OPTIMISING THE FREQUENCY ASSIGNMENT PROBLEM UTILIZING PARTICLE SWARM OPTIMISATION

by

WILLIAM BEZUIDENHOUT

DISSERTATION

submitted in the fulfilment
of the requirements for the degree

MAGISTER SCIENTIAE

in

INFORMATION TECHNOLOGY

in the

FACULTY OF SCIENCE

at the

UNIVERSITY OF JOHANNESBURG

SUPERVISOR: DR G.B. O'REILLY

MAY 2012

Contents

List of Figures

List of Algorithms

List of Tables

Part I

Background

Chapter 1

Introduction

1.1 Introduction

In the technology age, life is almost unfathomable without the mobile phone. It is hard to believe how the business world managed to function in the pre-mobile phone era[1].

The invention of the mobile phone fulfilled the need to always be connected and within reach of the modern world. This need can actually be attributed to feeling part of something and this something is deemed as being part of a network. This is similar to how mobile phones are able to provide connectivity at almost any location within a country.

Behind the connectivity of mobile phones lies an intricate layer of communication technology which has evolved since the invention of the normal landline phone. This technology is referred to at the top level as wireless communication, which includes inventions such as the radio. To enable communication at the level needed by mobile phones, wireless communication had to evolve and go a step further than normal radio communication. Hence the concept of cellular mobile networks was developed.

Without cellular networks mobile phones have no means of providing communication. The purpose of the cellular network is to provide the necessary functions to connect and facilitate communication between two entities (mobile phones) over a wireless network.

Cellular networks achieve this level of communication through expensive equipment and "smart" algorithms. By "smart" algorithms, what is actually

[1] Pre 1980s where the mobile phone did not yet exist

meant is algorithms that utilise artificial intelligence concepts to perform a certain function in an attempt to either automate a task within the network or improve a certain part of the network. Therefore, the correct term for these types of algorithms is actual artificial intelligence (AI) algorithms.

AI algorithms have a wide spread of functions that they can perform ranging from making informed intelligent decisions to optimising the operation of certain processes. In particular computers are indeed very apt at optimising certain procedures. This is because a computer is able to test and evaluate a huge number of different alterations and combinations of a certain procedure in a short amount of time, thereby allowing it to find the best combination out of those tested.

In this dissertation an algorithm will be presented which concentrates on cellular phone networks. This particular algorithm falls into the latter part of the AI algorithms discussed, namely an AI optimisation algorithm. Thus the algorithm presented in this dissertation operates on a cellular phone network to optimise a certain part of the network.

In this section a introduction was presented to the dissertation. The main themes of the dissertation were outlined namely, cellular phone networks and optimisation algorithms. In the next section research plan and chapter breakdown will be presented which will be followed by this dissertation.

1.2 Research Questions and Objective

As previously discussed, an artificial intelligent algorithm will be presented in this dissertation that generates plans for use by cellular networks. The purpose of creating such an algorithm is to determine the applicability of modern swarm-based algorithms with regard to finding better solutions to real-world problems that exist in domains such as cellular communication.

The research approach followed in this dissertation was first and foremost to understand the cellular network domain. More importantly, the focus is on how and what frequencies are used to facilitate communication as well as what affects the quality of the communicational link between two entities in a cellular network.

The cellular domain is the problem domain, but to develop a solution in the problem domain a study needs to be conducted in the artificial intelligence domain, namely a study of optimisation algorithms. In order for

a new optimisation algorithm to be developed, other algorithms that have achieved success in the respective optimisation problem domains in which they have been applied need to be investigated.

Therefore in an attempt to develop and apply a modern viable optimisation-based algorithm, a series of research questions has been identified which need to be answered. The research questions are as follows:

- **Question 1** — What is cellular technology, how it got developed and what improvements have been made since its initial development?
- **Question 2** — What is the architecture behind a modern cellular network, how do the various hardware entities within a network communicate with each other and how is a communicational link established between two users of the network?
- **Question 3** — What exactly is the frequency problem and how does it affect modern wireless communication?
- **Question 4** — What variants of the frequency problem exist and which are most applicable to cellular networks?
- **Question 5** — What are the most popular optimisation algorithms and what characteristics make them unique?
- **Question 6** — With algorithms that achieved success in their respective optimisation problems, what particular technique is used by the algorithms that allowed them to achieve better performance?

Throughout the course of this dissertation the aim is to answer each of these identified research questions. In the next section a chapter breakdown of this dissertation is presented.

The aim of this dissertation is first and foremost to answer the identified research questions. Answering these questions will aid in reaching the research objective of this dissertation. The research objective is therefore to develop and determine the viability of an algorithm based on modern swarm optimisation techniques that will operate on the frequency assignment problem in an attempt to provide better frequency plans for use in cellular networks.

1.3 Chapter Breakdown

1.3.1 Part I - Background

The first part of this dissertation is concerned with the domain and problem the algorithm presented in this dissertation addresses, and finally to also understand the intricate details of how the algorithm operates.

Below is an outline of the chapters in the first part of this dissertation.

Chapter 1

This chapter provides an introduction to the dissertation as well as a broad overview of the topics in the dissertation will discuss.

Chapter 2

This chapter is concerned with providing information on how a modern cellular network functions. Within this chapter a brief history is presented on how cellular network technology was developed. The chapter also provides an overview of the architecture of a cellular network,and each part of the network's intended purpose and function to facilitate wireless communication is discussed.

Chapter 3

This chapter presents the problem that the dissertation addresses, namely the frequency assignment problem. The chapter provides a discussion on why the problem exists, the causes of the problem and what it means for a problem to be NP-Complete. Furthermore the variants of the problem and how they differ depending on the wireless domain that is being considered are alos discussed. Finally the chapter also provides a formal definition of the problem which is later utilised by the algorithm developed in this research.

Chapter 4

This chapter marks the beginning of a discussion on various optimisation algorithms in this dissertation. The algorithms presented in this chapter

were chosen due to their widespread usage as well as success on NP-Complete problems. Each algorithm is discussed in depth providing an outline of the core features that make the algorithm unique as well as each core feature in detail. For each algorithm the chapter also presents an analysis on related work of the particular algorithm being applied to the frequency assignment problem.

Chapter 5

This chapter is concerned with providing algorithms that are new in the research domain of optimisation algorithms. The algorithms presented in chapter 4 are fairly old and have been applied to a wide variety of problems. Algorithms in this chapter are relatively new in the optimisation domain and have not been applied to the same number of problems as the algorithms in chapter 4. In this chapter swarm algorithms are presented and the algorithms have the particular characteristic that they are based generally on processes observed in nature. Each algorithm is discussed in depth with its core characteristics outlined. Furthermore for each algorithm an analysis is given if the algorithm were to be applied to the frequency assignment problem. Finally it is formally stated what algorithm this dissertation is applied to the frequency assignment problem in this research.

1.3.2 Part II - Implementation

Chapter 6

In this chapter various optimisation problems are presented. The problems were chosen due to their usage in the literature to test other optimisation algorithms. The problems are used to test two variants of the particle swarm optimisation algorithm. The tests were done not only to determine the general viability of the algorithm, but also to understand the algorithm better. The chapter also presents the results obtained by applying the algorithms to the test problems, together with a discussion on the performance of the algorithms.

Chapter 7

This chapter provides a discussion of the algorithm developed to be applied to the frequency assignment problem. Within this chapter an outline is given of the process in developing a specialised particle swarm algorithm for the frequency algorithm. Each specialised technique developed is discussed in depth, along with an explanation of why the technique is needed as well as why it is used by the algorithm.

Chapter 8

This chapter is concerned with providing the results after applying the algorithm to a specialised set of benchmark problems for frequency assignment algorithms. The particular selected benchmark problems were discussed in chapter 2.

Chapter 9

This chapter concludes this dissertation. In this chapter it is determined whether the research goal was reached as well as whether any future work can be done to improve the presented algorithm.

Chapter 2

Cellular Technology

2.1 Introduction

In the current information age almost every device has a wireless technology of some sort that it uses to provide a specific service. Examples are radios for audio entertainment; television remotes to change channels; cellular phones for communication; wireless access points to create wireless LANs [3]. Wireless technology is now part of any modern human everyday life.

The popularity and rapid adoption of wireless technology has made it difficult to plan, manage and operate wireless networks [3, 32, 53, 81, 88]. Wireless technology facilitates communication between two entities by transmitting data or voice via a radio frequency [3, 32, 53, 81, 88].

As the popularity and use of services that use wireless technology increase, the need for these services to use different radio frequencies to communicate becomes greater [3, 32, 53, 81, 88]. This need arises due to an effect called *interference*, which occurs when two or more connections between entities use the same radio frequency to facilitate communication [3,32,53,81,88]. This effect is discussed in more detail in chapter 3.

Therefore the use of frequencies by services must be carefully considered to avoid interference when entities communicate with each other, which is a problem since the number of entities that communicate far outstrips the amount of frequencies available for communication [3, 32, 53, 81, 88]. Thus assigning frequencies to entities for communication is a difficult problem and is referred to as the frequency assignment problem (FAP) [3, 32, 53, 81, 88].

Initially manual techniques were used to assign frequencies in an attempt to solve the FAP [3, 32, 53, 81, 88]. As a result, the assignment of frequencies was either too complex or just too daunting because of the shear number of entities that need to be assigned frequencies [3, 32, 53, 81, 88]. Also, because of the rapid adoption of wireless technology the assignment of frequencies needed to be dynamic and hence automated [3, 32, 53, 81, 88].

When a task needs to be automated an algorithm needs to be developed to tell the machine how it must solve the problem it is destined to work on [3, 32, 53, 81, 88]. Early algorithms developed to solve the FAP utilised brute force[1] techniques to assign frequencies [3, 32, 53, 81, 88].

Since the FAP has been proven to be an NP-Complete problem, using an algorithm in this research that tried to brute force a solution was futile as no solution could be found in a reasonable time frame. Hence, algorithms based on heuristic techniques are utilised in an attempt to either solve the FAP or come close to a solution. Heuristic algorithms will be discussed in chapter 4.

In this chapter an overview of GSM networks as well as a brief history of GSM will be given. A discussion on the topology of GSM will then follow. This chapter concludes with the problems that are present in a GSM network, namely the FAP.

2.2 GSM Networks

The General System for Mobile Communications (GSM) is a system for multiservice cellular communication that is capable of providing voice as well as data services [3, 32, 53, 81, 88]. Most cellular networks in operation are GSM based [3, 32, 53, 81, 88]. The primary service that GSM caters for is voice communication, but other data services such as Short Message Service (SMS), Multimedia Message Service (MMS) and Internet connectivity services, e.g. *general packet radio system* (GPRS), *enhanced data rate for global evolution* (EDGE) and *high speed circuit switched data* (HSCSD), are becoming more important [32, 53].

GSM is one of the most widely used radio communication technologies, which is why one needs to look at the history behind it in order to understand

[1] Brute force refers to a method which sequentially tries every possible combination in the hope of finding a solution

the domain of radio communication better [53]. A brief history of the GSM network specification will now be presented in the next section.

2.2.1 A Brief History of GSM Networks

In early 1981 a group known as the Groupe Speciale Mobile (GSM) was established to develop a Europe-wide radio communication system using the reserved 900 MHz band[2] [3, 81].

At the start of the GSM specification in the early 1980s it was initially thought that the system would be analogue based, but this soon changed with the *integrated service digital network* (ISDN) specification nearing completion. As such the GSM specification started following many of the same design principles and access protocols that ISDN exhibited [3, 53, 81].

After the completion of the ISDN specification, the advantages of switching to digital instead of analogue for communication became clear. One of the primary advantages of ISDN is that it is capable of transmitting data at higher speeds. GSM would therefore be based on digital transmission and speech would be represented by a digital stream of 16 kbits/s [3, 53, 81].

The primary benefit that one can deduce from the use of ISDN over slower analogue connections is that because data can be transmitted faster, more data can be sent for the same amount of time it would have taken on an analogue connection [3, 53, 81]. Hence, since more data can be sent it has the net effect of increasing the quality and efficiency of the connection between two entities. The network therefore does not need to remove as much information from a packet to be sent to fit within the data transmission constraints [3, 53, 81].

Before the switch to digital transmission was finalised the GSM first wanted to evaluate the spectral efficiency of analogue and digital-based transmission [3,53,81]. Spectral efficiency plays an important part in wireless communication since the radio spectrum is a limited resource and whichever transmission technology is used, the utilisation of the spectrum should be maximised [3, 53, 81].

Maximum utilisation is an important problem that will be discussed in detail in later sections of this chapter. After an evaluation of spectral

[2] In 1990 the United Kingdom requested that 1800 MHz band be added to the scope of the GSM standard group. This variant of the GSM specification became known as the *Digital Cellular System 1800* (DCS1800) [3, 81].

efficiency it was decided that the GSM system would be digitally based using *time division multiple access* (TDMA) [32, 77, 81].

By the early 1990s GSM became an evolving standard and the first GSM-based network was demonstrated in 1991[3] [3,32,53,81]. The following year a number of GSM networks were operating in Europe due to mobile terminals and equipment capable of operating on the networks becoming more widely available to the general public [3, 32, 53, 81]. In the same year an operator in Australia became the first non-European operator to implement a GSM-based network [32].

The collective subscriber base of GSM networks surpassed the million subscriber mark in 1993. Due to this phenomenal growth in GSM network use, numerous extensions were made to the GSM specification. Some of the extensions that were made are the following [3, 32, 53, 81]:

- Half rate speech telephony
- Improved SMS
- Line identification
- Call waiting
- Call holding

The specification with these extensions is known as GSM Phase 2. As the world shifted towards more digital and data-intensive services it became difficult to deliver these services over GSM networks. This difficulty was due to the restriction that data could only be transmitted at 9.6 kbps [3, 81].

The new specification defined new technologies such as GPRS, EDGE and HSCSD, which were designed with the primary goal of making more bandwidth available for data transmission [3, 53]. Data transmission was improved by these technologies by enabling more than one GSM slot to be used for a terminal or service at a time [3, 53].

If more than one GSM slot is to be used by a terminal or service, transceivers are required to have a higher signal-to-noise (SIR) ratio [53, 77]. This requirement affects radio interfaces as there is a higher likelihood that

[3]Near the end of 1991 the GSM group was renamed *Speciale Mobile Group* (SMG) to eliminate confusion between the standard and the group

interference might occur; hence it makes it more difficult to generate a low interference frequency plan [32, 77].

The actual SIR at a receiver is dependent on a number of factors that include [3, 53]:

- Frequency used at the transceiver
- Strength of the signal
- Weather conditions
- Shape of the surrounding environment
- Direction of the transmission

Even taking these factors into account, the calculation of the SIR at a transceiver is not trivial. This calculation of the SIR as well as its impact on mobile radio frequencies will be discussed in section 3.4.

As the GSM standard matured as a cellular technology, industry experts began specification of the next generation of cellular networks, which would in time replace the GSM cellular system.

The *Universal Mobile Telecommunications System* (UMTS) can be considered the third generation (3G) of cellular networks. UMTS was designed from the beginning to operate in parallel with the legacy GSM system. The first standard of UMTS was issued in the beginning of 2000 and subsequently most modern networks are based on it or are migrating their networks to it.

UMTS is a major improvement over the GSM in two areas, namely data tansmission bandwidth and frequency planning due to UMTS utilising *DS-CDMA* (direct sequence code division multiple access) and *WCDMA* (wide band code division multiple access). The higher data transmission speed (2 Mb/s) can be attributed to UMTS using the DS-CDMA scheme. The scheme also allows more users to be served than previous generations of networks [32, 116].

A direct consequence of UMTS utilising DS-CDMA and WCDMA, which sends data over a wide band of 5 MHz, is that no frequency planning problem comparable to GSM has to be solved [32, 116].

Code division multiple access (CDMA) is a technology primarily used in broadband systems. Users do not gain access to only a small portion

of the bandwidth, but rather use the entire band for the duration of a connection. Users also do not gain exclusive access to the whole band, but instead share the usage of the bandwidth with other users simultaneously, hence the name *multiple access*. Users using a band simultaneously are separated using orthogonal codes [53].

With CDMA a user's signal is not transmitted as its original signal. Instead the signal is spectrally spread over a multiple of its original bandwidth using a spreading factor [53]. The spreading factor fluctuates between values of 10 and 1 000 [53]. Using these spreading factors less interference and fewer disturbances are encountered because the broadband signal is generated from a narrowband signal [53]. UMTS may be a major improvement, but its adoption does not spread very far from busy city centres that contain a large concentration of clients in a small geographical area. The reason for this is that, as mentioned previously, UMTS caters for larger data usage and therefore more clients can be serviced simultaneously [53].

Most network operators do not implement entirely new backbone architecture for UMTS to operate on, but instead utilise the same backbone used for GSM and GPRS. This not only extends the lifetime of previous infrastructure investment by the operator, but also builds upon the redundancy provided by the GSM network [53]. Thus even with new technological improvements such as UMTS, GSM as a wireless technology is still used for communication and is therefore still relevant today.

In this section a brief overview of the history of the GSM network specification was presented. In the next section an explanation of the topology of GSM network will be given. This will broaden the understanding of GSM networks.

2.3 Topology of a GSM Network

A GSM network consists of a variety of different subsystems to realise the goal of establishing a radio communication link between two parties. The hierarchy of systems and their respective connections to each other are illustrated in figure 2.1. Each subsystem will now be discussed.

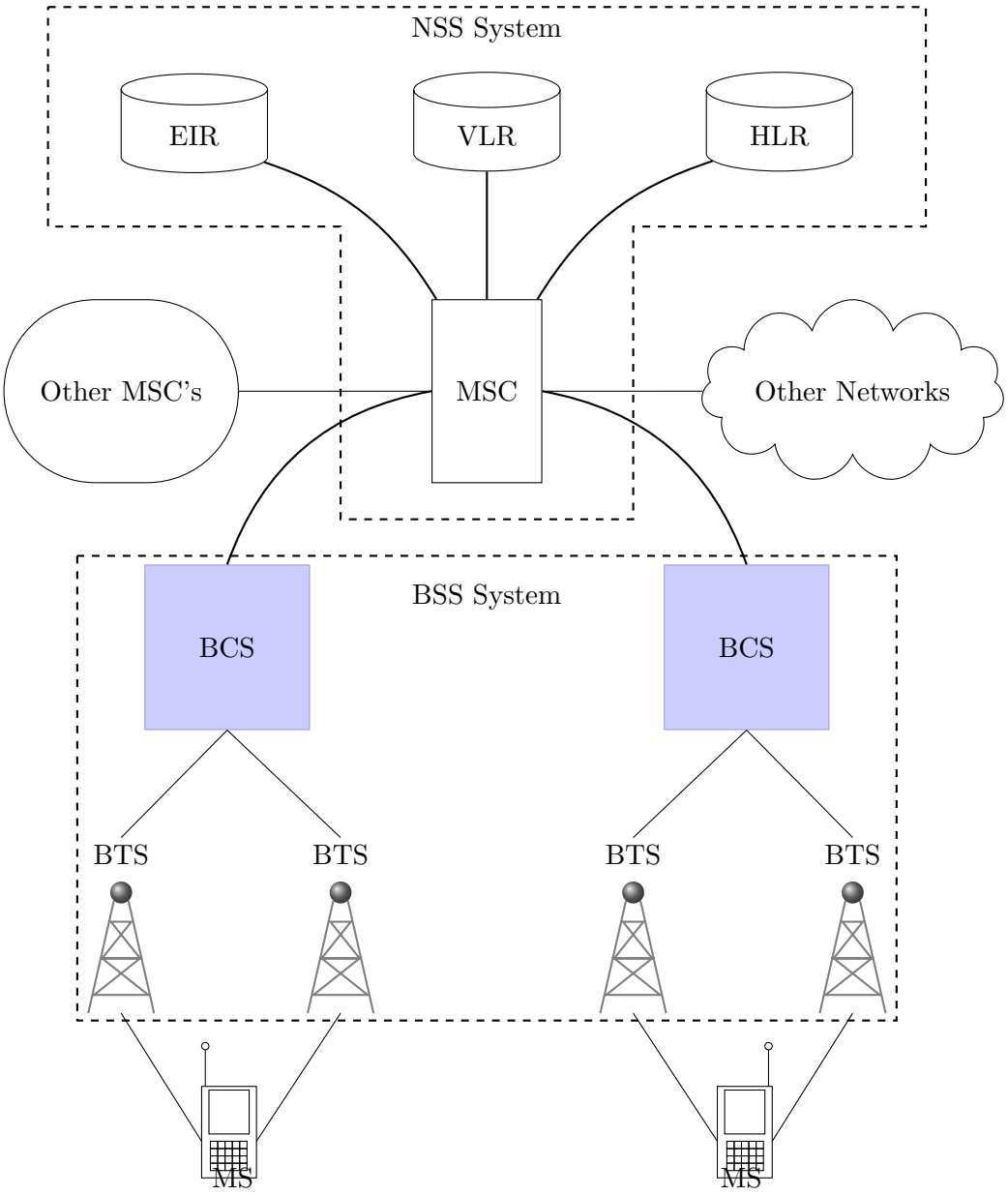

Figure 2.1: GSM architecture

Mobile Station (MS)

A mobile station (MS) as it is defined in the GSM specification refers to any mobile device that is capable of making and receiving calls on a GSM network. The MS is the main gateway for a user to gain access to the GSM network [32, 53]. Typical devices that fall under the category of MS are cellular phones, smart phones and point of sale (POS) devices. The MS has two features that play an important role throughout the GSM network, namely:

Subscriber Identification Module (SIM) — Usually inserted into a mobile device. The SIM contains the *international mobile subscriber identity* (IMSI) and is used throughout the network for authentication as well as being a key part in providing encrypted transmissions [32].

International Mobile Equipment Identity (IMEI) — Used to identify mobile station equipment. Primarily used in the denial of service to equipment that has been blacklisted[4] and tries to gain access to the network [32].

The MS has the capability to change the transmission power it uses from its base value to a maximum value of 20 MW. The change in transmission power is automatically set to the lowest level by the base transceiver station (BTS) to ensure reliable communication after evaluating the signal strength as measured by the MS [53, 77]. The power adjustment also minimises co-channel interference because the transmitted signal is less powerful and therefore less likely to interfere with other signals [77].

2.3.1 Base Station Subsystem (BSS)

According to the GSM Phase 2+ specification, this system is viewed by the *mobile switching centre* (MSC) through an Abis radio interface as the system responsible for communicating with mobile stations in a particular location area [32]. The BSS usually consists of one *base station controller* (BSC) with one or more *base transceiver stations* (BTS) that it controls [32]. The communication link between the MSC and BSC is the called the A-interface and that between the BSC and BTS is called the Abis interface [32]. A

[4]Equipment can be blacklisted for a variety of reasons, e.g. theft

BTS has similar equipment to that of a MS [77]. Both have transceivers, antennae and the necessary functions to perform radio communication.

Communication between the various GSM entities occurs over these interfaces via radio channels, which are in fact certain frequencies that are used to transmit information wirelessly. A more in-depth discussion on the types of channels that are used in a GSM network as well as their purpose will be presented in section 2.5.

In a GSM network the service area (SA) is subdivided into location areas (LAs) which are then further divided into smaller radio zones called cells [96]. When a cellular network is modelled, cells are modelled as hexagonal shapes. Each cell in the modelled network is served by only one BTS[5] and is usually regarded to be in the centre of a cell as can be seen in figure 2.2 [53]. Even though cells are modelled as being hexagons (see figure 2.2), the actual coverage area of a cell has no predefined regular shape [53].

With the network modelled as a series of interconnecting hexagons it allows one to more easily take constraints into account. For a cell to serve its geographical area, it needs to be allocated frequencies to operate on. Therefore, for each cell i in the modelled network a subset S_i of frequencies from the total frequencies F allocated to the GSM network is assigned [53]. Neighbouring cells must at all costs avoid having the same subset of frequencies allocated to them, since such a scenario would lead to severe interference on any communication and thus degrade quality [53]. Since the number of cells in a network greatly outnumber the available of subset frequencies available, one is forced to start reusing frequency subsets in cells. To ensure that the reused frequency subsets do not interfere with their neighbouring cells, a reuse distance D is defined [53]. The reuse distance means that a certain number of cells must be between the cell already assigned the frequency subset S_i and the cell to be assigned a frequency subset [53]. The amount of cells is the distance value D.

The size of a cell determines the amount of potential traffic that the cell will be required to handle [32,53,81]. Therefore, if the size of a cell is chosen to be small, fewer channels will be required to be allocated to that cell as it would not be required to handle as much traffic as a larger cell.

By making the size of cells smaller, the network operator is required to invest more into its network infrastructure. Smaller cells have a direct

[5]Also referred to as a site

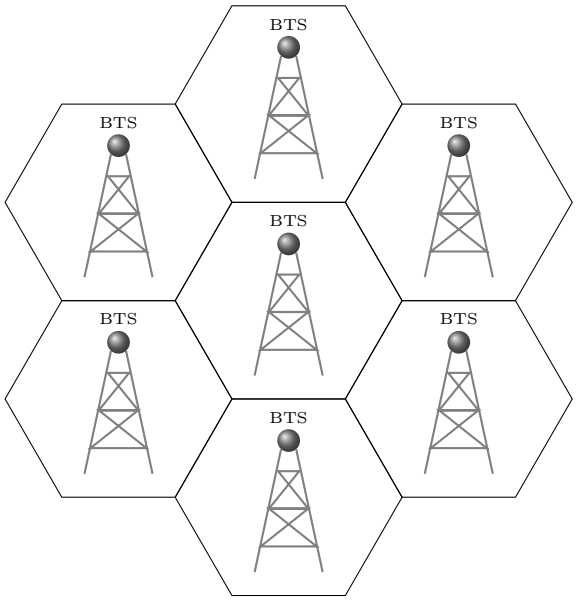

Figure 2.2: Cells with BTSs

consequence that more cells would be required to serve the same geographical area, and more cells means that more BTSs etc. need to be built and maintained, more locations need to be rented [53]. Hence, making a cell smaller has a compounding effect on the amount of infrastructure needed to support it.

Fortunately all the extra infrastructure investment by the operator can be greatly scaled down if cells are divided into sectors. Each sector performs the exact same function as a traditional cell and is therefore regarded to also be a cell, just smaller in size and not omnidirectional [53, 77, 81].

A cell is divided into 3 to 6 service sectors and each sector is allocated an antenna/transceiver [77]. Depending on how many sectors are at a cell, the operating angles of the antennae need to be adjusted accordingly to ensure 360 degree service. If there is only one sector, an omnidirectional antenna is used, otherwise the antennae operating angles are adjusted to $\frac{360\,^\circ}{n}$ where n is the number of antennae [32].

By dividing the cell into sectors the amount of co-channel interference that would occur in a cell is greatly reduced [53]. It is important to note that the reduction of co-channel interference is only applicable when the angle of the antenna by which transmission occurs is restricted [53].

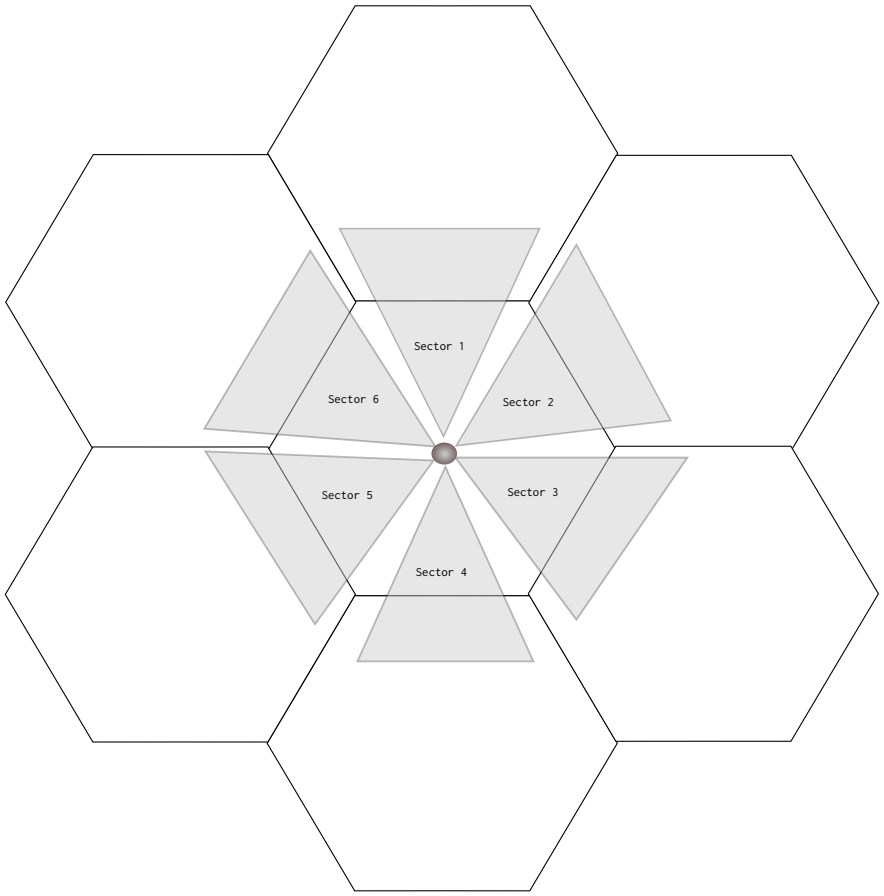

Figure 2.3: Cell sectorisation. It can be observed that the centre cell has been sectorised into 6 distinct sectors using directional transceivers. With the a transceiver servicing each sector it can be seen in the figure that the transceiver will interfere with only one cell.

Suppose, if a cell using an omnidirectional transceiver is assigned 6 channels. If the cell were to be divided into 3 sectors, where the sectors' antennae are 120° apart, the number of interfering co-channels shrinks from 6 to 2 and from 6 to 1 in the case when the cell is divided into 6 sectors [53,77,81].

Each sector operates one or more elementary transceivers called TRXs. The number of TRXs per sector is determined by the expected peak traffic demand that the cell must be able to handle. Each TRX can handle 7 to 8 communication links or calls in parallel except the first TRX, which handles fewer calls than normal because it is responsible for transmitting cell organisation and protocol information [32]. TRXs are able to handle

7-8 calls in parallel due to the use of *frequency division multiplexing* (FDM) and *time division multiplexing* (TDM) schemes.

TRXs are assigned channels, which enable them to provide conversion between digital traffic data on the network side and the radio communication between MSs and the GSM network [15, 71]. The various channels that are used by a cell for communication are discussed in section 2.5.

2.3.2 Mobile Switching Centre (MSC)

The MSC is at the heart of a cellular switching system and forms part of the *network switching subsystem* (NSS). The MSC is responsible for the setting up, routing and supervision of calls between GSM subscribers [77, 81]. The MSC has interfaces to communicate with GSM subscribers (through the BSS) on the one had and with external networks on the other [81]. The MSC interfaces with external networks to utilise their superior capability in data transmission as well as for the signalling and communication with other GSM entities [81].

The most basic functions that an MSC is responsible for in a network are the following [88]:

- Voice call initialization, routing, control and supervision between subscribers
- Handover process between two cells
- Location updating
- MS authentication
- SMS delivery
- Charging and Accounting of services used by subscriber
- Notification of other network elements
- Administration input or output processing functions

To achieve most of these functions the MSC has an integrated *visitor location register (VLR)* database that stores call setup information for any MS that is currently registered for service with the MSC [81, 88].

The VLR retrieves this information from the *home location register (HLR)* that contains all the registered GSM subscriber information for the network. This information enables the MSC to quickly retrieve the necessary information to set up a call between two entities [77, 96].

A requirement for being able to communicate with other network elements such as *Public Switching Telephone Networks* (PSTN) is the ability to multiplex and demultiplex signals to and from such network elements. This operation is a necessity, since the incoming or outgoing connection bitrate from the source entity might be either too low or too high for the receiving entity.

A typical scenario where this operation proves vital is when a mobile subscriber makes a call to a subscriber on a PSTN. The connection bit rate needs to be changed at the MSC from a wireless connection bitrate to a bitrate suitable for transmission over a PSTN.

2.3.3 Network databases

The HLR, authentication centre (AUC) and equipment identity register (EIR) are the three 'back-end' databases which store and provide information for the rest of the GSM network. In this subsection each one of the databases that form part of the 'back-end' will be discussed briefly and a description will be given of the core functions that each database performs in the network.

Home location register (HLR) — The HLR is a database that permanently stores information pertaining to a given set of subscribers. It needs to store a wide range of subscriber parameters because it is the reference source for anything GSM subscriber related in the network.

Subscriber parameters that are stored in the database include billing information, routing information, identification numbers, authentication parameters and subscribed services. The following information is also stored, but the information is of a temporary nature and can change at any time: Current VLR and MSC the subscriber is registered with; whether the subscriber is roaming [77].

Authentication centre (AUC) — The AUC is the entity in the GSM network that performs security functions and thus stores information that

enables it to provide secure over-the-air communication [77, 81]. The information that is stored contains authentication information as well as keys that are used in ciphering information [77, 81].

During an authentication procedure no ciphering key is ever transmitted over the air; instead a challenge is issued to the mobile that needs to be authenticated. This challenge requires the mobile station to provide the correct *signed response* (SRES) with regard to the random number generated by the AUC [77, 81]. The random number and ciphering keys that are used change with each call that is made; thus an attacker would gain nothing by intercepting a key, since it will change with the next call [77].

Each mobile that is registered in the HLR database needs to be authenticated and each call that is instantiated needs to retrieve keys from the AUC to establish a secure communication link [77, 81]. The AUC is sometimes included with HLR to allow for fast communication between the two entities [77].

Equipment identity register (EIR) — The EIR is a database that stores the IMEI numbers of all registered mobile equipment that has accessed the network. Only information about the mobile equipment is stored, nothing about the subscriber or call is stored in the database.

Typically there is only one EIR database per network and interfaces to the various HLR databases contained in the network. The IMEIs are grouped into three categories: *white list*, *black list* and the *gray list*. The white list contains only the IMEI numbers of valid MSs; the black list stores the IMEI numbers of equipment that has been reported stolen and the gray list stores the IMEI numbers of equipment that has some fault (faulty software, wrong make of equipment).

2.3.4 GSM Network Management Entities

In a GSM network most of the elements that form part of and make the network function are often distributed in a wide geographical area to provide the best network coverage for the customer.

For a network to function properly and efficiently network engineers need to be kept up to date on the state of the network and be alerted if *any* problems occur. For this purpose there are two systems in the GSM

network architecture that allow for this functionality required by network engineers.

One system is called the operations and management centre (OMC) which is responsible for centralised regional and local operational and maintenance activities. The other system is called the network management system (NMC) and unlike the OMC it provides global and centralised management for operations and maintenance of the network supported by the OMCs [77].

In the following sections a more in depth discussion on the critical functions the OMC and NMC perform will be presented.

Operational and Management Centre — The OMC is capable of communicating with GSM entities using two protocols, namely SS7 and X.25. The SS7 protocol is usually used when the OMC is communicating within the GSM network over short and medium distances. The X.25 protocol is used for large external data transfers. All communication where the OMC is involved typically occurs over fixed line networks and/or leased lines. The OMC is usually used for day-to-day operation of a network [77].

The OMC has support for alarm handling. An alarm in a GSM network goes off whenever a predefined expected condition does occurs. Engineers are able to define the severity of an alarm, which defines who or what is further alerted and if the alarm needs to be escalated to a higher level [77].

The OMC is also capable of fault management in the GSM network. It is able to activate, deactivate, remove and restore a service manually or automatically on network devices [81]. Various tests can be run and diagnostic information can be retrieved on the network devices to detect any current or future defects [77].

Network Management Centre — The NMC is similar to the OMC but it is not restricted to only regional GSM entities as it is in charge of all the GSM entities in the network. The NMC provides traffic management for the global network and also monitors high priority alarms such as overloaded or failed network nodes. It is usually used in long-term planning of a network, but it has the capability to perform certain OMC functions when an OMC is not staffed.

2.4 GSM Interfaces

In the GSM network all the various entities communicate with each other through various predefined interfaces. In this section an overview will be given of these various interfaces between the entities.

Um interface — This interface is the link between an MS entity and a BTS and is also referred to as the *Air* interface since communication occurs wirelessly. The primary protocol used on this interface is the *Link Access Protocol D Channel modified* (LAPDm), which is an extension of the ISDN LAPD protocol to accommodate the mobile nature of MS devices as well as for the shorter TDMA frames which are used in GSM networks [88, 96].

Abis interface — Between the BTS and BSC the interface used for communication is known as the Abis interface. The only messages that the BTS is interested in are those that have to do with management of radio resources [88, 96]. All other messages are left alone and merely pass through the BTS to the BSC transparently.

A interface — The interface between the BSC and MSC is known as the A interface. This interface is used for the transfer of information, which is used by the MSC to manage BSSs, control connections and manage the mobility of MS in its administrative area [53, 88].

Other Interfaces — The MSC has various interfaces going from itself to the various databases and other external networks. Each interface connects to a specific database, MSC or network and is therefore very specific as to what function it performs [53, 88]. An interface going from one MSC to another will typically convey information regarding handling the administration of handover of an MS device leaving one administrative area to another MSC's administrative area. The handover procedure is a very delicate process which will be described in section 2.6.

In this section a brief overview was given of the most critical interfaces used by all the entities of the GSM network involved. In the next section the difference between a logical channel and a frequency is described. Additionally an outline and overview of all the logical channels defined in the GSM will be presented.

2.5 GSM Channels

GSM defines a series of logical channels, which are used for communication over these interfaces. A distinction needs to be made between channels and frequencies. As discussed earlier, a network is licensed in a certain section of the wireless spectrum for use for commercial communication. This piece of spectrum is referred to as bandwidth and is measured in Hz, therefore W Hz, where W denotes the allocated bandwidth [128].

This bandwidth W is then divided into N smaller chunks of bandwidth called narrowband chunks. Each N narrowband chunk is a channel and has a width of W/N Hz [128]. This channel of W/N Hz width is also referred to as a frequency.

The terms channel and frequency can therefore be used interchangeably. In this dissertation the term channel will be used.

Using TDMA the GSM system is able to provide additional transmission capacity by dividing the channel into 8 equal timeslots [88]. As can

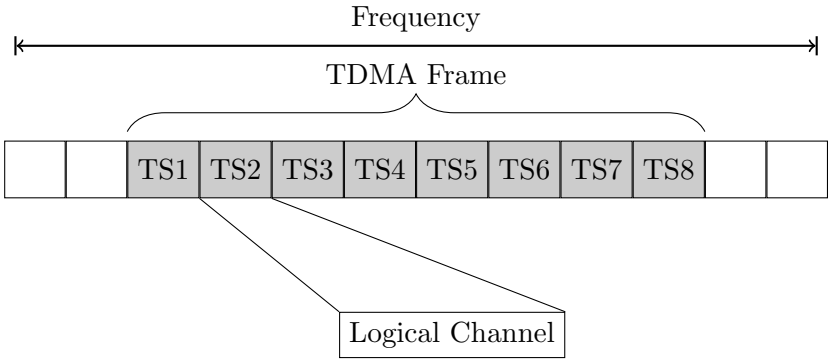

Figure 2.4: TDMA frame and logical channels [88]

be observed from figure 2.4 each TDMA frame has a series of consecutive timeslots. Each timeslot can be used for both uplink and downlink transmission. A GSM *channel* is a logical channel and refers to a single timeslot within a TDMA frame [53, 88].

The GSM system is therefore able to use the same physical frequency in 8 different timeslots without interference as these *logical* channels are used at different times. Therefore using TDMA the available channels that can be used for communication in GSM are increased eightfold [88].

Frequencies are assigned to the uplink and downlink portion of the connection with a certain duplex separation in the frequency band [6] to avoid interference between uplink and downlink. There are two types of channels, traffic channels and control channels. Traffic channels (TCHs) primary purpose is to enable communication of user speech and data and therefore carry no control information [53].

A TCH is assigned to an MS device when the device indicates that it needs to communicate with another device either with speech or data. When an MS has finished with the TCH the allocated TCH is reclaimed for use by other MS devices on the network. This request by the MS device occurs using the control channels [53].

Control channels are much more actively used in a GSM network since they are the primary means by which control and management of the network occurs [53]. These channels are used even when the MS has no active connection and is in idle mode. This constant activity on the control channels is to keep the network updated with information such as the position of the MS (location updating) and signal strength [32, 53, 77].

The control channels are divided into three main channel groups namely broadcast channel (BCH), common control channel (CCCH) and dedicated control channel (DCCH) [53]. Each of these channel groups contains other channels that aid in the control and management of the network. Each group along with the associated channels will now be briefly discussed.

The first group, BCH, consists of three channels:

Broadcast control channel (BCH) — This channel is broadcast using the very first frequency assigned to a cell. Using this frequency, the channel broadcasts information regarding the network. This information includes radio channel configuration of the current and neighbouring cells, synchronisation information, registration identifiers and most importantly the format of the CCH used by the local BTS [53].

Frequency correction channel (FCCH) — Synchronisation information is broadcast to the MSs to enable them to perform frequency correction on the transmission. Typical synchronisation information on this channel, for instance, is the exact frequency the local BTS is

[6] Separation is usually 45 MHz

using for transmission to enable the MSs to attune themselves to the same frequency [53].

Synchronization channel (SCH) — On this channel, identifying information regarding the BTS is transmitted. Also on this channel information regarding synchronisation of frames is sent which aids an MS to, for example, structure the time frames of TDMA frames.

The FCCH and SCH are always broadcast with the BCH since these channels are needed for the operation of the radio subsystem [53]. The CCCH is a point-to-point signalling channel that is used to localise an MS through the use of paging. The channel is also used to assign dedicated channels [53]. The CCCH is made up of the following channels:

Random access channel (RACH) — The RACH forms the uplink portion of the CCCH and is randomly accessed by the MSs to request a dedicated channel for a single signalling transaction [53].

Access grant channel (AGCH) — The AGCH froms the downlink part of the CCCH. It is used by the radio subsystem to assign a nSDCCH or TCH to an MS [53].

Paging channel (PCH) — The PCH also forms part of the downlink portion of the CCCH. This channel is used by the radio subsystem to page specific MSs, which aids in the process of locating an MS [53].

Notification channel (NCH) — This channel is used to inform an MS of any incoming group calls or calls that are being broadcast [53].

The last group of signalling channels is referred to as dedicated/associated control channels (D/ACCH). This group of channels has the characteristic that it is a bidirectional point to point channels [53].

Stand-alone dedicated control channel — This channel is used for communication between the BSS and MS even when there is no active connection. Hence the 'stand-alone' since it means that there need not be a TCH assigned for communication to occur between the BSS and MS [53].

Slow associated control channel (SACCH) — When a TCH or SDCCH is assigned, an accompanying SACCH is also assigned. The SACCH is used to transmit information for optimal radio operation, which can include information on power control of the radio transmitter and synchronisation information. Packets must be continuously sent over the SACCH as it is used as proof that there is still a physical radio connection [53]. When the MS has finished using the SACCH channel, it transmits a report regard the current results of the radio signal level, which is continuously measured [53].

Fast associated control channel (FACCH) — When more bandwidth is required for signalling purposes, the signal of the TCH is modified using dynamic pre-emptive multiplexing. The additional bandwidth comes at the expense of the user data transport. When a channel is created in this manner it is called a FACCH [53].

Finally, one last channel is defined named the cell broadcast channel (CBCH), which shares the same physical channel that the SDCCH uses. On this channel messages of the short message service cell broadcast are broadcast [53].

In this section an overview was given of how *logical* channels are used for communication between an MS and BSS/MSC. These three groups of channels collectively enable the GSM network to facilitate wireless communication, a very important function for a telecommunication network. The following section dealts with how the network is able to keep a connection to an MS alive and allow the MS to make calls while the device is moving around geographically within the network.

2.6 Handover

The handover process in a GSM network is initiated when an MS with an active call moves outside the coverage area of a cell, BSS or MSC [32,53,88]. A handover might also be initiated because of measurements indicating bad channel quality [53].

Various information needs to be migrated across and shared between the entities to ensure a smooth handover, which results in the MS active call not ending suddenly. It is just not the GSM architecture entities that need

to continuously share information but also the MS. The MS is required to continuously observe and measure signal strength of up to 6 neighbouring cells. The MS does this by monitoring the BCCH [53,88]. This information is of course relayed to the MSC and BTS as it plays a critical role in the decision process as to which entity will be the best in taking over the administration of the active call [53, 88].

An MS can in some cases receive the same BCCH from different cells, which are most likely neighbours [53]. This problem of duplicate BCCH from different cells can be attributed to the frequency reuse in the cellular network as well as to the smaller sector cells forming clusters and therefore overlapping coverage area [53]. As discussed in section 2.3.1 page 18, cells are typically divided into sectors, which lessen the problem of co-channel interference. This division into sectors can cause clustering of cells and overlapping of coverage area.

This creates a problem for the MS since it needs to distinguish between the two cell measurements as it does not know which measurement belongs to which cell [53]. To distinguish between different cells, the MS also tries to determine the identity of each cell it is monitoring. Only cells whose identity can be determined reliably are included in the report sent to the BTS [32, 53, 88][7]. Using this report the handover algorithm is able to decide how to handle the handover, and which cell needs to take over the call [32, 53, 88].

Once the cell has been selected, the actual handover process starts. Which has to take into account that the frequency allocated to the incoming call from the other cell does not interfere with other present active calls in the cell receiving the handover [32, 53, 88]. A frequency interfering with calls currently active in the cell can cause users to hear other conversations from the interfering call, or experience their call being "dropped" i.e. disconnected [32].

There are three core handover procedures when a handover needs to occur in a GSM network: *intra-BSC*, *inter-BSC* and *inter-MSC*, each of which involves different GSM entities [88].

Intra-BSC — Also known as *intercell handover*, this handover is concerned with the transfer of an active call/connection from an MS to

[7] When cell identity cannot be determined this can be due to environmental factors like signal strength or to packets getting lost/dropped

another cell which is controlled by the same BSC as the current cell. The current BTS that manages the active call of the MS constructs a report containing measurement information from the MS, as well as measurements the BTS has taken on signal strength and error bit ratio of the current connection [53, 88].

The constructed report is forwarded to the managing BSC, which analyses the report to determine the necessity of handing over the active call to another BTS. If a handover is deemed necessary, the BSC starts by initialising the BTS to prepare it to handle the new connection [53, 88].

The BSC then notifies the MS through the old BTS of the new BTS identity, and various properties of the new connection such as TCH frequency, power output etc. After receiving the information about the new connection, the MS makes the necessary adjustments for it to continue operating and handling the call on the new connection [53,88].

Once all the adjustments have been made the MS sends a confirmation to the BSC of the successful handover through the new BTS. The BSC instructs the old BTS that it must relinquish the use of the TCH and its associated SDCCH used by the old MS call. The BSC notifies the MSC of the handover as it is used in network operation reports [53,88].

Inter-BSC — In an inter-BSC handover a call of an MS that is being managed by a BTS is transferred to another BTS which has a *different* controlling BSC. Thus, the call is essentially moved between two BSCs, and it is up to the new control BSC to select a suitable BTS that will actually handle the call of the MS being handed over [53, 88].

This handover occurs because the MS is moving or is about to move into a cell that is not controlled by the current BSC. The current BSC detects this and therefore takes the necessary precautions to ensure that the new BSC is able to make suitable provision to assume control of the call within one of its BTSs [53, 88].

The BSC informs the managing MSC that a handover to another BSC must occur. The request sent to the MSC contains the identity of the cell managed by a different BSC. The MSC determines the managing BSC of the cell in question and notifies it that it must select and prepare the cell for handover. The new BSC informs the cell to create

a new connection, which will be used by the MS once the handover is complete [53, 88].

The new BSC informs the MSC of the new connection details which the cell will use to handle the handover and maintain the active connection of the MS. The MSC forwards the connection information to the old BSC, which forwards it to the MS. The MS makes the necessary adjustments for it and then moves on to the new connection. The MS informs the new BSC that the handover has been completed successfully. The new BSC informs the MSC, which then instructs the old BSC to relinquish the old connection used by the MS [53, 88].

Inter-MSC — With an inter-MSC handover, control and management of an active call on an MS must be transferred to another BTS, which resides in a different area that is managed by a different MSC. The handover process follows the same basic formula as the intra-BSC and inter-BSC handovers once the handover request is made [53, 88]. The BSC managing the BTS to which the MS is going to be handed over is to be determined by the MSC. The BSC is notified by the new managing MSC that certain BTSs must bring a new connection online for the incoming MS; the entities upstream[8] are notified [53, 88].

The new MSC informs the old MSC of the new connection details. The old MSC transfers the connection information to the MS in question that is going to be a handover to another BTS. The MS makes the corresponding adjustments and then starts operating on the new connection. The MS informs the BSC of the successful handover [53, 88].

The BSC in turn informs the new MSC, which in turn informs the old MSC that the handover was successful. The MSC then instructs the BSC to ensure that the old connection resources are relinquished by the BTS.

In this section a discussion was presented on the process the GSM network follows as an MS device moves around geographically in the network to keep an active call on the MS active and not be disconnected. The effect this has on channel selected as well as the different handover procedures were highlighted.

[8] Entities upstream are the entities which are higher up in the managing structure. In this instance, BTS – BSC – MSC.

2.7 Summary

In this chapter a broad discussion was given of modern cellular technology, specifically GSM cellular network technology. A brief history on how GSM was developed to be the most widely used cellular technology in use today was provided. The various GSM architecture entities followed. In the GSM architecture all the entities present in a modern GSM network were identified.

Following the discussion of the GSM entities, a broad overview was given of the various communication interfaces used between the various GSM entities to communicate with each other. This was followed by a definition of the GSM channels which are used on the interfaces to communicate information.

The chapter ended with the handover process which is used to allow an MS device to move freely geographically within the network.

Chapter 3

The Frequency Assignment Problem

3.1 Introduction

The frequency asignment problem (FAP)[1] is a generalisation of the graph-colouring problem and is consequently an NP-Complete problem [30]. The FAP is an NP-Complete problem due to fact that only a finite number of frequencies can be assigned to antennae/transceivers (TRXs), where the number of transceivers to be assigned frequencies greatly outweighs the number of available frequencies [30]. A more thorough definition of what it means for a problem to be NP-Complete will be given in section 3.2.

In wireless communication a huge concern is a notion known as interference which occurs when frequencies used for communication are close to each other in the frequency spectrum [3]. Interference and its effects will be discussed in detail in section 3.4. Essentially for the FAP the primary concern is to develop an approximate plan on assigning frequencies in such a way that interference is kept to a minimum.

Using exact algorithms to find a solution is not practical since the time to find a solution will be polynomial. Generally metaheuristic algorithms are used to find optimal solutions to NP-Complete problems [71]. In the chapter 4, a discussion will be presented on algorithms that are generally used to find solutions to NP-Complete problems.

[1] Also known as automatic frequency planning (AFP) or channel assignment problem (CAP) [71]

A wireless cellular operator is not allowed to operate on just any frequency. A governing body licenses a certain piece of the available wireless spectrum to the operator for use in their network [30]. These frequencies that need to be licensed for use are known as *commercial* frequencies and are very scarce as it is immensely expensive to license these frequencies [30].

Usually a licensed piece of spectrum contains a series of consecutive frequencies as well as gaps. Gap frequencies are barred from being used by any device within the network as they may have already been allocated to another operator for use [14]. By barring frequencies, a scenario is avoided where the different networks' equipment interferes with their respective operations [14].

Due to the whole spectrum not being available to network operators and only a subset being available for commercial communication as per the frequencies allocated to them, networks opt to reuse their frequencies [14]. The networks do this to maximise the use of their allocated frequencies and to minimise their licensing fees, since if the network needs more frequencies, these need to be licensed [30].

It is not always possible to simply allocate more frequencies to a network even if the network pays the associated fees. The whole commercial spectrum may already have been licensed to various entities. Hence, licensed frequencies are a very valuable and scarce commodity [3, 14, 30, 32].

As will be discussed in section 3.4, when frequencies are reused the probability that interference will occur on the communication link increases. Interference is a very important factor for networks to consider and keep to a minimum as it influences the quality of the communication occurring on the network.

In the next section an overview of what it means when a problem is NP-Complete is given.

3.2 NP-Complete

The term NP stems from the field known as complexity analyses. Algorithms are typically measured for their worst running time using $O(n)$ notation. The field of complexity analyses is more interested in measuring complexity of a problem than the running time of an algorithm [106].

The field of complexity analyses makes a keen distinction between problems that can be solved by algorithms in polynomial time and problems that cannot be solved in polynomial time using any available algorithm [106]. Polynomial time refers to a timespan that is reasonably feasible, for instance 8 hours or 1 day.

A distinction needs to be made between *finding* a possible solution and determining whether a result is a valid solution to an NP problem. Verifying whether some result is indeed a solution to an NP problem is a quick operation. Finding a solution through the use of an algorithm is what polynomial time refers to.

Problems that can be solved in polynomial time usually have worst case running times of $O(n), O(\log n)$ and $O(n^2)$. These classes of problems are relatively easy to solve using most algorithms available today. Problems that can be easily solved with these kind of running times are classified as being in the P range of complexity problems [106].

Another range of problems consists mostly of problems that cannot be solved in polynomial time. Hence, if an algorithm were to try each and every possible solution, it would take an arbitrarily long time, which cannot be determined, which is why these problems also have the characteristic of being non-deterministic. Problems in this range are referred to as being in the NP range of complexity problems [106].

There is another range of NP problems that are a subset of NP problems, which are referred to as the "most extreme" problems within NP. These problems are collectively known as NP-Complete problems and are the most difficult problems to determine feasible solutions for in the NP problem range [106]. The FAP is one such problem [3, 31, 32, 85, 144].

3.3 Frequency Assignment Types

In this section the different methods used to allocate frequencies to cells in a cellular network are discussed. Furthermore the method that relates to the specific FAP variant in this dissertation will be described.

Within the FAP domain there are different types of FAP, which have emerged over the years as the domain of wireless communications matured and technological requirements changed. These FAP variants will be discussed in section 3.5.

There are a variety of FAPs in the wireless communication domain but each individual problem can be classified into one of the following two categories based on the way frequencies are assigned to cells:

- *Fixed frequency/channel assignment (FFA/FCA)* is where channels assigned to cells are static; therefore they cannot be changed until a new assignment plan is calculated.

- *Dynamic frequency/channel assignment* (DFA/DCA) is the process of allocating channels to cells as required to meet the current traffic demand imposed on them by clients.

3.3.1 Fixed Frequency/Channel Assignment (FFA/FCA)

Fixed Frequency/Channel Assignment (FFA/FCA) is the process of permanently assigning frequencies to cells (cellular towers). The frequencies assigned are fixed and cannot be changed immediately while the network is active, since the frequencies assigned to the cell form part of a delicate frequency plan designed to keep interference on communication links to a minimum [117].

When the channel used by a particular cell for communication is suddenly changed, the cell might start interfering with neighbouring cells' communication links, if the assigned channels of the neighbouring cells are close to each other on the frequency spectrum. Hence, if the cell is sectored[2], it can interfere with a minimum of 3 and up to a maximum of 6 neighbouring cells [117].

When an FCA plan is created, cells are assigned frequencies based on the estimated traffic that cell will be expected to handle during peak network usage. FCA is ideally suited for macro cellular networks since the nature of the traffic encountered in such networks has the characteristic of being homogeneous, stationary and predictable [117]. Cellular networks can be classified as being in the macro cellular group of wireless networks.

With FCA, networks are able to permanently allocate a certain subset of frequencies to cells since the nature of the traffic on their network allows them to predict with reasonable certainty the call blocking probability [117].

[2]Sectorisation of cells is discussed in chapter 2, section 2.3.1

A call is blocked on the network when a cell has no available channels to use when establishing a communication link [117].

In situations where the nature of the traffic is neither homogeneous nor stationary, using the FCA allocation scheme is not feasible as its use of available frequencies is grossly inefficient [117]. With FCA, if there is a new call or hand off in a cell where all the permanently assigned frequencies are in use, the call will be blocked, even if adjacent cells have available frequencies, which can be used to handle the call [117].

3.3.2 Dynamic Frequency/Channel Assignment (DFA/DCA)

DFA/DCA is a channel allocation scheme where frequencies assigned to cells are not permanent but rather assigned to cells as the need arises [117]. Therefore all the frequencies licensed by a particular network are available to each and every cell to establish a communication link as long as the channel does not violate the co-channel reuse constraint [117].

The co-channel reuse constraint must be adhered to otherwise the amount of interference occurring on the communication link will be too much. This constraint forms part of the electromagnetic constraints, which are described in section 3.4.

The DCA allocation scheme is ideally suited for micro cellular wireless networks since the traffic on these networks has the characteristic of being immensely unpredictable as traffic demand varies constantly [40, 110, 117].

As the name indicates, micro cellular wireless networks have much smaller cell sizes than macro cellular networks. Thus a cell in a micro cellular network must handle a lot more hand-off traffic than a cell in a macro cellular network, since an MS with an active connection is much more likely to move out of the coverage area of a micro cell than a macro cell [40, 110, 117].

Since a micro cellular network has increased hand-off traffic compared with a macro cellular network, a DCA scheme must rapidly allocate frequencies to requesting cells that must handle the hand offs [40, 110, 117].

DCA is much more efficient than FCA when the amount of mobile traffic on the network is relatively low. On the other hand, when the network is under heavy mobile traffic load, the FCA scheme outperforms the DCA scheme, since the DCA usually allocates frequencies to cells in an inefficient

arrangement that might affect the amount of interference encountered on the network [40, 110, 117].

Finally DCA inherently requires a great deal more computational power than FCA, since the frequencies need to be selected and allocated with great speed, otherwise the cell requesting a channel will not be able to handle the call and will therefore block the call or drop the call [40, 110, 117].

Most researchers have concentrated on solving the FFA using heuristic approaches like neural networks, local search techniques and more recently metaheuristic approaches, which include genetic algorithms, simulated annealing, ant colony optimisation and particle swarm optimisation.

This concludes the discussion on the different allocation schemes used in modern cellular networks. In the next section a description will be given of what interference is and why it is important for cellular networks. An overview will also be given of when interference occurs.

3.4 Interference

Interference can be defined as any unwanted signal that is received along with a signal of interest. The unwanted signal is said to *interfere* with the original signal and as a consequence degrades the original signal quality with unwanted information [36].

Interference usually occurs when two or more entities communicate independently on the same channel or on adjacent channels [36, 40]. Other external factors can also contribute to interference on a communication link, such as machines, which inherently produce some sort of electromagnetic distortion, for instance a car's ignition or a big turbine [36, 40].

Interference that occurs when two signals operate on the same channel can be seen in figure 3.4 and interference that occurs as a consequence of two signals operating on adjacent frequencies can be seen in figure 3.4.

The impact interference will have on an entity that has established a connection and that operates on the same channel or adjacent channel as another entity decreases as the geographic distance between them increases [32, 36, 40, 107]. Therefore, to minimise the impact interference will have on communication links a *separation* is defined [32,36,40,107]. More specifically, this separation is known as the channel reuse or frequency reuse distance within wireless networks [32, 36, 40, 107].

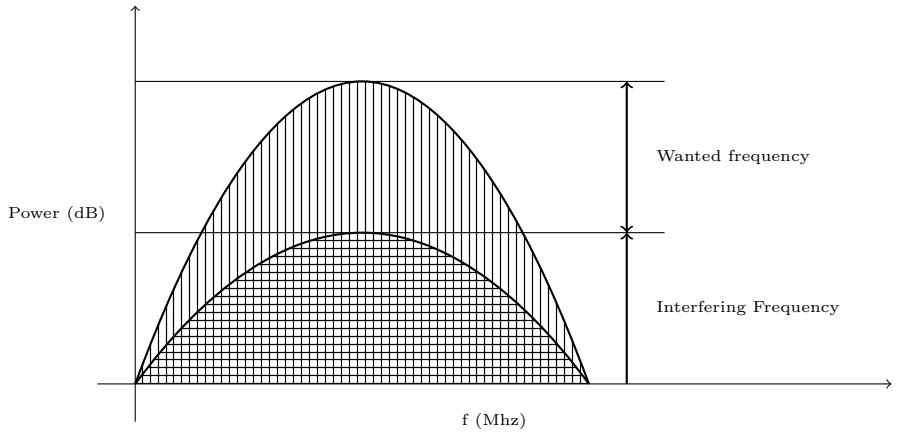

Figure 3.1: Co-channel interference

This separation is defined as the minimum number of cells (which must all use different channels) between one cell, which has been allocated a channel, and another cell before a cell is allowed to reuse the same channel that another cell has been allocated [32, 36, 40, 107].

The separation can be depicted visually as in figure 3.3 where f_a, f_b, f_c, f_d are different frequencies that are assigned to the specific cells. The frequency f_a is allowed to be reused since the two cells it is assigned to are separated by 3 cells (shaded in gray) because the separation for this network was set to 3.

As discussed earlier, cellular networks are forced to reuse their licensed

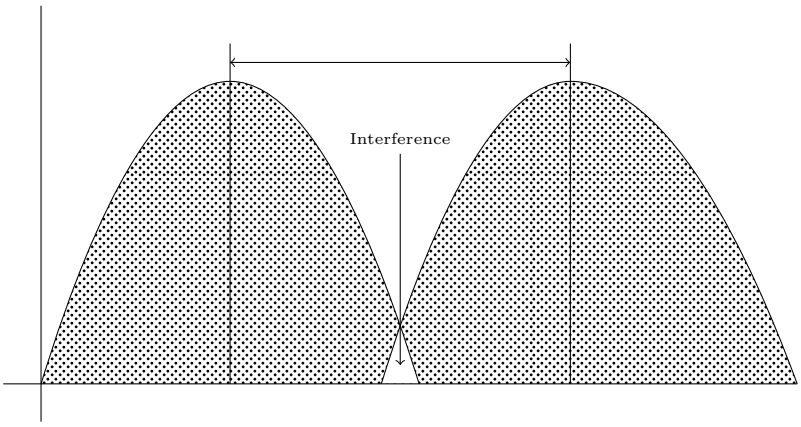

Figure 3.2: Adjacent channel interference

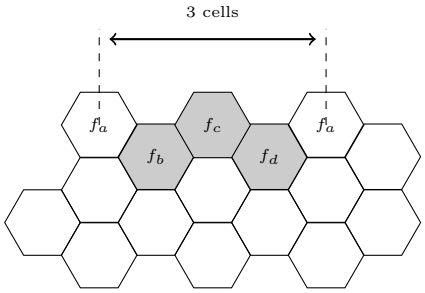

Figure 3.3: Frequency Separation

frequencies multiple times to keep costs to a minimum. Therefore, the design of a cellular network is limited to the defined separation distance between cells as it defines the size of cells that will be in the network. Smaller cells can lead to a larger separation distance compared with when cells are larger [32, 36, 40, 107].

Cellular networks use the amount of interference on their networks as a qualitative measure for their quality of service (QoS). A network with high interference would experience a lot of dropped connections/calls, which occurs when the interference is too high to sustain a connection or call for communication; consequently their QoS degrades as interference increases [36, 40].

Even though interference can cause a call or connection to be lost, i.e. dropped, there are other situations where a call can be dropped due to other factors. For instance, a call can be dropped when a handover procedure occurs between two cells and one cell receiving the call is at full utilisation of its allocated frequencies [36, 40, 77].

In the literature a variety of methods are used to calculate the amount of interference in a network. The SIR ratio is the recommended way of calculating the potential interference at a certain point [3].

The SIR equation is actually based on the signal-to-interference-plus-noise power ratio (SINR) but since cellular networks are interference limited, the noise is not considered in the interference calculation [40]. Noise can be disregarded since the power of interference is much larger than the power of noise [36, 40].

A formulation of the SINR and SIR is as follows:

$$sSINR = \frac{P_r}{N_0 + P_I} \tag{3.1}$$

$$SIR = \frac{P_r}{P_I} \tag{3.2}$$

Where P_r is the power of the received signal and P_I is the power associated with interference from within a cell (intracell interference) and interference from outside a cell (intercell interference) [40].

This calculation can be considered a best guess as it models the environment, weather and other factors which may influence the potential interference at a point with a Gaussian distribution for noise represented by the N_0 [3, 40].

Using the SIR formula cellular networks are able to determine the *bit-error rate* (BER) users on the network will experience on their connections [36, 40]. The BER is defined as the probability that a received bit on the connection will be incorrect [36, 40, 110].

As the BER increases voice quality on the connection decreases since more bits that are used to describe the voice information are incorrectly received. SIR and BER probability are interlinked. As SIR increases, i.e less interference is encountered on the communication link, the probability that bits will be received incorrectly decreases [36, 40, 110].

Whether precise measurements are taken or the interference is calculated based on the SIR formula, the end result of both methods is that all the calculated or measured values are put into a matrix to produce an *interference matrix* [71].

An interference matrix consists of a number of cell pairs (*i,j*), where i is the cell receiving interference and j the cell whose allocated channel is providing the interference. Each cell pair in the matrix has two corresponding values that indicate the level of interference if the *electromagnetic constraints* are violated [3, 31, 32, 71].

Primarily interference occurs when the electromagnetic constraints are violated. These constraints are defined as:

Co-channel — As discussed earlier, when cell i and cell j operate on the same channel interference will occur [3, 32, 36, 40, 71, 77, 107, 109, 117]. When this type of interference occurs it is referred to as *co-channel* interference.

Adjacent channel — When cell i and cell j operate on adjacent channels, their allocated frequencies differ by one, i.e. cell i operates on channel *f*, then if cell j operates on either channel *f - 1* or *f + 1*, then interference will occur [3, 32, 36, 40, 71, 77, 107, 109, 117]. This type of interference is referred to as *adjacent channel* interference.

The electromagnetic constraints defined above are applicable in any wireless network. With regard to mobile telecommunication networks, such as cellular networks, there are additional constraints that are imposed due to technological requirements, availability, location and size of area with unacceptable interference [3, 31, 32]. These constraints are defined as the following:

Co-site — If cell i and cell j are located at the same site, then their allocated channel ranges must differ by a certain distance in the frequency domain. This distance is known as the reuse distance where cell i and cell j serve different sectors [3, 31, 33, 144]. In the benchmarks which are discussed in section 3.7 this distance is also referred to as the *separation variable*.

Co-cell — Channels used on the same antennae of a cell must differ by a certain number. This is typically set to 3 but can be any number greater than 0 that the network operator deems necessary to avoid unwanted interference [3, 31, 32].

Handover — This constraint means that frequencies must differ by a predefined margin, i.e. 2 or 3, when one cell hands over a call to another cell. If this constraint is violated a mobile subscriber will experience a dropped call since the handover between cells fails [3, 31, 32].

Within the licences of wireless networks there are two hard constraints which forbid networks from using certain frequencies. Hard constraints means that under no circumstances are these constraints allowed to be violated.

The first set of hard constraint frequencies is known as *globally blocked channels*. Channels that are in the set of globally blocked channels are usually frequencies that have been licensed to other networks [3, 32, 107].

The second set of hard constraint frequencies is known as *locally blocked channels*. These channels are not allowed at certain geographic areas but

are free for use at any other area [3, 32, 107]. A typical area where certain frequencies will be forbidden to be used is near a country border [3, 32, 107]. The locally blocked frequencies are most likely in use by another network resident to the bordered country.

In this section a description was given of what interference is and what the consequences are of too much interference in a network. This section further elaborated on the circumstances in which interference can occur in a wireless network. In the next section an overview will be given of the various different subproblems in the FAP domain.

3.5 Frequency Assignment Problem types

In this section each of the problem variants for the FAP will be discussed, starting with one of the first and oldest problems in the FAP domain. This section will conclude with a on the particular variant of FAP focussed on in this research.

3.5.1 Minimum Order FAP

The Minimum Order FAP (MO-FAP) was the first FAP that emerged in the 1970s. The MO-FAP is concerned with assigning frequencies to transmitters while interference is minimised as well as minimising the number of different frequencies that are used [3, 85].

In MO-FAP channel reuse is prioritised and the usage of a channel has a certain cost associated with it. The reason for this is that when the wireless network industry started, operators were billed according to the number of different frequencies they used. In the beginning frequencies were not cheap since they were sold per unit [3, 85].

Over the years as the law governing the wireless spectrum changed and new technology as well as standards emerged, MO-FAP lost its relevancy [3, 85]. Companies are no longer billed according to the different frequencies they use, but they purchase licences from a regulatory body [3, 85]. This licence usually stipulates what channel band the network is allowed to use.

In some instances a certain band of frequencies is put up for auction by a regulatory body, on which interested parties can bid to own the specified

spectrum [3, 85]. Due to the shift in how frequencies are allocated to networks, neither the regulatory bodies nor the network operators care about the number of different frequencies are used [3, 85].

3.5.2 Minimum Span FAP

The Minimum Span FAP (MS-FAP) is a problem that is very relevant today, especially when network operators want to deploy a new network in a region [3]. The MS-FAP is concerned with keeping the interference below a certain level during assignment as well as minimising the span. The interference threshold used is specified by the network designer as the minimum allowable interference on the network [3, 85, 105].

The span is defined as an interval on the frequency domain. This interval is the difference between the maximum and minimum frequencies used during assignment [3, 85, 105]. With the span value, network operators are able to request certain frequency bands and know their network will be able to operate at suitable interference levels [3, 85, 105].

The MS-FAP and MO-FAP are two very similar problems, the only difference being that MO-FAP focuses on minimising different frequencies and MS-FAP focuses on minimising the interval of frequencies used during assignment [3]. The Philadelphia benchmark is usually used to gauge how well the algorithm performs.

3.5.3 Minimum Interference FAP

The Minimum Interference FAP (MI-FAP) or Fixed Spectrum FAP (FS-FAP) is typically encountered after the network operator has obtained a frequency band from a regulatory body. Other problems use matrices to forbid certain frequencies from being used by certain transmitters [3, 32, 44, 85].

Unlike the previous problems, in MI-FAP any available channel in the allocated band may be used even though it produces interference. The other problems are concerned with the frequencies used, even though they might be violating some constraints that incur a huge amount of interference [3, 32, 44, 85]. The interference value does not play a large role in their respective objective functions [3, 32, 44, 85]. In MI-FAP the objective is to minimise the

total amount of interference on the network. It is important to note that this amount of interference might not necessarily be zero [3, 32, 44, 85].

The MI-FAP is the problem currently most encountered in cellular networks, since there are more operating networks than new networks being designed in the cellular industry today. This particular problem forms the focus of this research.

Since MI-FAP is very close to real-world instance problems, authors tend to use real-world instances or benchmarks to test the quality and efficiency of their algorithms [3,32,44,85]. The quality and efficiency of the solution in this research will be benchmarked against the COST 259 benchmark ,which is discussed in section 3.7.

In the following section a formal mathematical definition for the fixed spectrum MI-FAP will be set out. The definition is important as it forms the basis for the objective/cost function that the algorithm in this research uses.

3.6 Fixed Spectrum MI-FAP Mathematical Formulation

A Mathematical definition of the FAP is given in this section. The mathematical definition will be used by the algorithm discussed in this dissertation to evaluate the amount of interference that generated frequency plans exhibit.

The FAP can be represented as a graph colouring problem is known to be NP-Complete. Before a mathematical definition can be formally given for the FAP, some symbols and their respective definitions need to be introduced.

$$G = (V, E) \tag{3.3}$$

$$V = \{v_0, v_1, ..., v_i\} | i \in \mathbb{N} \tag{3.4}$$

$$E = \{v_0v_1, v_0v_2, ..., v_iv_j\} | v \in V, \forall ij \in \mathbb{N}, i \neq j \tag{3.5}$$

$$D = \{d_{01}, d_{02}, ..., d_{ij}\} | \forall \{i, j\} \in E, \exists d_{ij} \in \mathbb{N}^+ \tag{3.6}$$

$$P = \{\{p_{\bar{0}0}, p_{\bar{=}01}\}, \{p_{\bar{1}0}, p_{\bar{=}11}\}, \ldots, p_{\bar{i}0}, p_{\bar{=}i1}\}\} | \forall \{i, j\} \in E, \exists p_{ij} \in \mathbb{N}^+ \tag{3.7}$$

$$F = \{0, 1, 2, 3, ..., k\} | \forall k \in \mathbb{N}, \forall v \in V \exists f \in F \tag{3.8}$$

$$d_{ij} < |f(i) - f(j)|, \forall ij \in \mathbb{N}, i \neq j \tag{3.9}$$

Let G (see equation 3.3) be a weighted undirected graph, where V (see equation 3.4) is a set of vertices. Each $v \in V(G)$ represents a transmitter in the FAP.

E (see equation 3.5) is a set of edges. An edge consists of two vertices v_i and v_j that are joined because there is a constraint on the frequencies that can be assigned between the two vertices or transmitters. Each edge has two associated labels d_{ij} and p_{ij} [15, 86].

The label d_{ij} that is part of the set D (see equation 3.6) denotes the maximum separation that is required between frequencies assigned to two transmitters v_i and v_j. $f(i)$ denotes the frequency assigned to i. Using equation 3.9 the amount of interference that is generated between transmitters v_i and v_j can be determined. By evaluating the amount of interference generated it can be decided whether the interference lies within an acceptable range, which is predetermined by a network operator [15, 86].

The other label, p_{ij}, forms part of the set P (see equation 3.7) which is referred to as the interference matrix[3]. Each label p_{ij} contains two values which represent interference[4]:

- $p_{\bar{i}0}$ represents the value for co-channel interference [15, 86].
- $p_{\bar{=}i1}$ represents the value for adjacent channel interference [15, 86].

Finally the set F (see equation 3.8) denotes a set of consecutive frequencies for every transmitter in V [15, 86].

[3] Discussed in section 3.4

[4] Interference values can be zero in some cases

Formally the FS-FAP can now be defined as a 5-tuple $FS-FAP = \{V, E, D, P, F\}$ with a required mapping of $f : V \rightarrow F$ [86]. The objective of the FS-FAP is to find an assignment of frequencies to transmitters that minimise the sum of total interference (see equation 3.11).

$$c(p_i) = \begin{cases} \bar{p_{i0}} & \text{, if } |f(i) - f(j)| = 0 \\ \bar{\bar{p_{i1}}} & \text{, if } |f(i) - f(j)| \leqslant d_{ij} \\ 0 & \text{, if } |f(i) - f(j)| > d_{ij} \end{cases} \tag{3.10}$$

$$TotalInterference = \sum_{i=0}^{\mathbb{P}} c(p_i), p \in P \tag{3.11}$$

In the following section a brief discussion on the different FAP benchmarks that exist will be provided and the benchmark against which the algorithm developed in this research will be evaluated is defined.

3.7 FAP Benchmarks

Some of the most used benchmarks in the FAP domain are now discussed. The first benchmark was introduced in the 1970s.

3.7.1 Philadelphia Benchmarks

The Philadelphia benchmarks are derived from an instance that was introduced in 1973 by Anderson. Each instance is a hexagonal grid of cells that overlaps the area of interest. At the centre of each cell there is a transmitter. Past approaches used these hexagonal systems to model modern cellular networks [3, 74].

In this benchmark interference is measured by a co-channel reuse distance. This distance stipulates that the difference between the frequencies assigned to two cells must be greater than or equal to a certain value d. A channel cannot be assigned to a cell if it violates this minimum distance [3, 74].

These benchmarks are typically used to test algorithms developed for MS-FAP, since there is no concept of cost or penalty for interference incurred by violating constraints.

3.7.2 CELAR

In 1994 EUCLID introduced a project called CALMA, which was a combined effort by several European governments that were part of EUCLID to investigate algorithms for military applications. The project was granted to 6 research groups. Within the project 36 instances were made available by CELAR for radio link frequency assignment [3, 29].

All the CELAR instances have the constraint that the difference between frequencies assigned to interfering radio links must be greater than a certain predefined distance in the frequency domain. This is a soft constraint and may be violated. Another constraint in the CELAR instances is that each pair of parallel links must differ by an exact predefined distance. This constraint is a hard constraint and may not be violated [29].

These instances were initially not available to the general public as they were contained to be within the CALMA project. In 2001 the CELAR launched the International ROADEF challenge, where certain instances from the CALMA project were made available for the research teams taking part in the challenge. The instances made available had been modified to take polarisations and controlled relaxations of certain EMC constraints [49].

3.7.3 COST 259

The COST (COoperation europene dans le domaine de la recherche Scientifique et Technique) 259 is a set of real-world GSM instances made available by the European Union. The instances are publicly available and can be downloaded for free at http://fap.zib.de/ (FAP Web 2011). The website also contains the most recent results obtained by researchers using these instances [3, 32].

The instances are fairly difficult due to the large number of transmitters (900 - 4 000) that need to be assigned frequencies, with a relatively small number of spectrum of frequencies. The most important characteristic of these benchmarks are that they resemble real-world GSM network data. Due to these benchmarks' real-world applicability, they were selected as the main benchmarks to evaluate the algorithm presented in this dissertation.

More specifically this research concentrates on a small subset of the instances that are available, namely Siemens1, Siemens2, Siemens3 and Siemens4. In the paper by Montemanni and Smith [86] the same subset

of problems was used and to date their algorithm has produced some of the best results. The characteristics of each instance will now be discussed.

Siemens1

The Siemens1 instance resembles a GSM network that follows the GSM900 standard. This particular network has been allocated a spectrum set of frequencies $F = 16 - 90$ which are allowed to be assigned to cells.

Not all 74 frequencies are available to be used by the network. The allocated frequency block is split into two blocks. Because according to the problem instance frequencies ranging from 36 to 67 are globally blocked; thus frequencies ranging from 16 - 35 and 68 - 90 are available for assignment.

This problem instance finally also defines this network as consisting of a total of 506 cells where on average each cell has 1.84 transceivers that need to be assigned a frequency. The co-site separation is stated to be 2 and the co-cell separation is stated to be 3.

Siemens2

The Siemens2 problem instance describes a GSM network based on GSM900 and has 86 active sites. The problem specifies that the network consists of 254 cells where each cell has on average 3.85 transceivers that need to be assigned frequencies.

For this problem, the network has been allocated two blocks of frequencies: one block of 4 frequencies ranging from 42 - 46 and a second block of frequencies ranging from 53 - 124. The frequencies allocated to the network have been split into two blocks because frequencies ranging from 47 - 52 are globally blocked. Finally the problem specifies that the co-site separation must be set to 2 and the co-cell separation must be set to 3.

Siemens3

The Siemens3 problem describes a network based on GSM900. This network has been allocated a continuous set of frequencies that start at 681 and end at 735. Thus the network has 55 frequencies, which can be allocated to transceivers in its networks.

The problem defines the network as consisting of 366 active sites and 894 cells. On average each cell has 1.82 transceivers that need to be allocated a frequency to handle communication.

Siemens4

The Siemens4 instance is similar to a GSM network that follows the GSM900 standard. According to this instance this network has been allocated 39 continuous frequencies starting at 56, thus $F = (56, 94)$. No frequencies are said to be globally or locally blocked in this network.

According to this problem instance this network has 276 active sites and consists of 760 cells. Where each cell is said to have on average 3.66 transceivers. The Co-site separation is set to be 2 and the co-cell separation must be 3.

In the next section a general overview will be given on the different industries where the FAP is encountered.

3.8 FAP in the Industry

In this section some of the industries where the FAP is encountered will be listed. For each industry listed, a brief overview is given of how the problem differs compared with other industries.

3.8.1 Satellite Communication

The FAP in the satellite communication domain occurs in the ground terminals that transmit and receive signals via a satellite. One would assume that the problem includes the satellite, but the problem is only concerned with the frequencies that the ground terminals use. It is interesting to note that the ground terminals can be a base station or a handheld device (e.g. GPS or satellite phone).

In Satellite communication, a signal is transmitted to one or more satellites via an uplink from a ground terminal. The signal is received by the recipient satellite and relayed to the interested ground terminals that receive the signals via a downlink.

A large distance in the frequency domain separates the frequencies used by the ground terminals for uplink and downlink communication. The typical distance is much larger than the bandwidth. When frequencies are assigned to transmitters, downlink transmitters are ignored and only uplink transmitters are considered [3].

A radical difference with regard to the use of frequencies compared with the standard FAP in cellular networks is that frequencies are only allowed to be used once. This is specific to the satellite domain to avoid interference [3].

3.8.2 Wireless Mesh Networks and Wireless Local Area Networks (WLANs)

Wireless mesh networks and WLANs[5] are the most recent applications where the FAP is encountered.

Multiple WLANs are increasingly being used to provide backbone support for large fixed line networks, enterprise networks, campuses and metropolitan areas. To be able to provide backbone support for these networks, a primary design goal when designing and deploying these networks is capacity. A limiting factor for WLAN capacity is interference, which affects multihop hop settings. Thus the overall network interference needs to be minimized to increase the capacity of the network [118].

Typical approaches allocating frequencies include using DCA and FCA[6]. DCA is not very popular because the dynamic switching of channels lowers the response time on commodity hardware since there is a delay in milliseconds when switching channels. Typical packet transmission times are in microseconds. To guarantee uptime and high responsiveness, FCA is the preferred approach [118].

The FAP in wireless mesh networks and WLANs differs from the standard problem in that it introduces an extra constraint. Channels assigned to links on a node cannot be more than the available interfaces on that particular node. This constraint is known as the *interface constraint* [118]. Another aspect to consider is the placement of access points (APs) in the network, which is similar to the problem cellular networks face with regard to base station placement [3].

[5]Both applications use the same standard and encounter similar problems in their respective domains.

[6]Discussed in section 3.3

3.8.3 Military Field Communication

In a military context the FAP is a very difficult problem to be solved due to its dynamic nature. During deployment, connections need to be established rapidly between nodes which guarantee that the nodes will stay static at locations. Usually nodes are military field phones or can be any transceiver device [2, 29].

Due to the nature of the problem the DCA scheme is used to allocate frequencies to nodes. The military FAP differs due to the property that any of the nodes is mobile and can move at any moment to a new location, potentially interfering with another connection [2,29]. Two frequencies need to be assigned to each connection that is established, one for each direction of communication. These allocated frequencies must also differ by a certain distance in the frequency domain to prohibit alternating directions of communication from interfering [2, 29].

A lot of literature can be found on Military field communication. This is due to two organizations CELAR[7] and EUCLID[8] making data available to various research groups and allowing them to develop algorithms for frequency assignment [2, 29].

3.8.4 Television and Radio Broadcasting

The FAP encountered in broadcasting very closely resembles the problem domain found in cellular networks. The only notable difference is that the required distance by which allocated frequencies must differ in the frequency domain are larger in broadcasting than in cellular networks [3].

Since the problem resembles the problem found in cellular networks, there are few articles that specifically discuss frequency assignment in broadcasting as a main topic. Research that specifically discusses FAP in broadcasting has been conducted by Idoumghar and Schott [52]. The authors present a distributed hybrid genetic algorithm and a cooperative distributed tabu search algorithm. They compare these algorithms with the sequential counterparts of their algorithms and with an ANTS algorithm. The benchmark instances they use were provided by the TDF-C2R Broadcasting and Wireless Research Centre.

[7] Centre Electronique de L'Armement
[8] European Cooperation on the Long Term Defense

3.8.5 Cellular Communication

Cellular communication[9] can be considered the main driving force behind research in the frequency assignment domain. As new standards are developed and used in 3G networks, in general an FAP still needs to be solved since these newer technologies still use GSM as their backbone architecture, as discussed in section 2.2.1. With new networks being deployed or current networks being expanded, standard GSM is used as it is cheaper than using the latest 3G technology. Therefore, standard GSM is still relevant and in use in modern networks.

There is a wealth of research that concentrates on the FAP within cellular networks. This is because cellular networks are used by millions of people around the world and as such this presents an interesting notion to produce better results since viable solutions have the possibility to impact millions of people. Most of the literature concentrates on this domain and one can find a lot of research in the literature presenting viable algorithms that produce real-world solutions [32].

Because the FAP problem is NP-Hard[10] most presented algorithms are either of the metaheuristic type or more recently of the swarm intelligence type. Both of these algorithmic types are discussed in chapters 4 and 5 respectively.

3.9 Summary

In this chapter a discussion was presented of the problem this dissertation is based upon. The problem was defined as being the frequency assignment problem (FAP) and is categorised as being part of the set of NP-Complete problems. The NP-Complete nature of the problem is an important concept to understand.

Within the FAP domain there are two different techniques when assigning frequencies. The two different techniques were discussed in section 3.3.

An important concept that needs to be understood to comprehend why the FAP exists is the concept of interference. This was discussed in depth in section 3.4.

[9] An overview of cellular communication technology called GSM is presented in Chapter 2.

[10] NP-Hard problems are the most difficult NP-Complete problems to solve [106]

The FAP is not just one problem but consists of various subproblems that have different goals for the resulting frequency plan. Some problems are concerned with the number of frequencies used, others are more concerned with the amount of interference that is generated on the network because of the assignment.

The various FAP subproblems were outlined and discussed in section 3.5.

A formal mathematical definition of the MI-FAP was also presented. Finally, the chapter concluded with an overview of the various benchmarks problems which are used to test the viability of frequency assignment algorithms.

Chapter 4

Metaheuristic Algorithms

4.1 Introduction

Metaheuristics is a subdomain of the artificial intelligence domain [106]. It evolved out of a need for more efficient search techniques with regard to hard problems.

Metaheuristics forms part of a collective body of algorithms that use heuristics to search a particular domain's problem space for the most optimal solution adhering to certain hard and soft constraints [106, 143].

A hard constraint is defined as a certain condition an algorithm or potential solution is not allowed to violate [3, 32, 106, 143]. A soft constraint is allowed to be violated but there is some sort of penalty or cost involved which is imposed onto the potential solution, which lowers its desirability [3, 32, 106, 143].

An optimal solution would therefore be any solution that violates no hard constraints and violates no or a minimum number of soft constraints [3, 32, 106, 143].

Some of the most famous algorithms that are classified as being part of the collective body of algorithms known as metaheuristic algorithms are:

- Tabu search
- Simulated annealing
- Genetic algorithm

The above-mentioned algorithms are not the only algorithms to form part of this subdomain, but they are the algorithms that have received the most attention in the literature and generally produce good results [79].

The main focus of this chapter will be each of the above algorithms. Before each of the algorithms is discussed, a brief overview is given of the various characteristics that metaheuristic algorithms exhibit.

4.2 Characteristics of Metaheuristics

NP-Complete[1] problems have been proven to not be solvable in polynomial time by traditional search methods such as A* search, Breath First Search and Depth-First Search [106].

These traditional search algorithms are concerned with the path taken from a starting solution to a final solution. Hence, the algorithms test each and every possible solution and store any alternatives in memory since a final solution constitutes the path taken from a start node to a final node [106].

It is not always viable to test every possible solution in a given problem search space, especially in NP-Complete problems, since their search spaces are usually huge or infinite. This is why traditional algorithms are not able to produce optimal solutions in polynomial time [106].

With NP-Complete problems, the path taken from an initial possible solution to a final optimal solution is irrelevant [106]. Only the final optimal solution is needed. Algorithms that disregard the path taken to a solution and that only care about the final solution are classified as *local search* algorithms [106].

Local search algorithms tend to use a small amount of memory since the algorithm is only concerned with the current state and only moves to a neighbouring state when searching for a potential solution. This makes local search algorithms a good fit for optimisation problems as an optimised solution is just a solution which constitutes to the current best state found by the algorithm [48, 106].

In its most basic form, a local search algorithm is just an algorithm that moves from one state to another. How it generates neighbouring states to

[1] A discussion of NP-Complete problems is appears in section 3.2

move to and how and why it moves to a certain state makes the local search algorithm unique [35, 48, 106].

Metaheuristic algorithms are considered to be *general-purpose* algorithms and can thus be applied to a wide variety of optimisation problems with only small modifications that need to made to the algorithm model [72].

As can be gathered from the name, a metaheuristic algorithm uses some sort of heuristic. A heuristic in its most basic form is a decision rule. Algorithms use heuristics to make decisions by applying the heuristic to the data the algorithm is currently working on [106, 143].

A heuristic is able to dictate what an algorithm must do for its next iteration by evaluating the current internal state of the algorithm, i.e. whether it should move to a different point in the solution space, generate new data, or select the current data as the most optimal solution [106, 143].

A metaheuristic differs slightly from a normal heuristic. In general "meta" means *beyond* or *higher level* [106, 143]. A metaheuristic therefore refers to a heuristic that is more complex with regard to the decisions it is able to make compared with a standard heuristic [106, 143].

In the literature algorithms are generally classified as being of the metaheuristic variant when they satisfy the following criteria [106, 143]:

- Use randomisation
- Use a local search
- Are stochastic[2]

Metaheuristic algorithms do not search the solution space statically by testing and evaluating every possible permutation in the solution space. This is inefficient and a normal path-based algorithm can be used instead [9].

Metaheuristic algorithms make use of certain strategies and heuristics (specific to the problem domain) to search the solution space intelligently through trial and error [9]. Intelligent searching is accomplished by combining the local search strategy along with a heuristic [35, 106, 143].

These algorithms iteratively move through the solution space, using a heuristic to guide the search to move to more desirable regions in the solution

[2]Exhibit behaviour that is non-deterministic.

space where there is a high probability of obtaining high quality candidate solutions [79, 86].

Metaheuristic-based search methods are not guaranteed to find the most optimal solutions in the solution space; instead these methods are usually used to find near-optimal solutions. Thus most algorithmic development in the metaheuristic domain focuses on developing new techniques that will increase the probability that a good solution will be obtained in difficult combinatorial problems [9].

Similarly, metaheuristics is not guaranteed to find suitable solutions or perform well in each problem domain it is applied to. The quality of the solution and performance of the metaheuristic is very much dependent upon on the expertise of the algorithm designer [141].

The standard metaheuristic algorithm will not take advantage of specific domain knowledge as the algorithm is designed to be general purpose. Therefore for the algorithm to exploit the search domain the algorithm designer needs to embed domain-specific knowledge [106, 141]. Otherwise the algorithm will search the solution space generally and will produce relatively poor results [106, 141].

Although heuristics plays a key role in the performance of metaheuristic algorithms, it is not the only factor that has an impact on performance and results. Algorithms also use techniques and concepts from other system paradigms like multi-agent systems [78].

In multi-agent systems, multiple agents have to communicate with each other and the system as a whole has to perform some sort of autonomous self-organisation [78].

These social and self-organisation concepts enable these systems to be distributed, robust and flexible. This is why in metaheuristic algorithms that are population-based, hybrid and/or distributed, these same concepts are used to better exploit the solution space [78].

In this section the characteristics of metaheuristics that set these algorithms apart from the conventional algorithms used on difficult problems was introduced. In the next section of this chapter the tabu search algorithm is discussed.

4.3 Tabu Search

Algorithm 1 Basic Tabu Search Algorithm [86, 100]

```
1: Initialize parameters
2: x_0 ← Initialize starting solution
3: while stopping criteria not met do
4:    y_i ← Determine x_i neighbourhood solutions
5:    Evaluate neighbouring solutions with fitness function f(y_i)
6:    z_i ←Select best neighbour from y_i
7:    if Move to z_i is Tabu then
8:       if z_i meets Aspiration Criterion then
9:          x_i ← z_i
10:      end if
11:   else
12:      Add x_i to Tabu List
13:      x_i ← z_i
14:      if x_i repeated ≥ max repeats then
15:         diversify()
16:      else
17:         intensify()
18:      end if
19:   end if
20: end while
21: Return x_i as best found solution
```

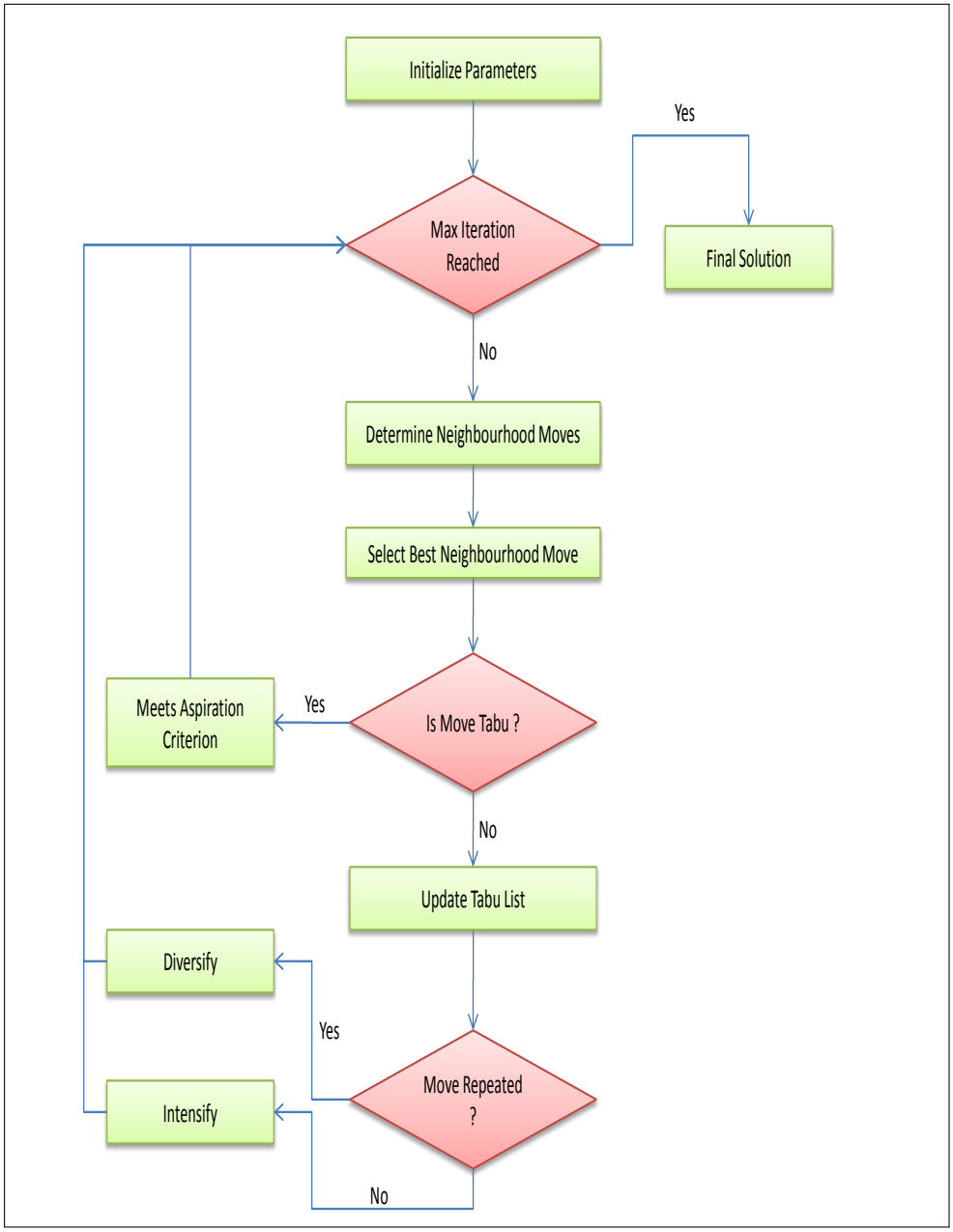

Figure 4.1: Flow chart for tabu search algorithm

4.3.1 Introduction

Tabu search (TS) was first proposed by Glover as a new searching technique to help algorithms avoid getting trapped in local optima in combinatorial and optimisation problems [100]. Since Glover introduced the algorithm in the 1980s, tabu search has been applied to a wide range of problems such as the vehicle routing problem [82], FAP [86], capacitated-lot sizing problem [43], Nurse Scheduling [28] and the Resource Constrained Assignment Problem [100].

Even though the problems mentioned differ by a large margin, the algorithm has been relatively successful in most optimisation problems it has been applied to. If one observes the results obtained in research [17, 43, 82, 86, 92, 93, 100, 116, 120, 134], it can be deduced that tabu search has on average obtained the best results compared with previous attempts with other algorithms.

Tabu search (TS) resembles in its most basic form the hill-climbing search algorithm [120]. The hill-climbing search algorithm starts from an initial solution and then iteratively moves from the current solution to a neighbouring solution [106]. Each neighbour is rated based on its attractiveness as a possible optimal solution that the algorithm is being applied to [106].

The hill-climbing algorithm moves to the neighbour with the highest rating without considering whether the neighbour might lead the algorithm astray, to a position where the neighbours are in fact *worse* than previously encountered possible solutions [106].

The TS algorithm addresses this shortcoming by introducing the concept of memory [120]. The memory of the algorithm is actually a history of previous solutions that the algorithm has moved to in its search for the problem space for a solution [120].

General search algorithms like hill-climbing, random-restart[3] or scatter search tend to get trapped in local optima [106]. The local optima might be a very attractive solution and thus general search algorithms will not move to better solutions since, according to the algorithm's built-in strategy, it has found the best solution.

[3]Random-restart is a search algorithm where once a certain trend of repeated moves is noticed, the algorithm restarts by generating a new initial solution to start from and then continues its search process from that generated solution [106].

In actual fact the solution that was found is the best solution in the *local* search space, but not in the *global* search space [35, 106]. Therefore an important characteristic of algorithms being applied to optimisation problems is breaking out of local optima [35, 106].

In an attempt to better understand the algorithm, the flow of the TS algorithm is set out below.

4.3.2 Flow of the algorithm

The general flow of the TS algorithm is described using algorithm 1 as a reference point.

Before the algorithm can actually start searching, it first needs to initialise various parameters. These parameters include, but are not limited to, the tabu list size, the aspiration criterion and the starting solution. The initialisation can be observed to occur from lines 1 - 2.

Once all the various parameters that are needed by the algorithm have been initialised, the algorithm is ready to enter the actual search phase, which ranges from lines 3 – 21.

The search phase starts off by first generating possible solutions that neighbour the current solution x_i as can be observed in line 4. Generating neighbouring solutions are a critical process in the TS algorithm as tehy are the means by which the algorithm is able to move from one possible solution to the next in the search space. Neighbourhood search will be discussed in detail in section 4.3.3.

After all the solutions that neighbour the current possible solution have been generated, the algorithm needs to decide which of the possible neighbours is the most lucrative. The algorithm therefore determines the fitness of each neighbour y_i by applying a fitness function $f(y_i)$.

Once all the neighbours have been evaluated, the algorithm selects the best neighbour that not only has the best fitness out of all the generated neighbours, but also has a better fitness than the current solution held by the algorithm. The best neighbour selection can be seen to occur in line 6.

The algorithm has now determined a possible neighbour z_i to move towards. Before moving on to the next iteration, it first needs to perform a series of checks that will aid it in the search process.

The first check that needs to be performed is whether the neighbour z_i is in the tabu list and this occurs in line 7. A solution is only in the tabu list if the algorithm has in a previous iteration had the particular solution as its current solution. Tabu lists will be discussed in section 4.3.3.

If the neighbour z_i is in the tabu list, then another check is performed where the aspiration criterion is calculated as can be seen in lines 8 – 10. The aspiration criterion determines whether the algorithm can make neighbour z_i its current solution once more even though it is tabu.

If the aspiration criterion has been met, the algorithm makes neighbour z_i its current solution x_i. More will be discussed on the aspiration criteria in section 4.3.3.

When a solution is not in the tabu list, it could possibly mean that it is the first time the algorithm has encountered the solution, or the solution has been previously encountered, but has been removed from the tabu list. A solution can be removed from the tabu list once it has been tabu for a certain number of iterations. More will be said on how the tabu list is updated in the next section.

In the algorithm, if a neighbour z_i is found not to be in the tabu list, the algorithm then adds the currently held solution x_i to the tabu list. The current solution is added to prohibit future movements to the same solution in an attempt to avoid cycling of solutions. After x_i has been added to the tabu list, the algorithm makes z_i the current solution x_i. This process can be observed from lines 12 – 14.

Before the algorithm continues to the next iteration, it performs one last final check. The purpose of this check is to determine whether the algorithm is repeating solutions. As can be observed from lines 15 – 19, the algorithm calculates whether the new selected solution has been repeated for a certain number of iterations.

If the solution has indeed been repeated for a predetermined number of iterations, the algorithm activates its diversification strategy or intensifies its search. These two strategies are discussed later.

4.3.3 Important Tabu Search Characteristics

Various characteristics are important to the TS algorithm. The first characteristic is exactly how the TS algorithm iteratively improves upon the initial

start solution.

Initial Solution Generation

The core feature of the TS algorithm is to sequentially improve an initial solution [146]. An initial possible solution is a point in the problem space where the TS algorithm will *start* exploring in search of a more optimal solution [106, 146].

An important consideration one has to make is how initial solutions are generated for the TS algorithm to start on [106, 146].

Random initial solutions might seem to be a good starting point, but by introducing randomisation it becomes hard to control the quality of the end solution [146]. Hence the generation of starting solutions must be controlled to limit the infeasibility of potential solutions [146].

Control of the randomly generation solutions can be achieved by simply constraining the random solution generator to only generate initial starting points in a bounded subset of the entire search space. For example: instead of letting the random initial starting point be any number between positive infinity and negative infinity, the random number generator is constrained to only generate numbers between 5 and -5.

Neighbourhood Search

TS uses a neighbourhood local search process to explore the solution space. There is no set process of how neighbourhood candidate solutions are selected. Depending on the problem to which the TS is applied, different neighbourhood solution selection strategies are needed. The overall quality of the solution produced by TS is also dependent on the neighbourhood search strategy used [146].

As can be observed from the TS flow chart in figure 4.1 the neighbourhood search phase is the first operation performed after the algorithm has been initialised, which is to say the algorithm has generated an initial starting solution from which the exploration process can start.

The neighbourhood search phase is the primary means for the TS algorithm to search the solution space for an optimal solution. It is within this phase that new possible solutions must be presented for the TS heuristic to allow the algorithm to decide to which solution it must move next.

The new possible solutions that are generated are called neighbouring solutions; hence the TS algorithm always moves to a neighbouring solution. When the TS algorithm moves to a neighbouring solution, the current solution is replaced by the neighbouring solution. Therefore, in the next iteration, neighbours for the new solution need to be generated.

Generation of new neighbours can range from a simple increment option to a complex operation that incorporates additional intelligence by means of a more heuristic approach to generate new neighbours.

The TS algorithm is not limited to just one neighbourhood search strategy. In the paper by Gopalakrishnan et al. [43] five neighbourhood move strategies are developed and are used interchangeably; in some cases a strategy is used three times in a row due to stagnation in the search space.

Stagnation occurs when the algorithm does not move to a better solution; instead it opts to stay on the current solution, as no neighbouring solution is better than the current one. However, to combat this stagnation, the authors opted to use all the move strategies 15 per cent of the time, and the last four move strategies for 85 per cent of the time when generating neighbourhood solutions.

Other neighbourhood strategies developed is that by N. A. Wassan [134]. In that author's paper a neighbourhood selection strategy is used that exchanges route nodes from initial vehicle routes for the vehicle routing problem. This route exchange enables the TS algorithm to search much more broadly due to the constant supply of different solutions.

Since initial solutions are constantly modified, it enables the TS procedure to be a very fined-grained process, because often a small change in a potential solution can have a big impact on the overall proposed solution by the TS algorithm.

In the research done by Zhang et. al. [146] an interesting neighbourhood selection scheme called *dynamic penalty* is developed. When the algorithm moves onto an infeasible solution a penalty is imposed. By dynamically changing the penalty that is imposed the "feasibility" of solutions produced is influenced.

Therefore, if and when the algorithm continually produces infeasible solutions, the penalty imposed is increased to guide the algorithm to produce more feasible solutions. Finally, when the algorithm gets trapped at lo-

cal optima, the penalty is reduced, which allows the algorithm to consider moving onto infeasible solutions thus escaping local optima.

Considering all the research done to develop new neighbourhood selection strategies that improve TS to search the solution space more efficiently and produce better, faster solutions, TS still has some drawbacks, especially with problems that have very large solution spaces [8].

TS is an iterative algorithm, executing a set of operations sequentially until a stopping criterion is met as can be seen in the flow diagram. At each iteration the algorithm has to determine feasibility of the immediate neighbourhood candidate solutions [8, 82].

Therefore each candidate must be evaluated by some function, which may be a costly operation in terms of computational cycles as well as in terms of time. This constant evaluation can drastically reduce the overall performance of the algorithm, since it is spending more time calculating feasibility than actually searching the solution space [8, 82].

Memory Structures of Tabu Search

The hill-climbing and random-restart algorithms are able to break out of local minima, but there is nothing stopping these algorithms from avoiding the local optima with their second or n-pass in the search space. TS addresses the shortcoming of these algorithms by incorporating an important concept: the notion of memory.

In its most basic form TS keeps a local memory of all its recent best moves, and puts them into a *tabu list* that has a predefined size. In the literature the Tabu list is also referred to as the *tabu tenure* [43,56,134,146]. The algorithm is not allowed to move to any solution that is in the tabu list unless a solution that is *tabu* is better than any current moves available in the immediate search neighbourhood [43, 56, 134, 146]. The process of overriding a solution's tabu status in the tabu tenure is called the *aspiration criterion* [43,56,134,146]. With the use of the tabu tenure and the aspiration criterion, the algorithm is able to avoid cycling, local optima as well as searching in a too narrow region [7, 93].

Research done by Ashish Sureka and Peter R. Wurman makes an important distinction with regard to the memory scheme that is used in the TS algorithm. Two memory schemes are discussed: *explicit memory* and

attribute-based memory [120, 121]. Of the two memory schemes the explicit memory scheme is the most used in the literature [82].

With explicit memory the algorithm stores a complete solution in the tabu tenure; hence the algorithm is prohibited from moving to that position in the solution for as long as the solution is in the Tabu tenure [120, 121]. With attribute-based memory the algorithm stores the *operation* used to move from the previous solution to the current solution [120,121]. Therefore with attribute-based memory the tabu tenure intended function is changed from prohibiting certain solutions already encountered to rather prohibiting making changes to the current solution that would lead to solutions already present in the tabu tenure [120, 121].

In research conducted by D. M. Jaeggi, G.T. Parks, T. Kipouros and P. J. Clarkson [54], the authors add two additional memory structures called *medium term memory* (MTM) and *long term memory* (LTM) besides the standard *short term memory* (STM), typically referred to as the tabu List [54]. Each additional structure remembers a different set of solutions for use by the diversification and intensification phases in the algorithm.

STM is similar to the traditional tabu list: to store the most recent solutions produced by the algorithm. MTM is designed to remember optimal or near-optimal solutions. These solutions are therefore used later in the intensification phase. Finally, the LTM structure stores all the regions that the algorithm has already explored and is thus used in the diversification phase of the algorithm [54].

Search Phases

As TS searches through the solution space, it goes through two cycles of search phases called *diversification* and *intensification* [17, 37, 48, 56].

The diversification phase in the TS algorithm is the phase where the algorithm is directed to areas in the solution space that has not yet been explored. The algorithm usually applies diversification as mechanisms monitoring the memory; note that solutions being produced are being repeated [37, 134].

Fescioglu-Unver and Kokar [37] researched a strategy is presented that consists of two components namely the *observer* and the *diversifier*. The goal of the observer is to continually monitor the best solution obtained by

the algorithm as to whether it violates the *stagnation period*. The stagnation period is defined as the number of iterations where the current best obtained solution has not changed [37].

As soon as the algorithm exceeds the stagnation period the observer component activates and transfers the necessary information needed by the diversifier component. The diversifier component dynamically changes the size of the tabu tenure based on the information the observer gathered. The diversifier mainly targets older moves to diversify, but for short bursts of time it decreases the tabu list size to a very small value in an attempt to combine new and old moves [37].

The specific mechanism used to define a new position where the algorithm can continue to search, should ideally select areas in the solution space that have not been explored yet. Therefore, the diversification phase typically makes extensive use of the knowledge present in the long-term memory structures as an indication of what areas of the solution space have been previously explored and which areas have not [17, 37, 48, 56].

Intensification is usually the first phase of the TS algorithm, since it is responsible for building up a history in memory on which the diversification phase can act. Fescioglu-Unver and Kokar also present an intensification strategy based on control theory in their research [37]. The authors identify repetition length as a critical value for their intensification strategy to be based upon. The repetition length is a control measure that defines how many times a solution may be repeated.

In the following section, an overview will be presented of literature that used the TS algorithm on the FAP.

4.3.4 Tabu Search on the FAP

As discussed, the TS algorithm is an optimisation algorithm and has been applied to a wide variety of optimisation problems (see page 61). In the literature the TS algorithm has also achieved relatively good results.

In a study conducted by Robert Montemanni and Derek Smith [86] the TS algorithm is used on the FS-FAP. Since the TS algorithm is generic, the authors had to make some alterations to the algorithm to suit their needs as well as to make the algorithm more efficient in exploring in the FAP solution space.

Problem instance	TS with HMT	Best COST 259
Siemens1	2.7692	2.200
Siemens2	14.9360	14.280
Siemens3	6.6496	5.19
Siemens4	110.9725	81.89

Table 4.1: Results of applying TS with HMT on COST 259

The TS algorithm used by the authors is the multistart TS algorithm, which randomly starts on different initial solutions [86].

The authors developed a technique called heuristic manipulation technique (HMT). HMT first monitors an underlying heuristic being used on the problem by the algorithm [86]. It then identifies certain characteristics that good solutions exhibit. In the FAP, it is transmitters that are assigned different frequencies which results in an overall lower interference value [86].

The HMT then uses the identified characteristics to add *additional* constraints to the problem [86]. By adding constraints, the search space is reduced, which therefore makes it easier for the algorithm to find an optimal solution, as there are fewer solutions to consider. However, by reducing the search space, other near-optimal solutions which might be far better are excluded [86]. It is for this reason that the authors opted not to add the constraints permanently, instead the constraints are replaced by other constraints [86].

The authors applied their TS algorithm together with HMT to the COST 259 family of benchmarks, specifically the Siemens1, Siemens2, Siemens3 and Siemens4 problems. As can be observed from the results obtained by the authors, the TS algorithm with HMT produces results that rank very favourably against other algorithms also applied to the COST 259.

When critically evaluating the TS algorithm with regard to applying it to the FAP, the following disadvantages can be identified:

Search based on a single solution — The TS algorithm at any moment in time only searches in the vicinity of *one* current solution for possible neighbours that might be the current solution for the next iteration. FAPs typically have huge search spaces due to their NP-Hard nature. Therefore, only searching for possible lucrative neighbours from only potential solutions

seems to be terribly inefficient. A better strategy would be to use the notion of population-based algorithms and have multiple solutions from which lucrative neighbours are searched.

Neighbourhood generation — The TS algorithm defines no set process for generating a neighbouring solution given a starting solution. Generating neighbours from a solution is a critical process in the TS algorithm, for it is the only means by which the algorithm considers other solutions, i.e. it is the mechanism by which the algorithm searches. Generating a new neighbour can be as simple as changing only one value from the current solution or it can be very complex and incorporate other algorithms together with mathematics formulae. Regardless of the complexity of the neighbour generation that is used, care must be taken to ensure that the algorithm is able to produce a wide diversity of neighbours and is also able to intensify on the most optimal solution.

Tabu lifetime — The TS algorithm only operates on a single solution at a time and at most only considers one potential neighbour as its next possible current solution. Therefore a difficult choice needs to be made as to how long a solution stays tabu. In the FAP a solution might be entered into the tabu list early in the search process of the algorithm. A large majority of the neighbours of this solution are vastly superior solutions compared with any of the current solutions produced by the algorithm. Due to the solution with these neighbours being in the tabu list, these neighbours will not be reconsidered until much later when the solution is removed from the list. The only other option is if the aspiration criterion is met, which is another parameter which needs to be fine-tuned. A high aspiration criterion and the algorithm might be too eager to select just any solution even though it is Tabu. A low aspiration criterion and the algorithm will be too strict in selecting a tabu solution.

In the next section the simulated annealing algorithm is discussed.

4.4 Simulated Annealing

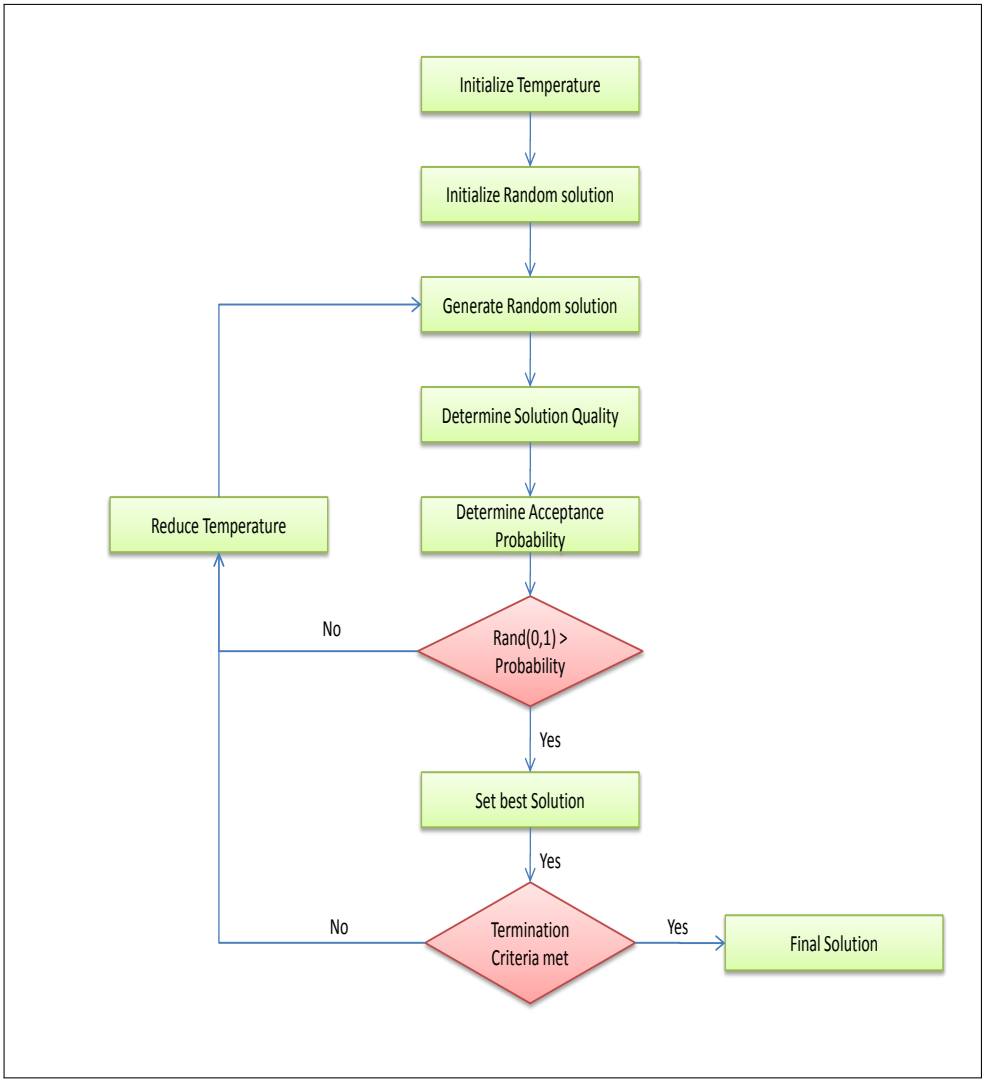

Figure 4.2: Flow chart for simulated annealing algorithm

Algorithm 2 Basic Simulated Annealing Algorithm [90, 91]

```
Initialize parameters
Set starting temperature T(0)
x_0 ← Generate initial starting solution
while Stopping criteria not met do
    y_i ← Generate neighbouring solutions to x_i
    Evaluate y_i neighbours with fitness function f(y_i)
    Calculate probability p_i of y_i neighbours with equation 4.1
    x_i ← Select y_i neighbour based on probability p_i
    Reduce temperature T(i) based on cooling schedule
end while
Return best solution x_i
```

4.4.1 Introduction

Simulated annealing (SA) is a heuristic search technique proposed in the 1980s by Kirkpatrick to solve combinatorial optimisation problems. The technique is based on a natural process which is known in metallurgy as annealing [65, 91, 119, 129]. Kirkpatrick was the first to use SA to solve optimisations problems but the basic algorithm structure was defined by Metropolis et al. in 1953 [90, 129].

Annealing is the natural process of crystallisation when a solid is heated to a high temperature and then systematically cooled to a lower temperature to reach a crystallised form [6, 68, 80, 129]. This crystallised form of the solid is known to be the global minimum of the solid's internal energy state.

When the solid is rapidly cooled from a high temperature, the molecules have no time to reach a thermodynamic equilibrium stage [6, 68, 80, 129]. Therefore the molecules of the solid have high energy and the resultant structure has no real crystalline form; thus the solid energy is at a local minimum [68, 80, 129]. When the solid is slowly cooled in a controlled manner, the molecules are able to reach a thermal equilibrium at each temperature [6, 68, 80, 91, 129].

In the algorithm the energy state is the *cost function* that needs to be minimised, and the molecules are the *variables*, which represent the solutions, and thus their state needs to be optimised to reach the desired energy state.

The following equation is the standard probability function that is used to determine when an uphill move is performed by the algorithm. This function is known in the literature as the *metropolis criterion*.

$$M_{AC} = \begin{cases} 1, & \text{if } f(y) \leq f(x) \\ exp(-\frac{\Delta E}{T_k}), & \text{otherwise} \end{cases} \tag{4.1}$$

The function f is the objective function or a function that determines the state of a given position in solution space [140]. The parameter T_k is the temperature of the algorithm at iteration k [140]. Finally, ΔE is the change in "energy" between two solutions x and y [140].

The main purpose of the SA algorithm (like most optimisation algorithms) is to minimise or maximise the cost function [119]. This cost function typically evaluates a solution desirability compared with other solutions in the immediate *neighbourhood* of the algorithm's current position [23].

The immediate neighbourhood of solutions is generated based on some heuristic implemented by the algorithm designer [106]. This heuristic, as with the TS algorithm, can be simple or complex. The neighbourhood generated is the first step right after the starting solution has been generated by the algorithm, as can be observed from figure 4.2.

Typically a neighbouring solution is only selected as the new best state if its desirability ranks higher than the current solution. When the algorithm moves to a better solution from the previous solution, the move is typically referred in the literature as a *downhill* move [129].

The best state is not always selected; in some cases the algorithm is also able to move to solutions that are worse than the current solution. A worse solution is only selected based on some probability which is controlled by the *annealing temperature* of the algorithm [23].

At a high annealing temperature the probability that the algorithm will select a bad solution is very good. As the annealing temperature decreases so does the probability that a bad solution will be selected [129]. When the algorithm moves to a worse solution, the move is typically referred to in the literature as an *uphill* move [129]. Uphill moves allow the algorithm to break out of local minima and can lead the algorithm down a different path, which may ultimately result in obtaining the global optimum [119].

The SA algorithm is also very popular due to its basic structure being generic [99]. As with the TS algorithm, the standard SA algorithm does not

define a set neighbourhood generation mechanism; instead it is up to the algorithm designer to implement a suitable generation mechanism that will allow the algorithm to adequately explore the problem space [99].

Applying the algorithm to other problems requires slight changes. These changes usually need to be applied to the *neighbourhood selection* scheme and the *cooling schedule* [99, 127]. Both of these concepts will be discussed below.

4.4.2 Flow of the Algorithm

In an attempt to better understand how the SA algorithm operates, a general discussion on the flow of the algorithm will now be given using algorithm 2 as a reference point.

From lines 1 – 3, the SA algorithm is initialised. The most important step here is setting the starting temperature for the annealing process to start. As mentioned in the introduction, the temperature of the annealing process plays a critical role in the potential solution selection process.

After the algorithm has been initialised the search phase of the algorithm starts which ranges from lines 4 – 10. Like the TS algorithm, the SA algorithm starts the search phase by generating a number of neighbours to the current solution held by the algorithm as can be observed in line 5.

Before selecting a neighbour the algorithm first needs to evaluate the generated neighbours. It evaluates each neighbour by applying a fitness function $f(y_i)$ in order to determine its fitness. Once the fitness of all the generated neighbours has been determined, the algorithm uses equation 4.1 to calculate the probability of selecting a particular neighbour for all the generated neighbours as well. The probability calculation can be observed to occur in line 7. The algorithm then selects the neighbour with the highest probability to be the current solution, as observed in line 9.

Before the algorithm advances to the next iteration the temperature needs to be lowered. The temperature is a critical variable as it has a role in the probability equation and therefore can affect solution selection. The temperature is not simply lowered with a subtraction operation, but rather according to a particular cooling schedule. In the algorithm the process of lowering the temperature occurs in line 10.

The characteristics that were briefly mentioned in the introduction section will now be discussed in more detail.

4.4.3 Important Simulated Annealing Characteristics

There are four characteristics of the SA algorithm that make the algorithm unique. One of the most important is the cooling schedule.

Cooling Schedule

The cooling schedule/annealing Schedule is the most defining characteristic of the SA algorithm. It is the procedure where the natural annealing process is mimicked. The temperature of the SA algorithm is a control parameter that defines how much the algorithm moves around in the solution space.

As can be observed from figure 4.2, after each iteration, whether the algorithm has selected a new best solution or not, the temperature is reduced by a certain amount. This amount is determined by the cooling schedule.

In general, when the SA algorithm temperature has a very high value most solutions that are produced from the neighbourhood are accepted [73]. Thus the algorithm moves freely in the solution space with little constraint. As the temperature decreases, the probability that the algorithm will select a bad or just any solution decreases. When the temperature is very low, the SA algorithm is similar to a greedy algorithm in the sense that it only accepts downhill movements [73].

In the literature there are three annealing schedules in common use, namely *the logarithmic schedule*, the *geometric schedule* and the *Cauchy schedule* [90, 119].

The standard and most commonly used schedule is known as the logarithmic schedule and is based on Boltzmann annealing [90]. The main disadvantage of this schedule is that is slow due to its logarithmic nature [90]. It also requires moves to be generated from a Gaussian distribution for it to be able to reach the global minimum [119]. The logarithmic annealing function has the following form:

$$T_k = \frac{T_0}{ln(k)}, \text{where k is the iteration value} \tag{4.2}$$

Where T_k is the temperature at iteration k.

The Cauchy schedule is faster than the logarithmic schedule. Similar to the logarithmic, this schedule also has a movement requirement. Moves must be generated from a Cauchy distribution for the algorithm to be able to reach the global minimum [90,119]. The Cauchy schedule is also typically referred to as fast annealing [90]. The schedule has the following form:

$$T_k = \frac{T_0}{k} \tag{4.3}$$

Finally, the fastest annealing schedule is known as the geometric or exponential annealing schedule [119]. This schedule introduces the concept of *re-annealing*. Re-annealing is a procedure by which all SA temperatures are rescaled [90]. This schedule has no move generation requirement to reach the global minimum, since there is no regorous proof in the literature [119]. The geometric schedule has the following form:

$$T(k) = T_0 exp(-C_k), \text{where C is a constant} \tag{4.4}$$

Initial Temperature

The initial temperature is a very important parameter to define in the SA algorithm, since it defines a point from which the cooling schedule will start. Therefore, depending on what the initial value of the temperature, is the final result that the algorithm will produce can be influenced [99, 114, 136].

When the initial temperature is set to a very high value, the algorithm takes a long time to reach a result. On the other hand, if the initial temperature is set to a very low temperature, the algorithm might converge too quickly and thus produce a result which may be the local minima [99, 114, 136].

The initial temperature together with the cooling factor allows the algorithm designer to define the time window for the algorithm to escape local minima, as well as the rate of convergence to an optimum solution [99, 136].

A low initial temperature together with a low cooling factor makes the time window for the algorithm to leave a local optimum very small [136]. With a high initial temperature and cooling factor value that is almost 1, the time window for the algorithm to leave the local optimum is much larger [136].

When the algorithm is near a global optimum, a low initial temperature and low cooling factor will allow the algorithm to reach the optimum faster

in the solution space. In contrast, if a high temperature and a very low cooling factor are used, the algorithm will take longer to reach the optimum even though it is near the global optimum [136].

Move Generation

Most of the research done on the SA algorithm focuses on the annealing schedule and not so much on the move/solution/neighbourhood generation. Typically an initial solution is generated and then small changes are made to the solution to represent a new solution. The solution is said to be perturbed to the next solution.

Move generation is the phase where neighbouring solutions to the current solution are generated. It is the ideal section for an algorithm designer to embed domain-specific knowledge, to generate attractive solutions, which the algorithm can consider moving to.

In research done by Tseung and Lin [129] an initial solution is not modified, but a move generation technique known as *pattern* search is used. Pattern search has two forms of movement, namely the exploratory move and the patten move. The exploratory move continually changes the certain variables of a solution [129]. This is done so that the technique can rapidly find and identify a "downhill" move. The pattern move uses the information gathered by the exploratory move to move towards the minimum of the function [129].

Algorithm Efficiency

The algorithm is also efficient with regard to CPU cycles when compared with the genetic algorithm because it only has to evaluate a certain number of moves each iteration, instead of a whole population each iteration. Unlike TS, the basic SA algorithm does not keep any memory and is therefore memory efficient, but in contrast suffers the risk that the solution will cycle. The more iterations spent at a temperature, the longer the algorithm spends at a certain temperature and therefore the higher the probability that solutions will cycle.

4.4.4 Simulated Annealing on the FAP

The SA algorithm, as with the TS algorithm, has achieved relatively good results in other optimisation problems, as mentioned in section 4.4.1. Due to its success on other NP-Complete optimisation problems, the SA algorithm has also been applied to the FAP as well.

In literature by Carlo Mannino and Gianpaolo Oriolo [75] the SA algorithm is applied to the FAP. The authors utilise *dynamic programming* together with the SA algorithm to alter the SA algorithm in such a way as to better explore the FAP solution domain.

Dynamic programming is a computer science technique that is usually applied to problems with big search spaces that are therefore difficult to solve [106]. The technique decomposes the problem into smaller problems that are easier to solve since the smaller problems have smaller solution spaces to explore [106, 139].

Utilising dynamic programming, the authors decompose the FAP into smaller interval graphs. They are able to do this since the FAP is similar to the graph problem, which means an FAP can be modelled as a graph [22]. The smaller interval graphs are graphs of vertices that are closely related and form a clique [22].

The resulting SA algorithm was benchmarked on the COST 259 Siemens benchmark instances. The results will now be presented. Note that at the time the above authors did the benchmark the best results obtained for the siemens problem were different from the latest results.

Cooling Schedule — Depending on the cooling schedule selected, the algorithm might converge too quickly. As discussed previously the cooling schedule reduces the temperature and the temperature plays a large part in the determination of whether a particular solution will be moved to or not in an iteration. Thus early on the algorithm will explore a lot more and later on will exploit more. In the FAP, the algorithm must not only be able to explore and exploit, but also be able to return to an exploration phase if need be. As the temperature gets colder the SA algorithm exploits more and therefore will not easily move to a worse off solution. In the FAP, it might be desirable to rather move a worse off solution later on, as the particular current solution is a local minimum and yields bad neighbours as potential

Problem instance	SA	COST 259 (old)	COST 259 (new)
Siemens1	22.96	23.00	2.200
Siemens2	14.72	14.75	14.280
Siemens3	52.43	52.55	5.19
Siemens4	80.96	80.80	81.89

Table 4.2: SA on COST 259 Benchmark

next solutions. With the cooling schedule this is simply not possible, unless the temperature and schedule are reset. Resetting the temperature and schedule is not ideal, since the algorithm keeps no history and might risk making the same faults as before the reset.

Neighbourhood generation — The SA algorithm, as with the TS algorithm, has no set process that defines how neighbours should be generated. As discussed, neighbourhood generation is the primary means by which the SA algorithm moves about the search space in search of an optimal solution. Therefore applying the SA algorithm would require a custom neighbourhood generation scheme, which would be difficult to control as the algorithm keeps no history and also does not really expose the temperature for use with neighbour generation. This control and history, as the FAP search space, is huge and a desirable quality would be to know if the algorithm is moving towards an area that has already been explored.

Single solution based search — The SA algorithm is similar to the TS algorithm in the sense that it only searches from one solution per iteration. It searches by generating neighbours around the current solution of the algorithm. The FAP search space is huge; hence it would be more efficient to have multiple current solutions from which neighbours are generated. This enables the algorithm to explore the search space much more efficiently at the expense of more computational resources.

4.5 Genetic Algorithm

Algorithm 3 Basic Genetic Algorithm Algorithm [39, 64, 126, 130]

```
1: pop_n ←Initialize population
2: Evaluate population with fitness function f(i)
3: while Stopping criteria is not met do
4:    y_k ← Select best performing genes from population
5:    Remove other genes from population
6:    repeat
7:       for Each gene g_i in y_{k−1} do
8:          z_i ← Perform crossover with g_i and g_{i+1}
9:          m_i ← Calculate Mutation probability for z_i
10:         if m_i ≥ Mutation threshold then
11:            Perform mutation on gene z_i
12:         end if
13:         Add gene z_i to new_pop
14:      end for
15:   until size(new_pop) = size(pop_n)
16:   pop_n ← new_pop
17: end while
18: x_i ← Determine best gene in pop_n
19: Return best solution x_i
```

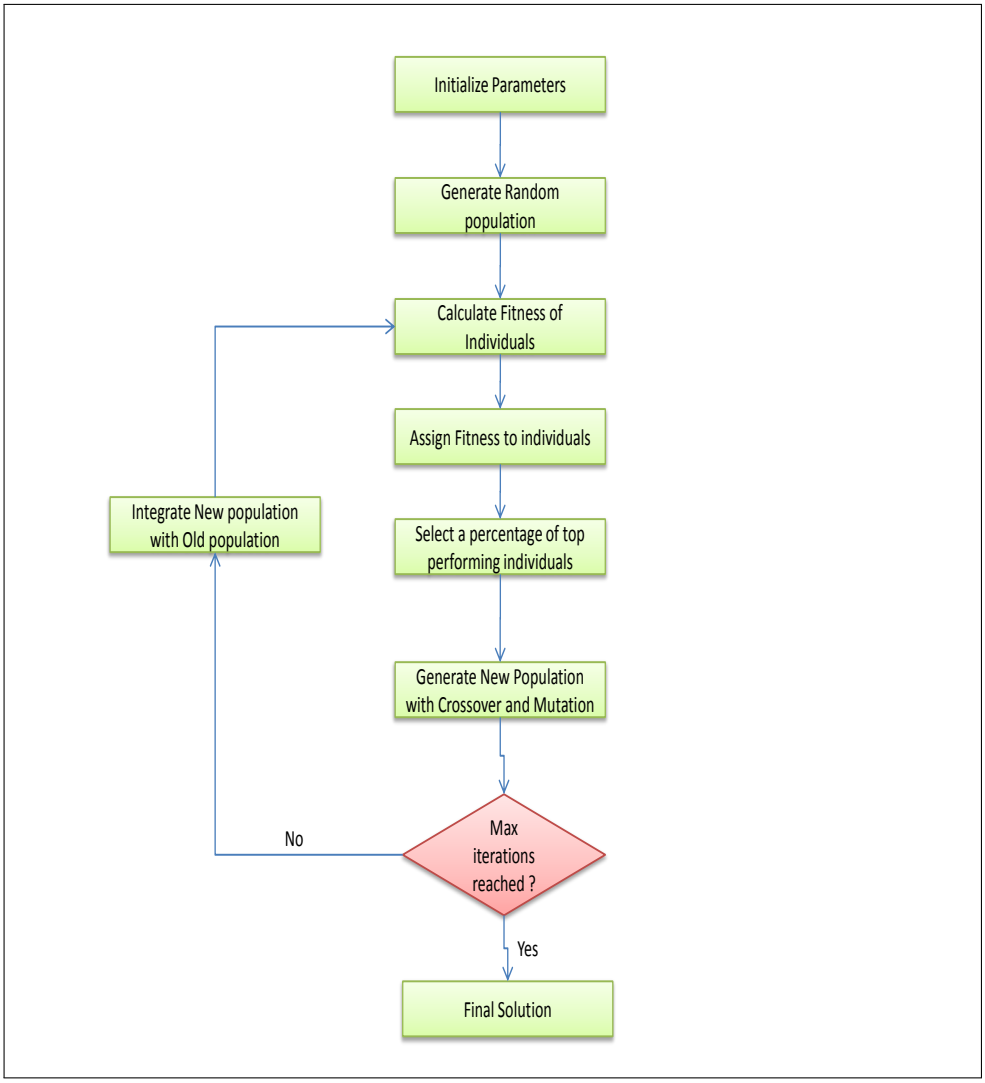

Figure 4.3: Flow chart for genetic algorithm

4.5.1 Introduction

The genetic algorithm (GA) is a stochastic search method that is based on the natural process of genetic evolution and the Darwinian concept of "survival of the fittest" [39, 46, 64, 130]. The GA was initially developed for adaptive systems by Holland, but has been widely used in the optimisation field of study due to its effective exploration of the solution space as well as its relative success in multidimensional problems [39, 130, 132].

The wide use of the GA can also be attributed to its generic algorithm structure as well as the ease of implementation [64, 130]. GA is application dependent and thus the designer needs to tailor the algorithm to their needs to obtain good results [46].

The GA is generic since it defines no specific crossover, selector and mutation operator. It defines that these operators exist, but exactly how they perform their respective operations is up to the algorithm designer.

The GA search procedure involves searching the solution space through artificial evolution and natural selection [55, 112, 130]. An individual or point in the solution space is known as a *chromosome* [18]. An initial set of chromosomes (referred to in the research as the *population*) is randomly generated [46, 55, 112, 130].

Unlike other algorithms, the GA does not concentrate on one point when searching the solution space, but rather on a wide range of points represented by the population [39, 55, 130]. Each chromosome in the solution space represents a string encoding of the problem parameters [130]. Encoding problem parameters also attributes to the wide use of the GA since difficult mathematical problems can now be easily modelled [46].

The population is artificially evolved each iteration by using a set of stochastic operators [126]. This set consists of a selection operator, crossover operator and mutation operator [112,126]. Each operator plays an important role in emulating the evolutionary process.

The selection operator is in charge of applying the *objective function* to each chromosome in the population [18, 64]. Depending on whether the selection operator is set up for maximisation or minimisation, each individual is ranked based on its "fitness" or objective function value.

In figure 4.3 the selection operator is the first operation applied after a random starting population has been generated. It is also the first operation

applied when the algorithm starts a new generation.

In accordance with Darwinian theory, only the fittest individuals are selected from the population [18]. Based on the algorithm flow in figure 4.3, the fittest individuals are only selected once the whole population has been assigned its respective fitness.

The fittest individuals are copied and sent to the next phase of the algorithm, which is known as the *reproduction* phase [18]. This phase can be observed in figure 4.3 as the operation where the new and old populations are integrated.

Depending on how complicated the objective function is and how large the population is, the selection phase may be the most computationally expensive as well as time consuming [46]. The reproduction phase mostly consists of string manipulations on the chromosome, to drive to search forward and it is thus a phase which is completed quickly, especially if the string encoding is of a binary nature [46, 64]. These string manipulations occur through the application of the crossover and mutation operators [103].

The reproduction phase is where a new population is generated for the next generation to be evaluated by the selection operator. The reproduction is said to generate "offspring" from the selected fittest population which in turn are known as the "parents" of the offspring [18, 103]. Reproduction will occur until a certain number of predefined generations is reached or a suitable solution is found to be greater/less than a certain fitness threshold that would indicate a good chromosome [98].

There are two basic forms of the GA and both forms differ in the way that offspring and parents are handled [130]. One form is called the *generational* GA and the other form is known as the *steady-state* GA [130, 135].

With the generational GA the offspring are not immediately used in the next generation; instead they are kept in a pool until the pool reaches a certain size [130]. The offspring are then used to replace the parents entirely in the next generation [130]. In the steady-state GA the offspring are continually integrated with the population; thus offspring and parents occupy the same population pool every generation [130, 135].

The GA search process moves around in the search space using probabilistic rules rather than deterministic rules [130]. The probabilistic transition rules aid the algorithm to avoid local optima regions in the solution space [55].

Some sequential search algorithms require that the objective function be differentiable [103]. These sequential algorithms use the derivative to obtain gradient information so that they can move in the solution space [103, 126]. The derivative of the objective function may be used to increase the efficiency of the GA, but is by no means a core requirement of the algorithm [55, 103, 126]. The GA makes no assumptions about the solution space and primarily works on the information provided by the objective function [55, 103].

4.5.2 Flow of the Algorithm

Most of the core concepts of the genetic algorithm were introduced in the previous section. To better understand the algorithm, a general overview of the algorithm will now be presented using algorithm 3 as a reference point.

The GA algorithm is a population-based algorithm and therefore needs to initialise its population. Each individual of the population represents a potential solution. Population initialisation occurs in line 1 in algorithm 3.

Before the algorithm can start *evolving* its population, it first needs to determine each individual in the population's fitness. The fitness of an individual is calculated using a fitness function $f(i)$.

Since the initial population has been evaluated, the algorithm is now able to start its searching process, which starts at line 3 and ends at line 17.

The first part of the search process that is performed by the algorithm is to select a certain subset of the population. Typically this subset is the individuals of the population that have the highest fitness values. High fitness individuals are preferred as the fitness is an indication of good genes, i.e. the solution the individual represents is of a high quality. This subset selection can be observed to occur in line 4.

All the individuals that are not in the subset are removed from the population. Once the subset selection has been completed, the algorithm is ready to enter the reproduction phase, which ranges from lines 6 – 15.

The first step in the reproduction phase is where various individuals of the subset are mated together to produce a new offspring individual. Mating occurs through the use of the crossover operator. The purpose of the crossover operator is to take certain characteristics of the two mating individuals and combine them to form a new individual referred to as the

offspring. Generation of new individuals with the crossover operator can be observed from lines 7 – 8

The second step of the reproduction phase is where mutation occurs. For each of the offspring generated by the crossover a mutation probability is calculated. If the calculated probability for a particular offspring is high enough, the algorithm enters the mutation phase. In the mutation phase, the offspring is mutated, i.e. the solution represented by the individual is altered slightly to be different. Mutation can be seen to occur in lines 9 – 10.

Regardless of whether the offspring has been mutated or not, the resulting offspring is added to the new population. The reproduction phase continually loops, until the new population equals the size of the initial starting population. Once the new population has reached sufficient size, the algorithm moves on to its next iteration.

The algorithm continually generates and evaluates new population until a predetermined stopping criterion has been met. Once the criterion has been met, the algorithm selects the individual with the highest fitness in the current population as its most optimal solution.

4.5.3 Important Genetic Algorithm Characteristics

Certain characteristics make the GA search procedure unique. The first is the initial population generation.

Initial Population Generation

Initial population generation is the very first activity that the GA performs. Out of this population potential mating candidates are selected based on their fitness, which indicates desirability. Generally the initial population is generated by means of randomisation [126]. Since the algorithm searches multiple points simultaneously in the solution space, it is desirable that the initial population have a wide diversity with regard to the solution it represents [39, 89]. By controlling the initial population generation we can control, to a small degree, the amount of exploration the algorithm does initially as well as avoid premature convergence [89]. Therefore care must be taken in the selection of the particular randomisation scheme that will be used to generate solutions.

In a survey done by Andrea Reese [102], two randomisation schemes which defined, namely pseudo-random number generators (PRNGs) and quasi-random number generators (QRNGs). PRNGs were found to be heavily problem dependent, improving the search efficiency in some instances and in other instances having no considerable impact. QRNGs, on the other hand, were shown to significantly improve the final solution produced by the GA as well as lower the number of generations for the solution to be obtained [102].

Not all GAs use randomisation entirely for their initial population generation. In research done by Amit Nagar, Sunderesh S. Heragu and Jorge Haddock [89] an algorithm is presented that generates an initial population through the aid of a branch-and-bound algorithm. The branch-and-bound algorithm provides the GA with an upper bound of acceptable solution in the solution space. The initial population is then randomly generated in the constrained space defined by the upper bound [89].

Selection Operator

The selection operator is the first operator to be applied to the population after each generation. This operator is in charge of evaluating the current population to determine which individuals will survive and which will be terminated [89, 103, 108]. The individuals who survive and who thus have the highest fitness are moved to a "mating pool" [18]. Individuals from this mating pool will be used in the reproduction phase to generate a new population [46, 64].

By favouring high fitness individuals above low fitness individuals the operator guides the search towards better high quality solutions [103]. Care must be taken if the operator is too eager regarding high quality individuals since this may eliminate diversity in the population and thus result in premature convergence for certain problems [103]. If the solution space is known to have only one optimum, then a strict selection policy may be used, thus directing the search into a gradient-based direction [103]. In contrast, with a solution space that is known to have multiple optima, a forgiving selection policy might be more favourable since it allows the solution space to be more widely explored [103].

The most widely adopted selection scheme is known as the *roulette wheel* selection scheme [103, 108, 135, 145]. With this scheme an individual is se-

lected based on a probability defined by the fitness of the individual divided by the collective fitness of the population [135].

Crossover Operator

The crossover operator is usually the first operator applied to the population in the reproduction phase. The crossover operates exclusively on the chromosomes in the mating pool. This operator is the main process by which by the GA is able to diversify as well as exploit certain optimal regions [89,108].

Crossover works by interchanging and matching two parent chromosomes randomly selected from the mating pool to produce a single chromosome known as the offspring [18, 108, 130]. Since two chromosomes are combined or partially changed, some historical information is retained in the new chromosome [130].

In some algorithms, like for instance the one presented by Nagar et al., before the crossover operator is applied to the two parent chromosomes, the parents are first evaluated to determine if they represent suboptimal regions [89].

If either of the parents is from a suboptimal region, a disruption operator is applied that interchanges certain domain-specific information between the parents. After the disruption operator is applied, the crossover operator is applied [89].

There are a variety of ways in which values are interchanged between chromosomes in the crossover operation i.e. fixed point crossover, two point crossover, uniform crossover and gaussian crossover. Fixed point crossover operates on binary parents whereby a point is selected in one parent and then all other bits are replaced by the other parents' bits [18].

Two point crossover generates two random indices which dictate a certain segment in the one parent to be interchanged with the other parent [103].

Uniform crossover is the most basic of all crossovers since it randomly selects bits from one parent to be replaced by another parent's bits and is usually used when a large solution space must be search [132, 135].

Finally, the Gaussian crossover interchanges bits between parents based on a Gaussian distribution [132, 135]. Depending on the state of the algorithm, crossover operators can also be interchanged or even paired if

the algorithm needs better search performance for large or small solution spaces [5, 132].

Note, in all the above crossovers it is assumed that the chromosomes are bit encoded, but these crossovers do require them to be. All these crossover operators are able to work on any encoding; it just depends on what is considered to be a "bit" if non-binary encoding is used.

Mutation Operator

The mutation operator is a probabilistic operator, which means it is applied infrequently and thus it will be applied to the parent solution only with a certain probability. The operator typically changes some small part in the solution regardless of the fitness of the chromosome [18, 145].

Given enough time, the mutation operator enables the algorithm to search the entire search space [130]. The operator also aids the algorithm with regard to escaping local optima in the solution space [130].

Due to the way the crossover works, some information may be lost when it is replaced by other chromosomes' bits [46, 103]. Mutation is a source of new information that is continuously inserted into the algorithm; hence it works against information loss [46, 103, 108].

Usually, the mutation operator has no previous information on the chromosome it is mutating; thus it is entirely possible that the mutation may modify the chromosome for the worse [46]. A worse solution might lead the algorithm out of local optima or lead it down a new path to find the global optima, but this is not always the case and thus the probability of the mutation operator is set to be very low [64, 103, 130].

In a survey done by Engelbrecht [35] another mutation operator is discussed. Instead of mutating a small part of randomly selected chromosomes, this operator generates new offspring to be inserted back into the population. The operator randomly generates a new chromosome and then uses0 any of the previously discussed crossover operators (see page 87).

The mutation operator isn't always simple random operations. In research done by Il-kwon Jeong and Ju-jang Lee [64], a mutation operator is presented that incorporates the SA algorithm. The SA mutation operator generates a new chromosome whose fitness is also calculated. If the new chromosome fitness is worse than the chromosome to be mutated, then

depending on the SA mutation temperature as well cooling schedule, the newly generated chromosome might replace the chromosome to be mutated; otherwise if the new chromosome has a better fitness than the chromosome to be mutated, it is replaced [64].

Elitist Operator

The elitist operator differs from the crossover and mutation operator in that it does not modify the chromosomes in any way. Instead, the operator works on the population [97]. The elitist operator ensures that the best chromosomes do not get lost from one population to the next, when the crossover and mutation operators are applied [5]. Thus the elitist operator is only applied after the crossover and mutation operators have generated a new population.

The elitist operator selects a certain number of high quality chromosomes from the parent population and transfers them to the new population without any modification from the other two operators [97].

The operator does not only move parents into the new population. It typically replaces subpar chromosomes in the new population with the higher quality parents [51]. Therefore the elitist operator helps the algorithm retain knowledge gained from previous generations and prevents the best solution from extinction [95].

Finally, the retainment of knowledge and best solution aids the algorithm in global convergence [115].

Algorithm Efficiency

The GA is a powerful, yet simple algorithm and tends to find good solutions given enough time, it does have its disadvantages. One of the major disadvantages occurs when the GA is applied to problems that have very large solution spaces. In these problems, the population size is a very sensitive parameter [5, 64, 97, 115]. If the population is too small the algorithm will not have enough diversity to search and will tend to converge prematurely.

A large population is preferred in large problem spaces, but then the algorithm is very computationally expensive since more time is spent evaluating than evolving new populations and the speed of the algorithm conver-

Problem instance	GA	COST 259
Siemens 1	2.60	2.20
Siemens 2	16.34	14.280
Siemens 3	6.37	5.19
Siemens 4	84.08	81.89

Table 4.3: GA on COST 259 Benchmark

gence decreases drastically. Hence, the population size must be fine-tuned to achieve optimal performance in large problem spaces [35, 64]

Another disadvantage of the GA is that it is memory intensive, most sequential algorithms search on a single point basis through the solution space. As discussed above, GAs search multiple points simultaneously and therefore require more memory to keep track of all the possible solutions.

In conclusion, the algorithm has some sort of hill-climbing ability through the mutation operator, but the probability of the mutation operator is far too low for the algorithm to be considered to have a real hill-climbing capability.

4.5.4 Genetic Algorithm on the FAP

Continuing the trend of the SA and TS algorithms, the GA has also been applied to a wide variety of optimisation problems and has on average obtained good results and in some cases better than all previous algorithms. Due to the success of the GA on other difficult problems, it has therefore also been applied to the FAP.

Colombo and Allen [22] have developed a GA to be applied on the FAP. The authors also utilide the notion of dynamic programming, by decomposing the FAP into smaller subproblems. On average the solution quality is improved by using the technique but at the expensive of more complex and taxing evaluations that have to be performed [22].

The evaluations are more complex, since in most instances the subproblems (which are subgraphs) overlap with other subproblems, and therefore more complex evaluations need to be performed taking into account the overlapping [22]. The authors report that the technique is viable to improve solution quality as long as the connectivity between the subproblem graphs is not too high [22].

By critically evaluating the GA if it were applied to the FAP, the following disadvantages can be identified:

Diversity — The GA continually operates on a set population that was randomly initialised at the beginning of the algorithm. It therefore only has this set of generated genes in the initialised population to therefore evolve successive populations. If one disregards mutation, the GA is a process by which the optimal combinations of the starting genes are found. Thus, the GA purifies the starting population genes in an attempt to find those individual genes, which if combined into a single individual, will produce an optimal individual, i.e. solution. Therefore the GA is very reliant on the quality of the random generator used. The probability of the algorithm finding a particular desirable gene that is exhibited by the starting population is directly related to the rate of mutation, as mutation is the only means by which more diversity is introduced in the GA.

Crossover — The crossover operation in the GA is the only means by which successive populations are generated and can therefore be regarded as the primary means by which the algorithm performs its search. As the crossover is defined in the standard algorithm, certain parts of both parents are copied and combined to form a new individual. With regard to the FAP, if each individual represents a frequency plan, the crossover operation would copy certain cells from the two parent plans. This is not desirable, since a single channel within a cell can generate major interference which overshadows the rest of the channels that generate low interference in the cell. Thus, for the GA to generate high quality solutions on the FAP, the algorithm would be better off utilising a crossover operation which works on individual channels assigned rather than cells. Crossover operation is also a memory and computationally expensive operation since individuals need to be constantly created and values need to be copied to these new individuals from the respective parents.

Mutation — Mutation is the primary means by which more diversity is introduced in the GA population. Typically the mutation value is set to a low probability due to mutation actually *modifying* individuals and not introducing *new* individuals. Therefore there is a possibility that a mutation

might alter an excellent solution in such a way that the solution is suddenly one of the worst solutions. With regard to the FAP, the low probability of mutation is not desirable, as the FAP search space is huge and therefore requires constant diversity to be introduced to accurately explore it. A possible good mutation would be one that is slightly more intelligent than the standard mutation operator, which just randomly modifies a selected individual. An intelligent mutation would be one that takes into account the recent history of the individual as well as the history of the population and, based on the collective knowledge alters or *mutates* a particular individual.

4.6 Summary

In this chapter a description was given of metaheuristic algorithms. What it means for an algorithm to be classified as being of a metaheuristic nature was explained and the characteristics these algorithms must exhibit were identified.

Three metaheuristic algorithms were discussed in this chapter. For each algorithm a flow chart visually depicted the general flow of the algorithm, and an explanation was given of how the algorithm works as well as the various characteristics that make the algorithm unique.

For each algorithm, a brief overview of studies using the particular algorithm was given as well as some of the disadvantages or challenges that would be faced when applying the particular algorithm to the FAP.

The first algorithm discussed was the tabu search algorithm and the second was the simulated annealing algorithm. The chapter concluded with the genetic algorithm. In the next chapter the class of algorithm that is relatively new to optimisation research, namely the swarm intelligence class of algorithms, will be discussed.

In the table below, the algorithms discussed in this chapter and their performance on the COST 259 set of benchmarks have been summarised.

Problem instance	TS	SA	GA	COST 259
Siemens1	2.7692	23.00	2.60	2.20
Siemens2	14.9360	14.75	16.34	14.280
Siemens3	6.6496	52.55	6.37	5.19
Siemens4	110.9725	80.80	84.08	81.89

Table 4.4: Summary of algorithm performance on the COST 259 benchmark

Chapter 5

Swarm Intelligence

5.1 Introduction

The research field of artificial intelligence stands a lot to gain by the study of the inner workings of nature itself; this is why there is a branch of artificial intelligence that incorporates some of nature's processes, like genetics, which can be seen being applied in practice in the GA (see section 4.5).

There are other approaches in artificial intelligence which also have their rotos in nature, sfor instance animal learning or the study of how dogs learn [13]. These approaches only look at a single agent thought process when it maps percepts (senses) to actions in its particular environment [13].

A percept can be said to be a process that is specifically designed to take data from the surrounding environment, process it and present it as information upon which decisions can be made [13, 106]. For instance, eyes are percepts, which take visual data presented by the environment [13, 106]. The visual data is processed by the brain into information to enable decisions to be made on navigating the environment [13, 106].

The research field of swarm intelligence is an approach more concerned with the underlying processes and behaviour patterns when multiple agents (insects, animals) come together and perform a task as one collective entity [13, 21, 70, 106]. This study of animals or insects in groups has already contributed to artificial intelligence as a whole [21,70]. Each individual in the swarm might not be able to solve the problem on its own, but by interacting with other individuals and performing primitive actions, the individual can contribute to solving the problem as a whole entity [21].

New algorithms that incorporate the lessons learned from the study of the collective intelligence are now being used and form part of the Meta-heuristic algorithm group. A discussion on some of the more popular meta-heuristic algorithms was presented in chapter 4 (see page 55).

Swarm intelligence algorithms are also meta-heuristic algorithms, with the distinction being made that swarm intelligence algorithms use multiple agents as a collective entity of knowledge to search the problem space [16, 21, 34, 35, 70].

The initial algorithms developed with regard to swarm intelligence, where based on the coordination and behaviour exhibited by schools of fish and flocks of birds. The newer generation of algorithms include [16, 21, 70]:

- ant colony optimization (ACO) [16].
- artificial bee colony (ABC) optimization [21].
- particle swarm optimization (PSO) [70].

These newer generations of algorithms are primarily being used in combinatorial NP-Complete problems, where there is no defined solution, only an optimisation that comes close to an optimal solution. Swarm intelligence works on a key feature observed in nature, the notion of emergent behaviour [16, 34, 35]. Whenever an entity does something unconventional and achieves success, it can be considered as exhibiting emergent behaviour as if it is emerging out of the mass's way of doing things [16, 34, 35].

Typical emergent properties exhibited by swarms are self-organisation and synchronisation [16]. In swarm intelligence, when an agent exhibits emergent behaviour, the behaviour of the agent needs to be shared with the whole swarm in order for the swarm to adapt and use the newly gained knowledge [16, 34, 35, 70].

The information that is shared among the individuals of the swarm can be anything that contributes to the swarm as a whole, for instance the information might be about a new solution that is better than all previous solutions [16, 34, 35, 70].

The behaviour propagates from one agent to another through social interaction, which brings forth information exchange [16]. Social interaction is but one component of self-organisation. Other components that form part of self-organisation are [16]:

- Positive and negative feedback
- Increasing the magnitude of fluctuations

Individuals within the swarm are able to accomplish social interaction with each other to achieve information sharing by using a concept referred to as *stigmergy* [16,34,35]. Stigmergy as well as the different forms of stigmergy will be discussed in section 5.2.

Utilizing stimergy along with self-organisation swarm intelligence algorithms like ACO, PSO and ABC optimization are able to produce relatively good results for NP-Complete problems [16, 34, 35].

Swarm intelligence based algorithms are able to achieve this since they have small individuals searching for more optimal values for a potential optimal solution on multiple fronts. As one individual makes progress on a certain front, its knowledge is shared [34, 35].

Traditional single agent based metaheuristic algorithms like TS (section 4.3) and SA (section 4.4) only have in essence one "individual" searching for a solution [34, 35].

With algorithms like TS and SA, the information is not shared since there is only one individual search [34, 35, 119, 146]. The information is kept to influence future decisions [82, 86, 106, 129, 140]. Swarm intelligence algorithms also store information for use by the various individuals that make up the swarm to make more informed decisions as the solution space is explored [34, 35].

NP-Complete optimisation problems are but one of the fields where swarm intelligence algorithms have been applied. Other fields where swarm intelligence has been applied include neural network training [35], vehicle routing [25], clustering [47], search engines and electrical power systems [10]. Swarm intelligence thus seems more suited towards problems with combinatorial complexity [24].

In this chapter the ant colony optimization will be covered first in section 5.3. This is follow by the artificial bee colony in section 5.4 and then particle swarm optimisation in section 5.5.

Before the algorithms are discussed a small overview of stigmergy is given, as it forms a core concept upon which swarm intelligence algorithms are based.

5.2 Stigmergy

Stigmergy is defined as the method animals and insects use to facilitate communication through interaction [12, 27, 34, 35]. Through the use of stimergy animals or insects are able to socially interact with their own species to convey information to each other [12, 27, 34, 35].

Interaction occurs through signals that the individuals receive which might require them to perform a specific action [12, 27, 35].

Two forms of stigmergy can be observed in nature. One form, *sematectonic stigmergy*, is a direct and physical form of interaction since it relies on altering the environment or by using some form of physical action to communicate [35].

Examples of this type of stigmergy are nest building and brood sorting by ants [35]. Schools of fish also use this type of stigmergy to communicate direction and speed by visually observing their closest partner in the school. Besides using visual information, birds use sound to communicate with and alert each other [16].

The other form, *sign-based stigmergy*, is an indirect form of interaction, where communication occurs through some sort of signal mechanism [35]. Ants use sign-based stigmergy to communicate with each other. More on how ants communicate with this type of stigmergy will be discussed in section 5.3.1.

Other species that use sign-based stigmergy are bees [45]. When a bee determines that an entity poses a threat to the hive, it might decide to sting the entity. The sting of a bee not only injects a toxin into the entity, but also releases a pheromone [45]. This pheromone alerts nearby other bees from the hive of the presence of an entity that is a potential danger to the hive [45].

The other bees of the hive pick up this pheromone that is released by the initial bee's stinger and attack the entity by also stinging it [45]. As more bees sting the entity, more bee stingers emit the danger pheromone identifying the entity [45]. Hence the pheromone is reinforced and becomes stronger, which persuades more bees into action [45].

Stigmergy is a powerful mechanism that is able to alter the behaviour of a collective entity efficiently, as can be gathered from the above-mentioned examples of stigmergy in nature [12, 27, 35]. Stigmergy is therefore a core

concept upon which swarm intelligence algorithms are based as these communication techniques are exploited to aid the algorithm in finding better solutions [12, 27, 35].

In the forthcoming sections three swarm intelligence algorithms will be discussed. Each section is divided into four subsections.

First, an overview of the algorithm will be given, where basic concepts about the algorithm will be introduced as well as a general outline given of the search process the algorithm uses.

The second subsection will provide a step-by-step discussion of the algorithm using pseudocode as a reference.

The third subsection will give an in-depth discussion of some of the core characteristics that make the algorithm unique.

Finally, for each algorithm studies using the algorithm on the FAP are mentioned and the various considerations that need to be made to apply the algorithm to the FAP are identified.

5.3 Ant Colony Optimisation (ACO)

5.3.1 Introduction

ACO is a class of algorithms incorporating different behavioural aspects that ants exhibit when they perform certain activities, i.e. gather food, build nests and construct cemeteries [27,35]. The first ACO algorithms that were developed were based on the foraging behaviour that was exhibited by ants when finding the most optimal path towards a food source. Deneubourg noticed the foraging behaviour when he performed the bridge experiment [27, 35].

Deneubourg wanted to know how ants were able to find the shortest path towards a food resource and communicate it to the whole nest [27]. Thus an experiment was set up to study the ants' behaviour. In the experiment a food source was placed a certain distance away from the nest [27, 35]. Two paths were established towards the food resource, and one path was purposely made longer than the other path as can be seen in figure 5.2 [27].

Ants would exit the nest in order to forage food [27, 35]. They first started exploring the area outside the nest in an attempt to locate a suitable

Algorithm 4 Basic Ant Colony Optimisation Algorithm [35]

Initialize τ_{ij} with small starting values
$t \leftarrow 0$
Place n_k ants on starting node
while stopping condition not reached **do**
  for each ant $k \leftarrow 0$ to number of ants n_k **do**
    $p^k(t) \leftarrow$ Initialize path $p^k$for time step t
    repeat
      Select next node based on probability equation 5.1
      Add link (i,j) to path $p^k(t)$
    until Final node reached
    $x^k(t) \leftarrow$ Remove loops from path $p^k(t)$
    Calculate length of path $f(p^k(t))$
  end for
  for each link (i, j) in graph **do**
    $\tau_{ij} =$ Reduce pheromone of link (i, j) with equation 5.5
  end for
  for each ant $k = 0$ to number of ants n_k **do**
    for each link (i, j) in $p^k(t)$ **do**
      $\triangle\tau_{ij} = \frac{1}{f(p^k(t))}$
      Update the pheromone τ_{ij} with equation 5.3
    end for
  end for
  $t \leftarrow t + 1$
end while
return path $x^k(t)$ with the smallest $f(x^k(t))$ as the solution

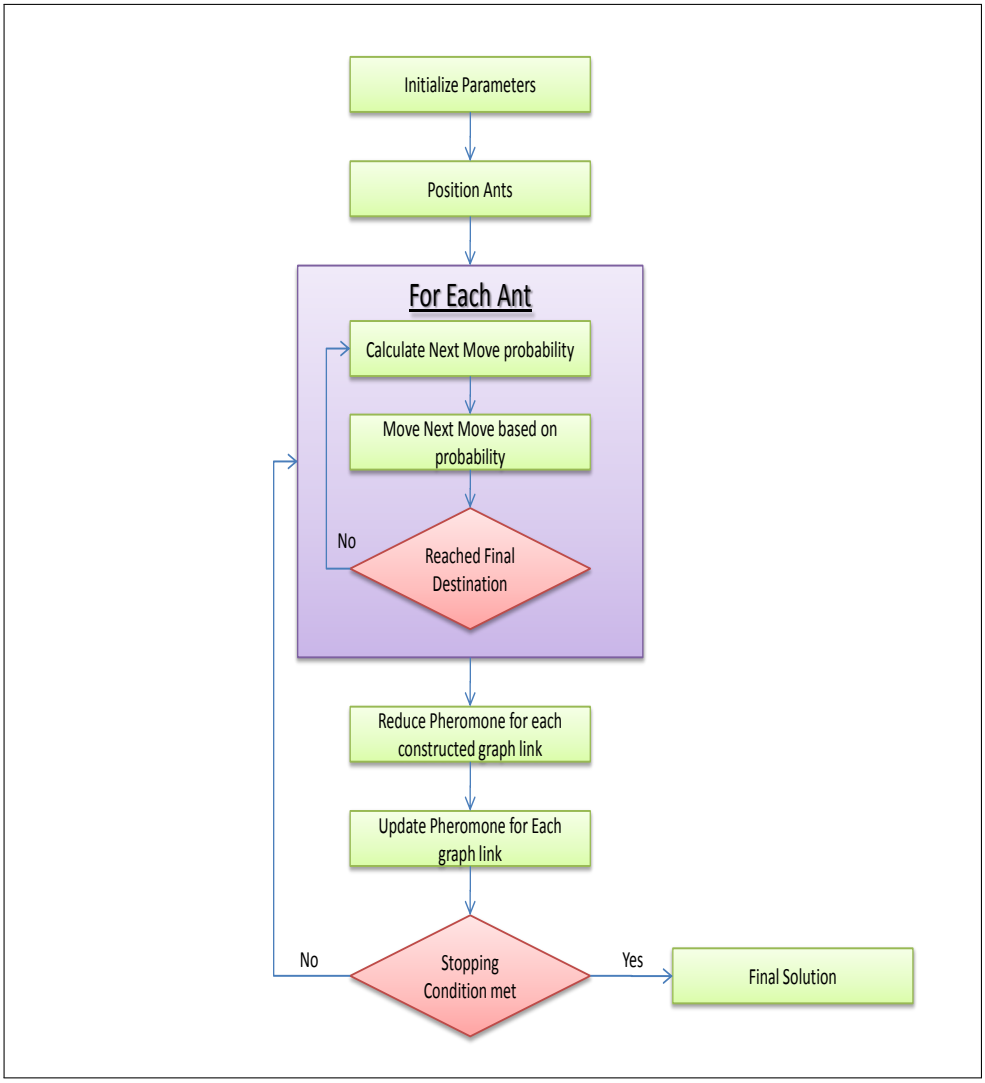

Figure 5.1: Flow chart for ACO algorithm

food source [27, 35]. As the ants in the experiment explored the immediate space outside the nest to the food source, they had to take one of the paths provided to the food source [27].

The ants initially oscillated between the paths with no clear distinction of the more dominant route to take to retrieve food from the food source [27, 35]. After a certain amount of time one path to the food source became the preferred route for the ants and it was indeed the shortest path towards the food source [27].

Naturally the question that was asked was how the ants were able to communicate to each other that one food source was closer than the other. The answer lies in the use of *stigmergy* by ants [27, 35]. Ants use sign-based stigmergy (discussed in section 5.2) when they retrieve food [12, 27, 35]. As the ant moves along a particular path, it marks the path with a chemical signal that alerts other ants to the desirability of the path [35]. The chemical signal that ants use to indicate optimal paths is called *pheromones* [27, 35].

The concept of pheromones and how the ACO goes about in updating the pheromones is a critical concept of ACO hence, an in depth discussion on pheromones will be provided in subsection 5.3.3.

A path is made up of a series of links between nodes [42, 106]. A link between two nodes represents a movement from one node to the other [42, 106]. Therefore, a path can be considered as the traversal of the interlinked nodes, from a starting node to some final node [42, 106]. A path differs from another path by the order in which the nodes are interlinked between a start and end node [42, 106].

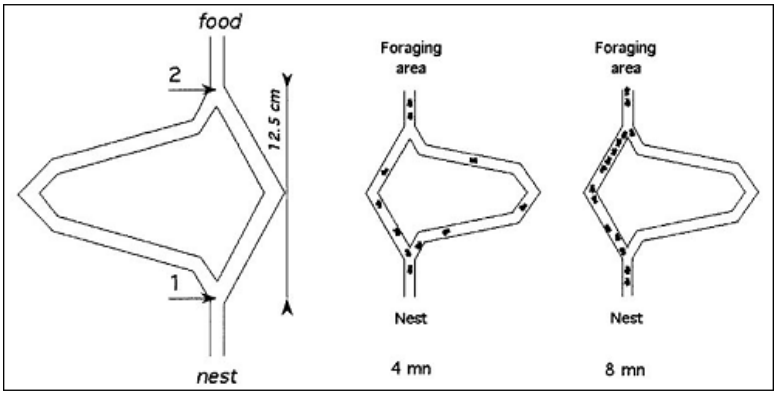

Figure 5.2: The Ant bridge experiment [27]

The ACO algorithms are considered stochastic search procedures due to their innate use of randomness when exploring the solution space [25, 142]. The ACO class of algorithms has been applied to a wide range of problems that include single machine scheduling [50],weapon target assignment [66], flow shop scheduling [20] and image thresholding [147]. Variants of the standard algorithm have been developed, but all of the algorithms still follow the basic structure listed in algorithm 4 [34, 35].

The ACO algorithm has achieved relatively good success in the problems it has been applied to, but it does have some disadvantages [25, 142]. One of the primary disadvantages of ACO is that it tends to get trapped at local minima in the solution space and therefore the solution space is not adequately explored and the algorithm prematurely converges to local optima [142].

To combat this shortcoming, the MAX-MIN ant optimization algorithm was developed. The MAX-MIN algorithm is based on the original ants system algorithm (discussed below). MAX-MIN addresses the local minima problem by imposing dynamically changing bounds on the pheromones of the ants such that they always fall within the limit of the heuristic and best current path of the colony [131].

The first algorithm to be based on the behaviour of ants was called the ants system (AS) [12, 35]. The AS was initially applied to the traveling salesman problem (TSP) as a proof of concept but its performance on TSP was lacklustre compared to other algorithms applied to the same problem [12, 35].

Subsequently, various algorithms have been developed that improve on the AS algorithm. These improved algorithms include the ant colony system (ACS), ANTS system, Ant-Q and AntTabu [12, 35]. Where applicable, reference to these algorithms will be made to indicate how they have improved the AS system.

In an attempt to better understand the ACO algorithm the general flow of this algorithm is discussed below.

5.3.2 Flow of the Algorithm

In this section the process the algorithm uses to explore the solution space will be described using algorithm 4 as a reference point.

The ACO algorithm initialises by creating a set population of ants and placing them on random starting nodes as well as initialising the pheromones to starting values as can be observed from algorithm 4, lines 1 – 3. The main purpose of the ant is to explore the solution space and to ultimately produce a solution that might be optimal. The ant explores the solution space by performing a series of moves from one node to another. Each move is a link that is added to the path. This process can be seen in lines 5 – 10.

The ant selects which node to move to next based on a probability. The probability is calculated taking into account the amount of pheromone that is on the current link representing the movement from the current node to the next node [34, 35]. This decision process can be seen to occur in line 8.

As the ant moves it records each link between the nodes it traverses until it reaches the final node. All the links the ant has traversed represent a path taken through the solution space [34, 35]. Thus, as the ant is moving it is actively building an optimal solution.

Before the ant deposits pheromone on the links it traversed to construct its solution, the pheromones first need to be decayed. This is why in lines 14 – 16, the algorithm traverses all links that contain pheromones and reduces the amount of pheromones by applying equation 5.5.

Once an ant has constructed a path and the pheromone evaporation has occurred, the ant is ready to inform the rest of the ants of what movements it made to construct its solution. The ant needs to share this movement information in order for the rest of the colony to know which movements worked well and which did not. The ant therefore needs to signal the other ants, which is accomplished with pheromones. Therefore, in the next phase of the algorithm, pheromones are deposited on all the links that make up the path the particular ant constructed. In the algorithm pheromones are deposited on lines 17 – 22 in algorithm 4.

After all the ants have deposited pheromones on all the links represented by the each individual ant's constructed solution, the algorithm is ready to continue to its next iteration. This process occurs until some defined stopping criterion occurs.

5.3.3 ACO Characteristics

Various characteristics are important and unique to the ACO class of algorithms, namely pheromones, state transition rules and pheromone evaporation.

Pheromone Trail

The pheromone technique used by ants forms part of the core methodology used by the ACO algorithm [38]. As an ant moves it lays down pheromones to mark the path it is walking. This step can be observed in algorithm 4 in lines 17 – 21.

Pheromones decay over time. As the ant follows the pheromone trail it reinforces the pheromone that is already on a path [38]. Thus the more ants following a path, the stronger the pheromone trail for that path and the stronger the pheromone, the higher probability that an ant will follow the path [38]. Pheromone decay can be seen in algorithm 4 in lines 14 – 16.

The pheromone reinforcement is the last phase the algorithm enters before continuing to the next iteration as seen in figure 5.1.

In the bridge experiment the ants started following the shorter path, because an ant following the shorter path would reinforce its pheromone trail faster than on the longer path. Initially, the ants oscillated between the two paths since there was no clear indication to the colony as to what the shortest path was; therefore the ants initially randomly selected the paths they would follow [38].

Once a path has been marked with pheromones the ant no longer selects that path based solely on randomness [27, 35, 38]. Instead the ant selects that path based on a probability transition rule [27]. Transition rules will be discussed in the next subsection.

With the use of pheromones ants are able to communicate the best and shortest path [27,35,38]. The more ants following a preferred path the more pheromones would be laid on that specific path. This increases the strength of the pheromones [142]. The increase in strength of the pheromones on a path would thus let ants more clearly distinguish between paths they should and should not take [142]. Therefore, a pheromone provides positive feedback to the colony [27, 35, 38].

Initially, all the ants will choose random paths [27, 35, 38]. After all the ants have completed their paths, each path is evaluated [35]. The amount of pheromone marking a path in the standard ACO is related to the cost function [27, 35, 38]. Therefore, a low cost function value will have a high pheromone dosage indicating the path the ant took from each node, and a high cost function value will have a low dosage [35].

The pheromone of each path therefore allows the colony to remember good and bad decisions from previous iterations [35]. This is a form of local pheromone updating which will be discussed later.

In the iterations following the initial one, the ants will at each node decide, based on a probability function, whether they should follow the pheromone trail laid down in previous iterations or choose a new path to another node [27, 35, 38]. Thus as the ACO iterates through more iterations the stronger that particular path's pheromone trail will become, hence signalling the path out as the best found [35]. The probability function will be discussed in the next section.

The pheromone trail was initially developed with only one colony in mind [35]. In the research done by Tiwari et al. [125] pheromones in multiple colonies are considered. The basic principle of how pheromones are used by the ants stays the same, but the meaning of the pheromone changes if an ant of another colony encounters the pheromone trail [27, 35, 38]. The ant will not follow or even consider the pheromone trail since any pheromone encountered from other colonies repulses the ant [125]. Thus pheromones only provide positive feedback if the ant is from the same colony, otherwise the pheromone gives negative feedback, in a way warning the ant to stay away [125]. This repulsion strategy promotes exploration among the multiple colonies [125].

There are different transition rules that each algorithm in the class of ACO algorithms uses, and these are set out below

State Transition Rules

As discussed in the previous subsection, the ants select which path to follow next based on a probability. This probability is also known as the *transition*

probability.

$$p_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}}{\sum_{u \in N_i^k(t)} \tau_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)}, & \text{if } j \in N_i^k(t) \\ 0, & \text{if } j \notin N_i^k(t) \end{cases} \tag{5.1}$$

The first transition probability is formulated in equation 5.1 and is used by individual ants of the AS algorithm [26, 34, 35, 38, 125]. An ant k uses this equation to decide with what probability it will move from node i to node j [34,35,38,125]. τ_{ij} is the amount of pheromone on the link between nodes i and j [27, 34, 35, 38, 125]. Heuristic information is incorporated into the equation through the symbol η_{ij}, which is the desirability of the path from node i to node j as evaluated by a heuristic function [27, 34, 35, 38, 125].

As can be observed from figure 5.1, the algorithm individually moves each ant based on these defined transition probabilities.

Through the use of parameters α to represent pheromone intensity and β to represent heuristic information the algorithm is able to achieve a good balance between exploration and exploitation when $\alpha = \beta$ [34, 35, 38, 125]. When $\alpha = 0$ no pheromone is taken into account; hence, any history that the algorithm has on the path between node i and node j is neglected and the algorithm degrades to a stochastic greedy search procedure. If $\beta = 0$ then the algorithm does not take into account the amount of desirability of the path between node i and node j as dictated by the problem-specific heuristic function.

The set $j \in N_i^k(t)$ contains all the valid neighbourhood moves ant k is allowed to make when moving from node i to node j. A tabu list is kept by each ant to trim the set of moves already performed previously, and thus cycling is prevented.

The intention here was not to give an exhaustive survey of different transition rules in the literature. Therefore, only the first transition rule that was developed is discussed, since most of the other rules can simply be considered derivatives of the first.

Pheromone Update

Pheromones start to evaporate[1] over time, and so the path marked by the pheromone trail becomes less attractive to the ants. Therefore, a path that

[1]Pheromone evaporation is discussed in section 5.3.3

represents a good solution needs its pheromone trail to be continuously updated. Certian rules govern when and by how much pheromones are updated.

Most of the variants that have been developed differ in what pheromone update rules they employ. In the literature pheromone update rules are classified into two groups [35]. One group is called the global update rule. The other group is called the iteration-based or local update rule, an example of which has been given on page 106 [35].

The first local pheromone rule was presented in the AS algorithm [26, 27, 35]. The ants would retrace their path after each iteration, depositing pheromones on each link that makes the complete path. The following equation is used to update the pheromone:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{5.2}$$
$$\text{where } \Delta\tau_{ij} = \sum_{k=1}^{n_k} \Delta\tau_{ij}^{k}(t)$$

In equation 5.2 $\tau_{ij}(t+1)$ represents the amount of pheromone that will be on the link for the next time step $(t+1)$. τ_{ij} represents the amount of pheromone currently on the link (i, j). $\Delta\tau_{ij}$ is the actual amount of pheromone that needs to be added to the current pheromone τ_{ij}.

Pheromone update rules that are in the global update group only allow the pheromone trail of the path representing the best found solution by the algorithm since the first iteration to be updated [35]. Thus the global rule favours intensification where the algorithm exploits the global knowledge gained by the ants to find a better solution. By updating pheromone what is actually occurring is that the pheromone is reinforced.

ACS was the first to use the global update rule together with the local update rule [35]. By using both types of rules the algorithm is able to efficiently exploit the history provided by the pheromones [35]. The global update rule used by the ACS is formulated in the following equation [35]:

$$\tau_{ij}(t+1) = (1-p_1)\tau_{ij}(t) + p_1\Delta\tau_{ij}(t), \tag{5.3}$$
$$\text{where } \Delta\tau_{ij} = \begin{cases} \frac{1}{f(x^+(t))} & \text{if } (i,j) \in x+(t) \\ 0 & \text{otherwise} \end{cases}$$

The parameter $f(x^+(t))$ represents the best/shortest path found so far by the algorithm [35]. $p1$ is the variable that controls that rate of evaporation.

$\Delta\tau_{ij}(t)$ is the amount of pheromone at the current time step t for the link ij.

By using the global update rule the algorithm is able to direct the search more, which is to say the algorithm exploits the solution space more. This exploitation is achieved as the best path is continually used in the update of the pheromone as can be observed in equation 5.3 [34, 35].

As can been seen in the following equation, the ACS uses a slight variant of the local update rule first used in AS [35]:

$$\tau_{ij}(t) = (1 - p_2)\tau_{ij} + p_2\tau_0 \tag{5.4}$$

In the above equation τ_0 is a small constant and $p_2 \in [0, 1]$ is the constant that defines the rate of evaporation [35]. With the local update rule, the algorithm is able to explore more as the individual ant constructed path is used to update the pheromone and no information from the best path found in the colony is incorporated, as with equation 5.3 [34, 35].

This phase of the algorithm is executed as the very last step, as can be observed from figure 5.1.

Pheromone Evaporation

Initially when the pheromone concept was first implemented the ants of the colony rapidly converged on a solution and did not adequately search the solution space for alternate paths that might lead to better solutions. To combat this premature convergence and force the ants to explore the solution space more, the concept of *pheromone evaporation* was introduced [12, 26, 27, 35].

As discussed previously the pheromone marking a trail evaporates over time until an ant reinforces it. The evaporation of pheromones is governed by equation 5.5 [12, 26, 27, 35]:

$$\tau_{ij}(t) \leftarrow (1 - p)\tau_{ij}(t), p \in [0, 1] \tag{5.5}$$

The constant p defines the rate at which the pheromone evaporates. If $p = 1$ the pheromone completely evaporates every iteration and the ants take no history into account with regard to their path selection and the search is completely random [27, 35]. Thus, the amount of exploration done by the algorithm can be controlled by the constant p [27, 35].

The above equation 5.5 was first introduced in the AS [12, 26, 27, 35]. Most subsequent algorithms that are a form of the ACO class of algorithms also use the concept of pheromone evaporation, but they either use the standard equation or develop their own variant [12, 26, 27, 35].

A more aggressive form of pheromone evaporation is added to the AS discussed in the research done by Gambardella, Taillard and Dorigo [38]. The more aggressive form works beside the already present pheromone evaporation, but this form seeks to add an additional search phase called *diversification* [38]. The aim of the diversification phase is to lead the algorithm into another direction of the search space [38]. This is done in an attempt to avoid local minima and stagnation [38].

In the ant system developed by the authors the algorithm continually monitors the current best solution and keeps a history of recent best solutions [38]. If the algorithm starts to notice that solutions are cycling or that the current best solution has not changed for a certain number of iterations, the algorithm activates the diversification phase [38]. The algorithm is forced to re-search the search space to create new solutions, as it cannot rely on previous historical information provided by the pheromone trails [38].

As can be observed from figure 5.1, the last phase is where all the pheromones laid by the ants are updated for the next iteration. Pheromone evaporation forms part of this phase.

5.3.4 ACO on the FAP

ACO has been applied to a wide number of problems and has produced good results. It has also been applied to the FAP instance of problems.

When using the ACO algorithm on the FAP the ants need to construct a path that represents a frequency plan and has low interference. With the ACO, a node is a cell that has a unique set of channels assigned to it. Thus the same cell may exist in the search space, but will have a different set of channels assigned to it, and will therefore represent an entirely different node to the ACO.

As an ant moves in the frequency planning domain, it is actually moving between two cells that are said to interfere. The interference between two cells occurs as a consequence of the channels that have been assigned to them.

As an ant completes a movement from one cell to another, i.e. it assigns channels to the cell, it measures the interference that occurs due to the assignment. The measured interference information is incorporated into the pheromone, which the ant will deposit on the link between the two cells.

An optimal frequency plan would therefore be a path through all the interfering cells marked with a high dosage of pheromone. As discussed in the previous sections, the pheromone indicates the desirability of a particular path. In the FAP, a desirable path would be one where interference is low; thus a path with a high dosage of pheromones would be the frequency plan with the lowest interference found by the algorithm.

When analysing the basic ACO algorithm 4 one can identify the following possible disadvantages if the algorithm were applied to the FAP:

Memory Usage — The algorithm requires a fair amount of memory. The memory is used to keep track of each permutation of a particular cell and its allocated frequencies until the pheromone that links to the cell has decayed enough to be discarded. As an ant moves from one cell to another, it might not select the previous cell (due to probability) to move to, but rather generates an entirely new cell to move towards. This newly generated cell would then be linked to the previous cell, and therefore the algorithm needs to keep track of the pheromone on that link until it has completely been decayed away. The algorithm needs to keep track of these pheromones on the links even if the new link to the generated cell is not even close to optimal and has very high interference.

Building a solution — The ACO *builds* an optimal solution. Therefore, early decisions made by the ants still influence the plan later for better or for worse. A good decision might seem to be good early on, but later the algorithm might be better off with a slightly worse decision. In the FAP, a cell can have multiple interfering cells, but a particular ant only knows about one link between two cells and not about the other interfering cells. Thus an ant will find the optimal path on the first interfering link between two cells, in other words it will optimise the channels allocated to these cells so that interference is low. The first interfering link is now optimised, and subsequent ants will reinforce this channel allocation since the interference is low. When the ants later reach the other cells that also interfere with

Time limit	ACO*	ACO
120s	104719.72	91140.04
600s	103752.12	89703.44
1 800s	103781.86	88345.94

Table 5.1: ACO and ACO* on custom GSM FAP benchmark [71]

the first cell that has been optimised, they will have difficulty changing the assignments that have already been made, since the pheromone representing that assignment is too strong to disregard.

The above disadvantages have only been identified by critically evaluating the algorithm as a possible point of interest to produce an optimal solution for the FAP for this dissertation. Even with these disadvantages the ACO has achieved success in producing high quality optimal solutions for the FAP.

In research conducted by Luna et al. [71] an ACO algorithm was applied to a custom cellular network instance. This network instance had 711 sectors with 2 612 transceivers, which needed to be assigned frequencies. For their particular network, only 18 channels were available for assignment. The channels started at 134 and ended at 151 [71].

The authors presented two versions of the algorithm. The first version used no heuristic information (henceforth referred to as ACO*) and the other version used heuristic information to update the pheromone laid done by the artificial ants [71].

With regard to the heuristic updating of the pheromone trails, the authors opted to increase the pheromone by some magnitude [71]. This magnitude was handtuned to be 100. The heuristic only increases a certain path's pheromone if the frequencies assigned to the transceivers represented by this path differ enough so as to not cause significant interference [71]. Thus, the heuristic aims to amplify good choices made previously by the algorithm for the next iteration of the algorithm.

If one observes the results the above authors obtained, one can clearly see that the ACO version that incorporates heuristic information to reinforce pheromone trails outperforms the version that does not [71]. The above values represent the amount of interference that will result if the plan is used in the network [71].

5.4 Artificial Bee Colony (ABC) Algorithm

Algorithm 5 Basic Artificial Bee Colony Algorithm [60]

$b_n \leftarrow$ Initialize bees
$s_n \leftarrow$ Initialize starting solutions
Evaluate starting solutions with fitness function $f(s_n)$
$t \leftarrow 0$
while stopping criteria not met **do**
  for each employed bee $eb_i = 0$ to max bees b_n **do**
    $v_i \leftarrow$ Generate new solution with equation 5.7
    Evaluate with fitness function $f(v_i)$
    Apply greedy selection between v_i and the current solution s_i of bee eb_i
  end for
  Calculate probability p_i for solutions s_i in s_n using equation 5.6
  for each onlooker bee $ob_i \leftarrow 0$ to max bees b_n **do**
    $x_i \leftarrow$ Select solution s_i based on p_i
    $v_i \leftarrow$ Generate new solution with x_i and p_i
    Evaluate v_i with fitness function $f(v_i)$
    Apply greedy selection between v_i and bee ob_i current solution
  end for
  if there is an abandoned solution for a scout bee **then**
    Replace with solution generated with equation 5.8
  end if
  $t \leftarrow t + 1$
end while

5.4.1 Introduction

The ABC algorithm is the most recent algorithm discussed in this chapter [60,61,113]. It was first proposed by Karaboga in 2005 who wanted to mimic the foraging behaviour exhibited by bees [60,61,113]. Like ants, bees need to gather food to support the colony. To understand how the ABC algorithm tries to mimic the foraging behaviour of bees, this behaviour of real bees needs to be described first [60].

In a bee colony there are numerous bees, each with a specific role that

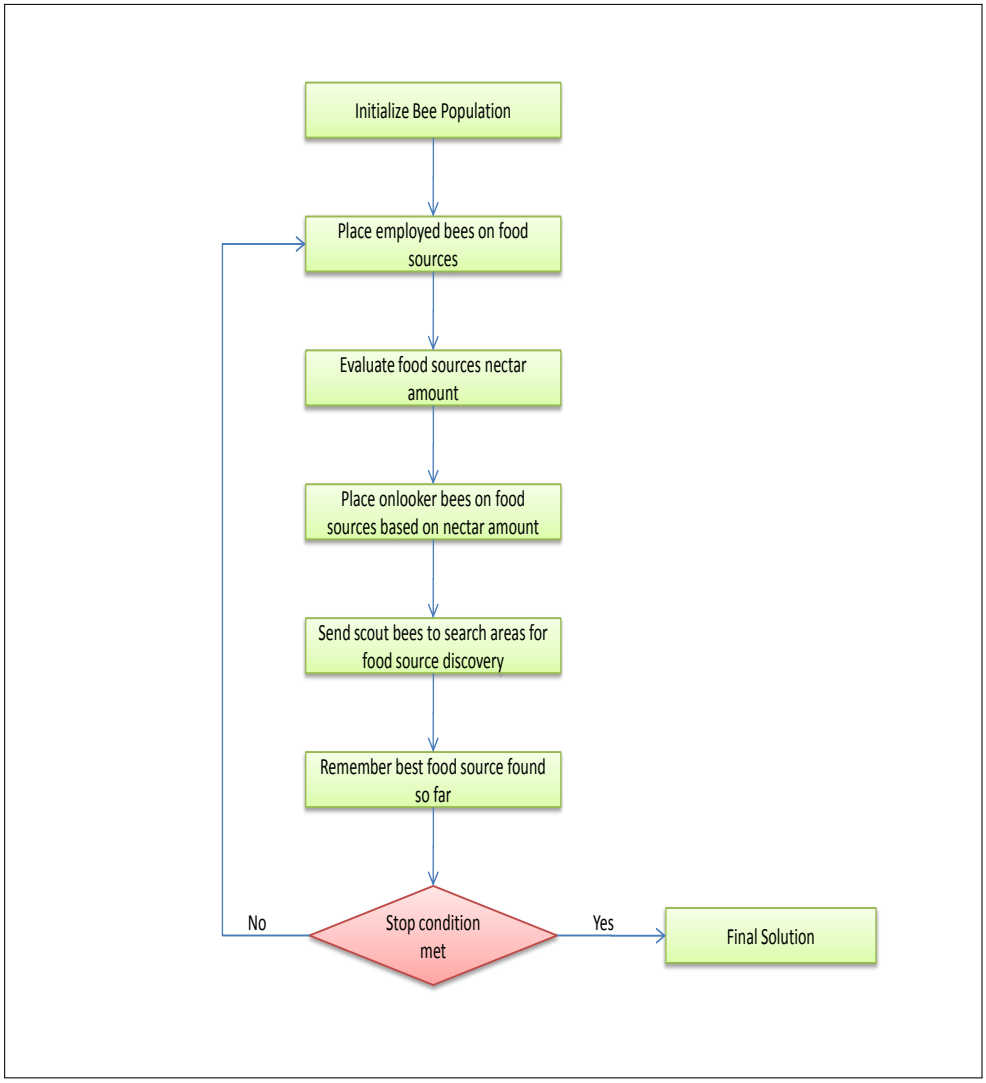

Figure 5.3: Flow chart for the ABC algorithm

performs certain actions for the colony. There are bees that protect the queen, maintain the colony, scout for resources and gather food, i.e the worker bees. The most important bees for foraging are those that scout and gather food [60].

The scout bees are sent out and, as their role implies, they are responsible for exploring the surroundings of the hive to find suitable food sources [60]. If a scout bee has found a food source it needs to return to the colony to share the information with the worker bees [60]. When the bee enters the colony it needs to communicate to the other bees by using some form of stigmergy (see section 5.2) [60].

The scout bee accomplishes this communication by performing a dance known as the *waggle dance* in the colony for all the bees to see [60]. This is not a dance as in the traditional sense, since through certain movements the bee is able to communicate a variety of characteristics about the food source including [60]:

- How far the food source is from the colony
- Quality of the food source
- Path towards the food source

It can be concluded that foraging bees use *sematectonic* stigmergy (discussed in section 5.2). This is deduced from the dance which is a physical form of communication.

The dance is observed by *onlooker* worker bees [11, 60]. These onlooker bees are initially *unemployed* in the colony [11, 60]. Once the information of the scout has been transferred to the onlooker bees, the onlooker bees become *employed* bees [11, 60]. They become employed bees when they operate on a particular food source to gather food [11,60]. Thus it is the job of the worker bees to *exploit* the information provided by the *exploration* done by the scout bees [60, 61].

Worker bees gather food from the designated food source, until the food source reaches a certain quantity with regard to nectar content [60,61]. Each time the bee returns to the colony it evaluates the current food source versus other food sources discovered [60,61]. If a better food source is found, the bee abandons the previous and starts gathering food from the new source [60,61]. On the other hand, if the food source has been exhausted, meaning there is

no more nectar content to gather, the bee returns to the colony and becomes "unemployed" again [60, 61].

In the ABC algorithm, possible solutions are considered to be food sources [60, 61]. Each food source has an employed bee associated with it. Onlooker bees either wait for new food sources to be communicated to them or become employed bees by moving to another, more attractive food source [60, 61].

A food source might be more attractive to a bee because its defined nectar content is more than that of the current food source the bee is operating on [60, 61]. The nectar content of a food source can be considered to be the fitness, which is determined using the fitness function of the specific problem domain [60, 61].

As with real honey bees, a *waggle dance* is performed to all the onlooker bees by employed bees that provide information on the nectar amount (fitness value) that they represent [11, 60, 69]. The onlooker bees choose food sources depending on the nectar amount [11, 60, 69]; therefore as the nectar amount of a food source increases, the probability that more onlooker bees will choose the source increases [11, 60, 69]. How and what affects the probability will be discussed in the next subsection.

Bees can transition to different roles depending on their situation [60,61]. An onlooker bee becomes employed when assigned to a food source and an employed bee can become a scout if its initial food source becomes exhausted [11,60,69]. Note that not all employed bees of a food source become scouts; only the first employed bee of a food source transitions to a scout [11,60,69]. Scout bees are sent to randomly generated food sources [11, 60, 69].

The more onlooker bees a food source attracts, the more the neighbourhood will be explored since the onlooker bees move to the food source and choose an immediate neighbouring food source to be employed upon [60,61]. Thus, this can be considered exploitation and the algorithm is therefore performing a local search [60,61,69]. Finally, the number of onlooker bees a food source has also indicates its desirability. A very good solution will have the majority of onlooker bees choosing it and searching for nearby better food sources [60,61,69]. More bees are lured towards a particular food source due to the high nectar amount that has been communicated to them by other employed bees [60, 61, 69].

When a food source is abandoned, the previous bee that occupied the

food source transitions to a scout bee [60, 61]. The scout bee is responsible for replacing the abandoned food source by finding a new one, and a new food source is generated and communicated back to the colony [11, 60, 61]. The generation of food sources will be discussed in the next subsection.

Karaboga was not the first to base an algorithm on the above foraging behaviour. Other bee foraging inspired algorithms have been developed such as the BeeHive algorithm, bee colony optimisation (BCO) and bee swarm optimization (BSO) [61, 76, 123].

The BeeHive algorithm is based on the dance communication used inside the colony of bees. In BCO solutions are randomly generated and assigned to bees [61, 76]. The solutions are then progressively modified using certain strategies. Finally, BSO solutions are iteratively constructed by forager (worker) bees and the best solution is communicated to the rest of the colony by performing a dance [61, 76]. All of the above-mentioned algorithms were developed to be primarily used on combinatorial problems [60].

Another bee algorithm is the virtual bee algorithm (VBA) which, like the previous algorithms, is also based on the foraging behaviour of bees, but it differs in that it is not designed for combinatorial problems [61]. Instead the VBA is a variant of the standard ABC algorithm which is designed for numerical function optimisation [61]. In VBA bees move around in the search space communicating to each other any target nectar food sources that are found [61]. Good food sources are function evaluations of particular coordinates in the numerical search space which produce low function evaluation values in the case of minimisation [61].

Karaboga developed the ABC algorithm based on the previous research done on bee colony optimisation and the above algorithms. The ABC algorithm is designed to be a multivariable optimisation algorithm and has to date been applied to the job scheduling problem, clustering [76], neural network training and reconfiguration of distribution networks [69]. Due to the nature of the algorithm being similar to that of the ACO, the ABC algorithm will most likely also be applied to a whole host other of problems.

In the next subsection the general flow of the ABC algorithm will be described, which will aid in the understanding of how the algorithm searches a particular problem space.

5.4.2 Flow of the Algorithm

Most of the concepts that are used in the ABC algorithm have been explained. The general search process of the algorithm will now be discussed using algorithm 5 as a reference point.

The algorithm starts off by generating a set number of possible solutions. The number of solutions is equal to the number of employed bees. Each starting solution is evaluated using a fitness function. These sets of operations can be observed from lines 1 – 3.

At first each bee is assigned to one of the initial generated solutions (food sources). Hence the bees start off as employed bees and each bee has in its memory a particular possible solution with an associated nectar amount. The algorithm can now be considered to be initialised, and therefore the algorithm enters the next phase, which is where the actual optimisation and search procedure occurs. This phase stretches from lines 5 – 22.

From lines 6 – 10, each employed bee modifies its particular solution based on local information, which is also referred to as visual information in the literature. The modified solution is then tested to determine its nectar amount, i.e. the fitness of the generated solution is calculated. The employed bee then compares the newly generated nectar amount with the nectar amount of the solution in the bee's memory. If the newly generated solution has a better nectar amount, the bee replaces the current solution in its memory with the newly generated solution.

After all the employed bees have determined whether to keep the newly generated solution or keep the one in memory, they then need to communicate to the rest of the bee hive the nectar amount of the food sources that they occupy. This phase is where the waggle dance occurs and can be observed in algorithm 5 from lines 12 – 17.

Each onlooker bee then selects which food source it will move towards based on a probability. The probability takes into account the nectar amount that was communicated through the waggle dance by a particular employed bee. The probability is calculated using equation 5.6 which will be discussed in section 5.4.3.

Once an onlooker bee has selected a food source based on the calculated probability, it then starts to search the immediate neighbourhood of the selected food source for other food sources. The neighbouring food sources

are generated using equation 5.7 which will be discussed in section 5.4.3. This procedure of generating neighbouring food sources by an onlooker bee can be observed in line 14 in algorithm 5.

The onlooker bee then applies the same procedure as an employed bee with regard to determining if the newly generated food source should be remembered or discarded. The bee does this by evaluating each generated neighbouring food source to determine its nectar amount, which is then compared to the nectar amount of the food source in the bee's memory.

In the last phase of the algorithm (before the next iteration starts) the algorithm determines which food sources have been abandoned by the bees. A food source in the algorithm can be abandoned if, for a certain number of iterations the food source has not improved, meaning its nectar content has not increased. When a food source is abandoned the employed bee that occupied the particular food source transitions to a scout bee.

A scout bee aims to replace the abandoned food source with a new food source. In algorithm 5 this occurs in lines 18 – 22. The scout bee uses equation 5.8 to generate a new food source. It then transitions to an employed bee and occupies the newly generated solution. The newly generated food source will now also be evaluated to determine its nectar amount as the rest of the employed bees do at the start of the next iteration.

5.4.3 ABC Algorithm Characteristics

Various characteristics of the ABC algorithm define the algorithm and make it unique. The first characteristic is how food sources are handled in the algorithm. The second is how information is communicated to the colony.

Food Sources

As discussed previously, food sources represent solutions to the problem the ABC algorithm is being applied to. When the algorithm starts, there are no defined food sources for the bees to evaluate and report on, and therefore initially a finite number of food sources are randomly generated [11, 21, 41, 60, 61, 69]. Since each food source needs an employed bee to evaluate the nectar amount of the source, the parameter that defines the number of food sources also defines the number of employed bees [11, 60, 61, 113].

Employed bees evaluate these food sources by determining their nectar amount [11,60,61,113]. The nectar amount is directly related to the fitness value calculated using a domain specific cost function [60,69,113]. After the amount is determined the employed bee advertises the food source to the colony by performing the waggle dance. In algorithm 5 lines 4 – 9 are where the nectar amount of a particular solution is calculated.

Onlooker bees witness a number of dances from a variety of employed bees [11,21,41,61,69]. They therefore need to select a food source that is the most attractive while maintaining some diversity in the pool of solutions. Thus, an onlooker bee selects a food source based on a probability function which is formulated in equation 5.6 [60]:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{5.6}$$

The parameter p_i is the ith food source under consideration by the onlooker bee. As discussed above, the fit_i parameter represents the value of the cost function and is directly related to the nectar amount of food source i. The parameter SN is the maximum amount of food source and hence the maximum employed onlooker bees [60].

In algorithm 5 the waggle dance can be seen being applied in line 11 where the probability of all the employed bee' solutions are calculated using equation 5.6. From lines 12 – 16 the onlooker bees evaluate the solutions of the employed bees based on the probability p_i that was calculated. P_i can be seen as the rating of the waggle dance that was performed by the employed bee.

Employed and Onlooker Bees

As previously outlined, when recruited onlooker bees reach the advertised food source that is stored in memory, they do not occupy the same food source [60, 61]. Instead the bees explore the immediate neighbourhood of the food source that was communicated to them [21,41,69]. They seek to find a food source that improves on the previous one [21, 60, 61, 113]. Equation 5.7 is used by the bees to generate new food sources in the neighbourhood of food source x_i [21, 41, 60].

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{5.7}$$

The subscripts $k \in \{1, 2, \ldots, SN\}$ and $j \in \{1, 2, \ldots, D\}$ are indices which are randomly chosen. D is the maximum dimensionality of the vector a solution represents. The index k has a constraint tied to it – whatever value is randomly assigned to k *must* differ from the value i. The position of the new food source in the neighbourhood of x_{ij} is controlled by the ϕ_{ij} parameter, which is a bounded random value between $[-1, 1]$.

From equation 5.7 it can be concluded that the randomness of the food source position decreases as the difference between $x_{ij} - x_{kj}$ decreases. Thus, as the algorithm moves closer to an optimal solution the finer grained the search process of the algorithm becomes [11, 60, 61].

After a new solution v_i is produced, the bee takes the new and old solutions from memory to compare their respective nectar contents. If the new solution is found to have higher quality nectar, the bee replaces the old solution in memory with the new solution [11, 21, 60, 69]. Otherwise, the bee abandons the new solution and keeps the old solution in memory [60,61,113]. Thus, the bee seeks to always move towards a better solution and therefore uses a greedy selection process [60, 69, 113].

One of the problems with the above approach is that little information about the food source is used in generating a neighbouring food source. In the research by Singh [113] a slight variation is proposed to generating food source neighbours by using more global information. The author adds a constraint to the algorithm that all neighbouring solutions generated by *employed* bees must be unique.

When an employed bee generates a neighbour and an identical solution already exists in the system, a *collision* is said to have occurred. A collision is solved by letting the employed bee transition to a scout bee so that a completely random solution can be generated [113]. Scout and onlooker bee generated solutions are not checked if they collide with other solutions in the system since the aim for them is exploring and not exploiting as with employed bees [21, 60].

Scout Bee

The artificial bees are modelled on the behaviour of real bees. Thus an employed bee can also abandon certain food sources when it has outlived its usefulness. Abandonment of a food source can occur for the following

reasons [11, 21, 61]:

- The employed bee has reached the maximum allowed cycles to improve the nectar amount. The maximum cycles spent on a food source allow the algorithm to avoid local optima [11, 60, 61].
- The bee cannot improve the solution represented by the food source any further [11, 60, 61].

When a food source in the algorithm is abandoned, it needs to be replaced by a new food source [11, 21, 60]. An employed bee under goes a role transition when it abandons a food source from an employed bee to a scout bee [11, 60, 61].

It is the responsibility of the scout bee to replace the abandoned food source with a new randomly generated one [11, 21, 60]. The scout bee uses equation 5.8 to produce a new food source that will replace food source x_i.

$$x_i^j = x_{min}^j + rand[0,1](x_{max}^j - x_{min}^j) \tag{5.8}$$

In research done by Gómez-Iglesias et al. [41] an extension is made to the scout bees. The scout bee individuals are divided into two types of bees, namely *rovers* and *cubs* bees [41].

- Rover bees are similar to traditional scout bees and hence use diversification strategies to explore the solution space.
- Cub bees explore the solution space relative to a good solution found by a rover by randomly changing configuration parameters.

By using two different scout bees a good balance is achieved when searching the solution space in the beginning where diversity is preferred and late in the algorithm where intensification is preferred [41].

5.4.4 ABC algorithm on the FAP

The ABC algorithm and all its variants are relatively new. To date it has only been applied to a select few problems such as the traveling salesman problem.

As yet, no research has been done to apply the ABC algorithm to the FAP. After critically evaluating the basic ABC algorithm listed in algorithm 5, the following obstacles can be identified if one were to apply the algorithm to the FAP:

Food source representation — Each food source can either represent a frequency plan or it can be a particular cell and a collective of food sources represents a frequency plan. If each food source is a frequency plan the algorithm will require a fair amount of memory, since as onlooker bees select it, they will start searching for neighbouring solutions. These neighbouring solutions are *also* frequency plans. However, if each food source is a cell, this would require less memory. The problem with the latter approach is that the bees would then single out one cell as the optimum, since they do not know that all food sources collectively represent a plan and each cell is actually unique.

Scout bee generation — When a food source is abandoned a scout bee needs to generate a new food source to take its place. In particular with the FAP, the newly generated solution cannot be completely random. The scout bee needs to incorporate knowledge already gained by the colony operating on different food sources, otherwise a completely random solution might not be even nearly lucrative enough for the rest of the bee colony to consider if it contains no knowledge gained by the algorithm.

Knowledge sharing — As a food source becomes more popular due to its high nectar amount, indicating a good fitness, more onlooker bees will select it to search in its neighbourhood for slightly better solutions. Therefore the bees are disregarding previously gained knowledge while searching for neighbouring sources on *other* food sources. If a food source represents a complete frequency plan, a previous food source a bee operated on might have had one or more cells that were assigned to their optimal frequencies, but due to the larger majority of the cells not being optimised, these *good* cells are overshadowed. Thus due to the *bad* cells overshadowing the good cells, the food source gets a low nectar amount, indicating a bad fitness, and therefore the bees might abandon the food source and those optimal cells are lost.

The above obstacles present real relevant challenges that would require new techniques to be developed. As the algorithm has not been applied to a wide variety of problems and taking into account the above obstacles, it is difficult to gauge if the ABC is well suited to be applied. In future, when the algorithm has been applied to a wider set of problems and has matured

in the variety of approaches to problems one can revisit the algorithm and attempt to apply it to the FAP.

In this section the various obstacles one would encounter when applying the ABC class of algorithms to the FAP were identified. This concludes the discussion on the ABC algorithm. The next section deals with the particle swarm optimisation algorithm.

5.5 Particle Swarm Optimisation (PSO)

Algorithm 6 Basic Global Particle Swarm Optimisation Algorithm [35]

```
1: Initialize s_n swarm
2: while Stopping condition not met do
3:    for each particle p_i ← 0 in s_n do
4:       Evaluate particle with fitness function f(p_i)
5:       if f(p_i) ≤ pbest(p_i) then
6:          personal best of p_i to f(p_i)
7:       end if
8:       if f(p_i) ≤ f(gbest) then
9:          gbest ← f(p_i)
10:      end if
11:   end for
12:   for each particle p_i ← 0 in s_n do
13:      update velocity of p_i with equation 5.9
14:      update position of p_i with equation 5.10
15:   end for
16: end while
```

5.5.1 Introduction

PSO is a self-adaptive, population-based stochastic search technique that was developed by Kennedy and Ebenhart in 1995 [111]. The basic model of the algorithm is based on simulations done to recreate the natural behaviour of a flock of birds [138].

In the early stages of the particle swarm development, simulations were developed to closely model the stigmergy (see page 98) exhibited when a

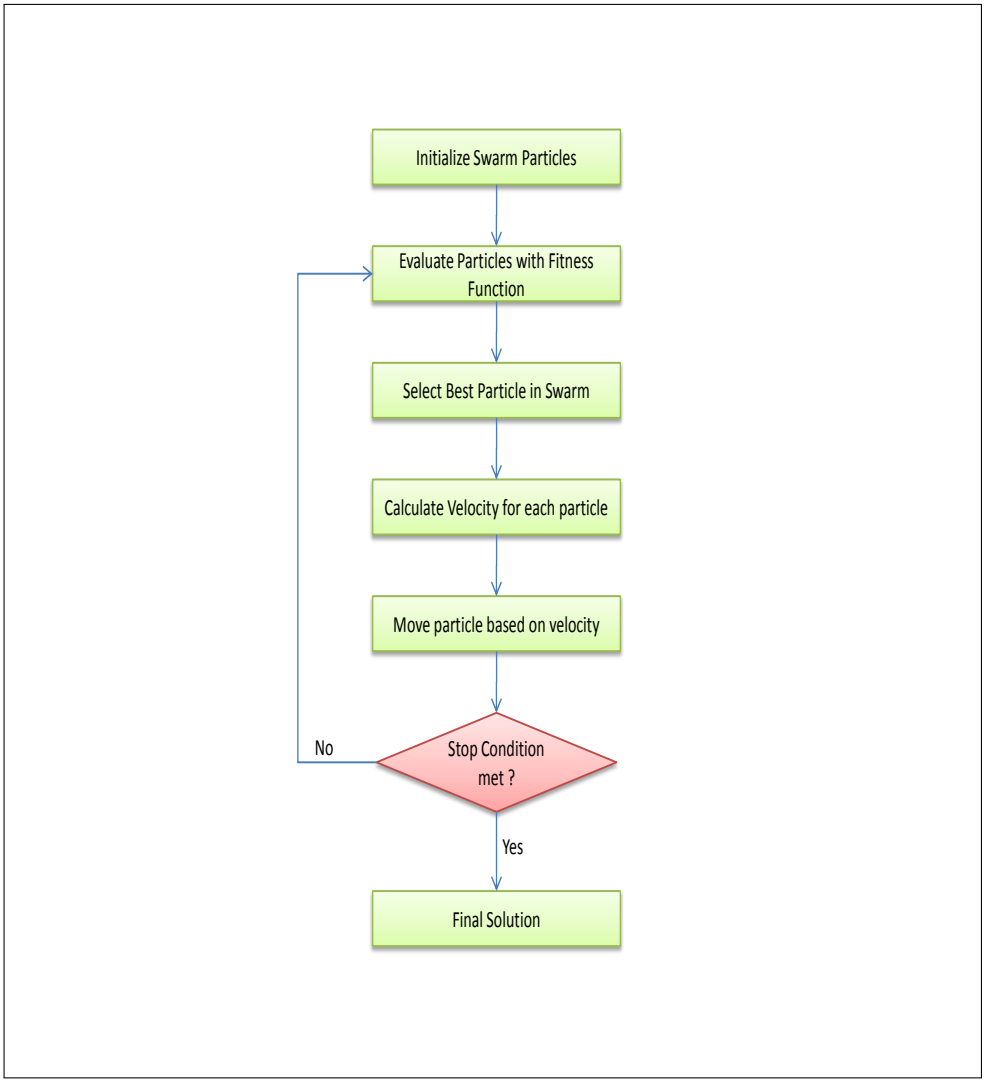

Figure 5.4: Flow chart for PSO algorithm

flock of birds cohesively move as one and are able to suddenly change direction in a unpredictable, graceful manner, only to regroup as one observed entity [57].

As the "leading" bird of the flock changes its movements the information is shared with all birds in the immediate vicinity of the leading bird. As the information is shared locally among birds, each bird modifies his own movement to that of the leading bird's movement [57].

Because birds obtain information by observing their neighbouring birds, the stigmergy can be deemed to be of a physical nature; therefore the particular stigmergy used by birds is sematectonic stigmergy (see page 98).

The simulations based on this behaviour of the flock allowed researchers to discover the underlying patterns that governed the way birds are able to share information about the general movement of the flock. Based on these patterns and simulations, the particle swarm algorithm emerged into an optimisation algorithm [35].

In the algorithm a particle is an individual. A group of particles, referred to in the literature as a swarm, fly through the solution space of the problem the algorithm is being applied to. Each particle changes its movement based on information shared with it by neighbouring particles in the swarm [34,35].

As information is shared among particles, the success of one particle ripples through the rest of the swarm and each particle is able to utilise shared information that leads to success of another particle. Thus, each particle's own personal experience and knowledge of the search space has an effect on its neighbouring particles [34, 35].

There exist two primary PSO algorithms: global PSO and local PSO. The only difference between the two algorithms is how they go about sharing information with the rest of the swarm. The sharing models of these two algorithms along with other sharing models will be discussed in the PSO characteristics subsection [94].

In the swarm, each particle is a potential solution that is encoded in a D-dimensional vector [57, 101]. As a particle moves through the solution space, it continually evaluates its current position and adjusts it accordingly to move in the general direction of the best particle of the swarm.

Depending on the sharing model used, the best particle in the swarm is denoted as either *gbest* of the global PSO or *lbest* for the Local PSO [34, 35, 94].

The global PSO uses a star neighbourhood to allow for information to be shared by everyone in the swarm. In contrast, the local PSO follows the process of natural birds more closely and uses the ring neighbourhood for information sharing. Hence, particles only share information with their immediate neighbourhood and not with the whole swarm.

A particle evaluates its current position by using a heuristic function, or in more evolutionary algorithm terms, a fitness function. The fitness value indicates to the particle how far it is from an optimal position [35].

As a particle moves through the solution space it keeps a memory of the personal best position it has achieved since the start of the algorithm. In the literature and in the algorithm this personal best position is referred to as *pbest* [94].

A particle moves with a certain velocity through the solution space. As the information is shared the particle must take advantage of the newly gained knowledge and therefore needs to adjust its own velocity to match the movement of the swarm. The particle updates its own velocity to move in the direction of the *gbest* shared position, its own *pbest* position and its current heading.

A particle needs to systematically explore the solution space; therefore when the particle needs to update its personal velocity, it does not use all the information it has available, otherwise it will start to cycle solutions. The particle uses a certain amount of global information together with a certain amount of local information to produce a direction and new velocity [34, 35, 94, 101].

The amount of global knowledge is referred to as the *social* component [34, 35, 94, 101] and is usually represented by the symbol c_1. The amount of personal information used by a particle is referred to as the *cognitive* component and is usually represented by the symbol c_2 [34, 35, 94, 101].

The PSO algorithm is quick to converge on an optimum, which might not necessarily be the global optimum [101]. Most of the research done on PSO has focused on the convergence of the algorithm as well as improving the diversity [34]. Most of these improvements and modifications will be discussed in the subsection on PSO characteristics.

5.5.2 Flow of the Algorithm

The general concepts that are evident in the PSO algorithm have been covered. Using these concepts a general overview will now be given of the PSO algorithm flow using algorithm 6 as a reference point.

The PSO algorithm starts off by initialising the swarm of particles. Each particle is randomly assigned a certain position in the problem space. After the swarm has been initialised the algorithm enters the optimization or search phase, which starts in line 2.

Before the swarm can start moving around in the problem space, it first needs to determine the gbest particle as well as each particle's own pbest position. Therefore as can be observed in line 3, each particle's current fitness $f(p_i)$ is determined using a problem-specific fitness function. The fitness determines the lucrativeness of the current position a particle occupies in the problem space.

Once the fitness of a particle's position is calculated, the algorithm needs to determine whether the current position of the particle is its pbest since the algorithm started. This comparison can be seen occur in line 5.

If the fitness of the currently held position is indeed better than the previous personal best of the particle, then the new position is remembered as the personal best for that particular particle, as can be observed in line 6.

Regardless of whether the personal best of a particle has been updated or not, the algorithm performs another comparison also utilising the calculated fitness of the current position of the particle. The algorithm uses this fitness to also determine whether the current position of the particle is the best in the *entire* swarm, i.e. whether it is the *global* best (gbest). This comparison occurs in line 8.

If the position of the particle is indeed the best position in the entire swarm, the algorithm replaces the current gbest with the position of the current particle being evaluated, as seen in line 9.

After the swarm has been evaluated, each particle should have a personal best and the swarm should have a global best. The swarm is therefore ready to move around in the problem space, which occurs in algorithm 6 from lines 12 – 15.

For each particle in the swarm the algorithm determines the particle's new velocity, as can be observed in line 13. The velocity of a particle is calculated using equation 5.9.

Once the velocity of a specific particle has been calculated, the particle is ready to move to a new position. Moving a particle from its current position to a new position using the calculated velocity is done by applying equation 6 and occurs in line 14 in algorithm 6.

After the whole swarm has been moved, the algorithm continues to the next iteration to evaluate the new positions. This process occurs until certain stopping criteria are met.

5.5.3 PSO Characteristics

Some of the characteristics of the PSO algorithm are *swarm size*, *particle velocity* and *inertia*.

Swarm Size

The swarm size dictates the search breadth the algorithm has to maintain each iteration [35, 144]. The initialising of particles in a swarm are the same as traditional population-based evolutionary algorithm use to initialise their respective populations [144]. At the start of the algorithm the swarm is initialised by randomly generating possible solutions that will represent the position of particles [35].

The PSO algorithm in some aspects resembles evolutionary algorithms like the genetic algorithm since it also has a population that operates in the problem space in search of an optimal solution. However unlike the GA the PSO does not continually generate new solutions to be reinserted into the swarm to increase diversity [124]. Thus the swarm size needs to be adjusted to get an optimal representation of the search space because as particles move in the swarm, the diversity among particles decreases rapidly as information is shared [34, 35].

Diversity decreases as the swarm moves because, with each iteration, the swarm converges towards the global best position [34, 35, 57] since the velocity equation directs the general movement of a particle in the direction of the global best [34, 35, 57]. The velocity equation will be discussed below [34, 35, 57]. If the global best position does not change as the algorithm

processes more iterations, a larger portion of the swarm will soon occupy the same position as the global best [34, 35, 57].

A large swarm might increase diversity, but at the expensive of computational time since the algorithm has to spend more time evaluating the fitness of particles [34, 35]. In a small swarm, there might be low diversity, but the algorithm requires little computational effort to evaluate the fitness of particles and move the swarm around in the solution space [34, 35].

Small swarms usually have a population that ranges from 10 – 5 000. Big swarms typically have a population of ≥ 10000 [58, 67, 101].

Diversity of the swarm lowers because as the swarm progresses, particles tend to converge toward the best particle shared with them. If no particle is able find a better position on its way towards the global position shared and the best particle is not able to improve its own position, then eventually all particles will converge on the best position [34, 35].

With a small swarm the algorithm will converge faster to an optimum, which is not guaranteed to be the global minimum [34, 35]. Therefore care must be taken with the selection of the swarm size, because if a large swarm size is selected the algorithm will be slower to converge to an optimum. It will, however, gain diversity, which allows for a higher probability of finding the global optimum [87].

Particle Velocity

The velocity calculation of each particle is where the optimisation procedure occurs in the PSO algorithm. It is the only means by which the PSO algorithm searches the solution space, since it is the only means by which particles are moved to new positions in the search space [35].

The velocity update is where the personal experience of a particle and the knowledge gained through social sharing are incorporated, to steer a particle into a certain direction by modifying its velocity. The most basic function to update a particle is formulated in equation 5.9.

$$v_i(t+1) = v_i(t) + c_1\phi_1(t)[pbest - x_i(t)] + c_2\phi_2(t)[gbest - x_i(t)] \quad (5.9)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (5.10)$$

where $v_i(t+1)$ is the new velocity of particle i for the next time step $t+1$. The cognitive component is represented by parameter c_1 and the social component is represented by the parameter c_2 (discussed on page 127) [34, 35].

Each of the respective components c_1 and c_2 controls how much information is used in the calculation of the new velocity.

The variables ϕ_1 and ϕ_2 are random numbers in the range $[0, 1]$. The current position of a particle in the solution space at time step t is represented by parameter $x_i(t)$ [34,35]. After the new velocity is calculated the position of the particle is updated for time step $t + 1$ using equation 5.10 [34, 35]. The velocity update can be visually depicted as shown in figure 5.5.

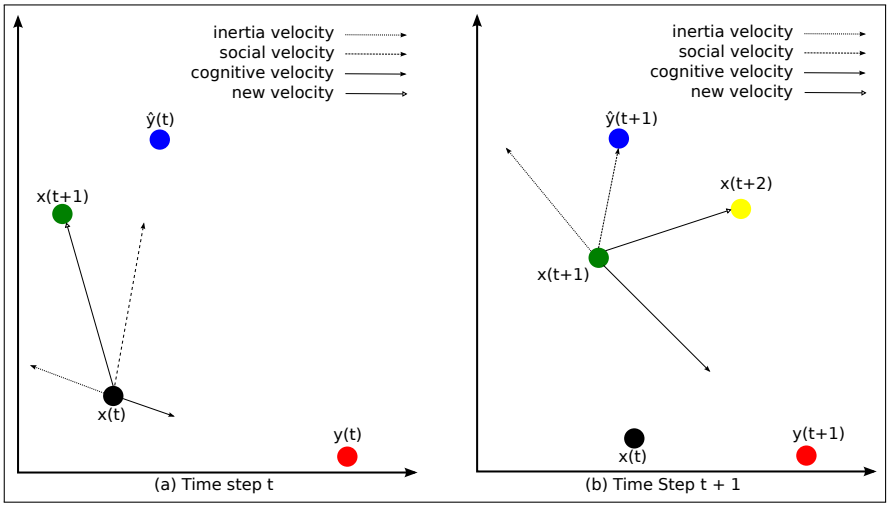

Figure 5.5: Visual particle velocity update [34, 35, 94, 101]

As discussed earlier gbest is the best position the particle has occupied since the start of the algorithm. In the literature there are two defined methods of determining gbest. The most common method used is where gbest is the best position obtained by a particle in the swarm since the start of the algorithm; thus long-term knowledge dictates the best position found which favours exploitation [34, 35].

The second method of determining the gbest is the best particle position occupied by any particle in the swarm in the *current* iteration of the algorithm; thus short-term knowledge dictates the best position found which favours exploration [34, 35].

As can be observed in equation 5.9 the new velocity is added to the old velocity. The velocity of particles can get very large, especially for those particles that are far from the pbest and gbest positions. Large velocities are necessary for early exploration, but if the velocity gets too large, the rate at

which the particle moves in the solution space is too high and good solutions might be missed [34]. Thus the velocity of a particle needs to be clamped to ensure its step size stays within acceptable bounds. Equation 5.11 is used to clamp the velocity of a particle *before* its position is updated [34].

$$v_i(t+1) = \begin{cases} v_i'(t+1), & \text{if } v_i'(t+1) < V_{max} \\ V_{max}, & \text{if } v_i'(t+1) \geq V_{max} \end{cases} \tag{5.11}$$

$$V_(max) = \delta(x_{max} - x_{min}) \tag{5.12}$$

Where V_{max} is the maximum allowed velocity and $\delta \in (0, 1]$. The values x_{max} and x_{min} are the respective minimums and maximums of the domain the algorithm is being applied to [34]. The value of δ is very problem dependent and must be carefully chosen to maximise the exploration-exploitation trade-off [34].

Most of the literature has concentrated on the velocity of the particle because it is the main function performing the optimisation. In research done by Ratnaweera et al. [101] particle positions in the solutions space are continually monitored. If the particle appears to be stagnant in the solution space, the velocity is first updated, and then the particle is reinitialised with a random position. The new position of the particle is then updated with the new velocity, thus knowledge of the discarded particle is retained by using the velocity it had [101].

In research done by Kalivarapu et al. [58] a PSO algorithm is developed that seeks to incorporate the pheromone notion of ACO algorithms into the velocity updating of particles. The premise of the method is to allow greater sharing of information about promising areas between particles. The algorithm developed by the authors achieved promising results, with it in certain cases finding solutions faster and also better solutions than other PSO algorithms [58].

Other research done by Monson and Seppi [84] is more concerned with how the particle is presented. In the general PSO algorithm, particles have no physical form or volume and so particles in the swarm move through each other. The authors changed this in their algorithm by letting each particle have a radius around itself. This means that as particles move through the solution space and another particle at a certain time step occupies the same space, the particles are said to collide. As one would expect, when a collision

occurs both particles are deflected into random directions [84]. At a greater expense of computational time due to constant collision detection, the PSO gains greater exploration in the solution space.

Finally, in the research by Lenin and Monan a PSO algorithm is developed that is called the attract and repulse PSO (ARPSO). The algorithm continually monitors the solutions in the swarm. If it picks up that a certain percentage of the swarm is stagnating, it activates the repulse state. In the repulse state particles are repelled from other particles in the swarm, which facilitates greater exploration. After a certain number of iterations, the algorithm returns to its default state, where particles attract each other. The state of attraction facilitates exploitation [67].

Inertia Weight

As an object moves with a certain velocity it carries momentum. If the object were to suddenly change direction, momentum would for a certain period still move the article in the previous direction. Inertia weight seeks to add type of behaviour to the particles of the PSO algorithm. It was initially developed to negate the use of clamping a particle velocity. The velocity update equation with added inertia is formulated in equation 5.13 [34].

$$v_i(t+1) = wv_i(t) + c_1\phi_1(t)[pbest - x_i(t)] + c_2\phi_2(t)[gbest - x_i(t)] \quad (5.13)$$

The addition of inertia (w in equation 5.13) to the general velocity update equation is simple and elegant, which is why it has been adapted to a wide variety of PSO algorithms. The inertia value allows one to regulate the amount of control the social and cognitive components have with regard to velocity updates [34].

For values of w that are greater than 1, a large amount of momentum is preserved as the particle's velocity is updated, and the particle explores more [34]. When $w < 1$, each time the particle updates its velocity it loses a certain amount of momentum. Hence the particle seems to slow down, allowing it to exploit the current solution space in finer detail [34].

Even though inertia was developed to remove the use of velocity clamping, it did not entirely achieve its goal. For values of w that are greater than 1, the particle keeps a lot of momentum and accelerates even more. Therefore as the particle accelerates, its step size through the solution space

becomes larger and larger, increasing the probability that it will miss a good solution. This disadvantage is only applicable for algorithms that keep the inertia value static [34, 35].

To allow for a greater trade-off between exploration and exploitation, the inertia value was made dynamic. Exploration is favoured early on in an optimisation algorithm and exploitation later on the algorithm when it is near an optimum; thus various methods that are either linear decreasing or non-linear decreasing have been developed that modify the inertia component as the algorithm moves around in the solution space [34, 35].

Finally, a similar inertia type component was developed from the analysis of particle dynamics [34]. This new component is called the *constriction coefficient* and, like the inertia above, also modifies the velocity update equation slightly [34, 35, 84].

This modification can be observed in equation 5.14, which is the standard velocity equation with the constriction coefficient. The constriction coefficient is formulated in equation 5.15 [34, 35, 84].

$$v_i(t+1) = \chi[v_i(t) + c_1\phi_1(t)[pbest - x_i(t)] + c_2\phi_2(t)[gbest - x_i(t)]] \quad (5.14)$$

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \quad (5.15)$$

The constriction coefficient is represented by the value *phi* and allows one to omit the usage of velocity clamping. The constriction coefficient evaluates to an ever-decreasing value between $[0, 1]$. By using the constriction coefficient the PSO algorithm is also guaranteed to converge for values of $phi \geq 4$ and $\kappa \in [0, 1]$. As with the inertia discussed above, high values of κ allow for greater exploration and slow convergence, whereas low values of κ force the algorithm to exploit the solution space and converge quickly [34, 35, 84].

5.5.4 PSO on the FAP

The PSO algorithm is also a relatively new algorithm and has been applied to only a handful of NP-Complete problems, including the FAP. In this dissertation the PSO algorithm will be utilized on the FS-FAP to try and produce optimal solutions.

Only two groups have conducted research where the PSO has been applied to the FAP to produce a near optimal solution. The research concentrated on the MS-FAP variant of the FAP, and so the aim of their algorithm

was to reduce the span of frequencies used. The problem this dissertation is concerned with is the FS-FAP where the amount of interference generated needs to be minimised.

To date, no PSO algorithm has been designed to operate on the FS-FAP variant. Therefore, the interest in the research mentioned above is more to do with how the authors went about encoding a particular frequency plan as a position for a particle, than with the actual optimisation procedure.

In the research presented by Elkamchouchi et al. [33] a PSO algorithm is applied to produce optimal solutions for the MS-FAP. The way the authors went about assigning frequencies in their algorithm is known as frequency exhaustive assignment (FEA). This method works by first generating a list of calls, called a *call list*, denoting calls that occur in the system [33].

The method then iterates over the calls in the list and assigns the lowest possible frequencies to the calls without violating interference constraints [33]. The authors note that the specific frequency that is assigned to a particular call depends heavily on the order the calls are in the list [33].

Because of the success of the PSO on the MS-FAP, for this dissertation the PSO algorithm was selected as the primary means by which to address the FAP. Other factors also influenced the final decision to utilize the PSO instead of another algorithm.

The PSO algorithm is short and elegant. As mentioned earlier, it utilises the "flying" approach to search the problem space. With this approach one point progressively moves towards another. Using this approach, the algorithm is able to adequately explore the problem space much more thoroughly.

The algorithm also makes extensive use of knowledge gained by the various particles as they search the problem space. Not only does each particle keep personal history (with pbest), but the swarm as a whole keeps a history of the best particle (with gbest). Thus with regard to FAP, it is possible that even though a particle might be in an overall bad position, it might have some small bit of good knowledge being overshadowed by bad knowledge. Through the extensive use of historical knowledge good information is more likely to be shared or kept slightly longer in the algorithms collective knowledge.

For this current research the PSO was applied to the FS-FAP and thus the approach by the authors in the above literature could not be used. FS-FAP is concerned with interference generated and there are some constraints

which cannot be broken, wheres for the MS-FAP, the performance measure is explicitly the number of constraints violated.

The PSO algorithm might be short and elegant, but applying it to the FS-FAP required various new techniques in response to the following questions:

- How can a particle best be represented as a frequency plan?
- How can one frequency plan "fly" to another?
- How can particles be prevented from using forbidden frequencies when they fly towards a particular plan?

The above questions were only the preliminary questions and were in fact a problem that had to be addressed for successful application of the PSO to the FAP. In chapter 7 a discussion will be presented on how these problems were solved as well as how other problems were solved that were not anticipated.

5.6 Summary

In this chapter three swarm intelligence algorithms were discussed. The general flow of the ant colony optimisation algorithm was described with the help of a diagram (see figure 5.1) as well as how the algorithm came about. The defining characteristics of the algorithm were identified and a literature review was given of the ACO being applied to the FAP.

The second swarm intelligence algorithm was the artificial bee colony optimization algorithm. How the algorithm was developed and how it performs its search in a problem space were explained. A diagram also outlined the general flow of the ABC algorithm.

A series of defining characteristics was explained. Each characteristic is a defining attribute of the algorithm that makes it unique with regard to other algorithms. No literature is available on the algorithm being applied to the FAP since to date no research has been conducted on such an ABC algorithm.

This chapter concluded with the most important algorithm, which is used in this dissertation on the FAP, namely the particle swarm optimization

algorithm. The flow of the PSO algorithm was illustrated (see figure 5.4). Furthermore, characteristics that make the algorithm unique were explained, and a literature review was given of the PSO algorithm being applied to the FAP.

Part II

Implementation

Chapter 6

PSO on Benchmark Functions

6.1 Introduction

This chapter deals with a series of optimisation benchmark functions. First all the benchmark functions are formulated and briefly classified. Finally this chapter will conclude with a presentation of the results obtained by the PSO algorithm.

The functions vary from being relatively easy to optimise, to functions that contain numerous local minima and slightly concealed global minima. In total 16 benchmark functions are formulated and benchmarked.

Various optimisation algorithms are benchmarked such as GA, ABC algorithm, TS and SA [34, 35, 133]. The algorithms were benchmarked using numerical optimisation functions.

Numerical optimisation functions are good candidates to test optimisation algorithms as, with a few slight changes, the function operates in more dimensions or can have more or less optima [34, 35, 133]. Being able to alter these functions is a desirable trait as it enables one to accurately benchmark an algorithm, not only with regard to how the algorithm handles the increased dimensionality, but also in the algorithm's accuracy in locating optima [34, 35, 133].

The reason why these functions have variable numbers of local and global optima, is to test various factors on how good the algorithm is that is being applied to the function. The factors that are tested are [34, 35]:

- Rate of convergence
- Exploration
- Exploitation
- Diversity
- Breaking out of local minima
- Information sharing

As discussed previously, the numerical functions have predetermined optima, which means researchers are able to produce statistical information on how the algorithm performs. For instance, researchers will not be able to measure accurately the performance of the algorithm on an NP-Complete problem [34, 35]. They can of course compare results with what other algorithms have produced, but cannot with absolute certainty say or measure the algorithm convergence, diversity etc. on NP-Complete problems as their problem spaces are huge [137].

With these benchmark functions, the optima have been mathematically calculated and their position is known within the problem space [137]. Thus, researchers can now with certainty measure the convergence rate and diversity and compare them with other algorithms, since the domain in which the algorithms operate it is not as specific as an NP-Complete problem [137]. Rather, the domain is mathematical and deterministic and therefore allows easy comparison [137].

In this current study, two PSO algorithms were developed specifically to measure the performance of the PSO algorithm and also to better understand the various underlying dynamics of the algorithm.

The first PSO that was developed was the standard PSO algorithm with no constriction coefficient or inertia weight. The second PSO algorithm differed in that it utilises the notion of inertia to move particles.

Both of these algorithms have been applied to all 16 benchmarks that are presented in the chapter and will be compared with each other to determine the benefit of adding inertia weight to the algorithm performance. The comparison of these algorithms appears in section 6.3.

In the next section, all the benchmarks that were used for testing the PSO algorithms are formulated. For the interested reader, 3D graphs along

with the python code that generated the graphs appear in the appendix. This chapter will conclude with the results of the PSO algorithms that were applied to the benchmarks and comparisons with other results that have been obtained by other algorithms.

6.2 Test Functions

For each test function a mathematical formulation is given and the global optimum is explicitly stated. In addition, each function is also classified as a unimodal or multimodal function, separable or non-separable, as explained below.

Unimodal — A particular problem is classified as being unimodal when there is only one clear solution. With only one clear solution, it means there is only one global optimum point in the solution space [34, 35, 62, 137].

Multimodal — A problem is multimodal when it has more than one defined solution; thus the particular problem space contains multiple global optima [34, 35, 62, 137].

Separable — These functions can be written as a series of summations of just one variable [62]. This quality makes the function easier to solve as the algorithm has only one variable to be concerned about [62,137]. Separable functions also have the inherent quality of being scalable, meaning that they can easily be adapted to higher dimensions [62,137].

Non-separable — Functions classified as non-separable cannot be rewritten into a series of summation functions, as the variables used in the functions have the characteristic of being interrelated [62, 137]. The interrelation of the variables makes non-separable functions more difficult to solve than separable functions since the algorithm has more interdependent variables to be concerned about [62, 137].

The DeJong test functions (F1, Shekel's Foxhole) are not considered to be the gold standard of testing optimisation algorithms [137]. The only reason for their extensive use in the literature is that they were the first to be developed and applied to test an optimisation algorithm, i.e. GA [133,137].

Since the inception of the De Jong test functions, additional functions have been developed which make it more difficult for an optimisation algorithm to locate the optimum [137]. These functions are more difficult because they have multiple local optima, which does in actual fact leads the algorithm astray, i.e. the problem space is deceptive [34, 35, 137].

Problems that have deceptive search spaces test how well the algorithm is resistant to hill-climbing[1], and hence how efficient the algorithm is in exploring the entire search space [137].

6.2.1 DeJong F1 Function

$$f(x) = \sum_{i=1}^{n} x_i^2, -5.12 \leq x_i \geq 5.12, i \in \mathbb{N} \tag{6.1}$$

The DeJong F1 function has the following global minimum when $f(x) = 0, x(i) = 0, i : n$ where n is the number of dimensions [59, 60, 62, 63, 83, 104]. In the literature the function is classified as being unimodal and separable [60, 83].

A graph of this equation is presented in the appendix, section A.1, page 220.

6.2.2 Shekel's Foxhole

$$f(x_1, x_2) = \{0.002 + \sum_{j=1}^{25} [j + (x_1 - a_{1j})^6 + (x_2 - a_{2j})^6]^{-1}\}^{-1} \tag{6.2}$$

where

$$a = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & ... & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & ... & 32 & 32 & 32 \end{pmatrix}$$

the variables x_1 and x_2 are usually restricted to the square represented by $-65.356 \leq x_1 \leq 65.357, -65.357 \leq x_2 \leq 65.356$ [19, 60, 63, 83]. The global optimum is when $f(x_1, x_2) = 0, \{x_1, x_2\} = \{-32, -32\}$ [19, 60, 63, 83].

The matrix controls the holes that appear in the search space. The interested reader is directed to the 3D graph rendering of this function in the appendix, section A.2 on page 220. In the literature the function is classified being multimodal and separable [60, 83, 84].

[1] Continously selecting what seems to be better moves, but in reality moving towards a local optima peak

6.2.3 Rastrigin

$$f(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)],\ i \in \mathbb{N} \tag{6.3}$$

The values of the variable x_i are bounded by the hypercube $-5.12 \leq x_i \leq 5.12$ [55, 59, 60, 62, 83, 84, 104]. The global optimum for the function is when $f(x_i) = 0$, $x_i = 0$, $i = 1, \ldots, n$ [55, 60, 62, 83, 84].

Rastrigin's function is based on DeJong's first function equation 6.1, adding a cosine term, which in turns alters the problem space by introducing many local minima [55, 59, 62, 83]. In the literature the function is classified being multimodal and separable [4, 55, 59, 60, 62, 83, 84, 104].

A graph of this equation is presented in section A.3 page 221.

6.2.4 Schwefel

$$f(x) = 418.9829n - \sum_{i=1}^{n}[x_i \sin\sqrt{|x_i|}], \qquad i \in \mathbb{N} \tag{6.4}$$

The variable x_i is restricted to be in the hypercube $-500 \leq x_i \leq 500, i = 1, \ldots, n$ [39, 55, 60, 62, 83]. The global optimum for the function is $f(x) = 0$ when $x_i = 420.9687$ [39, 55, 60, 62, 83].

If one observes the 3D rendering of the Schwefel function problem space in the appendix, section A.4 on page 221, one can see that the search space contains a great number of peaks which might be local optima. The function also has the characteristic of having a second best optimum far from the global optima in which many algorithms became trapped [39, 55, 60, 62, 83]. In the literature the function is classified as being multimodal and separable [55, 60, 62, 83].

6.2.5 Griewank

$$f(x) = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \qquad i \in \mathbb{N} \tag{6.5}$$

The variable x_i is bounded to within the hypercube $-600 \leq x_i \leq 600$ [59, 60, 62, 63, 83, 104]. The global optimum of the function is when $f(x) = 0$, which occurs when $x_i = 0, i = 1, \ldots, n$ [59, 60, 62, 63, 83, 104].

As with the Schwefel function, the Griewank function also has a great number of peaks and valleys in which many algorithm become trapped. A

particular quality of the Griewank function is that at low dimensions, the function is quite difficult to solve. It has been shown that at higher dimensions the function becomes much easier due to there being fewer peaks and valleys to navigate in the search space [59, 60, 62, 83, 137]. In the literature the function is classified as being multimodal and non-separable [4, 59, 60, 62, 83, 84].

A graph of this equation is presented in the appendix in section A.5 page 222.

6.2.6 Salomon

$$f(x) = -\cos\left(2\pi \sum_{i=1}^{n} \sqrt{x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2} + 1, \quad i \in \mathbb{N} \tag{6.6}$$

Unlike the previous functions discussed in this section, the Salomon function imposes no constraint on the x_i variable. The global optimum is when $f(x) = 0$ and $x_i = 0$ where $i = 1, \ldots, n$. This particular function does not seem to have been applied as a benchmarking function yet, as no literature can be found. Nonetheless, the function is indeed a numerical optimisation function that is classified as being multimodal and non-separable [1].

A graph of this equation is presented in the appendix in section A.6 page 222.

6.2.7 Ackley

$$f(x) = -20e^{-0.2\sqrt{\frac{1}{2}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{2}\sum_{i=1}^{n} \cos 2\pi x_i} + 20 + e^1, \qquad i \in \mathbb{N} \tag{6.7}$$

The variable x_i is restricted to the hypercube represented by $-32.768 \leq x_i \leq 32.768$ [60, 62, 83, 104]. The global minimum is when $f(x) = 0$ and is obtainable for $x_i = 0, i = 1, \ldots, n$ [60, 62, 83, 104].

As can be observed from the mathematical formulation of the function, the function utilises an exponential term. By using the exponential term the problem space contains numerous local optima which require an algorithm to search much wider to avoid getting trapped. The literature classifies this function as being multimodal and non-separable [60, 62, 83, 84].

A graph of this equation is presented in the appendix in section A.7 page 223.

6.2.8 Six-hump Camel Back

$$f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^{\frac{4}{3}})x_1^2 + (x_1x_2) + (-4 + 4x_2^2)x_2^2 \tag{6.8}$$

The variables x_1 and x_2 are subject to the following boundary constraints: $-3 \leq x_1 \leq 3$ and $-2 \leq x_2 \leq 2$ [39, 83]. The global minimum is when $f(x_1, x_2) = -1.0316$ and is obtained when $x_1 = -0.0898$ and $x_2 = 0.7126$ or when $x_1 = 0.0898$ and $x_2 = 0.7126$ [39, 83].

True to its name the function has six peaks, four of which are local minima and two of which are global minima as has already been defined. The literature classifies this function as being multimodal and non-separable [60, 83].

A graph of this equation is presented in the appendix in section A.8 page 223.

6.2.9 Shubert

$$f(x_1, x_2) = -\sum_{i=1}^{5}(i\cos(i+1)x_1 + 1)\sum_{i=1}^{5}(i\cos(i+1)x_2 + 1) \tag{6.9}$$

The search domain is constrained to $-10 \leq x_i \leq 10, i = 1, 2, \ldots, n$ [19, 60, 63, 83]. The global optimum is when $f(x_i) = -186.7309$ [19, 60, 63, 83].

As can be observed from the 3D graph in the appendix, section A.9 on page 224 the landscape of the Shubert function contains various peaks and slopes. Therefore the function is classified as being multimodal and non-separable [60, 83].

6.2.10 Himmelblau

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{6.10}$$

The variables x_1, x_2 are constrained to be within the hypercube represented by $-6 \leq x_1 \leq 6, -6 \leq x_2 \leq 6$ [60, 83]. The Himmelblau function contains no local optima, but on the contrary, it has 4 global optima when $f(x_i) = 0$ which can be obtained at the following points [60, 83]:

- $(x_1, x_2) = (-3.779310, -3.283185)$
- $(x_1, x_2) = (-2.805118, 3.131312)$
- $(x_1, x_2) = (3, 2)$

- $(x_1, x_2) = (3.584428, -1.848126)$

The literature classifies the function as being multimodal and non-separable [60, 83].

A graph of this equation is presented in the appendix in section A.10 page 224.

6.2.11 Rosenbrock Valley

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \tag{6.11}$$

The variable x_i is bounded to the following constraint $-2.048 \leq x_i \leq 2.048$ [55,59,60,62,104,122]. The global optimum is when $f(x) = 0$ and is obtained when $x_i = 1, i = 1, \ldots, n$ [55, 60, 62, 104, 122].

The Rosenbrock search space has a curving valley which forces algorithms to cope with the changing direction of the landscape [4, 55, 59, 60, 62]. If the algorithm does not adapt to the changing direction it will fail in locating the global optimum. The function is classified as being multi-modal and non-separable [4, 55, 59, 60, 62].

A graph of this equation is presented in the appendix in section A.11 page 225.

6.2.12 Dropwave

$$f(x) = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{\frac{1}{2}(x_1^2 + x_2^2) + 2} \tag{6.12}$$

The variables x_1 and x_2 are restricted to be within the following bounds: $-5.12 \leq x_i \leq 5.12$ [83]. The landscape of this function resembles that of a droplet falling into a pool of water, as can be observed from the 3D graph presented in the appendix in section A.12 on page 225. Due to its "wave" nature, the function has various local minima and only a single optimum which is when $f(x_1, x_2) = 0$. The function is classified being multimodal and non-separable [83].

6.2.13 Easom

$$f(x_1, x_2) = -\cos(x_1)\cos(x_2)e^{(-(x_1-\pi)^2-(x_2-\pi)^2)} \tag{6.13}$$

The variables x_1 and x_2 are restricted to be within the hypercube represented by $-100 \leq x_1 \leq 100, -100 \leq x_2 \leq 100$ [63, 83, 122]. The global minimum is when $f(x_1, x_2) = -1$ and is obtainable if $(x_1, x_2) = (\pi, \pi)$ [63, 83, 122].

The Easom function has a deceptive global minimum as it is very close to other local minima [19, 60]. As can be observed from the 3D graph of the function in the appendix in section A.13 on page 226, the search space is a flat with the global minima clearly visible. Flat search spaces are difficult to navigate by algorithms as the surrounding area gives no indication whether the algorithm is on the right track or not. The function is classified as being multimodal and non-separable in the literature [19, 60, 122].

6.2.14 Branins

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos x_1 + e \tag{6.14}$$

where

$$\begin{aligned} a &= 1 \\ b &= \frac{5.1}{4\pi^2} \\ c &= \frac{5}{\pi} \\ d &= 6 \\ e &= 10 \\ f &= \frac{1}{8\pi} \end{aligned}$$

The variables x_1 and x_2 are subject to the following boundary constraints: $-5 \leq x_1 \leq 10, 0 \leq x_2 \leq 10$ [19, 60, 63, 83]. The global optimum is when $f(x_1, x_2) = 0.397887$ and is obtainable when x_1 and x_2 have the following values [19, 60, 63, 83]:

1. $x_1 = -\pi,\ x_2 = 12.275$
2. $x_1 = \pi,\ x_2 = 2.275$
3. $x_1 = 9.42478,\ x_2 = 2.475$

The literature classifies this function as being multimodal and separable [19, 60, 63, 83].

A graph of this equation is presented in the appendix in section A.14 page 226.

6.2.15 Michalewicz

$$f(x) = -\sum_{i=1}^{n} \sin(x_i)[\sin(\frac{(1-x_i^2)}{\pi})]^{2m}, \qquad i, m \in \mathbb{N} \tag{6.15}$$

The variable x_i is usually constricted to the following defined boundary: $0 \leq x_i \leq \pi, i = 1, \ldots, n$ [60, 83]. The parameter m defines the steepness of the valleys in the function. The function has two approximated global minima that are difficult to locate since their size in comparison with the rest of the search space is relatively small [60, 83].

1. $f(x) = -4.687,\ n = 5$
2. $f(x) = -9.66,\ n = 10$

In the literature the function is classified as being multimodal and separable [60].

A graph of this equation is presented in the appendix in section A.15 page 227.

6.2.16 Goldstein

$$\begin{aligned} f(x_1, x_2) &= (1 + (x_1 + x_2 + 1)^2) \\ &= *(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \\ &= *(30 + (2x_1 - 3x_2)^2 \\ &= *(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2) \end{aligned}$$

The variables x_1 and x_2 are subject to the following boundary constraints: $-2 \leq x_1 \leq 2, -2 \leq x_2 \leq 2$ [19, 60, 63, 83, 122]. The function has only one global minimum and four local optima [19, 60]. The global optimum is when $f(x_1, x_2) = 3$ and is obtainable when $x_1 = 0$ and $x_2 = -1$ [19, 60, 63, 83, 122].

The literature classifies this function as being multimodal and non-separable [60].

A graph of this equation is presented in the appendix in section A.16 page 227.

6.3 Results

In the previous section 16 benchmark functions were mathematically defined and, where applicable, comments were given on the search space. For each function the following was explicitly stated:

- The global optimum and where it is located
- Whether the function is unimodal or multimodal
- Whether the function is non-separable or separable

In this section the results are given of how the two PSO algorithms that were developed performed on each of the benchmark functions.

The algorithms will be compared with the following criteria for each benchmark:

- The number of iterations the algorithm took to find the global optimum
- The PSO is a population-based algorithm, thus diversity is important. Diversity will be measured and compared for each benchmark function.

For all the results the algorithm was applied to the various test functions for 1 000 iterations. The limit on the number of iterations was chosen as it was observed through various tests that the algorithms were not able to find any measurable improved solutions after the 600th iteration. The additional 400 iterations were given as a buffer period in case the algorithm was indeed able to obtain a better result.

The following are the final resulting values the algorithms obtained after operating on the given problem for 1 000 iterations. The algorithms were written in C++ and were executed on the following configuration

- Intel Q9700 operating at the manufactured speed of 3.167 GHz
- 4 gigabyte of DDR2 RAM
- Ubuntu 11.04, gcc-4.2, compiled with -O3 optimisation

First, the actual final fitness values obtained by the algorithms will be discussed, followed by the diversity of the particle swarm as measured at the very last iteration of each algorithm. In each table the column with the heading "PSO" refers to the results obtained with the standard PSO algorithm. The column with the heading "PSO*" refers to the results obtained with the PSO algorithm that uses inertia with particle movement.

The inertia value utilised by the PSO* algorithm was set to 0.5 for these tests as it was found that 0.5 obtained the best results. It must also be noted that any number smaller than e-08 is regarded as zero as the number is very close to zero and can be regarded as such. For accuracy the results presented have been left as is.

6.3.1 Final values

Function	PSO	PSO*
DeJong F1	0	0
Shekel's Foxhole	0	0
Rastrigin	0.74622	0.74622
Schwefel	-1.19069e+43	-908.013
Griewank	1.00003	1
Salomon	-0.460561	-0.80003
Ackley	-4.5901	-4.5901
Six-hump Camel back	-1.03163	-1.03163
Shubert	-186.731	-186.731
Himmelblau	1.02103	0
Rosenbrock Valley	1.81101e-08	0
Dropwave	4.59843e-15	0
Easom	-1	-1
Branins	1.22875	0.397887
Michalewicz	-2.40391	-2.40391
Goldstein	infinity	infinity

Table 6.1: Final fitness values

In table 6.1 it can be observed that the standard PSO algorithm found the global optimum for the following benchmark functions:

1. DeJong F1
2. Shekel's Foxhole
3. Six-hump Camel Back
4. Shubert
5. Easom
6. Rosenbrock Valley
7. Dropwave

In all the other benchmark functions the standard algorithm clearly become trapped in local optima, where the optima were located extremely close to the global optima in the search landscape. One can observe from the 3D graph renderings presented in the appendix that some of the functions exhibited a search landscape that contained a lot of hills visually. These hills can also be interpreted as local optima and were specifically designed to exist within the search landscape to trap optimisation algorithms.

If one looks at the number of benchmark functions the algorithm was applied to and the number of test functions successfully solved, the performance of the algorithm seems quite below par. Statistically the standard PSO algorithm developed has a relative success rate of 43.75%.

This success rate of the standard algorithm is improved when one compares the algorithm with the PSO algorithm that uses inertia. Indeed, the PSO algorithm solves the following functions:

1. DeJong F1
2. Shekel's Foxhole
3. Six-hump Camel Back
4. Shubert
5. Easom
6. Himmelblau
7. Rosenbrock Valley

8. Dropwave

9. Branins

The PSO* algorithm solved two more benchmark functions than the standard PSO algorithm. It must also be noted that, for the following functions, the PSO* also obtained better results, which were thus much closer to the global optimum results:

1. Schwefel

2. Griewank

The observed reason why the PSO* algorithm was able to solve more test functions and get closer to the global optimum is inertia. With inertia, the particles of the swarm search the landscape much more thoroughly. Because the algorithm does not clamp the velocity of particles, the velocity of particles quickly becomes huge. With a high velocity on a particle the particle no longer takes systematic small steps towards another particle; rather the particle takes huge strides in the search space skipping entire regions. It is for this reason that two variations exist in the PSO algorithm: One that uses velocity clamping and one that uses inertia. The inertia variant was selected due to the ease of implementing the alteration on the algorithm as well as the success the variant of the algorithm has achieved in the current literature.

It is interesting to note that both algorithms completely failed to find a solution in the search landscape depicted by the Goldstein function. This can be attributed to the particular landscape of the function being very misleading and one can see by looking at the 3D graph rendering of the function (see page 227 in the appendix) that the global optimum (bottom-left corner) is surrounded by steep hills. These hills lead the PSO algorithms astray in their search for the global optimum.

6.3.2 Diversity

Diversity as measured in table 6.2 is the average difference in the fitness value each particle has obtained in the swarm. By observing the values obtained it becomes evident as to why the PSO* algorithm with inertia in general performed better than the standard PSO algorithm. The difference

Function	PSO	PSO*
DeJong F1	2.10378	1.41E-14
Shekel's Foxhole	31.2242	8.68362
Rastrigin	1.84952	0.33564
Schwefel	7.80E+41	145.668
Griewank	262.908	0.000438
Salomon	28973.4	8.06003
Ackley	14.6277	0.783741
Six-hump Camelback	1.26021	0.041858
Shubert	2.55078	0.245377
Himmelblau	71723.80	16.0674
Rosenbrock Valley	2.09851	0.20354
Dropwave	17.336	0.533049
Easom	98.6724	3.90901
Branins	31445.8	271.076
Michalewicz	1.01585	1.25E-09
Goldstein	5.61E37	2.19E+36

Table 6.2: Diversity values

in particle' fitness values is extremely low compared to their equivalent values in the standard PSO algorithm. This means that inertia lets the swarm be much more cohesive and thus particles are much closer to each other in the search landscape. Being closer to each other allows the swarm to search the landscape much more thoroughly.

The diversity values can also be interpreted as the rate of exploration. The standard PSO algorithm clearly has a high exploration rate. Another interpretation is that diversity can also depict the exploitation that the algorithm performs, which means that the PSO* has a high exploitation rate due to particles being much more cohesive with regard to their search.

It must also be noted the inertia value of the PSO* algorithm allows one to control the rate of exploration and exploitation of the swarm. For these benchmarks a conservative 0.5 was chosen. A 0.5 inertia value achieves a good balance between exploration and exploitation where a value of 1.0 means full exploration. Thus with a 1.0 inertia value the PSO* algorithm

is equivalent to the standard PSO algorithm.

Since the PSO algorithm with the inertia value not only achieved better results than its standard counterpart, but also exposed a parameter which controls the behaviour of the algorithm it becomes clear the particular PSO algorithm to be used for this research had to be the algorithm that uses inertia.

6.4 Summary

In this chapter a series of mathematical optimisation problems were presented. Each problem was mathematically formulated as well as categorised as multimodal or unimodal and separable or non-separable.

To get a better idea of how the PSO algorithm operates and performs, two PSO algorithms were developed. The first PSO algorithm uses the standard velocity equation with no alterations. The second PSO algorithm uses the notion of inertia with regard to velocity calculation.

The algorithms were tested on all the mathematical problems outlined at the start of this chapter. The chapter concluded with the results obtained by the algorithms.

Chapter 7

Applying the PSO to the FAP

7.1 Introduction

PSO, as discussed previously (see page 124), is an algorithm that is largely based on the flying behaviour exhibited by a flock of birds. This is why the core of the algorithm is based upon vector mathematics, with new positions and velocities being calculated after each iteration of the algorithm. Thus a D-dimensional vector represents each particle position and is then simulated by flying through the D-dimensional space using the velocity equation (see section 5.5.3 on page 130).

The performance of the global PSO was benchmarked and outlined in the previous chapter. As can be observed from the results, the PSO algorithm produced favourable results on most of the problems it was applied to, failing to locate the optimum on only a select few problems (see section 6.3).

Most of the problems to which PSO has been applied to date have been problems where the position of particles has a constant D-dimensional space. In formal terms, therefore, *the dimensionality of a particle position in its entirety is constant*. Therefore, each particle has a position which is defined in a set dimension like 2D where a position is represented at x and y coordinate pairs.

This constant dimensionality introduces an intriguing problem if one wants to apply the PSO to an inherent multidimension problem like the FAP. This chapter deals with how the PSO was applied to the FAP.

Firstly, the particle position is represented in the frequency planning domain is defined. This definition of the particle position is important because it plays a central part in the movement of particles through the Frequency planning domain. A description is then given of how each position is evaluated as well as the fitness function that the PSO will use in the FAP domain.

Arguably the most important part of the swarm is how the velocity of a particle is calculated and then moving it to a new position in the problem space. The velocity update is important, as it is the primary means by which the algorithm searches problem space.

As was discussed in section 5.5.4, applying the PSO to the FAP introduces a variety of challenges. One of the challenges is how exactly one "flies" a frequency plan towards another frequency plan. This is an important question that needs to be addressed as the PSO algorithms have no other way of searching the problem space by any other means.

To be able to allow the PSO to operate in the FAP space custom velocity functions had to be developed to enable the particles of the swarm to move. These velocity functions that were developed will expanded on in section 7.4.

Developing custom velocity functions for the PSO was simply not enough to achieve good results with the PSO. Therefore more innovations needed to be made to improve the solution quality of the PSO. In section 7.5 a new mechanism is presented for selecting the global best which enabled the PSO to get better fitness values and therefore direct the swarm more towards better solutions.

Finally, the chapter will conclude with how the swarm utilises history to produce better results to enable the PSO to further improve the solution quality.

7.2 Position in the Frequency Planning Domain

In this section a description will be given of what a position is in the frequency planning domain. First a frequency plan is defined and the general structure to represent such a plan is provided. The section will conclude with the hard and soft constraints and how the constraints aid in creating a frequency plan that is suitable for a network.

A frequency plan, is almost exactly what the name implies: A plan that

outlines channel[1](frequency) usage for a mobile telecommunication network. The benchmark problems that were used to test the developed PSO all pertained to cellular phone networks and were presented in section 3.7. For cellular networks, the frequency plan outlines which channel must be allocated to which transceiver.

With this basic definition, the problem is deceptive as one naturally assumes that there are an infinite number of channels that can be used or the number of channels available for assignment is more than the number of transceivers in the network.

The reality is that there are only a finite number of channels available for cellphone transmissions, as was discussed in chapters 2 and 3. Hence a regulatory body needs to assign wireless spectrum to cellphone network operators for use in their networks. A regulatory body is needed because, if a network operator uses just any channel it wants, it is bound to interfere with someone else also utilising the same channel.

A network is not assigned to the entire wireless spectrum for wireless communication, but rather only a subset is assigned to the network. If one observes the FAP benchmark problems the PSO was applied to (see section 3.7) for instance Siemens1, the allotted spectrum is from channels 16 to 90. Which gives the network operator 74 channels to use in its network without considering other constraints.

Besides the electromagnetic constraints that are also applicable here, there are regulatory constraints, for instance frequencies in the spectrum that are by no means allowed to be used. These frequencies are referred to as globally blocked frequencies and are hard constraints. An in-depth discussion of these constraints was given in section 3.4.

As discussed in chapters 2 and 3, a cellphone network is divided into a number of cells, and each cell requires a certain number of transceivers to service its corresponding area.

The number of transceivers is based on the expected volume of traffic that a particular cell will experience at peak network usage. Some cells might be located in highly populous areas, which means the potential traffic that cell might need to handle during peak network usage is very high and thus the cell has more than one transceiver to handle the potential traffic.

[1]See page 25 on why frequencies are referred to as channels

With cells that are located in areas that have a low population, the potential traffic the cell might experience during peak network usage is low and thus the cell only has one transceiver to handle potential traffic.

Based on the amount of traffic a cell needs to handle, the number of transceivers differs; thus in a frequency plan not all cells have the same number of transceivers, otherwise a frequency plan can be modelled as a series of constant D-dimensional vectors, where the D represents the number of transceivers. As can be seen in figure 7.1 a cellular network can have

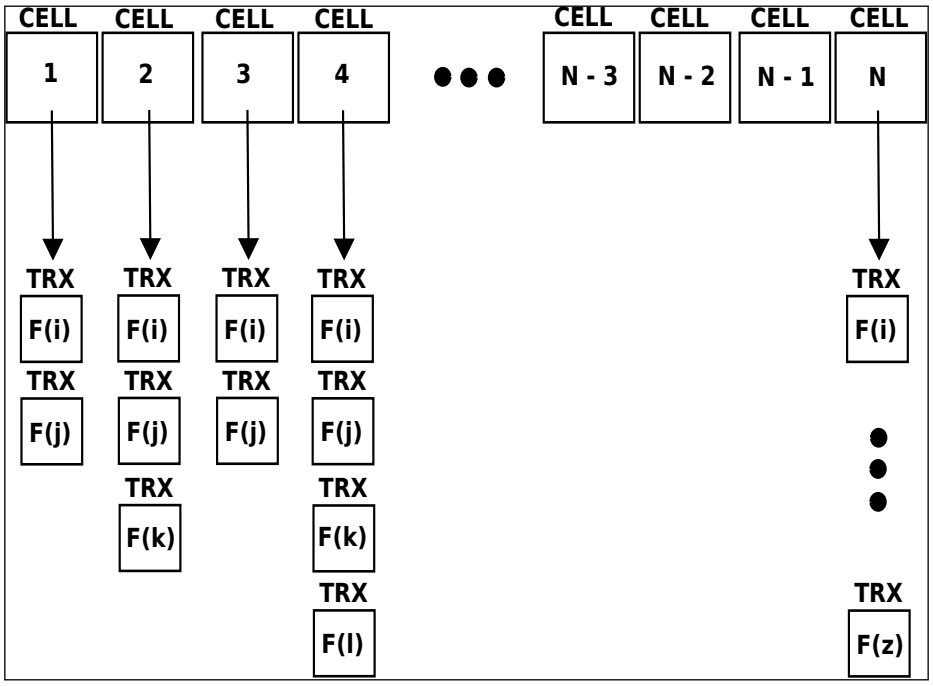

Figure 7.1: The structure of a frequency plan

any number (N in the figure) of cells to attain the desired coverage over the geographical landscape. In the COST 259 benchmark problems the cellular networks have a large number of cells that range from 500 to more than 1 000.

The most important part of the plan is the actual transceivers within each cell. In figure 7.1 it can be clearly seen how the number of transceivers (TRXs) varies from one cell to the next. $F(i)$ is a channel at position i from the available usable spectrum.

Based on the structure of the plan depicted in figure 7.1 there is no concept of which cell interferes with which other cell and if there is indeed

interference, the extent of this interference. Not all this information is part of the plan. Instead this information, for the purpose of this research is supplied by the COST 259 benchmark.

The interference information is referred to as the interference matrix. A definition of the structure of an interference matrix was given in section 3.4. As discussed, each entry references two cells' entries: Cell A and Cell B. Along with the entry the amount of interference that occurs when Cell B interferes with Cell A is also listed[2].

A frequency plan is a possible solution to the FAP. Therefore in the PSO that was developed each particle's position in the solution space is represented by a frequency plan. As illustrated in figure 7.1, a frequency plan is just a series of cells, where each cell has a set of transceivers; thus in the PSO algorithm a plan is actually represented as an array of cells. This enables the algorithm to access particular cells in a plan by index as can be observed in listed algorithms 9 and 14.

Algorithm 7 The FAP PSO Algorithm

```
s_n = Initialize Swarm s_n
while Termination criterion not met do
  EvaluateSwarm(s_n)
  UpdateGlobalBest(s_n)
  UpdateSwarmMovement(s_n,gbest)
end while
```

Before the particles can actually start to move around in the FAP space, they first need to be assigned positions. In the developed PSO listed in algorithm 7.2 line 1 the first operation that the algorithm executes is to initialise all the particles in the swarm. A particle position in the algorithm is initialised by assigning it a random position; thus a frequency plan (representing a position) is randomly generated by the algorithm.

The position is purely random in that the only considerations made by the position generator are that valid channels are assigned to transceivers installed at cells. Thus the generator does not check whether a channel has already been assigned in the current cell or any other considerations. The intended purpose of the generator is just to place a particle in the problem space, not the premature start of the optimisation process.

[2]Interference occurs based on the electromagnetic constraints as defined in chapter 3

Since particles are able to occupy positions in the FAP space the PSO algorithm is now able to move them around in the problem space. As mentioned previously, moving particles through the frequency plan solution space introduces an interesting problem due to the multidimensionality of a plan. A discussion of how particles are moved from one position to another through the solution space will be provided in section 7.4

In the next section an explanation will be given on the fitness function that determines the desirability of a particular particle's position or rather the frequency plan its position represents.

7.3 The Fitness Function

The fitness function rates the desirability of a particular particle's position in the problem space.

As discussed in the previous section, the COST 259 benchmark problems provide an interference matrix that lists the total amount of interference that occurs when a pair of cells interfere. As outlined in the structure definition (see section 3.4) each entry in the interference matrix defines a pair of cells that are said to interfere, along with two additional values. The first value is referred to as co-channel interference and is the total amount of interference that will occur on the communication link when the allocated channel of one transceiver is equal to a transceiver in the other cell that is listed in the interference matrix.

The second value is called adjacent channel interference and it is the total amount of interference that will occur on the communication link when the allocated channel of a transceiver in one cell differs by 1 from another channel allocated to the transceiver from the other cell that is listed in the interference matrix.

Particles move towards other particles because the other particles have indicated (through information sharing) that the positions they occupy are very lucrative and thus they have found potentially good solutions. The only way particles can know the lucrativeness of the position they occupy is if the position is evaluated with a fitness function. Thus the lucrativeness of a position is actually the fitness value obtained from the fitness function.

Since a particle position is defined as a frequency plan, a procedure is needed that calculates the fitness of a frequency plan. With the FS-FAP

the primary concern is to keep interference to a minimum. Therefore in the PSO that was developed the fitness value is the total amount of interference generated by all the cells with their allocated frequencies.

The evaluation procedure goes through each pair of cells defined in the interference matrix where it looks up both cells in the frequency plan. The second cell is said to interfere with the first cell. Therefore each transceiver in the first cell is checked against all the transceivers of the other cell. Depending on whether the frequencies differ from each other, the fitness procedure adds either co-channel or the adjacent channel interference to a summing variable. This procedure is mathematically defined in chapter 3 (see page 47 for the formal equation) and algorithm 8 is the pseudocode of the implement equation used by the PSO.

As can be seen in algorithm 8 not all interference values are added to the total amount of interference variable. The COST 259 benchmarks define a minimum tolerable interference variable. This means that if a given interference value is either equal to or less than this defined value the interference generated is negligible and can be disregarded as it will not have a noticeable impact on the communication link.

In the next section a description will be given of how particles are moved from one iteration to the next using frequency plans as positions in the solution space.

7.4 Velocity Function for Frequency Planning

The elocity function is arguably the core of the PSO algorithm. It is the procedure by which particles in the swarm move from one point to another in the solution space.

The velocity function does not blindly move a particle from one point to another, but instead it takes the particle history into account as well as the best particle in the swarm. Therefore the velocity function is the core means by which the swarm explores the solution space. A more thorough explanation is provided in section 5.5.3.

The development of a velocity function that is suitable for particles to move from one frequency plan to another is covered next. The section will start off with the first velocity function that was developed. With each method discussed, the problems associated with it will also be mentioned.

Algorithm 8 FAP Cost Function

Require: normalCell
Require: interferingCe;;

```
1: totalInterference ←0
2: for Each TRX trx_i in interferingCell do
3:   for Each TRX trx_j in normalCell do
4:     interference ← 0
5:     difference ← |trx_i − trx_j|
6:     if difference is 0 then
7:       if coChannelInterference ≤ minInterferenceThershold then
8:         interference ← interference + 0
9:       else
10:        interference ← interference + coChannelInterference
11:      end if
12:    else
13:      if difference is 1 then
14:        if adjChannelInterference ≤ minInterferenceThershold then
15:          interference ← interference + 0
16:        else
17:          interference ← interference + adjChannelInterference
18:        end if
19:      end if
20:    end if
21:    totalInterference ← totalInterference + interference
22:  end for
23: end for
```

This section will conclude with the second method that was developed and that is also the primary method the developed PSO uses.

7.4.1 Movement in the Frequency Planning Domain

The standard velocity equation works on the basis of vector mathematics. Each particle has a velocity and position, which is represented by a standard mathematical vector. The standard equation alters the direction of the particle to move to a more promising position in the solution space that is in the general direction of the global best particle and a previous personal best position the particle held..

Vector mathematics has standard basic operations defined for adding, subtracting and multiplying; hence applying the PSO to problems that are either mathematical functions or problems that map well to the vector domain is a defined process. With regard to the frequency planning domain an important question needs to be answered: How can one multidimension frequency plan be moved to another?

With any difficult problem it is better to break the problem down into its most basic constructs and then solve each piece individually until the problem as a collective is solved. This technique is also commonly known as divide and conquer. This technique was first applied to the nature of a frequency plan.

A frequency plan is a plan that consists of a series of different cells that are in use in the network. The plan specifies the channels that each individual transceiver installed at a cell must use for communication. Thus a frequency plan can broken up into three important constructs:

1. A plan is a list of different cells.
2. Each cell in a plan has a list of transceivers that it has installed.
3. Each installed transceiver has a single number allocated to it called the channel. This channel is used for communication.

Now that a frequency plan has been broken up into its constructs, the question of how to move one frequency plan to another can be rephrased. How does one move a channel allocated to a *transceiver* in a particular *cell* of one frequency plan to another channel of the *same* transceiver and cell in

another *different* plan? An important realisation needs to be noted here. In the PSO at any one time the algorithm is only considering two positions and for the FAP the two positions are frequency plans. Both plans are *identical* except for the specific that channels transceivers use. Thus a cell that exists in the one plan, also exists in another plan. Both the cells have exactly the same number of transceivers installed; only the channels each individual transceiver uses for communication differ.

Using this realisation, the conclusion can be made that the velocity equation can only work with the channels assigned to transceivers. Therefore a potential velocity equation mechanism needs to operate on the finest granularity of a frequency plan, i.e. the channels.

The principle on which the first velocity method developed is based, is for the movement of the swarm to be at a much finer granularity and hence movement is based on channels. Therefore when a particle needs to move towards a global best particle, the velocity procedure goes into the intricate details of the particle wanting to move and the global best particle. The procedure goes into each cell defined in the frequency plan represented by the standard particle as well as the global best particle to be able to access each installed transceiver.

To be able to move from one frequency plan to another by utilising the standard velocity equation, the equation needs to be broken up into its smaller operations. In this way, small operations can be developed that perform the same function as the individual parts.

The velocity equation in section 5.5.3 can be broken up into the following parts:

- **Subtraction** — $pbest - x_i(t)$ — (SubtractionResultPbest) and $gbest - x_i(t)$ — (SubtractionResultGbest)
- **Multiplication** — $c_1\phi_1 * SubtractionResultPbest$ — (MultiPbestResult) and $c_2\phi_2 * SubtractionResultGbest$ — (MultiGBestResult)
- **Addition** — $v_i(t) + MultiPbestResult + MultiGBestResult$

There are no mathematical constructs that define how two frequency plans are added together or subtracted, let alone multiplied. As discussed earlier, a frequency plan is just a series of cells that have channels and these

channels are numbers that internally are just integers and there are mathematical constructs that define how two integers should be added, subtracted or multiplied. Both velocity methods that were developed utilise the basic principle that on a fine granularity of a frequency plan one is merely working with integers.

Algorithm 9 Velocity Method 1

Require: currentParticle
Require: globalBestParticle
1: $pbest$ ←currentParticle position
2: $gbest$ ←globalBestParticle position
3: $localCoeff \leftarrow getLocalCoefficient()$
4: $globalCoeff \leftarrow getGlobalCoefficient()$
5: $pBestSubtractResult$ ←Subtract current Particle position from pbest
6: $gBestSubtractResult$ ←Subtract current Particle position from gbest
7: a ←Multiply localCoeff with pBestSubtractResult and a random value
8: b ←Multiply globalCoeff with gBestSubtractResult and a random value
9: **if** first time velocity is calculated for current Particle **then**
10: $currentParticle.Velocity$ ←Add a and b
11: **else**
12: $abAdditionResult$ ←Add a and b
13: $inertia \leftarrow getInertia()$
14: $newVelocity$ ←Add abAdditionResult and currentParticle.Velocity
15: $currentParticle.Velocity$ ←Multiply inertia with newVelocity
16: **end if**
17: $currentParticle.Position$ ←Add currentParticle.Velocity and currentParticle.Position
18: SanatizePosition(currentParticle.Position)

The first velocity method is listed in algorithm 9. Velocity method 1 works on the principle of moving one cell in a particular frequency plan to the same cell in a different frequency plan. As noted earlier, the cells are the *same*, but the channels that have been allocated to each transceiver within a cell differs. Thus in velocity method 1, each cell has an array of transceivers. The array of transceivers contains the individual channel numbers that have been allocated to a cell.

Velocity method 1 actually moves one array of transceivers to another array of transceivers. This is why, if one observes lines 5 – 17 in algorithm 9, each particle position is passed to methods that are named Subtraction, Multiplication and Addition. As discussed in the previous section, each position of a particle is actually a frequency plan.

Algorithm 10 Subtract One Position From Another

(Method 1)

Require: fromPosition

Require: toPosition

1: **for** Each cell c in fromPosition **do**
2: **for** Each transceiver f_{trx} in c **do**
3: $t_{trx} \leftarrow$ Get same f_{trx} in toPosition
4: $f_{trx} \leftarrow f_{trx} - t_{trx}$
5: **end for**
6: **end for**

The first method that is used to calculate the velocity of a frequency plan, is the first basic operation defined in the velocity equation, namely subtraction, as can be observed from algorithm 10. The particular algorithm expects two positions to be given to it: the position it must be moved from and the position it must be move to. Because each position is a frequency plan that contains an array of cells, a for-loop is started on line 1 that allows the algorithm to go through each cell that is defined in the frequency plan. Each cell has a number of transceivers, each with an assigned channel.

To be able to subtract two transceivers from each other, the algorithm needs access to a particular transceiver of a cell. The algorithm gains this access by entering another for-loop based on the number of transceivers a cell has, as can be observed to occur in line 2. Within this for-loop the actual subtraction of transceivers occurs.

The algorithm first obtains the exact same transceiver in the toPosition frequency plan. This operation is quick, as the two plans are identical except for the channels assigned to transceivers for each cell. Thus the algorithm is able to refer to the cell and specific transceiver in the toPosition plan by the same index it utilises to access the cell and transceiver in the fromPosition.

Once the toPosition plan transceiver channel value has been obtained, the algorithm performs a standard integer subtraction as seen in line 4 of

algorithm 10. After the subtraction algorithm is completed, a position is returned which is the result of subtracting the fromPosition from the toPosition. Using this result the velocity method 1 algorithm is now ready to apply the next operation in the velocity update equation, namely multiplication.

Algorithm 11 Multiply Position by a Value (Method 1)

Require: fromPosition
Require: value
Require: mustRandom
1: **for** Each cell c in fromPosition **do**
2: **for** Each transceiver trx in c **do**
3: **if** must multiply with random number **then**
4: $trx \leftarrow trx * value * random()$
5: **else**
6: $trx \leftarrow trx * value$
7: **end if**
8: **end for**
9: **end for**

The multiplication algorithm is where the local Coefficient and the global coefficient[3] defined for the PSO are multiplied into a position, i.e. frequency plan. As with the subtraction algorithm the multiplication algorithm enters two for-loops. The first for-loop is to access each cell defined in the frequency plan. The second loop allows the algorithm to access each transceiver within a particular cell. These for-loops can be observed to occur in algorithm 11 in lines 1 and 2.

Once the algorithm is able to access each individual transceiver and get its assigned channel, it is able to perform the multiplication operation. As can be observed in the algorithm in line 3 the algorithm checks whether it needs to multiply the channel by an additional random number besides the *value* passed to the algorithm. The reason this is done is that multiplication is required in the PSO in the following cases:

- The inertia case — $0.5 * v(t+1)$, where 0.5 is the inertia value and $v(t+1)$ is velocity that has already been calculated for a particular

[3]Local coefficient is the cognitive component and the global coefficient is the social component. These components are discussed in section 5.5.1

particle t. Note, a velocity that has been calculated is still a frequency plan.

- Standard velocity calculation randomisation case — As can be observed from the velocity equation 5.9 and also from the multiplication bullet point on page 166, the equation requires that a position be multiplied by a coefficient c_1 or c_2 and then a random value ϕ_1 or ϕ_2.

Regardless of which case is executed, the actual operation that is performed is integer multiplication, which therefore means that even though the inertia and random numbers are decimal, the fractional component of the result is discarded. Channels are integers so the loss of the fractional component is warranted as it is of no use.

As per the last bullet point on page 166 the last basic operation that occurs in the velocity equation is an integer addition operation. Algorithm 12 is the addition procedure used by velocity method 1 (algorithm 9).

Algorithm 12 Add One Position to Another (Method 1)

Require: fromPosition
Require: toPosition
1: **for** Each cell c in fromPosition **do**
2: **for** Each transceiver f_{trx} in c **do**
3: $t_{trx} \leftarrow$ Get same f_{trx} in toPosition
4: $f_{trx} \leftarrow f_{trx} + t_{trx}$
5: Bound(t_{trx})
6: **end for**
7: **end for**

In the addition algorithm, as with multiplication and subtraction, two for-loops are started as can be observed in lines 1 – 2. The algorithm first starts iterating through all the cell's present in the fromPosition frequency plan and then within the cell for-loop starts another for-loop which iterates through each cells transceivers. Inside the transceiver for-loop the algorithm actually performs the addition of two channels.

Two channels are added based on integer addition. Once the fromPosition cell transceiver is accessed, the algorithm retrieves the exact transceiver and cell in the toPosition in order to get its allocated channel value. After

both values have been retrieved the algorithm performs the addition as can be observed in line 4 of algorithm 12.

The addition operation is the last operation that occurs in the velocity equation and is also the only operation that occurs when the resultant velocity is applied to the current position of a particle as in equation 5.10. Since the addition operation is also used in the final position update of the particle, its last operation in the transceiver loop a bound operation.

The purpose of the bound operation is to keep the channels within valid value ranges.

7.4.2 Keeping Frequencies Bounded

In the previous section the first velocity method that was developed was discussed. The velocity method is important as it calculates the direction and next position of a particle in the problem space, where the problem space is the FAP and a position of a particle is a possible frequency plan. With the velocity method defined, the swarm is now able to move around in the problem space, but this alone is not enough. The swarm has no concept of the constraints[4] that are imposed inherently on the domain as well as the network for which a frequency plan is being created. The constraints

Algorithm 13 BoundValue Method

Require: Position

1: **for** Each cell c_i in Position **do**
2: **for** Each transceiver trx_j in c_i **do**
3: $trx_j = |trx_j|$
4: **if** $trx_j \geq maxChannelValue$ **then**
5: $trx_j = minChannelValue + (trx_j \bmod maxChannelValue)$
6: **else**
7: **if** $trx_j \leq minChannelValue$ **then**
8: BoundValue$(trx_j + minChannelValue)$
9: **end if**
10: **end if**
11: **end for**
12: **end for**

[4]Constraints in the FAP are discussed in chapter 3

that are imposed are usually some part of the spectrum that may under no circumstances be used anywhere in the network, which as discussed in chapter 3 is known as globally blocked channels. Some channels that are only allowed to be used in certain parts of the network are known as locally blocked channels. Then of course the swarm also needs to take into account the electromagnetic constraints[5].

The velocity function used by the PSO needs to be altered to make the swarm in some sense more aware of the domain it is operating in and hence keep the particle positions bounded within the allowable search space. By not bounding particle positions in the FAP problem space, transceivers of some cells might be assigned a channel that is not allowed or not even allocated to the network. The PSO will accept this assignment since the fitness function operates on the assumption that under no circumstances will these invalid channels be assigned and thus does not penalise invalid assignments.

To keep assigned channel values to transceivers in the allocated spectrum of the network a boundary check needs to be added to the velocity method. The purpose of the boundary check is to validate all assignments in a position, i.e. the frequency plan that a particle currently occupies, and if any assignments violate the defined boundary constraints then it must take the violating value and modify it to be in the acceptable value range.

The boundary check that is used by velocity method 1 operates on the basis that for any range of values there is a defined lower bound (a minimum value) and upper bound (a maximum value). The channel boundary check is only applied when one of the following conditions is met after the calculated velocity has been applied to the current position:

- If a channel allocated to a transceiver is above the maximum allowable channel (upper bound) given to the network.
- If a channel allocated to a transceiver is below the minimum allowable channel (lower bound) given to the network.

As can be seen in line 5 of algorithm 13, a mod operation is applied to the value to bring it within the allowable range. The integer mod operation is similar to integer division, the only difference being in the result that

[5]Discussed in section 3.4

is produced. Division produces the result of two numbers being divided. Mod produces the remainder of two numbers being divided. If two numbers divide perfectly into one another there will not be any remainder; if the numbers do not divide perfectly into one another there will be a remainder.

$$10 mod 50 = 10 \tag{7.1}$$

$$60 mod 50 = 10 \tag{7.2}$$

$$50 mod 50 = 0 \tag{7.3}$$

$$100 mod 50 = 0 \tag{7.4}$$

$$35 mod 50 = 35 \tag{7.5}$$

As can be seen in the above example mod operations, any value that is modded will be kept in the range $[0, 50]$. With regard to channels, the following occurs: If, for instance, the maximum allowable channel is 50, and the transceiver has a channel value (after velocity) of 56, the 56 value is modded by 50 to produce a value of 6. This modded value is then added to the minimum allowable channel. In essence, the value is wrapped around to always be within acceptable range.

The difficult case is when the channel value is lower than the minimum channel given to the network. This is because modding the channel value has no effect. For example, if the lowest allowable channel is 20 and the transceiver value after movement is 15, modding the transceiver value of 15 by 20 has no effect. To solve this, the following options are then considered:

1. First subtract the lower value from the minimum allowable channel. Then add the result to the minimum allowable channel. The resultant value is checked again as to whether it oversteps the bounds of the maximum allowable channel and is bounded accordingly.

2. Add the lower value to the minimum allowable channel. The resultant value is checked as towhether it oversteps the bounds of the maximum allowable channel and is bounded accordingly.

3. Repeatedly subtract the lower value from the maximum allowable channel until the resultant channel is within the acceptable channel range.

An important notion to consider is that, based on the velocity equation, it is entirely within the realm of possibility that a channel value after movement might contain a negative value. As can be seen in line 3 of algorithm 13 the algorithm solves this problem by first taking the absolute value of the negative channel value. The boundary check then treats the now positive channel value as a normal value that needs to be bounded.

It is better to use channel index values rather than actual channel values for frequency plans, and this is explained below.

7.4.3 Using Indices instead of Frequencies

As discussed in section 7.4 and as can be observed from algorithm 9, the first velocity method that was developed for the PSO worked with raw channel values. This is not ideal since upon closer inspection the channel range that the swarm used to move around was indeed incorrect. The bound value algorithm only keeps channels within a minimum and maximum allowable channel range, but globally blocked channels and locally blocked channels can be in between this minimum and maximum channel range.

With velocity method 1 and the PSO using raw channel values, the swarm increasingly moved towards allocating these blocked channels to transceivers since the fitness function does not penalise the use of blocked or invalid channel values. This is partly due to the fact that these values are under no circumstances allowed to be used and thus the fitness function is not designed to check for these values explicitly to impose a penalty.

Frequency plans that utilise these blocked channels are invalid and cannot be used. If a network were to use a plan that uses blocked channels, it could cause unexpected interference to other services and the network could be fined by the governing body that controls the spectrum. A bare minimum requirement then is that the PSO must generate valid frequency plans and hence swarm particles can only occupy valid positions. The following options were presented to solve the problem of particles moving towards invalid positions and hence having invalid frequency plans:

1. Modify the fitness function to penalise a frequency plan if it uses any globally blocked channels or locally blocked channels.

2. Instead of letting the swarm work with raw channel values, rather let it work with indices of an array. The array index values indicate

positions in an array that has been pre-filled with only *valid* channels. Thus the swarm then moves around in a range from 0 to F, where F is the size of the channel array.

With the first solution, the fitness function would have to be modified to impose a penalty if a prohibited frequency value were used. The first proposed solution was disregarded because it introduces complexity which can be completely avoided with the second proposed solution.

With the second solution the fitness function does not have to be modified and the boundary check is simplified since there is no need to check for a lower bound. The boundary check now only has to check for negative index values and whether the upper bound, which is now the size of the array, is violated. Working with index values rather than with raw channel values

Algorithm 14 Velocity Method 2

Require: currentParticle
Require: globalBestParticle
1: $currPos \leftarrow$ currentParticle position
2: $pbestPos \leftarrow$ currentParticle best position
3: $gbestPos \leftarrow$ global best particle position
4: **for** Each cell c in $currPos$ **do**
5: $c_{pbestPos} \leftarrow$ Same cell in $pbestPos$
6: $c_{gbestPos} \leftarrow$ Same cell in $gbestPos$
7: $movedIndices \leftarrow$MoveIndices$(c_{pbestPos}, c, c_{gbestPos})$
8: **if** First time velocity is calculated **then**
9: ApplyVelocity(c,movedIndices)
10: **else**
11: $newVelocity \leftarrow CalculateIndexVelocity(currPos.velocity, movedIndices)$
12: ApplyVelocity(c,newVelocity)
13: **end if**
14: **end for**
15: SanatizePosition(currPos)

led to a second velocity method. Algorithm 14 is the pseudocode for the second velocity method. The second velocity method differs from velocity method 1 due to it being designed to only work with indices of an array.

Velocity method 2 also differs from method 1 due to the manner in which it applies the velocity equation. Velocity method 1 applies the velocity equation in stages. Each stage is applied to the entire position, i.e. frequency plan before applying the next stage. With velocity method 2 the algorithm first enters a for-loop as in line 6. Within this for-loop the algorithm obtains the current cell, the cell stored in its previous best-held position *pbestPos* and the same cell stored in the global best position *gbestPos*.

Once all the correct cells have been obtained the algorithm calls a method named *MoveIndices* (which is presented in algorithm 15) to which all the cells are passed.

Algorithm 15 MoveIndices

Require: pbestCell
Require: currCell
Require: gbestCell
1: **for** Each trx_i in currCell **do**
2: $pbestTRX_i$ = Get trx_i in pbestCell
3: $gbestTRX_i$ = Get trx_i in gbestCell
4: $r1 \leftarrow$ Random double number
5: $r2 \leftarrow$ Random double number
6: $localCoeff \leftarrow getLocalCoefficient()$
7: $globalCoeff \leftarrow getGlobalCoefficient()$
8: $a \leftarrow localCoef * r1 * (pbestTRX_i - trx_i)$
9: $b \leftarrow globalCoef * r2 * (pbestTRX_i - gbestTRX_i)$
10: $trx_i \leftarrow a + b$
11: **end for**

In the MoveIndices algorithm three cells are passed to it. The current cell of the velocity algorithm that is being moved, as is the same cell that is stored in the previous best-held position of the particle and also the same cell in the global best position. The algorithm starts off by iterating through all the transceivers installed at the current cell as can be seen in line 1 of algorithm 15. In lines 2 – 3 the algorithm obtains the channel values from the *pbestCell* and *gbestCell*.

After the MoveIndices algorithm has obtained the channel values it is ready to apply the velocity equation. As can be seen in lines 8 – 10 the algorithm applies the standard velocity equation 5.9 formulated in chapter 5.

Once the MoveIndices algorithm has completef, the currCell variable now stores the calculated velocity. Algorithm 14 uses the return calculated velocity to update the cell position of the current particle. The algorithm accomplishes this by first determining whether the particle has had a previously calculated velocity. If it is the first time the velocity has been calculated, the algorithm simply applies the velocity by using algorithm 16.

Algorithm 16 ApplyVelocity

Require: currPos

Require: velocity

1: **for** Each index c_i in currPos **do**
2: $c_i \leftarrow c_i + velocity_i$
3: **end for**

If the particle does have a current velocity, the algorithm first executes algorithm 17 as can be observed to occur in line 11 of algorithm 14 before applying the calculated velocity with algorithm 16. Velocity method 2 first executes algorithm 17 to apply the concept of inertia[6].

Algorithm 17 CalculateIndexVelocity

Require: currVelocity

Require: newVelocity

1: **for** Each value v_i in currVelocity **do**
2: $inertia \leftarrow getInertia()$
3: $v_i \leftarrow v_i + (inertia * newVelocity_i)$
4: **end for**

As can be observed in algorithm 17 the CalculateIndexVelocity method loops the current velocity. Within the loop the algorithm retrieves the inertia value that needs to be applied, after which the inertia value is multiplied by the newVelocity value.

After the CalculateIndexVelocity algorithm has finished executing, the concept of inertia has been applied to the velocity. Velocity method 2 therefore applies the velocity using the ApplyVelocity method listed in algorithm 16 to the current cell found in the particle position to move it in order to obtain a new position.

[6]Inertia is discussed in chapter 5 in the subsection entitled "Inertia Weight" of section 5.5.3.

The observant reader might have noticed that in all the presented and associated velocity algorithms the procedures operate and store the result in the current particle position that is currently being worked on. Naturally one might think that this overwrites the position information currently in that position. In the developed PSO algorithm when velocity is calculated the respective velocity methods operate on clones of the original position. The original position is only used once the velocity has been calculated and the particle has actually moved.

Another important note, is that if one analyses velocity method 1, the switch to using index values instead of raw channels does not affect the method's operation. Since the method only requires explicit knowledge on the data to operate on in the BoundValue algorithm, the algorithm is still able to calculate velocity.

It is up to the algorithm designer to update the bound value method to use the array index bounds rather than the raw lower and upper bounds of the channels. The BoundValue algorithm needs to be updated since it is the primary means by which velocity method 1 ensures valid positions.

Both the velocity methods that are utilised by the developed FAP PSO algorithm have now been explained with corresponding pseudocode. Why the developed PSO was modified to operate on channel index values rather than channel values was also explained. All the algorithms that enable the main algorithm to accomplish particle movement with indices were also discussed. The PSO is hence able to move the particles in the FAP using two different methods, but for a particle to be moved it needs a personal best and most importantly, a global best to move towards. The next section deals with on how the developed PSO algorithm differs from the standard PSO with regard to selecting a global best.

7.5 Building a Global Best

Selection of the global best particle by the swarm is a very important procedure. After the swarm has determined which particle has achieved the best position, the swarm enters the velocity function phase.

As discussed previously each particle position is then modified to move in the general direction of the global best and personal best position. Therefore

the global best acts as a beacon for the rest of the swarm in the solution space to indicate where good solutions seem to be for the rest of the swarm.

Algorithm 18 Standard Gbest Selection in FAP PSO

Require: swarm
Require: gbest
1: $gbestCost$ = Evaluate(gbest)
2: **for** Each particle p_i in swarm **do**
3: $cost$ = Evaluate(p_i)
4: **if** $cost \leq gbestCost$ **then**
5: $gbestCost = cost$
6: gbest = p_i
7: **end if**
8: **end for**
9: **return** gbest

Initially the FAP PSO algorithm used the standard method for selecting the global best particle from the swarm and did not differ at all from the traditional global PSO algorithm. The standard global best selection is listed in algorithm 18.

As can be observed in lines 2 – 8 the FAP PSO algorithm loops through all the particles in the swarm and applies the fitness function to evaluate the fitness of the particle's position. The fitness value is also referred to as the cost. In the FAP PSO the cost or fitness value of a particle position is the amount of interference the frequency plan that represents the particles position generates.

A low cost value is preferred to a high cost value, since a low cost value indicates low interference. In lines 4 – 6 of algorithm 18 the FAP PSO algorithm determines whether the current particle position has a lower cost value than the current global best particle position.

If the current particle position evaluates to a lower cost value than the stored global best, the algorithm replaces the current global best with the current particle being evaluated, which in the algorithm is p_i.

Selecting the global best by evaluating the position as a whole seems to be a natural fit. As outlined in the critical evaluation of each algorithm in chapters 4 and 5, some of the algorithms had a problem with regard to some cells or even transceivers overshadowing better cells or transceivers.

For this dissertation, overshadowing is a term that describes a scenario where a bad value of one part of the frequency plan is so large that it causes other smaller values within the frequency plan not to be considered.

As per the following example a few cells in a frequency plan might have the worst possible channels assigned to their respective transceivers, and other might have the best. Now the few cells with the worst channels generate a great deal of interference, whereas the cells with the best channels generate almost nothing.

When the example frequency plan is evaluated, the bad cells push up the cost value. The high cost value of the frequency plan causes the PSO algorithm to disregard the whole plan. By discarding the whole plan the FAP PSO algorithm loses the knowledge gained on the few cells that had their best channels assigned to their respective transceivers.

In the traditional method of selecting the global best, a particle is actually selected as the swarm best because it contains fewer overshadowing cells or transceivers, and potentially good channel assignments are lost.

The FAP PSO therefore needed to exploit the knowledge that the fitness function exposes much more thoroughly. The information exposed by the fitness function allows one to see what effects certain channel assignments have on the interference of the cell when assigned to the individual transceivers. To make better use of this fitness information two methods were developed for the FAP PSO, each one being finer grained than the other.

1. Besides the particle storing its fitness or cost, the particle also needed to store the interference generated by an entire cell due to the channels allocated to its installed transceivers.

2. Besides the particle storing the total fitness, it also needed to store the interference generated by a channel allocated to a particular transceiver of a cell.

With both these methods, the global best selection scheme needs to be changed to allow the swarm to take advantage of this newly exposed information. As discussed, initially the FAP PSO used the standard global best selection scheme listed in algorithm 18, but now with these new methods, a global best position is no longer selected, but built.

Before the FAP PSO is able to build a global best, the way a particle stores its evaluated fitness needs to change. For the standard global selection scheme, the particle only needs to store one fitness value that is a result of evaluating the whole frequency plan. To be able to build a global best as described above, the fitness value cannot simply be one lump sum representing interference. Instead in the FAP PSO algorithm the interference generated by every transceiver is stored.

The FAP PSO is able to know the performance of every single channel allocated to a particular transceiver and also compare the allocation with other similar transceivers in other frequency plans.

Algorithm 19 Building Global Best with Cells

Require: gbest
Require: swarm
1: **for** Each particle p_i in swarm **do**
2: **for** Each cell c_j in p_i **do**
3: $gbestcell_j$ = Get cell c_j in gbest
4: $cellCost$ = Get total interference for cell c_j
5: $gbestCellCost$ = Get total interference for cell $gbestcell_j$
6: **if** $cellCost \leq gbestCellCost$ **then**
7: $gbestcell_j = c_j$
8: **end if**
9: **end for**
10: **end for**

Each global best scheme developed for the FAP PSO is more finely grained than the other with regard to what the scheme uses to build a gbest. Algorithm 19 uses interference information of cells to build a gbest. Algorithm 20 uses the interference generated by each transceiver installed at a cell to build a gbest. Since each cell has transceivers, the second algorithm is therefore more finer grained than algorithm 19.

Each algorithm will now be discussed since the difference between them is subtle. Algorithm 19 was the first global best building scheme which was developed and will be discussed first. The algorithm starts off in line 1 by iterating through all the particles in the swarm. For each particle the algorithm enters another loop as can be observed in line 2 of algorithm 19.

The second loop of the algorithm iterates through all the cells in the

Algorithm 20 Building Global Best with Transceivers

Require: gbest
Require: swarm
1: **for** Each particle p_i in swarm **do**
2: **for** Each cell c_j in p_i **do**
3: **for** Each transceiver trx_k in c_j **do**
4: $gbestTrx_k$ = Get trx_k in c_j in gbest
5: $trxCost$ = Get total interference for trx_k
6: $gbestTrxCost$ = Get total interference for cell $gbestTrx_j$
7: **if** $trxCost \leq gbestTrxCost$ **then**
8: $gbestTrx_k = trx_k$
9: **end if**
10: **end for**
11: **end for**
12: **end for**

frequency plan that represent a position of the particle. Once the algorithm has obtained a cell it looks for the same cell in the gbest frequency plan. Since the algorithm now has the current cell and the gbest cell it is able to compare the two cells based on the interference their respective transceivers generate.

The interference generated by the cell is retrieved and then compared as can be observed to occur in line 6. If the current cell has a lower interference (cost) value than the same cell in the global best plan, then the algorithm replaces the cell in the global best with the current cell.

The second algorithm is the finer grained algorithm and was developed because, after analysing the algorithm using cells, it was concluded that it is possible that a single bad channel allocation to a transceiver within a cell can overshadow other potentially good channel allocations to other cells within the cell.

Algorithms 19 and 20 have very similar flow, except that the second algorithm has an extra for-loop. In line 3 of algorithm 20 a third for-loop is started, which iterates through all the transceivers a particular cell c_j has installed. The algorithm then obtains the channel allocated to the same transceiver in the global best frequency plan.

Once both transceivers have been obtained, the algorithm determines

the interference (cost) their respective channel allocations generated. Using the cost values the algorithm determines whether the current transceiver channel allocation generates less interference than the channel allocated to the same transceiver in the global best frequency plan.

If the current transceiver channel generates less interference, the algorithm then proceeds to replace the transceiver channel in the global best with the current transceiver channel. Thus it can be seen that algorithm 20 utilises individual transceivers to build a global best.

Initially when the FAP PSO algorithm was tested using both of these global best schemes, the PSO did not produce noticeably better results. This was due to the algorithm at each iteration discarding the interference or cost information calculated in that iteration and making it zero. Making the cost values zero does initially seem correct, but effectively what is happening is that the algorithm is discarding knowledge gained by that iteration.

To enable this information to direct the swarm a bit more, the FAP PSO algorithm was modified to not reset the interference values for every transceiver and cell to 0. Instead, the interference values for an iteration are now added to the previous iteration interference values stored by the cell and transceiver.

By letting interference values compound after each iteration the PSO becomes much more aggressive. This is because as the interference compounds bad decisions made by the swarm for a particular particle become progressively worse as the swarm gets through more iterations.

With compounding interference values the FAP PSO was able to produce much better positions and had lower total interference (cost) than all previously generated positions by previous FAP PSO algorithms.

The FAP PSO algorithm is able to produce better results by allowing particles to keep a history of their previous movements. This is covered in the next section.

7.6 Keeping History

In the traditional PSO history is kept by using the particle personal best position to direct the next movement of the particle. Other methods such as inertia also allow history to direct the movement of the particle. With

regard to the developed PSO on the FAP, the algorithm also uses these concepts. But these concepts are not able to effectively exploit the history of a particle since they have no concept of what combinations of channel values have previously been used in a cell.

In the FAP PSO algorithm more historical information is kept. The algorithm accomplishes this by incorporating the concept of tabu lists from the TS algorithm. Using tabu lists a particle will be able to better exploit the problem space it currently finds itself in. In the FAP PSO algorithm, tabu lists were incorporated by adding to each cell a list which keeps track of each channel value that has been assigned to the transceivers in the cell for 20 iterations.

Initially the FAP PSO algorithm calculated the velocity of a particle and then applied this to the current position of the particle. This moved the particle to its next position in the problem space. With tabu lists this movement step becomes more complicated.

Tabu lists are there to prevent cycling of movements to the same position. Thus to stop the particle from moving to a position that was previously occupied, an extra check has to occur before the particle can occupy a new position. As can be seen in the two developed velocity methods in algorithms 9 and 14 the last step that occurs in both algorithms is that the SanitizePosition method is called.

Algorithm 21 SanitizePosition

Require: currPosition

1: **for** Each cell c_i in currPosition **do**
2: $tbList$ = Get Tabu List of currPosition
3: ResolveCollision(c_i,$tbList$)
4: AdhereToSeparation(c_i)
5: **end for**

The SanitizePosition method is listed in algorithm 21. Within this algorithm a particle's future position is first checked and sanitised before the particle is allowed to move to that position. The main purpose of this algorithm is to check if the future position has been occupied previously and hence is in the tabu list.

In the FAP PSO the tabu list check works slightly differently from what one would expect. As can be observed in algorithm 21, it enters a loop which

iterates through all the cells in the position of the particle. Note that the position passed to the SanitizePosition algorithm is a *future* position; thus the particle does not yet occupy the position yet. Within the for-loop in line 2 the method *ResolveCollision* is called which is listed in algorithm 22.

Algorithm 22 ResolveCollision

Require: cell
Require: tabuList
1: **for** Each trx_i in cell **do**
2: **while** trx_i exists in TabuList **do**
3: trx_i = Generate random channel
4: **if** Collision not resolved after 20 attempts **then**
5: Break out of while loop
6: **end if**
7: **end while**
8: **end for**

As can be observed in algorithm 22, when a channel value is found to exist in the tabu list a collision is said to occur. In the FAP PSO a collision means that the specific channel value that has been assigned to a transceiver for a particular cell was found in the tabu list. Once a collision occurs, the algorithm tries to generate a new random channel that can be assigned to the transceiver as can be seen to occur within the while-loop in lines 2 – 3.

The algorithm generates a new random channel value and then checks to see if the generated value collides with the tabu list. If collision still occurs, the algorithm will generate another random channel. As long as a collision occurs the algorithm will continually generate a new random channel until it has attempted 20 random channels with no channel colliding.

After 20 attempts the algorithm just accepts the last generated channel as the new channel. The maximum number of 20 attempts was selected through testing and can be increased at the expense of more computational time.

The resolution of collisions can be seen as a mechanism to increase the exploration of the PSO algorithm as well as to increase the diversity. By making certain channel assignments to transceivers tabu the algorithm is forced to try new channel assignments and thus explore more of the problem space.

Care must be taken to select a maximum size of the tabu List since one wants to keep enough history so that the problem space can be adequately exploited. The maximum tabu List size must be less than the number of available channels otherwise the algorithm will not be allowed to make any assignments.

Finally the maximum tabu List cannot be too large, since the checks the algorithm has to do to see if a value is tabu are very expensive. The operation is expensive, since the tabu list needs to be iterated through for each potential value to see if the channel value is tabu.

By incorporating tabu lists and the collision resolving procedure, the efficiency of the algorithm reduces dramatically. To increase efficiency of the operations in the algorithm, the FAP PSO algorithm utilises parallelisation. Since the collision resolving procedure is very expensive it was one of the first operations to be parallelised. Other procedures that were also parallelised to increase efficiency were the velocity and any other procedures which involved constraint checks.

By parallelising these operations the efficiency of the algorithm increased and it was able to produce results significantly faster. This is because parallelisation is a good fit to the now standard multicore CPUs in desktop computers.

With the parallelisation of the procedures a slight side effect was noticed. The randomness of the random number generator decreased. This effect was noticed because during testing the counter variable of the collision resolver was displayed on the console. When the value was being displayed on the console the FAP PSO algorithm produced much better results.

The reason for this is that outputting the variable inherently introduces a delay and therefore the random number generators in other threads have different seed values. Hence with a delay in each parallel thread the numbers generated by the random number generator are more distinct.

Due to how parallel threads are scheduled by the operating system, some threads might start off with similar seed values because in the FAP PSO algorithm the current time is used as a seed value[7].

Keeping the delay counter variable displayed on the console introduced a delay in the collision resolving procedure. The reason the particular procedure was selected was that it was where the effect of delay was first noticed.

[7]This is the default behaviour of the .Net 4.0 random number generator

After performing tests with delays of 5 milliseconds (ms), 10 ms, 15 and 20 it was found that 20 ms was the best-suited delay, as it gave just enough time for a reasonable distinction to be made between seed values used by other parallel threads.

In this section a discussion was presented on how the FAP PSO keeps additional history. The reason why the FAP PSO needs to keep more history was discussed as well as what mechanism the algorithm uses to store this information, namely tabu lists. Also covered was how the algorithm deals with collisions, which occurs when positions are in the tabu list. Finally collision resolution was explained with the aid of pseudocode of the algorithm that is utilised.

7.7 Summary

In this chapter an algorithm was presented based on the standard particle swarm optimisation algorithm to operate on the frequency assignment problem encountered in cellular networks.

At the beginning of the chapter it was explained how a frequency plan is represented by the algorithm for use internally. Reasons were given for choosing the particular representation in the algorithm.

One of the most important phases of the PSO algorithm is velocity calculation. The problem was outlined as to why the standard velocity calculation was unsuitable for the FAP. The customised velocity calculation used by the algorithm developed in this research was presented along with suitable pseudocode.

The chapter concluded with small additions made to the algorithm to improve performance and, most important of all, improve solution quality.

Chapter 8

Results

8.1 Introduction

In the previous chapter the FAP PSO algorithm was discussed. This algorithm was developed by modifying the standard PSO algorithm to operate on the FS-FAP. Thus far, no other PSO algorithms have been attempted on the FS-FAP. The only PSO algorithm that has been attempted in the FAP domain was discussed in chapter 5 and was not relevant to the study in this dissertation, as the PSO was applied to an entirely different FAP variant (MS-FAP). The PSO on the MS-FAP was not relevant because the performance measures and what the algorithms optimise differ.

With the FS-FAP, the main performance measurement is interference and the PSO aims to allocate channels in an optimal way to internally produce a frequency plan. On the other hand, the MS-FAP is concerned with the span of frequencies used and the performance measurement is based on the calls dropped. The main purpose of the PSO in the MS-FAP is to minimise the span of frequencies used and keep the number of dropped calls to a minimum.

In the previous chapter all the modifications that were made to the standard PSO were discussed. The modifications were made to enable to PSO to operate on the FAP. Two velocity methods were developed and in addition to the standard global best selection scheme, two additional global best selection schemes were put forward. The algorithm presented was benchmarked against the Siemens instances of the COST 259 benchmark.

In this chapter the results are given of applying the FAP PSO algorithm with its different velocity methods as well as different global selection schemes. This is followed by how the different velocity methods affected the PSO performance as well as how the global selection schemes affected the final produces results.

8.2 PSO COST 259 Siemens Results

The PSO was applied to the benchmarks on the following machine and frameworks:

- 4 GB RAM
- Windows 7
- Intel Quad Core CPU
- C# using .Net 4 Framework with Parallel Extensions

The FAP PSO was applied to Siemens1, Siemens2, Siemens3 and Siemen4 of the COST 259 benchmark suite. For more information about the nature of the benchmarks, the reader is directed to section 3.7.3.

For each benchmark, 12 results are presented. The following changes were made to the FAP PSO to obtain 12 different results:

- The two velocity method that were developed for the PSO were tested.
- Each velocity method also used the three different global best mechanisms that were developed.
- Together with different velocity methods and global best selection, the population size also alternated between 100 and 500.

The velocity methods and global selection schemes were discussed in the previous chapter.

8.2.1 Siemens1

Velocity method	GBest selection	Population	Interference
Method 1	Standard	100	104.64946291
Method 1	Built with cells	100	38.25217175
Method 1	Built with TRXs	100	126.63077549
Method 2	Standard	100	517.34796205
Method 2	Built with cells	100	252.82734335
Method 2	Built with TRXs	100	197.42134725
Method 1	Standard	500	106.13208087
Method 1	Built with cells	500	44.17120821
Method 1	Built with TRXs	500	118.91163136
Method 2	Standard	500	272.7178481399999
Method 2	Built with cells	500	123.42792595
Method 2	Built with TRXs	500	141.06015552
BEST	—	—	2.200

8.2.2 Siemens2

Velocity method	GBest selection	Population	Interference
Method 1	Standard	100	77.51130879
Method 1	Built with cells	100	55.4368279599999
Method 1	Built with TRXs	100	82.8570223200002
Method 2	Standard	100	228.486660409999
Method 2	Built with cells	100	195.921991349999
Method 2	Built with TRXs	100	100.29862742
Method 1	Standard	500	77.10497311
Method 1	Built with cells	500	54.88044032
Method 1	Built with TRXs	500	86.7949878400001
Method 2	Standard	500	236.22889478
Method 2	Built with cells	500	96.6051390999995
Method 2	Built with TRXs	500	96.8324254100002
BEST	—	—	14.271

8.2.3 Siemens3

Velocity Method	GBest Selection	Population	Interference
Method 1	Standard	100	138.77451397
Method 1	Built with cells	100	50.6521148400001
Method 1	Built with TRXs	100	157.55235973
Method 2	Standard	100	764.89452248
Method 2	Built with cells	100	311.646275819999
Method 2	Built with TRXs	100	316.998693
Method 1	Standard	500	141.1057882
Method 1	Built with cells	500	54.41708116
Method 1	Built with TRXs	500	148.64313238
Method 2	Standard	500	361.11838634
Method 2	Built with cells	500	167.18528179
Method 2	Built with TRXs	500	186.70589974
BEST	—	—	5.219

8.2.4 Siemens 4

Velocity Method	GBest Selection	Population	Interference
Method 1	Standard	100	541.617902899997
Method 1	Built with cells	100	293.23886995
Method 1	Built with TRXs	100	551.930406899999
Method 2	Standard	100	1877.7819344
Method 2	Built with cells	100	814.95027274
Method 2	Built with TRXs	100	930.63185040002
Method 1	Standard	500	538.812140599998
Method 1	Built with cells	500	302.44187813
Method 1	Built with TRXs	500	582.99206854
Method 2	Standard	500	1934.26035801001
Method 2	Built with cells	500	631.668030470002
Method 2	Built with TRXs	500	723.77245115003
BEST	—	—	77.246

8.3 The Performance of the PSO

In this section the effects of the changes made to the FAP PSO algorithm in terms of the results will be discussed.

Firstly, the effect of the two developed velocity methods in terms of the results is described. In section 8.3.2 the three global schemes that are used by the algorithm are discussed. Finally this section concludes with the effect of a larger population size on solution quality rather than a small population.

8.3.1 Velocity Method 1 vs Method 2

The FAP PSO algorithm is able to utilise two different velocity methods to move the swarm around in the FAP space. The algorithms that implement these two methods were presented in section 7.4 and section 7.4.3.

By analysing the results, it becomes abundantly clear that velocity method 1 is by far the superior method for moving in the problem space. In each of the results, when comparing end fitness values produced by algorithm variants that use method 1 one can easily come to the conclusion that method 1 performs better than method 2.

As discussed in chapter 7, method 1 uses a stage-based approach when applying the velocity function whereas method 2 applies the velocity function as is without it being broken up. Based on the results, using a stage-based approach to apply the velocity function is far better than applying the velocity equation directly to the transceiver in the plan.

Method 1 works by moving the whole swarm through to each stage before applying the next stage in the velocity equation. Thus after each stage, the whole swarm is at the same phase of the equation which keeps the swarm structured.

With method 2, the whole velocity equation is applied to whatever value is exposed to be operated on. Thus when method 2 moves a particle the frequency plan is moved piece by piece to some destination in the problem space.

By applying the velocity equation it is difficult to control the algorithm search process. With the FAP PSO control is necessary as there are various constraints which must not only be avoided but also adhered to for the generated plan to be usable. With method 2 adding domain knowledge is

difficult, since after the velocity equation has been calculated the particle is very close to being moved to a new position. All that still needs to be done, before a particle is moved, is to apply inertia, which means there is a little check that can be done to ensure that all the channels' values are within acceptable bounds.

By breaking the velocity equation up into smaller parts (stages) using method 1 the algorithm is able to direct and ensure that the swarm is moving in the general direction of valid channel allocations.

Also the algorithm is able to embed domain knowledge earlier into the calculation of the velocity and is therefore able to intercept early on movements that will result in invalid channel allocations at each stage of the velocity equation.

8.3.2 Different Global Schemes

In the previous chapter three global selection schemes were identified. The first global selection scheme uses the standard PSO selection and the particle with the best fitness is the global best. This scheme is called "Standard GBest".

As discussed in section 7.5, using the standard gbest selection scheme is not preferable as it can lead to the swarm losing out on good channel allocations due to overshadowing of channels[1].

Even with overshadowing the standard global scheme does not produce bad results, which seems to indicate that overshadowing of channel allocations does not impact the frequency plans as significantly as thought initially.

In addition to the standard gbest selection scheme, two other selection schemes were tested. It is actually incorrect to call these schemes selection schemes of gbest, since they build global best rather than select them.

By far the worst performing scheme is where the global best is built from transceivers. In every benchmark performed where this scheme was paired with a velocity method and population size, the algorithm was simply not able to produce any relatively good solutions. All possible solutions had high interference values, making them undesirable.

The bad performance of the build from transceivers scheme can be attributed to the granularity it uses to build a global best. As outlined in

[1] Overshadowing is discussed in chapter 7

section 7.5 the scheme only considers the interference generated by a single channel allocated to a transceiver. This would have worked well if there were some sort of guarantee that a particular transceiver would only be interfered with by one other transceiver.

In reality and in the Siemens4 benchmarks this is definitely not the case. More often than not, transceivers are interfered with by more than one other transceiver. Thus by only concentrating on a single case-by-case basis of channels allocated to transceivers, the scheme is discarding all other possible interferences.

It might select a channel at one point as the best, since in that scenario, the interference generated with the only other transceiver that is considered at that point is low. But this particular channel is too close on the spectrum to another channel allocated to some other transceiver that also interferes. Due to the algorithm only considering individual cases, this potential interference with the other transceiver will not be noticed by the algorithm and it will go ahead in selecting the channel as the best for the transceiver.

By analysing the results produced by the various FAP PSO algorithms, it can be concluded that the best global best selection scheme is by far the one in which cells are used to build a global best. With the cell selection scheme, the algorithm does not suffer the pitfall that is the reason for the transceiver gbest building scheme's bad performance.

As discussed the build from cells scheme uses cells to build a gbest, and thus each cell stores the interference that the channels allocated to its transceivers generate by interfering with other cells. As a cell interferes with other cells, the interference generated is added to the cell causing the interference.

After the PSO has calculated the fitness of all positions, each cell will contain the interference it personally has caused throughout the network to other cells. A cell with low interference means the channels that have been allocated to this particular cell are the best combination which causes the least amount of interference. Therefore, with the build gbest from cells scheme, the algorithm is able to make informed choices when selecting a cell to be included in the global best.

8.3.3 Population Size

As discussed in chapter 5 the population parameter of the PSO is a sensitive parameter. For problems with big search spaces, it is better to have a large population. A large population increases diversity, meaning more particles occupy different positions in the problem space and hence the space is better explored, which internally increases the likelyhood of finding a better solution. Thus, in the results the population size was also varied.

In the results using a larger population did not significantly improve the possible solutions. In only one benchmark, namely Siemens2, the algorithm was able to produce a better possible solution than an algorithm that used a lower population size. In Siemens1, Siemens3 and Siemens4 the algorithm using a lower population was able to produce the best overall results.

Due to the complexity of the FAP it is difficult to use the PSO with a large population. The algorithm's efficiency greatly decreased with the increase in population, taking significantly longer to iterate through the same number of iterations as the algorithms with lower populations. The decrease in efficiency was not unexpected due to the function evaluations, movements and considerations increasing fivefold over the lower population size. What was unexpected was the degree to which the efficiency degraded.

8.4 Summary

In this chapter the results produced by the algorithm discussed in chapter 7 were presented. The FAP PSO algorithm was applied to four COST 259 benchmarks namely Siemens1, Siemens2, Siemens3 and Siemens4. These four benchmarks were discussed in detail in chapter 3. For each of the benchmarks, 12 different variants of the FAP PSO algorithm were tested. Each variant used a different velocity function, global best selection scheme or population size. The chapter concluded with critical analyses of each of the different algorithms developed in this study to enable the PSO to operate in the FAP space.

Chapter 9

Conclusion

9.1 Introduction

The aim of this dissertation was to develop a modern artificial intelligence optimisation algorithm which would be applied to a problem in the cellular technology domain. This aim was achieved: an algorithm was developed based on the standard PSO algorithm which was then applied to operate on the FAP.

In current research no similar PSO algorithm to date has been presented that has been applied to the FS-FAP using interference as a performance measurement. In chapter 7 the innovation and new techniques that made it possible for an algorithm based on the PSO to operate in the FS-FAP domain were discussed.

9.2 Research Questions

At the start of the dissertation a series of research questions were outlined. These questions were identified and served as a roadmap to understand the problem domain as well as to gather enough background information on optimisation algorithmic techniques to allow innovation.

Six questions were identified in total. Each original question will now be listed together with a short discussion on how the question was answered and to which chapter it is related to.

- **Question 1** — *What is cellular technology, how was it got developed and what improvements have been made since its initial development?*

This question was answered in chapter 2, which provided the history of cellular technology and also highlighted improvements that have been made since the inception of cellular networks.

- **Question 2** — *What is the architecture behind a modern cellular network, how do the various hardware entities within a network communicate with each other and how is a communicational link established between two users of the network?* This question was answered in chapter 2 where the architecture of a modern-day cellular network was outlined and discussed in depth. Each entity used by the network to facilitate communication was identified and discussed in detail.

- **Question 3** — *What exactly is the frequency problem and how does it affect modern wireless communication?* This question was answered in chapter 3, which contains a description of the general problem of the frequency assignment problem as well as exactly what happens in a cellular network for this problem to occur.

- **Question 4** — *What variants of the frequency problem exist and which are most applicable to cellular networks?* The variants that currently exist in the problem domain were listed in chapter 3, together with their respective history and what domain they affect. The particular variant concentrated on in this dissertation concentrated on was formally stated.

- **Question 5** — *What are the most popular optimisation algorithms and what characteristics make them unique?* This question was answered in two chapters namely chapters 4 and 5. In these chapters the most popular and modern optimisation algorithms were evaluated.

- **Question 6** — *With algorithms that achieved success in their respective optimisation problems, what particular technique is used by the algorithms that them it to achieve better performance?* Chapters 4 and 5 covered modern optimisation algorithms, which were also evaluated. Exactly what made the algorithm unique and allowed it to achieve success in the domain it was applied to was also dealt with

9.3 Research Objective

As stated in the introduction, the objective of this research was to determine whether it is viable and possible to apply a modern-day swarm-based algorithm to the frequency assignment problem.

The question therefore is whether this research objective was achieved. The answer is yes. An algorithm based on the PSO algorithm was presented that operated on the frequency assignment problem. The only part still left is the viability of applying this algorithm to modern cellular networks.

Before the viability analysis is presented, the chapter breakdown in the introduction will be revisited in the next section.

9.4 Chapter Breakdown Revisited

The chapter breakdown presented in the introduction is provided once more but with a summary of exactly what was discussed in each particular chapter.

9.4.1 Part I - Background

Chapter 1

This chapter provided an introduction to the research as well as a broad overview of the topics the dissertation discussed.

Chapter 2

In this chapter a broad discussion was given on modern cellular technology, specifically the GSM cellular network technology. A brief history on how GSM was developed to be the most widely used cellular technology in use today was presented in section 2.2.1. This was followed by various GSM architecture entities.

A broad overview followed in section 2.4 on the various communication interfaces used between the various GSM entities to communicate with each other. The different GSM channels which are used on the interfaces to communicate information were covered.

The chapter concluded with the handover process which is used to allow an MS device to move freely geographically within the network.

Chapter 3

The problem defined in this dissertation is the frequency assignment problem (FAP). The problem where the problem was categorised as being part of the set of NP-Complete problems. The NP-Complete nature of the problem is an important concept to understand.

Within the FAP domain there exists are different techniques when assigning frequencies, discussed in section 3.3.

The FAP is not just one problem but consists of various subproblems that have different goals for the resulting frequency plan. Some problems are concerned with the number of frequencies used, others are more concerned with the amount of interference that is generated on the network because of the assignment. The various FAP subproblems were outlined in section 3.5.

A formal mathematical definition of the MI-FAP was also given. Finally, the chapter concluded with the various benchmark problems which are used to test the viability of frequency assignment algorithms.

Chapter 4

In this chapter a definition was given of what it means for an algorithm to be classified as metaheuristic and also what characteristics these algorithms must exhibit.

Three metaheuristic algorithms were discussed in this chapter. Each algorithm was accompanied by a flow chart depicting the general flow of the algorithm. How the algorithm works as well as the various characteristics that make the algorithm unique were described.

For each algorithm, a brief overview of literature using the particular algorithm was given as well as some of the disadvantages or challenges faced when applying the particular algorithm to the FAP.

Chapter 5

In this chapter a discussion was presented on three swarm intelligence algorithms. For each of the three algorithms a flow chart was given to allow the reader a more visual view of the operation of the algorithm.

Each algorithm was analysed in detail to identify its strengths and the techniques it uses to achieve good results. Finally the algorithm was critically evaluated in the terms of being applied to the FAP.

9.4.2 Part II - Implementation

Chapter 6

In this chapter a series of mathematical optimisation problems were presented. Each problem was mathematically formulated as well as categorised as to whether the particular problem was multimodal and separable. To get a better idea of how the particle swarm optimisation (PSO) algorithm operates and performs, two PSO algorithms were developed. Finally, this chapter concluded with the results obtained by the algorithms.

Chapter 7

This chapter contained an algorithm based on the standard PSO algorithm to operate on the FAP encountered in cellular networks.

All the various problems as well as how they were solved during the development of the main algorithm of this dissertation were described. For each new or old technique used by the algorithm pseudocode was given for a better idea of how the algorithm uses it.

The chapter concluded with small additions made to the algorithm to improve performance and, most important of all, improve solution quality.

Chapter 8

In this chapter results produced by the algorithm discussed in chapter 7 were presented. The FAP PSO algorithm was applied to four COST 259 benchmarks namely Siemens1, Siemens2, Siemens3 and Siemens4. These four benchmarks were discussed in detail in chapter 3. For each of the benchmarks, 12 different variants of the FAP PSO algorithm were tested. Each variant used a different velocity function, global best selection scheme or population size. The chapter concluded with a critical analysis of each of the different algorithms developed for this dissertation to enable the PSO to operate in the FAP space.

9.4.3 The Conclusion

In chapter 7 new velocity methods were created to enable the PSO swarm to move around in the problem space. To further improve the performance of the PSO additional techniques were developed.

The additional techniques were mainly concerned with how the gbest for the swarm is selected. The algorithms for these methods and techniques were also listed in the chapter. Not all the techniques used in the FAP PSO were entirely new; some techniques that were used can be easily identified like the use of tabu lists. Other techniques are a bit more difficult to identify. The building of the gbest actually borrows a concept used by ants in the ACO algorithm to build solutions.

The FAP PSO algorithm uses random collision resolution when a particle position is already in the tabu list. The random collision resolution works by randomly selecting a new channel until it is found not to be tabu. This procedure is very similar to the mutation operation used by the GA algorithm.

In chapter 8 the results of the FAP PSO algorithm applied to the COST 259 Siemens benchmark were discussed. By critically evaluating the results one can conclude that the best performing PSO variant is the particular algorithm using the first developed velocity method, which utilises cells to build the global best. The results achieved by this variant of the algorithm greatly outweigh those of the other algorithms, but when compared with the best achieved in the literature the algorithm still has some way to go.

Being so far off the best achieved results in the literature is not surprising, as applying the PSO to the FS-FAP and to the COST 259 benchmark is a first. Nonetheless this research has shown that it is indeed viable to apply the PSO to the FS-FAP, and with more research it is possible that the PSO algorithm might be able to either come near the best presented results or actually improve upon them.

9.5 Future Work

Most of the techniques developed for the FAP PSO, aim to stay true to the standard PSO algorithm. The next step is to hybridise the FAP PSO algorithm. A good candidate for hybridisation would be the GA as the

algorithm naturally "purifies" results in finding the right genes that make up a good possible solution.

In the PSO a good entry point for the GA to be incorporated would be twofold. The first point would be to take a certain global best selected with the standard procedure and then mate it with successive global bests, with the offspring global best being a best for a future iteration. With this method the PSO is able to use the history of the algorithm from the start.

The second method takes a swarm of a certain iteration and then mates all the particles with each other for a certain number of iterations. This method can be seen as a means to clean the swarm of inefficient genes and could serve as an intensification phase for the algorithm.

Another point of interest is to disregard the PSO and rather try and produce an ABC algorithm on the FS-FAP. As mentioned, the ABC algorithm was not chosen for this research, since it is new and has not been applied to a wide variety of problems.

By using some of the techniques developed in this research for instance the selection schemes, a viable ABC algorithm could be developed. The ABC algorithm is not that specific as to how new solutions are generated, as with the PSO which relies on vector mathematics, which allows an algorithm designer considerably more freedom.

Appendices

Appendix A

Plotting Functions in 3D

A.1 Introduction

In this section all the functions presented in chapter 6 page 141 will be plotted in 3D.

The following graphs were generated using Matplotlib[1] which is a python library that provides similar functionality to Matlab.

For each of the benchmark functions, the 3D graph along with the python code that was used to generate the graph will be presented.

A.2 Code

A.2.1 DeJong F1 Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150

def DeJongF1(x,y):
        return x**2 + y**2


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)
```

[1] address

```
print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = DeJongF1(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('DeJongF1')
print '----Complete----'
```

A.2.2 Shekel's Foxhole Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150

def insertValuesIntoMatrix(matrix,value,row,index,timesToInsert):
        if matrix.size / 2 >= index + timesToInsert:
                for i in range(timesToInsert):
                        matrix[row,index+i] = value


def createDeJongF5Matrix():
        a = np.array([])
        a.resize(2,25)
        for i in range(2):
                value = -32
                for j in range(25):
                        a[0,j] = value
                        value = value + 16
                        if j > 0 and (j+1) % 5 == 0:
                                value = -32
                                valueIndex = ((j + 1) / 5) - 1
                                startIndex = valueIndex * 5
                                insertValuesIntoMatrix(a,a[0,valueIndex],1,startIndex,5)
        return a

def DeJongF5(x,y,matrix):
        sumj = 0;
        for j in range(25):
                sumi = 0
                sumi = (x - matrix[0,j])**6 + (y - matrix[1,j])**6
                sumj = sumj + (j + sumi)**-1
        return (0.002 + sumj)**-1


fig = plt.figure()
ax = Axes3D(fig)
```

```
X = np.linspace(-65.356, 65.356, AMOUNT_OF_POINTS)
Y = np.linspace(-65.356, 65.356, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

dejongList = []
print 'Initializing Function'
deJongMatrix = createDeJongF5Matrix()
for i in range(AMOUNT_OF_POINTS):
        val = DeJongF5(X[i],Y[i],deJongMatrix)
        dejongList.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(dejongList)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

ax.set_zlim3d(-40,500)
#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Shekel Foxhole')
print '----Complete----'
```

A.2.3 Rastrigin Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150

def Rastrigin(x,y):
        rastriginSum = 0
        rastriginSum += x**2 + 10*np.cos(2*np.pi*x) + 10
        rastriginSum += y**2 + 10*np.cos(2*np.pi*y) + 10
        return rastriginSum


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Rastrigin(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'
```

```
#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Rastrigin')
print '----Complete----'
```

A.2.4 Schwefel Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
SCHWEFEL_CONSTANT = 418.9829 * 2
def Schwefel(x,y):
        schwefelSum = 0
        schwefelSum += (-1 * x)*np.sin(np.sqrt(abs(x)))
        schwefelSum += (-1 * y)*np.sin(np.sqrt(abs(y)))
        return SCHWEFEL_CONSTANT + schwefelSum


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-500, 500, AMOUNT_OF_POINTS)
Y = np.linspace(-500, 500, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Schwefel(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Schwefel')
print '----Complete----'
```

A.2.5 Griewank Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np
```

```
AMOUNT_OF_POINTS = 150

#http://www.geatbx.com/docu/fcnindex-01.html
def Griewank(x,y):
        griewankSum1 = (x**2)/4000 + (y**2)/4000

        griewankSum2 = np.cos(x / np.sqrt(1)) * np.cos(y / np.sqrt(2))
        return griewankSum1 - griewankSum2 + 1


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-600, 600, AMOUNT_OF_POINTS)
Y = np.linspace(-600, 600, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Griewank(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Griewank')
print '----Complete----'
```

A.2.6 Salomon Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 350

def Salomon(x,y):
        SalomonSum1 = 0
        SalomonSum1 += x ** 2
        SalomonSum1 += y ** 2
        SalomonSum1 = -np.cos(2*np.pi*np.sqrt(SalomonSum1))
        SalomonSum2 = 0
        SalomonSum2 += (x ** 2)# + 1
        SalomonSum2 += (y ** 2)# + 1
        SalomonSum2 = 0.1 * np.sqrt(SalomonSum2)+1
        return SalomonSum1 + SalomonSum2


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5, 5, AMOUNT_OF_POINTS)
Y = np.linspace(-5, 5, AMOUNT_OF_POINTS)
```

```
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Salomon(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('SalomonTest')
print '----Complete----'
```

A.2.7 Ackley Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
#http://www.geatbx.com/docu/fcnindex-01.html
def Ackley(x,y):
        a = 20
        b = 0.2
        c = 2* np.pi
        ndiv = 1.0 / 2.0
        AckleySum = -a * np.exp(-b * np.sqrt(ndiv*(x** 2 + y ** 2)))

        AckleySum -= np.exp(ndiv * (np.cos(c*x) + np.cos(c*y))) + a + np.exp(1)
        return AckleySum


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-32.768, 32.768, AMOUNT_OF_POINTS)
Y = np.linspace(-32.768, 32.768, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Ackley(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'
```

```
#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Ackley')
print '----Complete----'
```

A.2.8 Six-hump Camel Back Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
#http://www.geatbx.com/docu/fcnindex-01.html
def Camel(x,y):
        CamelSum = (4 - 2.1 * (x ** 2) + (x ** 4.0)/3.0) * x ** 2 + (x * y) + (-4 + 4 * y **2)* y ** 2
        return CamelSum


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-3, 3, AMOUNT_OF_POINTS)
Y = np.linspace(-2, 2, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Camel(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Camel')
print '----Complete----'
```

A.2.9 Shubert Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
```

```
def Shubert(x,y):
        answ1 = 0.0
        for i in range(4):
                answ1 += i * np.cos((i+1) * x + i)
        answ2 = 0.0
        for j in range(4):
                answ2 += i*np.cos((i+1) * y + i)
        return answ1 * answ2


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Shubert(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Shubert')
print '----Complete----'
```

A.2.10 Himmelblau Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Himmelblau(x,y):
        answ1 = (x ** 2 + y - 11)**2 + (x + y ** 2 - 7) ** 2
        return answ1


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Himmelblau(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
```

```
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Himmelblau')
print '----Complete----'
```

A.2.11 Rosenbrock Valley Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Rosenbrock(x,y):
        answ1 = 100 * ((x - y**2) ** 2) + (1-x)**2
        return answ1


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-2.048, 2.048, AMOUNT_OF_POINTS)
Y = np.linspace(-2.048, 2.048, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Rosenbrock(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('RosenbrockTest')
print '----Complete----'
```

A.2.12 Dropwave Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
```

```
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Dropwave(x,y):
        powsum = x ** 2 + y ** 2
        answ = 1 + np.cos(12 * np.sqrt(powsum))
        answ /= 0.5 * powsum + 2
        return answ


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Dropwave(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Dropwave')
print '----Complete----'
```

A.2.13 Easom Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Easom(x,y):
        answ = -np.cos(x)*np.cos(y)*np.exp(-1 * ((x - np.pi) ** 2) - ((y - np.pi) ** 2))
        return answ


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-100, 100, AMOUNT_OF_POINTS)
Y = np.linspace(-100, 100, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Easom(X[i],Y[i])
```

```
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Easom_-100_+100')
print '----Complete----'
```

A.2.14 Branins Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Branin(x,y):
        a = 1
        b = 5.1 / (4 * np.pi) ** 2
        c = 5 / np.pi
        d = 6
        e = 10
        f = 1 / 8 * np.pi
        answ = a * (y - b * (x**2) + c*x - d)**2 + e * (1 - f)*np.cos(x) + e
        return answ


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5, 10, AMOUNT_OF_POINTS)
Y = np.linspace(0, 15, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Branin(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
```

```
plt.savefig('Branin')
print '----Complete----'
```

A.2.15 Michalewicz Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Michalewicz(x,y):
        m = 20
        sumAnswx = np.sin(x) * (np.sin((1 - x ** 2) / np.pi))**(2 * m)
        sumAnswy = np.sin(y) * (np.sin((1 - y ** 2) / np.pi))**(2 * m)
        return -1 * (sumAnswx + sumAnswy)


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(0, np.pi, AMOUNT_OF_POINTS)
Y = np.linspace(0, np.pi, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Michalewicz(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Michalewicz')
print '----Complete----'
```

A.2.16 Goldstein Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Goldstein(x,y):
        answ = (1 + (( x + y + 1) ** 2) * (19 - 14 * x + ((3*x)**2) - 14 * y + 6 * x * y + ((3 * y) ** 2)))
        answ *= (30 + ((2 * x - 3 * y)**2) * (18 -32 * x +  (12 * x)**2 + 48 * y - 36 * x * y + (27 * y)**2))
        return answ  / 1000000
```

```
fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-2, 2, AMOUNT_OF_POINTS)
Y = np.linspace(-2, 2, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Goldstein(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Goldstein')
print '----Complete----'
```

A.3 Graphs

This section contains 3D graphs of all the formulated functions. These enable one to see more clearly the problem space the algorithm is searching in.

Each of the graphs has a coloured scale on the right-hand side and the graph follows this coloured scale. The scale ranges from the maximum value to the minimum value encountered in the particular function's problem space. The maximum value in the problem space is indicated in red and the minimum value is indicated in blue.

Finally the graphs are in 3 dimensions, and the x and y dimensions represent any numerical number. The z dimension represents the value produced by using the function to evaluate the particular x and y coordinates.

A.3.1 DeJong's First Function

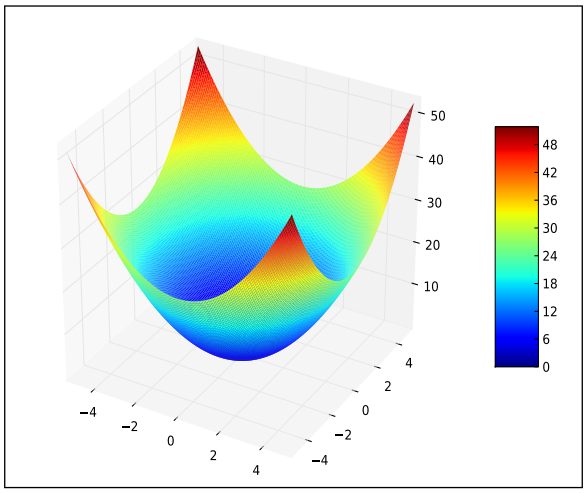

Figure A.1: DeJong's first function

A.3.2 Shekel's Foxhole Function

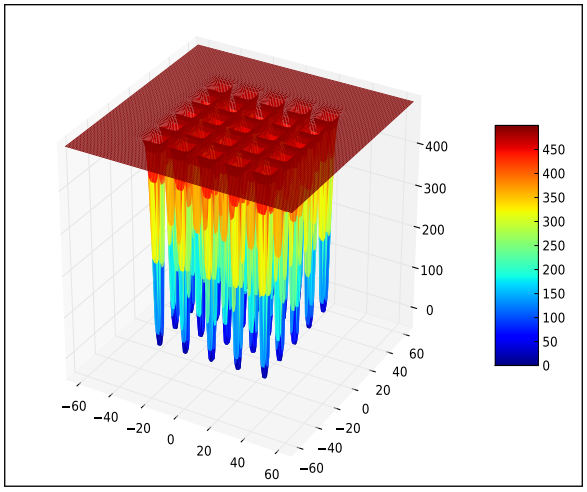

Figure A.2: Shekel's foxhole function

A.3.3 Rastrigin Function

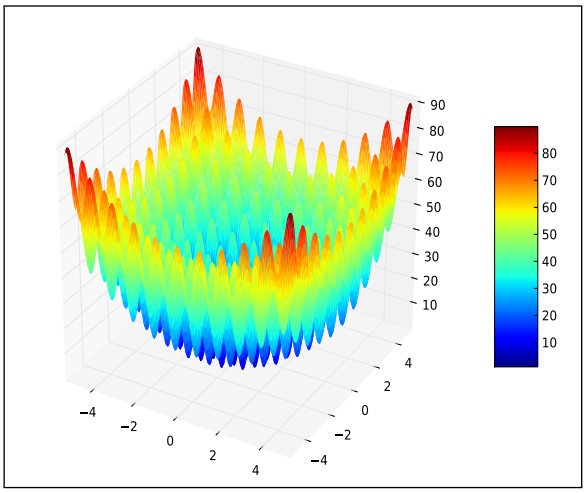

Figure A.3: The Rastrigin function

A.3.4 Schwefel Function

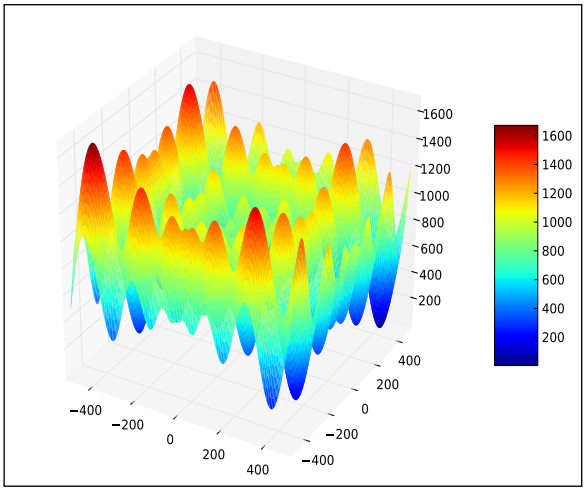

Figure A.4: Schwefel function

A.3.5 Griewank Function

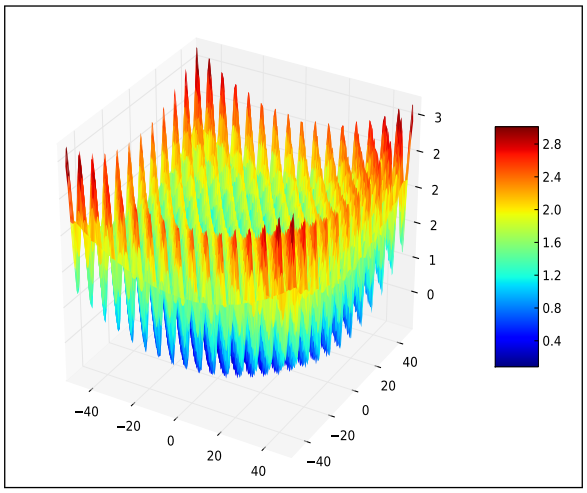

Figure A.5: Griewank function

A.3.6 Salomon Function

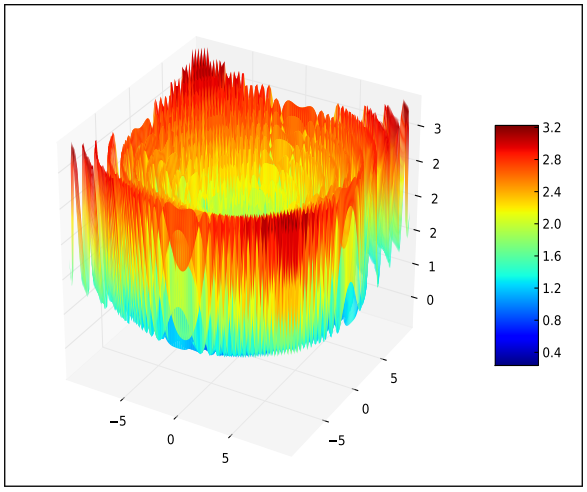

Figure A.6: Salomon function

A.3.7 Ackley

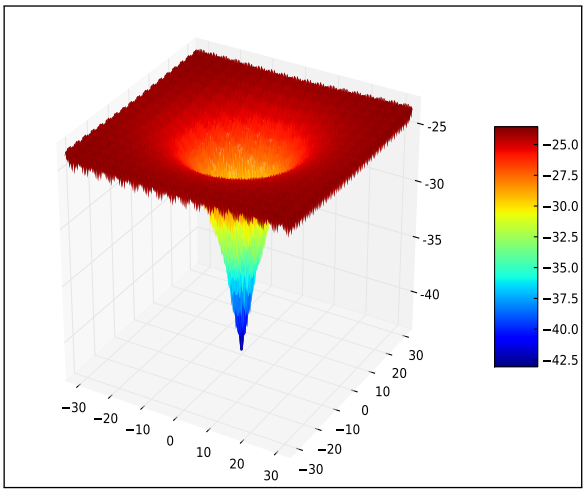

Figure A.7: Ackley function

A.3.8 Six-Hump Camel Back Function

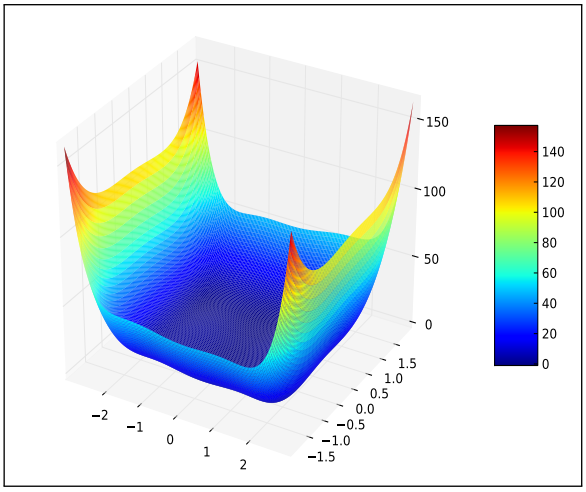

Figure A.8: Six-hump camel back function

A.3.9 Shubert Function

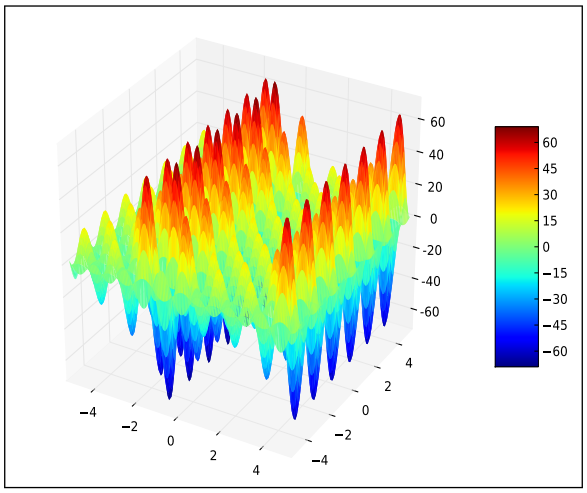

Figure A.9: Shubert function

A.3.10 Himmelblau Function

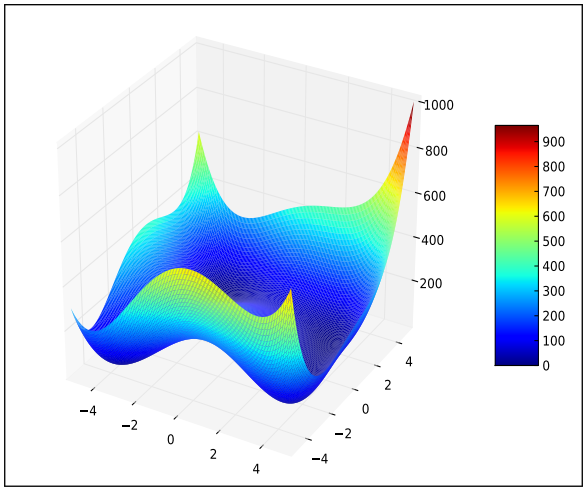

Figure A.10: Himmelblau function

A.3.11 Rosenbrock Valley Function

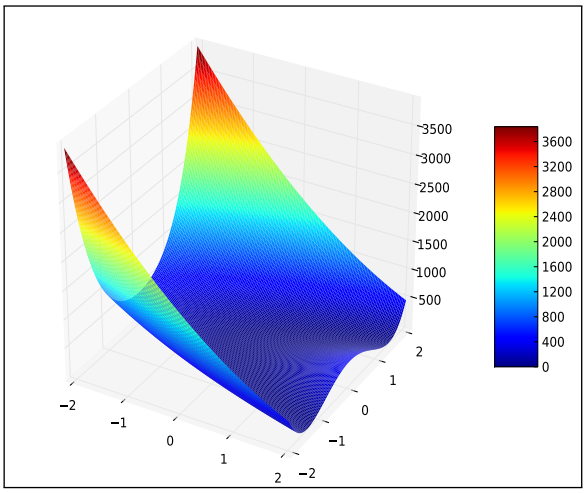

Figure A.11: Rosenbrock valley function

A.3.12 Dropwave Function

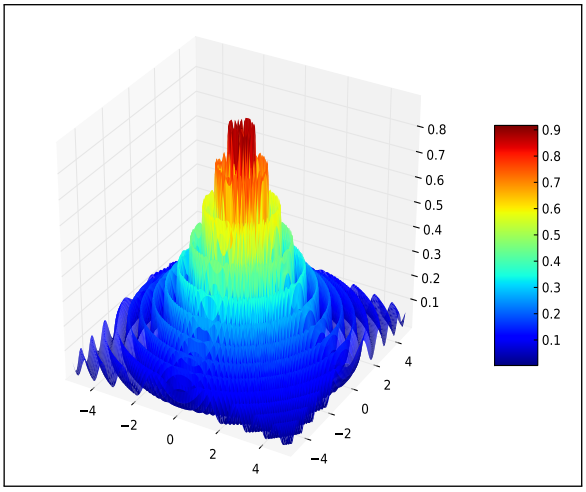

Figure A.12: Dropwave function

A.3.13 Easom Function

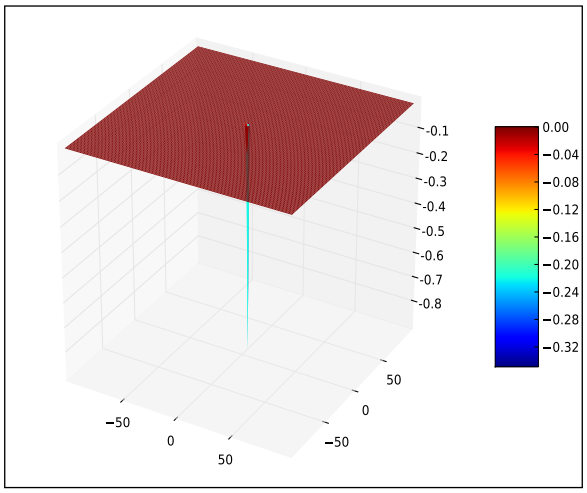

Figure A.13: Easom function

A.3.14 Branin Function

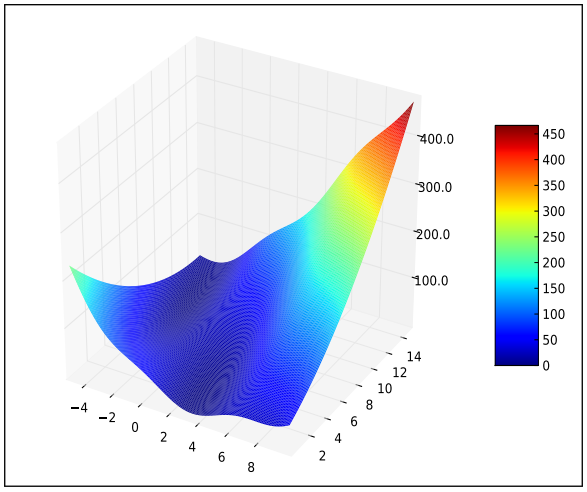

Figure A.14: Branin function

A.3.15 Michalewicz Function

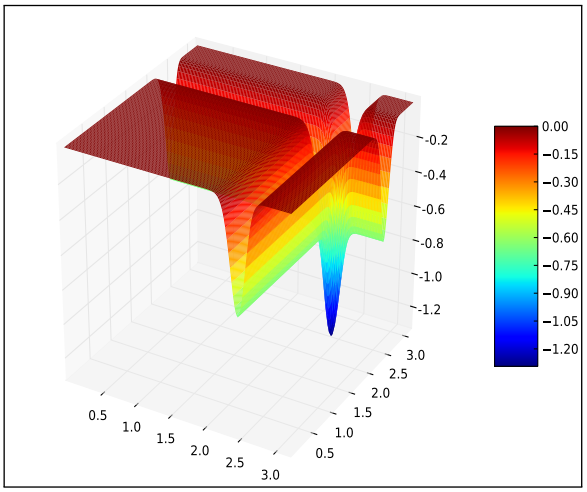

Figure A.15: Michalewicz function

A.3.16 Goldstein Function

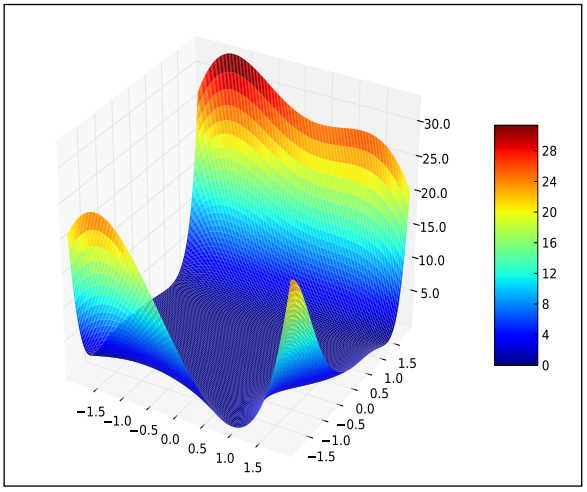

Figure A.16: The Goldstein function

Bibliography

[1] LAPPEENRANTA UNIVERSITY OF TECHNOLOGY: EVOLUATIONARY COMPUTATION PAGES (Accessed 2011/06/09). http://www.it.lut.fi/ip/evo/functions/node12.html.

[2] Karen Aardal, Cor Hurkens, Jan Karel Lenstra, and Sergey Tiourine. Algorithm for radio link frequency assignment: The calma project. *Operations Research*, 50(6):968–980, Nov - Dec 2002.

[3] Karen I. Aardal, Stan P. M. van Hoesel, Arie M. C. A. Koster, Carlo Mannino, and Antonio Sassano. Models and solution techniques for frequency assignment problems. *4OR: A Quarterly Journal of Operations Research*, 1(4):261–317, December 2004.

[4] Bilal Alatas. Chaotic bee colony algorithms for global numerical optimization. *Expert Syst. Appl.*, 37:5682–5687, August 2010.

[5] E. Alba, F. Luna, A. J. Nebro, and J. M. Troya. Parallel heterogeneous genetic algorithms for continuous optimization. *Parallel Comput.*, 30(5-6):699–719, 2004.

[6] Mahmoud H. Alrefaei and Sigrún Andradóttir. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science*, 45:748–764, 1999.

[7] S. Areibi and A. Vannelli. Circuit partitioning using a tabu search approach. In *Circuits and Systems, 1993., ISCAS '93, 1993 IEEE International Symposium on*, pages 1643 –1645, 3-6 1993.

[8] A. Augugliaro, L. Dusonchet, and E. R. Sanseverino. An evolutionary parallel tabu search approach for distribution systems reinforcement planning. *Advanced Engineering Informatics*, 16(3):205 – 215, 2002.

[9] I. Badarudin, A.B.M. Sultan, M.N. Sulaiman, A. Mamat, and M.T.M. Mohamed. Metaheuristic approaches for optimizing agricultural land areas. In *Data Mining and Optimization, 2009. DMO '09. 2nd Conference on*, pages 28 –31, 27-28 2009.

[10] Hua Bai and Bo Zhao. A survey on application of swarm intelligence computation to electric power system. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 2, pages 7587 –7591, 0-0 2006.

[11] T.R. Benala, S.D. Jampala, S.H. Villa, and B. Konathala. A novel approach to image edge enhancement using artificial bee colony optimization algorithm for hybridized smoothening filters. In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 1071 –1076, 9-11 2009.

[12] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353 – 373, 2005.

[13] Bruce M. Blumberg. *Exploring Artificial Intelligence in the New Millenium*, chapter D-Learning: what learning in dogs tells us about building characters that learn what they ought to learn, pages 37–67. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[14] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin. Frequency assignment in cellular phone networks. *Annals of Operations Research*, (76):73–93, 1998.

[15] Ralf Borndörfer, Andreas Eisenblätter, Martin Grötschel, and Alexander Martin. The orientation model for frequency assignment problems. Technical report, Konrad-Zuse-Zentrum Berlin, 1998.

[16] Jeffrey E. Boyd, Gerald Hushlak, and Christian J. Jacob. Swarmart: interactive art from swarm intelligence. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 628–635, New York, NY, USA, 2004. ACM.

[17] L. Cavique, C. Rego, and I. Themido. Subgraph ejection chains and tabu search for the crew scheduling problem. *The Journal of the Operational Research Society*, 50:608–616, 1999.

[18] A. Chawla, S. Mukherjee, and B. Karthikeyan. Characterization of human passive muscles for impact loads using genetic algorithm and inverse finite element methods. *Biomechanics and Modeling in Mechanobiology*, 8(1):67–76, 2009.

[19] Rachid Chelouah and Patrick Siarry. Tabu search applied to global optimization. *European Journal of Operational Research*, 123:256–270, 2000.

[20] Ruey-Maw Chen, Shih-Tang Lo, Chung-Lun Wu, and Tsung-Hung Lin. An effective ant colony optimization-based algorithm for flow shop scheduling. In *Soft Computing in Industrial Applications, 2008. SMCia '08. IEEE Conference on*, pages 101 –106, june 2008.

[21] Chin Soon Chong, Appa Iyer Sivakumar, Malcolm Yoke Hean Low, and Kheng Leng Gay. A bee colony optimization algorithm to job shop scheduling. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 1954–1961. Winter Simulation Conference, 2006.

[22] G. Colombo and S.M. Allen. Problem decomposition for minimum interference frequency assignment. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 3492 –3499, sept. 2007.

[23] Agnieszka Debudaj-Grabysz and Zbigniew J. Czech. Theoretical and practical issues of parallel simulated annealing. In *PPAM'07: Proceedings of the 7th international conference on Parallel processing and applied mathematics*, pages 189–198, Berlin, Heidelberg, 2008. Springer-Verlag.

[24] B. Denby and S. Le Hgarat-Mascle. Swarm intelligence in optimisation problems. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 502(2-3):364 – 368, 2003. Proceedings of the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research.

[25] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243 – 278, 2005.

[26] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243 – 278, 2005.

[27] Marco Dorigo, Eric Bonabeau, and Guy Theraulaz. Ant algorithms and stigmergy. *Future Gener. Comput. Syst.*, 16(9):851–871, 2000.

[28] K A Dowsland and J M Thompson. Solving a nurse scheduling problem with knapsacks, networks and tabu search. *J Oper Res Soc*, 51(7):825–833, 07 2000.

[29] Audrey Dupont, Andrea Carneiro Linhares, Christian Artigues, Dominique Feillet, Philippe, and Michel Vasquez. The dynamic frequency assignment problem. *European Journal of Operational Research*, 195:75–88, 2009.

[30] A. Eisenblätter, M. Grötschel, and A. Martin. Frequency planning and rammifications of colouring. Technical report, Konrad-Zuse-Zentrum fu r Informationstechnik Berlin, 2000.

[31] Andreas Eisenblätter. Assigning frequencies in gsm networks. Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), 2001.

[32] Andreas Eisenblätter. *Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2001.

[33] H.M. Elkamchouchi, H.M. Elragal, and M.A. Makar. Channel assignment for cellular radio using particle swarm optimization. In *Radio Science Conference, 2006. NRSC 2006. Proceedings of the Twenty Third National*, volume 0, pages 1 –9, 14-16 2006.

[34] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006.

[35] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, 2007.

[36] Dr. Kamilo Feher. *Wireless Digital Communications: Modulation & Spread Sprectrum Applications*. Prentice Hall, 1995.

[37] N. Fescioglu-Unver and M.M. Kokar. Application of self controlling software approach to reactive tabu search. In *Self-Adaptive and Self-Organizing Systems, 2008. SASO '08. Second IEEE International Conference on*, pages 297 –305, 20-24 2008.

[38] L. M. Gambardella, ÉD Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *The Journal of the Operational Research Society*, 50(2):167–176, Feb 1999.

[39] Gautam Garai and B. B. Chaudhuri. A distributed hierarchical genetic algorithm for efficient optimization and pattern matching. *Pattern Recogn.*, 40(1):212–228, 2007.

[40] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[41] Antonio Gómez-Iglesias, Miguel A. Vega-Rodríguez, Francisco Castejón, Miguel Cárdenas-Montes, and Enrique Morales-Ramos. Artificial bee colony inspired algorithm applied to fusion research in a grid computing environment. In *PDP '10: Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 508–512, Washington, DC, USA, 2010. IEEE Computer Society.

[42] Michael T. Goodrich. *Data Structures and Algorithms in Java*. John Wiley & Sons, 2005.

[43] Mohan Gopalakrishnan, Ke Ding, Jean-Marie Bourjolly, and Srimathy Mohan. A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. *Manage. Sci.*, 47(6):851–863, 2001.

[44] J.S Graham, R. Montemanni, J. N J. Moon, and D. H. Smith. Frequency assignment. multiple interference and binary constraints. *Wireless Networking*, 14:449–464, 2008.

[45] C. Grosan and A. Abraham. *Stigmergic optimization: inspiration, technologies and perspectives*, volume 31 of *Studies in Computational Intelligence*, chapter 1, pages 1–24. Springer Berlin / Heidelberg, 2006.

[46] Timo Hämäläinen, Harri Klapuri, Jukka Saarinen, Pekka Ojala, and Kimmo Kaski. Accelerating genetic algorithm computation in tree shaped parallel computer. *J. Syst. Archit.*, 42(1):19–36, 1996.

[47] Julia Handl and Bernd Meyer. Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113, December 2007.

[48] Abdel-rahman Hedar and Masao Fukushima. Tabu search directed by direct search methods for nonlinear global optimization. *European Journal of Operational Research*, 170:329–349, 2006.

[49] Alan Herz, David Schindl, and Nicolas Zufferey. Lower bounding and tabu search procedures for the frequency assignment problem with polarization constraints. *4OR: A Quarterly Journal of Operations Research*, 3:139–161, 2005.

[50] O. Holthaus and C. Rajendran. A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs. *The Journal of the Operational Research Society*, 56(8):pp. 947–953, 2005.

[51] Shun-Fa Hwang and Rong-Song He. Improving real-parameter genetic algorithm with simulated annealing for engineering problems. *Adv. Eng. Softw.*, 37(6):406–418, 2006.

[52] Lhassane Idoumghar and René Schott. Two distributed algorithms for the frequency assignment problem in the field of radio broadcasting. *IEEE Transactions on Broadcasting*, 55:223–229, 2009.

[53] C. Bettstetter J. Eberspächer, H.-J. Vögel and C. Hartmann. *GSM - Architecture, Protocols and Services*. Wiley Publishing, third edition, 2009.

[54] D.M. Jaeggi, G.T. Parks, T. Kipouros, and P.J. Clarkson. The development of a multi-objective tabu search algorithm for continuous optimisation problems. *European Journal of Operational Research*, 185(3):1192 – 1212, 2008.

[55] A.A. Javadi, R. Farmani, and T.P. Tan. A hybrid intelligent genetic algorithm. *Advanced Engineering Informatics*, 19(4):255 – 262, 2005. Computing in Civil Engineering.

[56] I. Jovanoski, I. Chorbev, D. Mihajlov, and I. Dimitrovski. Tabu search parameterization and implementation in a constraint programming library. In *EUROCON, 2007. The International Conference on Computer as a Tool*, pages 459 –464, 9-12 2007.

[57] Wei jun Xia and Zhi ming Wu. A hybrid particle swarm optimization approach for the job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 29(3):360–366, June 2006.

[58] Vijay Kalivarapu, Jung-Leng Foo, and Eliot Winer. Improving solution characteristics of particle swarm optimization using digital pheromones. *Structural and Multidisciplinary Optimization*, 37:415 – 427, 2009.

[59] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Appl. Soft Comput.*, 8:687–697, January 2008.

[60] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108 – 132, 2009.

[61] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *J. of Global Optimization*, 39(3):459–471, 2007.

[62] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *J. of Global Optimization*, 39:459–471, November 2007.

[63] Akbar Karimi, Hadi Nobahari, and Patrick Siarry. Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions. *Comput. Optim. Appl.*, 45:639–661, April 2010.

[64] Il kwon Jeong and Ju jang Lee. Adaptive simulated annealing genetic algorithm for system identification. *Engineering Applications of Artificial Intelligence*, 9(5):523 – 532, 1996.

[65] Sergio Ledesma, Miguel Torres, Donato Hernández, Gabriel Aviña, and Guadalupe García. Temperature cycling on simulated annealing

for neural network learning. In *MICAI 2007: Advances in Artificial Intelligence*, pages 161–171, 2007.

[66] Zne-Jung Lee, Chou-Yuan Lee, and Shun-Feng Su. An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem. *Applied Soft Computing*, 2(1):39 – 47, 2002.

[67] K. Lenin and M.R. Mohan. Attractive and repulsive particle swarm optimization for reactive power optimization. *Journal of Engineering and Applied Sciences*, 1(4):288–292, 2006.

[68] Yuming Liang and Lihong Xu. Mobile robot global path planning using hybrid modified simulated annealing optimization algorithm. In *GEC '09: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 309–314, New York, NY, USA, 2009. ACM.

[69] Nguyen Tung Linh and Nguyen Quynh Anh. Application artificial bee colony algorithm (abc) for reconfiguring distribution network. In *Computer Modeling and Simulation, 2010. ICCMS '10. Second International Conference on*, volume 1, pages 102 –106, 22-24 2010.

[70] Hongbo Liu, Ajith Abraham, and Maurice Clerc. Chaotic dynamic characteristics in swarm intelligence. *Applied Soft Computing*, 7(3):1019 – 1026, 2007.

[71] Francisco Luna, Christian Blum, Enrique Alba, and Antonio J. Nebro. Aco vs eas for solving a real-world frequency assignment problem in gsm networks. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 94–101, New York, NY, USA, 2007. ACM.

[72] H. Mahmoudzadeh and K. Eshghi. A metaheuristic approach to the graceful labeling problem of graphs. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 84 –91, 1-5 2007.

[73] S. Mallela and L.K. Grover. Clustering based simulated annealing for standard cell placement. In *Design Automation Conference, 1988. Proceedings., 25th ACM/IEEE*, pages 312 –317, 12-15 1988.

[74] Vittoria Maniezzo and Roberto Montemanni. An exact algorithm for the min-interference frequency assignment problem. Technical report, Department of Computer Science, University of Bologna, 2000.

[75] Carlo Mannino and Gianpaolo Oriolo. Solving stability problems on a superclass of interval graphs. Technical report, Centro Vito Volterra, 2002.

[76] Y. Marinakis, M. Marinaki, and N. Matsatsinis. A hybrid discrete artificial bee colony - grasp algorithm for clustering. In *Computers Industrial Engineering, 2009. CIE 2009. International Conference on*, pages 548 –553, 6-9 2009.

[77] Asha Mehrotra. *GSM System Engineering*. Arteh House, Inc, 1997.

[78] David Meignan, Jean-Charles Créput, and Abderrafiaa Koukam. A cooperative and self-adaptive metaheuristic for the facility location problem. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 317–324, New York, NY, USA, 2009. ACM.

[79] George Jiri Mejtsky. The improved sweep metaheuristic for simulation optimization and application to job shop scheduling. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 731–739. Winter Simulation Conference, 2008.

[80] P.R.S. Mendonca and L.P. Caloba. New simulated annealing algorithms. In *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on*, volume 3, pages 1668 –1671 vol.3, 9-12 1997.

[81] Marie-Bernadette Pautet Michel Mouly. *The GSM System for Mobile Communications*. Cell & Sys, 1992.

[82] Qi Ming-yao, Miao Li-xin, Zhang Le, and Xu Hua-yu. A new tabu search heuristic algorithm for the vehicle routing problem with time windows. In *Management Science and Engineering, 2008. ICMSE 2008. 15th Annual Conference Proceedings., International Conference on*, pages 1648 –1653, 10-12 2008.

[83] M. Molga and C. Smutnicki. Test functions for optimization needs, 2005.

[84] Christopher K. Monson and Kevin D. Seppi. Adaptive diversity in pso. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 59–66, New York, NY, USA, 2006. ACM.

[85] Roberto Montemanni. *Upper and Lower bounds for the fixed spectrum frequency assignment problem.* PhD thesis, School of Tecnology, University of Glamorgan, 2001.

[86] Roberto Montemanni and Derek H. Smith. Heuristic manipulation, tabu search and frequency assignment. *Comput. Oper. Res.*, 37(3):543–551, 2008.

[87] Angel E. Mu noz Zavala, Arturo Hernández Aguirre, and Enrique R. Villa Diharce. Constrained optimization via particle evolutionary swarm optimization algorithm (peso). In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 209–216, New York, NY, USA, 2005. ACM.

[88] Garry Mullet. *Wireless telecommunications systems and networks.* Thomsan Delmar Learning, 2006.

[89] Amit Nagar, Sunderesh S. Heragu, and Jorge Haddock. A combined branch-and-bound and genetic algorithm based approach for a flowshop scheduling problem. *Annals of Operations Research*, 63(3):397–414, June 1996.

[90] B. Natrajan and B.E. Rosen. Image enhancement using very fast simulated reannealing. In *Image Analysis and Interpretation, 1996., Proceedings of the IEEE Southwest Symposium on*, pages 230 –235, 8-9 1996.

[91] Lin Ni and Hong-Ying Zheng. An unsupervised intrusion detection method combined clustering with chaos simulated annealing. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 6, pages 3217 –3222, 19-22 2007.

[92] 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(2-3):599 – 623, 1998.

[93] E. Nowicki and S. Zdrzalka. Single machine scheduling with major and minor setup times - a tabu search approach. *The Journal of the Operational Research Society*, 47:1054–1064, 1996.

[94] Michael O’Neill and Anthony Brabazon. Self-organising swarm (soswarm). *Soft Comput.*, 12(11):1073–1080, 2008.

[95] W. Paszkowicz. Properties of a genetic algorithm equipped with a dynamic penalty function. *Computational Materials Science*, 45(1):77 – 83, 2009. Selected papers from the E-MRS 2007 Fall Meeting Symposium G: Genetic Algorithms in Materials Science and Engineering - GAMS-2007.

[96] Thomas La Porta Patrick Traynor, Patrick McDaniel. *Security for Telecommunications Networks*, volume 40 of *Advances in Information Security*. Springer US, 2008.

[97] Pedro Pereira, Fernando Silva, and Nuno A. Fonseca. Biored - a genetic algorithm for pattern detection in biosequences. In *2nd International Workshop on Practical Applications of Computational Biology and Bioinformatics (IWPACBB 2008)*, pages 156–165, 2009.

[98] Jean-Yves Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63:337–370, 1996.

[99] Nilesh B. Prajapati, Rupal R. Agravat, and Mosin I. Hasan. Comparative study of various cooling schedules for location area planning in cellular networks using simulated annealing. *Networks & Communications, International Conference on*, 0:146–150, 2009.

[100] Abraham P. Punnen and Y. P. Aneja. A tabu search algorithm for the resource-constrained assignment problem. *The Journal of the Operational Research Society*, 46(2):214–220, Feb 1995.

[101] A. Ratnaweera, S.K. Halgamuge, and H.C. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration co-

efficients. *Evolutionary Computation, IEEE Transactions on*, 8(3):240 – 255, june 2004.

[102] Andrea Reese. Random number generators in genetic algorithms for unconstrained and constrained optimization. *Nonlinear Analysis: Theory, Methods & Applications*, 71(12):e679 – e692, 2009.

[103] D. J. Reid. Genetic algorithms in constrained optimization. *Mathematical and Computer Modelling*, 23(5):87 – 111, 1996.

[104] J. Riget and J.S. Vesterstrm. A diversity-guided particle swarm optimizer - the arpso. Technical report, Aarhus Universitet, 2002.

[105] Angelos N. Rouskas, Michael G. Kazantzakis, and Miltiades E. Anagnostou. Minimizing of frequency assignment span in cellular networks. *IEEE Transactions on Vehicular Technology*, 48:873–882, 1999.

[106] Stuart Russel and Peter Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall, second edition, 2003.

[107] P. Demestichas E. Tzifa S. Kotrotsos, G. Kotsakis and V. Demesticha. Forumlation and computationally efficient algirithms for an interference-orientated version of the frequency assignment problem. *Wireless Personal Communications*, 18:289–317, 2001.

[108] Mohsen Saemi and Morteza Ahmadi. Integration of genetic algorithm and a coactive neuro-fuzzy inference system for permeability prediction from well logs data. *Transport in Porous Media*, 71(3):273–288, February 2008.

[109] H.G. Sandalidis, P.P. Stavroulakis, and J. Rodriguez-Tellez. An efficient evolutionary algorithm for channel resource management in cellular mobile systems. *Evolutionary Computation, IEEE Transactions on*, 2(4):125 –137, nov 1998.

[110] Mischa Schwartz. *Mobile Wireless Communications*. Cambridge University Press, 2005.

[111] Matthew Settles and Terence Soule. Breeding swarms: a ga/pso hybrid. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 161–168, New York, NY, USA, 2005. ACM.

[112] Patrick Siarry, Alain Pétrowski, and Mourad Bessaou. A multipopulation genetic algorithm aimed at multimodal optimization. *Adv. Eng. Softw.*, 33(4):207–213, 2002.

[113] Alok Singh. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing*, 9(2):625 – 631, 2009.

[114] J. R. Slagle, A. Bose, P. Busalacchi, and C. Wee. Enhanced simulated annealing for automatic reconfiguration of multiprocessors in space. In *IEA/AIE '89: Proceedings of the 2nd international conference on Industrial and engineering applications of artificial intelligence and expert systems*, pages 401–408, New York, NY, USA, 1989. ACM.

[115] K.G. Srinivasa, K.R. Venugopal, and L.M. Patnaik. A self-adaptive migration model genetic algorithm for data mining applications. *Information Sciences*, 177(20):4295 – 4313, 2007.

[116] Marc St-Hilaire, Steven Chamberland, and Samuel Pierre. A tabu search algorithm for the global planning problem of third generation mobile networks. *Comput. Electr. Eng.*, 34(6):470–487, 2008.

[117] Gordon L. Stuüber. *Principles of Mobile Communication.* Kluwer Academic Publishers, 1996.

[118] Anand Prabhu Subramanian, Himanshu Gupta, Samir R. Das, and Jing Cao. Minimum interference channel assignment multiradio wireless mesh networks. *IEEE Transactions on Mobile Computing*, 7(7):1459–1473, December 2008.

[119] B. Suman and P. Kumar. A survey of simulated annealing as a tool for single and multiobjective optimization. *The Journal of the Operational Research Society*, 57(10):1143–1160, 2006.

[120] Ashish Sureka and Peter R. Wurman. Applying metaheuristic techniques to search the space of bidding strategies in combinatorial auctions. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 2097–2103, New York, NY, USA, 2005. ACM.

[121] Ashish Sureka and Peter R. Wurman. Using tabu best-response search to find pure strategy nash equilibria in normal form games. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1023–1029, New York, NY, USA, 2005. ACM.

[122] Y. S. Teh and G. P. Rangaiah. Tabu search for global optimization of continuous functions with application to phase equilibrium calculations. *Computers & Chemical Engineering*, 27(11):1665 – 1679, 2003.

[123] Dusan Teodorovic, Panta Lucic, Goran Markovic, and Mauro Dell' Orco. Bee colony optimization: Principles and applications. In *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*, pages 151 –156, 25-27 2006.

[124] T.O. Ting, M.V.C. Rao, and C.K. Loo. A novel approach for unit commitment problem via an effective hybrid particle swarm optimization. *Power Systems, IEEE Transactions on*, 21(1):411 – 418, feb. 2006.

[125] Raj Gaurang Tiwari, Mohd. Husain, Sandeep Gupta, and Arun Pratap Srivastava. A new ant colony optimization meta-heuristic algorithm to tackle large optimization problem. In *COMPUTE '10: Proceedings of the Third Annual ACM Bangalore Conference*, pages 1–4, New York, NY, USA, 2010. ACM.

[126] Vedat Toğan and Ayşe T. Daloğlu. An improved genetic algorithm with initial population strategy and self-adaptive member grouping. *Comput. Struct.*, 86(11-12):1204–1218, 2008.

[127] Nguyen Thanh Trung and Duong Tuan Anh. Comparing three improved variants of simulated annealing for optimizing dorm room assignments. In *Computing and Communication Technologies, 2009. RIVF '09. International Conference on*, pages 1 –5, 13-17 2009.

[128] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[129] Hsien-Yu Tseng and Chang-Ching Lin. A simulated annealing approach for curve fitting in automated manufacturing systems. *Journal of Manufacturing Technology Management*, 18:202 – 216, 2007.

[130] Roger L. Wainwright. A family of genetic algorithm packages on a workstation for solving combinatorial optimization problems. *SIGICE Bull.*, 19(3):30–36, 1994.

[131] Gang Wang, Wenrui Gong, Brian DeRenzi, and Ryan Kastner. Design space exploration using time and resource duality with the ant colony optimization. In *Proceedings of the 43rd annual Design Automation Conference*, DAC '06, pages 451–454, New York, NY, USA, 2006. ACM.

[132] Z. G. Wang, Y. S. Wong, and M. Rahman. Development of a parallel optimization method based on genetic simulated annealing algorithm. *Parallel Comput.*, 31(8+9):839–857, 2005.

[133] Z. G. Wang, Y. S. Wong, and M. Rahman. Development of a parallel optimization method based on genetic simulated annealing algorithm. *Parallel Comput.*, 31:839–857, August 2005.

[134] N. A. Wassan. A reactive tabu search for the vehicle routing problem. *The Journal of the Operation Research Society*, 57(1):111–116, Jan 2006.

[135] Xian-Huan Wen, Tina Yu, and Seong Lee. Coupling sequential-self calibration and genetic algorithms to integrate production data in geostatistical reservoir modeling. In *Geostatistics Banff 2004*, volume 14 of *Quantitative Geology and Geostatistics*, pages 691–701. Springer Netherlands, 2005.

[136] Dennis Weyland. Simulated annealing, its parameter settings and the longest common subsequence problem. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 803–810, New York, NY, USA, 2008. ACM.

[137] Darrell Whitley, Soraya Rana, John Dzubera, and Keith E. Mathias. Evaluating evolutionary algorithms. *Artif. Intell.*, 85:245–276, August 1996.

[138] Andreas Windisch, Stefan Wappler, and Joachim Wegener. Applying particle swarm optimization to software testing. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1121–1128, New York, NY, USA, 2007. ACM.

[139] Wayne L. Winston and Munirpallam Venkataramanan. *Introduction to Mathematical Programming.* Thomson Learning, 2003.

[140] Lihua Wu and Yuyun Wang. An introduction to simulated annealing algorithms for the computation of economic equilibrium. *Computational Economics*, 12:151–169, 1998.

[141] Yiliang Xu, Meng Hiot Lim, and Yew-Soon Ong. Automatic configuration of metaheuristic algorithms for complex combinatorial optimization problems. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2380 –2387, 1-6 2008.

[142] Jingan Yang and Yanbin Zhuang. An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem. *Appl. Soft Comput.*, 10(2):653–660, 2010.

[143] Xin-She Yang. *Nature-Inspired Metaheuristic Algorithms.* Luniver Press, 2008.

[144] Dominic C O'Brien Yangyang Zhang. Fixed channel assignment in cellular radio networks using particle swarm optimization. In *Proceedings of the Internation Symposium of Industrial Electronics*, June 2005.

[145] Quan Yuan, Feng Qian, and Wenli Du. A hybrid genetic algorithm with the baldwin effect. *Information Sciences*, 180(5):640 – 652, 2010.

[146] L. Zhang, S. Guo, Y. Zhu, and A. Lim. A tabu search algorithm for the safe transportation of hazardous materials. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 940–946, New York, NY, USA, 2005. ACM.

[147] Xin Zhao, Myung-Eun Lee, and Soo-Hyung Kim. Improved image thresholding using ant colony optimization algorithm. In *Proceedings of the 2008 International Conference on Advanced Language Processing and Web Information Technology*, pages 210–215, Washington, DC, USA, 2008. IEEE Computer Society.