, pages 875–882. ACM, 2009.

[134] S. Singh, A. Kodali, K. Choi, K. R Pattipati, S. M Namburu, S. C. Sean, D. V Prokhorov, and L. Qiao. Dynamic multiple fault diagnosis: Mathematical formulations and solution techniques. *Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(1):160–176, 2009.

[135] Peter Slavík. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25(2):237–254, 1997.

[136] Lawrence V. Snyder. Facility location under uncertainty: a review. *IIE Transactions*, 38(7):547–564, 2006.

[137] Mauricio Solar, Víctor Parada, and Rodrigo Urrutia. A parallel genetic algorithm to solve the set-covering problem. *Computers & Operations Research*, 29(9):1221 – 1235, 2002.

[138] John Stewart, Francisco J Varela, and Antonio Coutinho. The relationship between connectivity and tolerance as revealed by computer simulation of the immune network: some lessons for an understanding of autoimmunity. *Journal of autoimmunity*, 2:15–23, 1989.

[139] Masruba Tasnim, Shahriar Rouf, and M. Sohel Rahman. A clonalg-based approach for the set covering problem. *JCP*, 9(8):1787–1795, 2014.

[140] Gianni Tedesco, P Bull, A Knowles, G Tedesco, and A Hone. Diophantine benchmarks for the b-cell algorithm.

[141] Jon Timmis, Camilla Edmonds, and Johnny Kelsey. Assessing the performance of two immune inspired algorithms and a hybrid genetic algorithm for function opti-

misation. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 1044–1051. IEEE, 2004.

[142] R. Tinós and S. Yang. A self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genetic Programming and Evolvable Machines*, 8(3):255–286, 2007.

[143] K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 1843–1850. IEEE, 1999.

[144] Berna Haktanirlar Ulutas and Sadan Kulturel-Konak. A review of clonal selection algorithm and its applications. *Artificial Intelligence Review*, 36(2):117–138, 2011.

[145] Rasmus K. Ursem. Multinational gas: Multimodal optimization techniques in dynamic environments. In L. Darrell Whitley, David E. Goldberg, Erick Cantú-Paz, Lee Spector, Ian C. Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Las Vegas, Nevada, USA, July 8-12, 2000*, pages 19–26. Morgan Kaufmann, 2000.

[146] David A Van Veldhuizen. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, DTIC Document, 1999.

[147] F. Vavak, K. Jukes, and T. C. Fogarty. Learning the local search range for genetic optimisation in nonstationary environments. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 355–360, Apr 1997.

[148] Dalessandro Soares Vianna and José Elias Claudio Arroyo. A GRASP algorithm for the multi-objective knapsack problem. In *XXIV International Conference of the Chilean Computer Science Society (SCCC 2004), 11-12 November 2004, Arica, Chile*, pages 69–75. IEEE Computer Society, 2004.

[149] Joanne H Walker and Simon M Garrett. Dynamic function optimisation: Comparing the performance of clonal selection and evolution strategies. In *Artificial Immune Systems*, pages 273–284. Springer, 2003.

[150] Rong-Long Wang and Kozo Okazaki. An improved genetic algorithm with conditional genetic operators and its application to set-covering problem. *Soft Computing*, 11(7):687–694, 2007.

[151] X Wang. Clonal selection algorithm in power filter optimization. In *Soft Computing in Industrial Applications, 2005. SMCia/05. Proceedings of the 2005 IEEE Mid-Summer Workshop on*, pages 122–127. IEEE, 2005.

[152] Andrew Watkins, XINTONG Bi, and AMIT Phadke. Parallelizing an immune-inspired algorithm for efficient pattern recognition. *Intelligent Engineering Systems through Artificial Neural Networks: Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Complex Systems and Artificial Life*, 13:225–230, 2003.

[153] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[154] Yonas G. Woldesenbet and Gary G. Yen. Dynamic evolutionary algorithm with variable relocation. *IEEE Transactions on Evolutionary Computation*, 13(3):500–513, 2009.

[155] Mutsunori Yagiura, Masahiro Kishida, and Toshihide Ibaraki. A 3-flip neighborhood local search for the set covering problem. *European Journal of Operational Research*, 172(2):472–499, 2006.

[156] S. Yang, H. Cheng, and F. Wang. Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks. *Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(1):52–63, 2010.

[157] Shengxiang Yang. A comparative study of immune system based genetic algorithms in dynamic environments. In Mike Cattolico, editor, *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, pages 1377–1384. ACM, 2006.

[158] Shengxiang Yang. On the design of diploid genetic algorithms for problem optimization in dynamic environments. In *IEEE International Conference on Evolutionary Computation, CEC 2006, part of WCCI 2006*, pages 1362–1369. IEEE, 2006.

[159] Shengxiang Yang. Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. *Evolutionary Computation*, 16(3):385–416, 2008.

[160] Shengxiang Yang, TrungThanh Nguyen, and Changhe Li. Evolutionary dynamic optimization: Test and evaluation environments. In Shengxiang Yang and Xin Yao, editors, *Evolutionary Computation for Dynamic Optimization Problems*, volume 490 of *Studies in Computational Intelligence*, pages 3–37. Springer, 2013.

[161] Shengxiang Yang and Xin Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, 9(11):815–834, 2005.

[162] Shengxiang Yang and Xin Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Trans. Evolutionary Computation*, 12(5):542–561, 2008.

[163] Yang Zhang, Michael Meyer-Hermann, Laura A. George, Marc Thilo Figge, Mahmood Khan, Margaret Goodall, Stephen P. Young, Adam Reynolds, Francesco Falciani, Ari Waisman, Clare A. Notley, Michael R. Ehrenstein, Marie Kosco-Vilbois, and Kai-Michael Toellner. Germinal center b cells govern their own fate via antibody feedback. *The Journal of Experimental Medicine*, 210(3):457–464, 2013.

[164] Z. Zhang and S. Qian. Artificial immune system in dynamic environments solv-

ing time-varying non-linear constrained multi-objective problems. *Soft Computing*, 15(7):1333–1349, 2011.

[165] Aimin Zhou, Yaochu Jin, Qingfu Zhang, Bernhard Sendhoff, and Edward Tsang. Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 832–846. Springer Berlin Heidelberg, 2007.

[166] Aimin Zhou, Yaochu Jin, Qingfu Zhang, Bernhard Sendhoff, and Edward P. K. Tsang. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In *IEEE International Conference on Evolutionary Computation, CEC 2006, part of WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pages 892–899. IEEE, 2006.

[167] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland, 1999.

[168] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. TIK Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2001.

[169] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2001. International Center for Numerical Methods in Engineering (Cmine).

[170] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Compara-

tive Case Study and the Strength Pareto Approach. *IEEE Trans. on Evolutionary Computation*, 3(4):257–271, 1999.

[171] Eckart Zitzler. Evolutionary multiobjective optimization. In Grzegorz Rozenberg, Thomas Bck, and JoostN. Kok, editors, *Handbook of Natural Computing*, pages 871–904. Springer, 2012.

[172] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[173] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.