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

FEBRUARY 2014

Abstract

A new particle swarm optimisation (PSO) algorithm that produces solutions to the fixed spectrum frequency assignment problem (FS-FAP) is presented. Solutions to the FS-FAP are used to allocate frequencies in a mobile telecommunications network and must have low interference. The standard PSO algorithm's velocity method and global selection is ill suited for the frequency assignment problem (FAP). Therefore using the standard PSO algorithm as base, new techniques are developed to allow it to operate on the FAP. The new techniques include two velocity methods and three global selection schemes. This study presents the results of the algorithm operating on the Siemens set of COST 259 problems and shows that it is viable applying the PSO to the FAP.

Contents

List of Figures

List of Algorithms

List of Tables

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.

The invention of the mobile phone fulfilled the need to always be connected and be 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.

Cellular networks provide mobile phones with the means to facilitate communication. Cellular networks achieve this level of communication with the use of expensive equipment and 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 processes. In particular computers are indeed very apt at optimising procedures.

This is because a computer is able to test and evaluate a huge number of different alterations and combinations of a procedure in a short amount of time, thereby allowing it to find the best combination out of those tested.

This dissertation presents an algorithm which is based on swarm intelligence and aims to be find an optimal solution to the frequency assignment problem. More on how the research of this dissertation is done is discussed in chapter 2.

The particular algorithm presented in this research is based on the Particle Swarm Optimisation algorithm (PSO). The PSO has been previously applied to one of the variants of the FAP but in the research presented here, the PSO is applied to a variant of the FAP which closely resembles real world cellular network problems and is the first of its kind. Further detail about the specific varaint of the FAP and the PSO is provided in chapter 6.

The next section, gives an outline of the chapters that are presented in this dissertation along with a brief summary on what each chapter discusses.

1.2 Chapter Breakdown

1.2.1 Part I - Background on the problem domain and influential algorithms

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.

Note that the particular algorithms discussed in chapters four and five, were chosen as they influenced the development of the algorithm presented in this study. Chapter 6, where applicable, refers to how these algorithms have influenced the development of a technique used by the developed algorithm. 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 that are discussed in this dissertation.

Chapter 2

This chapter defines the research methodology that is utilised through out the entire dissertation. Within the chapter a discussion is presented on the research approach followed, the hypothesis is defined and a plan is presented with which the hypothesis will be tested with.

Chapter 3

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 4

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 mobile telecommunications domain that is being considered are also discussed. Finally the chapter also provides a formal definition of the problem which is later utilised by the algorithm developed in this research.

Chapter 5

This chapter marks the beginning of a discussion on various optimisation algorithms in this dissertation. 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 6

This chapter is concerned with providing algorithms that are new in the research domain of swarm intelligence optimisation algorithms. 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 of the algorithm was to be applied to the frequency assignment problem.

1.2.2 Part II - Discussion and results of implementing a PSO algorithm on the FAP

The second part of the dissertation discusses the algorithm and all the different variants that were developed. The results are presented and discussed and finally the conclusion to this dissertation is presented.

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.

1.2.3 Appendix

FAP PSO Source code

The source code for the developed FAP PSO and all its variants is presented in C#.

Chapter 2

Research Methodology

2.1 Introduction

This dissertation presents an algorithm that operates on a cellular phone network to optimise the allocation of frequencies used for communication in the network. In this chapter an outline will be given on the process that is followed by this dissertation for the research presented.

The chapter is structured as follows. The next section discusses the specific research methodology that is followed. The hypothesis which this research aims to investigate is then presented. After the hypothesis, a series of research questions are presented. The research presented in this dissertation aims to answer these questions. Following the research questions a section on how the presented research will be measured and against what it will be measured. Finally the chapter will conclude with a summary.

2.2 Methodology

The research presented in the dissertation follows a combined approach between applied research and empirical research [74].

Applied research is a good fit since it is concerned with an immediate problem that is faced by an organisation / business / industry or society [74]. As stated in the introduction with the problem statement, the particular problem this research is about is the FAP which is a problem experienced by wireless communication businesses. A more exhaustive description of the FAP is presented in chapter 4.

Part of the applied research is to better understand the domain of the problem as well as what exactly the FAP problem is [74]. This research is presented in chapter 3 and chapter 4.

When the problem is understood, a study needs to be presented on other algorithms being applied to similar problems. This needs to be done in order to gain a better understanding of the type of techniques that are required to make algorithms successfully operate on these type of problems.

The empirical research part to the research presented in this dissertation comes into affect with the main purpose of this dissertation, which is to observe and experiment the effectiveness of applying the PSO to the FAP. The effectiveness of the algorithm presented in this research is measured using the benchmark COST. The benchmark closely resembles real world scenarios. More information regarding this benchmark is presented in chapter 4.

The results of the algorithm operating on the COST benchmark is presented in chapter 8.

2.3 Hypothesis

The hypothesis for this research is to discern two key factors. The first factor, is it possible the apply the PSO algorithm on the FAP. It is no coincidence that the first factor is also the problem statement for this dissertation.

The second factor is, if the first factor is successful, is to gauge the quality of the solutions that the algorithm produces.

The hypothesis presented here forms the main theme for the research. In the next section a series of research questions is presented.

2.4 Research Questions

- **Question 1** — What is cellular technology and what components are involved when devices communicate in the network ?
- **Question 2** — What factors influence the quality of communication and what is interference ?
- **Question 3** — What exactly is the frequency problem and how does it affect modern mobile communication?

- **Question 4** — What are optimisation algorithms and what characteristics make them unique?
- **Question 5** — With regard to particle swarms, how can a particle best be represented as a frequency plan ?
- **Question 6** — With regard to particle swarms, how can one frequency plan be moved towards another frequency plan ?
- **Question 7** — With regard to particle swarms, how can particles be prevented from using forbidden frequencies when they move towards a particular plan ?

Throughout the course of this dissertation the aim is to answer each of these identified research questions.

2.5 Measurements

As discussed previously, the algorithm will be compared against the best produced results for the COST 259 benchmark suite. Specifically the Siemens suite of benchmark instances. Chapter 4 discusses in more detail why the Siemens instances were chosen.

In addition to comparing the algorithm to the best COST 259 results additional measurements is required. In particular, statistical detail on the frequency plan produced by the algorithm. The statistical detail helps in identifying where the frequency plan is generating too much interference and thus gives an outline where the algorithm needs to perform better optimisation.

The COST 259 benchmarks also provide these interference statistics for all the results published, therefore it will also provide valuable insight where this dissertations algorithm needs to improve.

All these measurements are presented in chapter 8.

2.6 Summary

This chapter outlined the research methodology that is used to conduct research in this dissertation. The hypothesis was stated and a series of research questions which, when answered, will aid in proving the hypothesis.

Finally this chapter concluded with a discussion on the kind of measurements that will be required to assess the effectiveness of the algorithm presented in this dissertation. The next chapter provides a discussion on the cellular network domain.

Part I

Background on the problem domain and influential algorithms

Chapter 3

Cellular Communication Technology

3.1 Introduction

Wireless technology is used by modern electronic devices to establish communication over which information is exchanged [2]. Wireless technology facilitates communication between two devices by transmitting data or voice via a radio frequency [2]. Examples of devices exchanging information wirelessly are radios for audio entertainment; television remotes to change frequencies; cellular phones for communication; wireless access points to create wireless LANs [2].

The popularity and rapid adoption of wireless technology contributes to the difficulty of planning, managing and operating wireless networks [2]. 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 [99]. This need arises due to an effect called *interference*, which occurs when two or more connections between devices use the same radio frequency to facilitate communication [99]. This effect is discussed in more detail in chapter 3.

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 [37]. 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 [37].

The use of frequencies by services must be carefully considered to avoid interference when devices communicate with each other. This is a problem as the amount of devices that communicate far outstrips the amount of frequencies available for communication [99]. Thus, assigning frequencies to devices for communication is a difficult problem and is referred to as the frequency assignment problem (FAP) [2, 39].

Initially manual techniques were used to assign frequencies in an attempt to solve the FAP [2]. As a result, assigning frequencies was either too complex or just too daunting because of the sheer number of devices that needed to be assigned frequencies [2]. Also, because of the rapid adoption of wireless technology the assignment of frequencies needed to be dynamic and hence automated [2].

By understanding how a general system for mobile communication (GSM) network operates, especially how communication is facilitated, the reader is able to understand the problem this dissertation addresses. Therefore, this chapter will start off with section 3.2 within which a brief history of GSM networks is presented. Section 3.3 follows the history discussion and presents an overview on the different architectural components that are part of GSM network. In section 3.4 the different interfaces used by GSM network components for communication is presented. Section 3.5 presents an overview on the logical frequencies that frequencies get divided into when used in a GSM network. Finally this chapter concludes with a discussion on the handover process in section 3.6.

3.2 GSM Networks

The GSM is a system for multiservice cellular communication that is capable of providing voice as well as data services [65, 99]. Most cellular networks in operation are GSM based [2, 99].

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 global evolution (EDGE) and high speed circuit switched data (HSCSD), are becoming more important [39,65]. 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 the

domain of radio communication better [65]. A brief history of the GSM network specification will now be presented in the next section.

3.2.1 A Brief History of GSM Networks

In early 1981 a group known as the *Groupe Speciale Mobile* was established. The aim of this group was to develop a Europe-wide radio communication system using the reserved 900 MHz band[1]

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 [92]. As such the GSM specification started following many of the same design principles and access protocols that ISDN exhibited [92].

After the completion of the ISDN specification, the advantages of switching to digital instead of analogue for communication became clear [92]. One of the primary advantages of ISDN is that it is capable of transmitting data at higher speeds [92]. GSM would therefore be based on digital transmission and speech would be represented by a digital stream of 16 Kbits/s [92].

Before the switch to digital transmission was finalised the Groupe Speciale Mobile first wanted to evaluate the spectral efficiency of analogue and digital-based transmission [92]. 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 [92].

Maximum utilisation is an important problem that is discussed in detail in later sections of this chapter. After an evaluation of spectral efficiency it was decided that the GSM system would be digitally based using time division multiple access (TDMA) [88,92]. TDMA is discussed in section 3.5.

By the early 1990s GSM became an evolving standard and the first GSM-based network was demonstrated in 1991[2] [39, 65]. 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

[1] In 1990 the United Kingdom requested that 1800 MHz band be added to the scope of the Groupe Speciale Mobile standard group. This variant of the GSM specification became known as the digital cellular system (DCS1800) [2, 92].

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

available to the general public [39, 92]. In the same year an operator in Australia became the first non-European operator to implement a GSM-based network [39].

The collective subscriber base of GSM networks surpassed the million-subscriber mark in 1993 [92]. 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 [65, 92]:

- Half rate speech codec
- 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 [2, 92].

This restriction also applies to voice, as voice is encoded into data that is transmitted over the network [2, 92]. Before the extensions were made, voice was encoded using the Full Rate codec [65]. The Full rate codec was the first codec used to encode voice in the GSM standard and required the full 9.6 kpbs that was available [65]. With half rate speech telephony, only half of the 9.6 kpbs is required for the same voice quality to be delivered [65].

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 [2, 65]. 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 [2, 65].

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 ratio (SIR) [65, 88]. This requirement affects radio interfaces, as there is a higher likelihood that interference might occur; hence it makes it more difficult to generate a low interference frequency plan [39, 88]. The actual SIR at a receiver is dependent on a number of factors that include [2, 65]:

- 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 is discussed in section 4.5. 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) is considered the third generation (3G) of cellular networks [39,126]. 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 [39, 126].

UMTS is a major improvement over the GSM in two areas, namely data transmission bandwidth and frequency planning due to UMTS utilising direct sequence collision detection multiple access (DS-CDMA) and wide band collision detection multiple access (WCDMA) [39, 126]. The higher data transmission speed (2 Mb/s) can be attributed to UMTS using the DS-CDMA scheme [39,126]. The scheme also allows more users to be served than previous generations of networks [39, 126].

DS-CDMA and WCDMA sends data over a wide band spectrum of 5 MHz. The wider spectrum allows for more frequencies to be utilised and with more frequencies being available; the FAP is easier to solve compared to a GSM network [39, 126]. An in-depth discussion on how a range of spectrum is divided into frequencies for communication in GSM networks is discussed in section 3.5. DS-CDMA and WCDMA will not be discussed in depth as this discussion focuses on GSM. The interested reader is directed to [126] and [99] for more information.

Code division multiple access (CDMA) is a technology primarily used in broadband systems [65]. 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 [65]. 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* [65].

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 [65]. The spreading factor fluctuates between values of 10 and 1 000 [65]. Using these spreading factors less interference and fewer disturbances are encountered because the broadband signal is generated from a narrowband signal [65]. UMTS may be a major improvement, but its adoption does not spread very far from busy city centres where there is a large concentration of clients in a small geographical area [65]. The reason for this is that, as mentioned previously, UMTS caters for larger data usage and therefore more clients can be serviced simultaneously [65].

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 [65]. This not only extends the lifetime of previous infrastructure investment by the operator, but also builds upon the redundancy provided by the GSM network [65]. 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 [65].

The most up-to-date cellular network technology is referred to as the fourth generation of cellular networks (4G) [140]. The purpose of 4G is to improve upon 3G and bring true broadband speeds that are delivered over the cellular network [140]. There are currently two competing technologies that aim to provide 4G capabilities to cellular networks which is long term evolution (LTE) and Mobile worldwide interoperability for microwave access (WiMAX) [140]. Both technologies provide broadband speeds in the range of 14 - 40 MBits/s [140]. According to the 4G standard both technologies cannot be considered as "true" 4G technologies as the standard defines that broadband speeds in excess of 100 Mbits/s must be provided to be considered true LTE [140].

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

3.3 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 3.1. Each subsystem will now be discussed.

Mobile Station

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 [39, 65]. 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) — 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 [39].

International mobile equipment identity (IMEI) — Used to identify mobile station equipment. Primarily used in the denial of service to equipment that has been blacklisted[3] and tries to gain access to the network [39].

The MS has the capability to change the transmission power it uses from 0.8, 2, 5, 8 MW to a maximum value of 20 MW [88]. The change in transmission power is automatically set to the lowest level by the base transceiver stations (BTS) to ensure reliable communication after evaluating the signal strength as measured by the MS [65, 88]. The power adjustment also minimises co-channel interference because the transmitted signal is less powerful and therefore less likely to interfere with other signals [88]. Co-channel interference is discussed in chapter 4

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

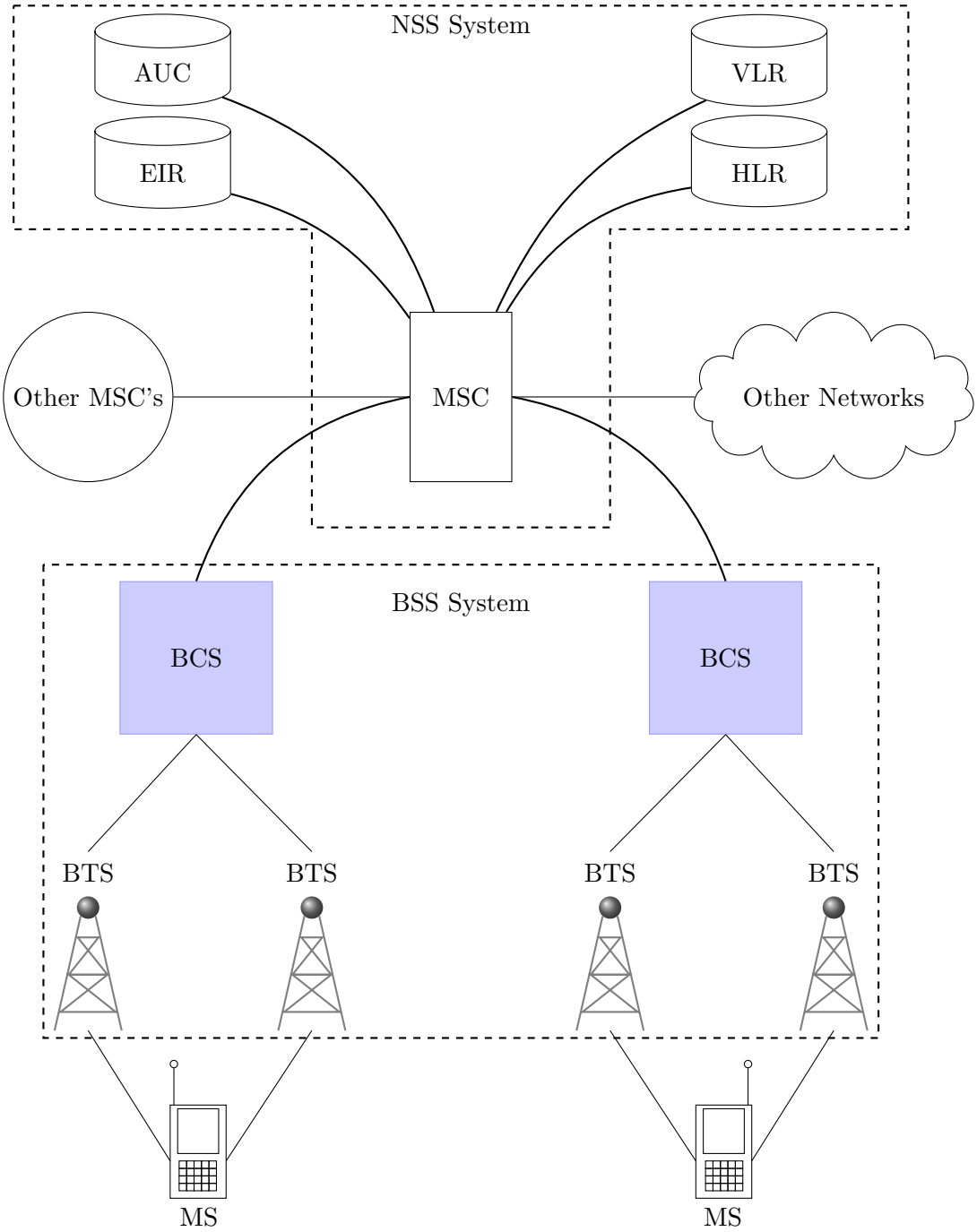

Figure 3.1: GSM architecture [65]

3.3.1 Base Station Subsystem

According to the GSM Phase 2+ specification, this system is viewed by the mobile switching centre (MSC) through an Abis radio interface. The base station subsystem (BSS) is responsible for communicating with mobile stations in a particular area [39]. The BSS usually consists of one base station controller (BSC) with one or more BTSs that it controls [39]. A BTS has similar equipment to that of a MS [88]. Both have transceivers, antennae and the necessary functions to perform radio communication.

The communication link between the MSC and BSC is the called the A-interface. The interface between the BSC and BTS is called the Abis interface [39]. The A and Abis interface as well as other interfaces between other GSM architecture components are discussed in section 3.4.

In a GSM network a service area is defined as a geographical area where the networks wants to deliver its cellular network service to potential clients [2,65]. The service area is subdivided into location areas (LAs), which are then further divided into smaller radio zones called cells [108]. When a cellular network is modelled, cells are modelled as hexagonal shapes. Each cell in the modelled network is served by only one BTS and is usually regarded to be in the centre of a cell as can be seen in figure 3.2 [65].

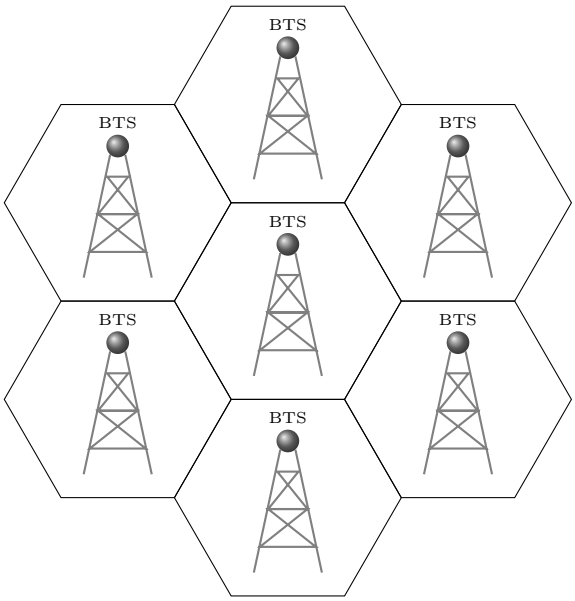

Figure 3.2: Cells with BTSs

Even though cells are modelled as being hexagons (see figure 3.2), the actual coverage area of a cell has no predefined regular shape [65]. For the purpose of this research a BTS is considered to be assigned frequencies that are used for communication. No discussion is presented on the electromagnetic wavelength properties of frequencies, as it has no effect on the results presented or problem being investigated.

With the network modelled as a series of interconnecting hexagons it allows one to more easily take constraints into account [39]. 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 [65]. 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 [65].

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 [65]. To ensure that the reused frequency subsets do not interfere with their neighbouring cells, a reuse distance D is defined [65]. 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 [65]. 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 [39,65,92]. Therefore, if the size of a cell is chosen to be small, fewer frequencies will be required to be allocated to that cell as it would not be required to handle as much traffic as a larger cell [65].

By making the size of cells smaller, the network operator is required to invest more into its network infrastructure. Smaller cells have a direct 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 [65]. 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 does not provide service in all 360 degree

directions. Instead each sector provides service in a particular direction [65, 88, 92].

A cell is divided into 3 to 6 service sectors and each sector is allocated an antenna/transceiver [88]. Figure 3.3 illustrates a cell divided into sectors.

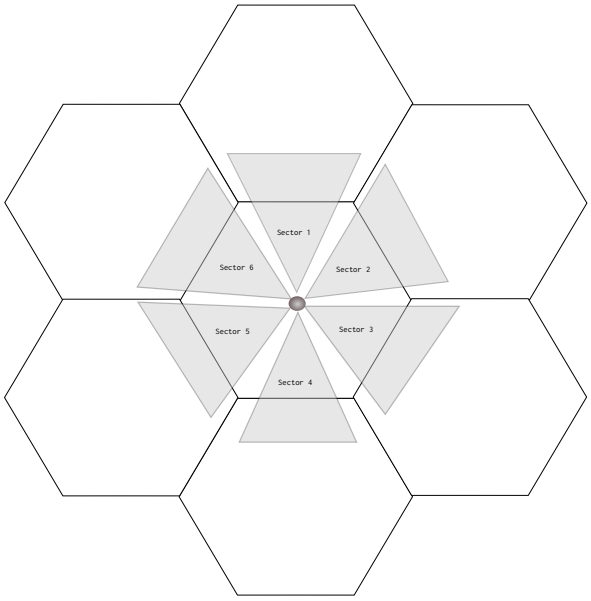

Figure 3.3: Cell sectorisation [88].

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. An omnidirectional antenna is used since it is able to provide service in 360 degrees and a single sector cell needs to provide service in all directions. Otherwise the antennae operating angles are adjusted to $\frac{360^\circ}{n}$ where n is the number of antennae [39].

By dividing the cell into sectors the amount of co-channel interference that would occur in a cell is greatly reduced [65]. 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 [65].

Suppose, if a cell using an omnidirectional transceiver is assigned 6 frequencies. If the cell were to be divided into three sectors, where the sectors' antennae are 120° apart, the number of interfering co-frequencies shrinks from 6 to 2 and from 6 to one in the case when the cell is divided into 6

sectors [65, 88, 92].

Each sector operates one or more elementary transceivers called transceiver (TRX). The number of TRX 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 [39]. 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 frequencies, which enable them to provide conversion between digital traffic data on the network side and the radio communication between MSs and the GSM network [17, 82]. The frequencies that are used by a cell for communications are discussed in section 3.5.

3.3.2 Mobile Switching Centre

The mobile switching centre (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 [88, 92]. The MSC has interfaces to communicate with GSM subscribers (through the BSS) on the one hand and with external networks on the other [92].

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 network components [92]. The most basic functions that an MSC is responsible for in a network are the following [99]:

- 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 [92, 99]. Call setup information includes the following [99]:

- IMSI
- Cypher key if Authentication is enabled
- Services subscribed to
- Whether the MS is active or not (turned on or not)
- Location
- BSC servicing MS
- Current status of a MS i.e. Is the MS ringing, in call, roaming etc.
- Temporary Mobile Subcriber Identity (TMSI) which is used when a MS originates from a different network which might be a public switching telephone network (PSTN) or competitors network.

The VLRs retrieve 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 clients that want to communicate with each other [88, 108].

A requirement for being able to communicate with other network elements such as 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 bit rate 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 bit rate to a bit rate suitable for transmission over a PSTN . In chapter 4 section 4.5 more information regarding bit rates is presented.

3.3.3 Network databases

The home location register (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' is discussed briefly and a description is given of the core functions that each database performs in the network.

Home location register — 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 [88].

Subscriber parameters that are stored in the database include billing information, routing information, identification numbers, authentication parameters and subscribed services [88]. The following information is also stored, but the information is of a temporary nature and can change at any time: The current VLRs and MSC the subscriber is registered with and whether the subscriber is roaming [88].

Authentication centre — 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 [88, 92]. The information that is stored contains authentication information as well as keys that are used in ciphering information [88, 92].

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 [88, 92]. 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 [88].

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 [88, 92]. The AUC is sometimes included with HLR to allow for fast communication between the two databases [88].

Equipment identity register — The EIR is a database that stores the international mobile equipment identity (IMEI) numbers of all registered mobile equipment that has accessed the network [88]. Only information about the mobile equipment is stored, nothing about the subscriber or call is stored in the database [88].

Typically there is only one EIR database per network and interfaces to the HLR databases contained in the network [88]. The IMEIs are grouped into three categories: *white list*, *black list* and the *grey list* [88] 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 grey list stores the IMEI numbers of equipment that has some fault (faulty software, wrong make of equipment) [88].

3.3.4 GSM Network Management Entities

In a GSM network the elements that form part of and make the network function are distributed in a wide geographical area. By distributing the network components in such a manner the network is able to provide the best network coverage for the customer [88].

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 [88]. For this purpose there are two systems in the GSM network architecture that allow for this functionality required by network engineers [88].

One system is called the operations and management centre (OMC), which is responsible for, centralised regional and local operational and maintenance activities [88]. The other system is called the network management centre (NMC) and unlike the OMC it provides global and centralised management for operations and maintenance of the network supported by the OMCs [88]. In the following paragraphs more in depth discussions on the critical functions of the OMC and NMC perform is presented.

Operational and Management Centre — The OMC is capable of communicating with GSM network components using two protocols, namely SS7 and X.25 [88]. The SS7 protocol is usually used when the OMC is communicating within the GSM network over short and medium distances [88]. The

X.25 protocol is used for large external data transfers [88]. All communication where the OMC is involved occurs over fixed line networks and/or leased lines. The OMC is usually used for day-to-day operation of a network [88].

The OMC has support for alarm handling [88]. An alarm in a GSM network goes off whenever a predefined expected condition does occur. 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 [88].

The OMC is also capable of fault management in the GSM network [88]. It is able to activate, deactivate, remove and restore a service manually or automatically on network devices [92]. Various tests can be run and diagnostic information can be retrieved on the network devices to detect any current or future defects [88].

Network Management Centre — The NMC is similar to the OMC but it is not restricted to only regional GSM network components as it is in charge of all the GSM network components in the network [88]. The NMC provides traffic management for the global network and also monitors high priority alarms such as overloaded or failed GSM network components [88]. 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.

3.4 GSM Interfaces

In the GSM network all the network components communicate with each other through predefined interfaces. In this section an overview is given of these interfaces between the components.

Um interface — This interface is the link between an MS device 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 [99, 108].

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 [99, 108]. 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 area of service [65, 99].

Other Interfaces — The MSC has interfaces going from it to 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 [65, 99]. An interface going from one MSC to another will convey information regarding handling the administration of handover of an MS device leaving one administrative area to another MSC's administrative area. The administrative area of an MSC is the section of BSS components the MSC is responsible for. The handover procedure is a very delicate process which is described in section 3.6.

In this section a brief overview was given of the most critical interfaces used by all the network components 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 is presented.

3.5 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 the spectrum is referred to as bandwidth and is measured in Hz, therefore W Hz, where W denotes the allocated bandwidth [137]. 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 [137].

Using TDMA the GSM system is able to provide additional transmission capacity by dividing the frequency into eight equal timeslots [99]. As can be observed from figure 3.4 each TDMA frame has a series of consecutive timeslots.

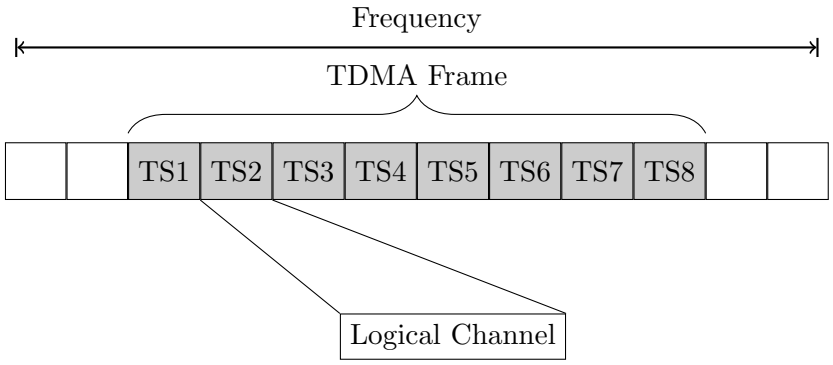

Figure 3.4: TDMA frame and logical channels [99]

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 [65, 99]. The GSM system is therefore able to use the same physical frequency in eight different timeslots without interference as these *logical* channels are used at different times. Therefore using TDMA the available frequencies that can be used for communication in GSM are increased eightfold [99].

Channels are assigned to the uplink and downlink portion of the connection with a duplex separation of 45 MHz in the frequency band 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 [65].

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 frequencies [65].

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 [65]. 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 [39, 65, 88].

The control channels are divided into three main channel groups namely broadcast channel (BCH), common control channel (CCCH) and dedicated control channel (DCCH) [65]. 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 (BCCH) — This channel is used for broadcasting and uses 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 CCCH used by the local BTS [65].

Frequency control 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 using for transmission to enable the MSs to attune themselves to the same frequency [65].

Sychronisation channel (SCH) — On this channel, identifying information regarding the BTS is transmitted. Also on this channel information regarding synchronisation of TDMA 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 BCCH since these channels are needed for the operation of the radio subsystem [65]. The CCCH is a point-to-point signalling channel that is used to localise an MS through the use of paging [65].

When a MS is paged in a GSM network, a CCCH message is sent to all the cells in the surrounding area of the last known cell the MS was recorded to be at [65]. The message is sent to surrounding cells in case the MS changed location since it was last in communication with the network [65].

The MS receives the paging through the CCCH and responds if it is intended for the particular MS [65]. The channel is also used to assign dedicated channels [65]. 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 [65].

Access grant channel (AGCH) — The AGCH forms the downlink part of the CCCH. It is used by the radio subsystem to assign a stand-alone dedicated control channel (SDCCH) or TCH to an MS [65].

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 [65].

Notification channel (NCH) — This channel is used to inform an MS of any incoming group calls or calls that are being broadcast [65].

The last group of signalling channels is referred to as dedicated/associated control channel (D/ACCH). This group of channels has the characteristic that they are a group of bidirectional point-to-point channels [65].

Stand-alone dedicated control channel (SDCCH) — This channel is used for communication between the BSS and MS even when there is no active connection [65]. Hence the 'stand-alone' since it means that there need not be a TCH assigned for communication to occur between the BSS and MS [65].

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 [65]. Packets must be continuously sent over the SACCH as it is used as proof that there is still a physical radio connection [65]. When the MS has finished using the SACCH channel, it transmits a report regarding the current results of the radio signal level, which is continuously measured [65].

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 [65]. When a channel is created in this manner it is called a FACCH [65].

Finally, one last channel is defined namely the cell broadcast control 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 [65].

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 deals 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.

3.6 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 [39,65,99]. A handover might also be initiated because of measurements indicating bad channel quality [65].

Various information needs to be migrated across and shared between the components handling the calls to ensure a smooth handover. Not only do the GSM architecture components continuously share information but also the MS shares information with the GSM in return. 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 [65, 99]. This information is of course relayed to the MSC and BTS. The information 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 [65, 99].

An MS can in some cases receive the same BCCH from different cells, which are most likely neighbours [65]. 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 [65]. As discussed in section 3.3.1, cells are 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 [65]. 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 [39,65,99][4]. Using this report the handover algorithm is able to decide how to handle the handover, and which cell needs to take over the call [39, 65, 99].

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 [39, 65, 99]. 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 [39]. 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 network components [99].

Intra-BSC — Also known as *intercell handover*, this handover is concerned with the transfer of an active call/connection from an MS to another cell, which is controlled by the same BSC as the current cell [65, 99]. 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 [65, 99]. The error bit ratio is defined as how often part of the data being sent over a connection needs to be retransmitted because an error occurred [65, 99]. An error can be caused by interference, faulty hardware or the signal strength is fading [65, 99]. The constructed report is forwarded to the managing BSC. The report is analysed to determine the necessity of handing

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

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 [65, 99]. The BSC then notifies the MS through the old BTS of the new BTS identity, and 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 [65, 99]. 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 [65, 99].

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 [65, 99]. 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 [65, 99]. 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 [65, 99]. 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 [99]. The MSC forwards the connection information to the old BSC, which forwards it to the MS [99]. 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 [99]. The new BSC informs the MSC, which then instructs

the old BSC to relinquish the old connection used by the MS [65, 99].

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 [99]. The handover process follows the same basic formula as the intra-BSC and inter-BSC handovers once the handover request is made [65, 99]. The BSC managing the BTS to which the MS is going to be handed over is to be determined by the MSC [99]. The BSC is notified by the new managing MSC that certain BTSs must bring a new connection online for the incoming MS; the network components upstream[5] are notified [65, 99]. The new MSC informs the old MSC of the new connection details. The old MSC transfers the connection information to the MS that is going to be a handed over to another BTS [99]. The MS makes the corresponding adjustments and then starts operating on the new connection. The MS informs the BSC of the successful handover [65, 99]. 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 handover process in GSM network. The effect this has on the channel selected as well as the different handover procedures was highlighted.

3.7 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 GSM architecture components followed. In the GSM architecture all the network components present in a modern GSM network were identified.

Following the discussion of the GSM network components, a broad overview was given of the communication interfaces used between the GSM network

[5]components upstream are the network components which are higher up in the managing structure. In this instance, BTS – BSC – MSC.

components 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 concluded with the handover process, which is used to allow an MS device to move freely geographically within the network.

Chapter 4

The Frequency Assignment Problem

4.1 Introduction

The frequency assignment problem (FAP) is a generalisation of the graph-colouring problem and is consequently an NP-Complete problem [37]. The FAP is an NP-Complete problem due to fact that only a finite number of frequencies can be assigned to TRXs, where the number of transceivers to be assigned frequencies greatly outweighs the number of available frequencies [37]. A more thorough definition of what it means for a problem to be NP-Complete is given in section 4.2.

In wireless communication a huge concern is interference, which occurs when, frequencies used for communication are close to each other in the frequency spectrum [2]. Interference and its effects are discussed in detail in section 4.5. 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 is polynomial. Generally metaheuristic algorithms are used to find optimal solutions to NP-Complete problems [82]. Chapter 5 presents a discussion on algorithms that are used to find solutions to NP-Complete problems.

As discussed in chapter 3, network operators are licensed a range of frequencies from the available spectrum. A licensed piece of spectrum con-

tains 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 [16]. By barring frequencies, a scenario is avoided where the different networks' equipment interferes with their respective operations [16].

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 [16]. 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, they need to be licensed [37].

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 [16, 37].

The chapter is organised as follows. The first section presents a discussion on NP-Complete problems. In section 4.3 constraint handling mechanisms are presented. Section 4.4 presents an overview of different methods for allocated frequencies. Interference is discussed in depth in section 4.5. After the interference discussion a section is presented on the different types of frequencies assignment problems in section 4.6. A mathematical formulation of the fixed spectrum frequency assignment problem is presented in section 4.7. Section 4.8 provides an overview of the different benchmarks used to evaluate frequency assignment problems. After the discussion on the different benchmarks that are available in the domain a section on where the FAP is encountered in the industrial domain is presented.

This chapter concludes with a section that summarises the contents presented. The next section presents an overview of what it means when a problem is NP-Complete.

4.2 NP-Complete

The term NP stems from the field known as complexity analysis. Algorithms are measured for their worst running time using $O(n)$ notation when solving a particular problem. An algorithm is said to be of polynomial time if the number of steps it needs to perform to solve the problem is given by

$O(n^k)$ where k is a nonnegative integer and n refers to the complexity of the input [116]. Problems that can be solved in polynomial time are referred to be in the P class of problems. Problems that form part of the NP class of problems differ to P problems in the sense that only their solutions can be verified to be correct in polynomial time [26]. Informally stated problems that are NP are decision problems where a solution contains a collection of answers. The collection of answers can be verified to be correct in polynomial time.

A distinction needs to be made between when a problem is NP-Complete and when a problem is NP-Hard. A problem can be NP-Hard but not necessarily NP-Complete, consequently NP-Hard means that the problem is at the very least as difficult as the most difficult problems in NP. Thus a NP-Hard problem does not have to be a decision problem and therefore part of NP. The "NP-hardness" of the problem only gives an indication to its difficulty [26]. A NP-Complete problem is a problem that is in NP and is also considered to be NP-Hard. Currently there are no known algorithms that exist to solve NP-Complete problems in polynomial time but there is also no proof that exists that proves that no such algorithms exist [26].

4.3 Constraint handling mechanisms

Not only is the FAP an NP-Complete problem but it is also a constraint problem. Constraint problem restrict the search space of possible solutions with boundaries. These boundaries are referred to as constraints [41]. The FAP domain has specific constraints defined for each of the different individual problems. Before the various ways in which frequencies can be allocated is discussed a brief discussion needs to be presented on constraint handling mechanisms.

Constraints define the boundaries of the search space. Solutions found to be violating the constraints are considered to be outside of the search space. Solutions that are outside of the defined search space and thus violate the defined problem constraints can be handled using various methods [41]. These methods are known to be *constraint handling mechanisms* and are discussed below. In research done by Engelbrecht [41] the following constraint handling mechanisms are defined.

- **Reject infeasible solutions** — Solutions that violate the defined constraints are not even considered and rejected.
- **Penalty function methods** — Modifies the objective function to add a penalty that is enforced if a solution violates constraints. By adding a penalty the infeasible solutions are discouraged.
- **Convert the constrained problem to an unconstrained problem** — By converting any constraints defined by inequalities to boundary constraints the problem is converted to an unconstrained problem [42]. Solving unconstrained problems can lead to better solutions being found, although not guaranteed.
- **Preserving feasibility methods** — Candidate solutions are initialised in the search space and satisfy all constraints. These solutions are continuously moved or transformed with specialised operators which ensure that the solutions continue to satisfy all the constraints. The operators ensure that all solutions stay within the bounded search space.
- **Pareto ranking methods** — Solutions are ranked based on the severity of their constraint violations. Ranking is achieved by using concepts from multi-objective optimisation. For more information the reader is directed to literature by Engelbrecht [41].
- **Repair methods** — Operators are used to transform solutions that violate constraints to solutions that adhere to all the constraints and are thus feasible solutions.

Constraint handling methods need to be used when solving constrained problems like the FAP. Without these methods, the search for solutions is undirected and solutions that cannot be used are presented as most optimal. Reference to these methods are made where applicable. The following section discusses two methods that define how frequencies are allocated.

4.4 Different methods used to allocate frequencies

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 is 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 are discussed in section 4.6. 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 allocation (FFA)* is where frequencies assigned to cells are static; therefore they cannot be changed until a new assignment plan is calculated [127].
- *Dynamic frequency allocation (DFA)* is the process of allocating frequencies to cells as required to meet the current traffic demand imposed on them by clients [127].

4.4.1 Fixed Frequency/Channel Assignment

FFA is the process of permanently assigning frequencies to cells [127]. 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 [127].

When the channel used by a particular cell for communication is suddenly changed, the cell might start interfering with neighbouring cells' communication links. This interference is caused because the assigned frequencies of the neighbouring cells are close to each other on the frequency spectrum. Hence, if the cell is sectored (refer to section 3.3.1 in chapter two for a discussion on cell sectorisation) it can interfere with a minimum of three and up to a maximum of six neighbouring cells [127].

When an FFA plan is created, cells are assigned frequencies based on the estimated traffic that a cell will be expected to handle during peak network usage. FFA 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 [127]. Macro cellular networks are networks where the size of a single cell typically spans 5 - 30 km [51].

With FFA, 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 [127]. A call is blocked on the network when a cell has no available frequencies to use when establishing a communication link [127]. In situations where the nature of the traffic is neither homogeneous nor stationary, using the FFA allocation scheme is not feasible as its use of available frequencies is grossly inefficient [127].

A problem that occurs with FFA when all assigned frequencies are in use is that any new call or call handed over will be blocked [127]. The call will be blocked even if adjacent cells have suitable capacity to handle the call [127].

4.4.2 Dynamic Frequency/Channel Assignment

DFA is a channel allocation scheme where frequencies assigned to cells are not permanent but rather assigned to cells as the need arises [127]. 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 [127]. Briefly, the co-channel reuse constraints dictates that frequencies allocated on the same transceiver must differ by a certain margin. A more thorough definition is given in section 4.5.

The co-channel reuse constraint must be adhered to otherwise the amount of interference occurring on the communication link will be too much [127].

The DFA allocation scheme is ideally suited for micro cellular wireless networks. Micro cellular networks are networks where the size a single cell typically spans less that 1km. The traffic on these networks have the characteristic of being immensely unpredictable as traffic demand varies constantly [119, 127].

As the name indicates, micro cellular wireless networks have much smaller cell sizes than macro cellular networks. The size of cells matter when one considers handover requests. A handover request occurs when a cell handling a connection of an MS needs to transfer control of the connection to another cell due to the MS moving out of the geographical area the cell is responsible for. For more information on the handover process the reader is directed to section 3.6. Due to the size a cell in a micro cellular network must handle a lot more handover requests 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 [51, 127]. Since a micro cellular network has increased handover requests between cells compared with a macro cellular network, a DFA scheme must rapidly allocate frequencies to requesting cells that must handle the handovers [51, 127].

DFA is much more efficient than FFA when the amount of mobile traffic on the network is relatively low [51, 127]. On the other hand, when the network is under heavy mobile traffic load, the FFA scheme outperforms the DFA scheme [51]. The DFA scheme is outperformed because it allocates frequencies to cells in an inefficient arrangement that might affect the amount of interference encountered on the network [119].

Finally DFA inherently requires a great deal more computational power than FFA, 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 [51, 127].

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

4.5 Interference

Interference can be defined as any unwanted signal that is received along with a signal of interest [43]. The unwanted signal is said to *interfere* with the original signal and as a consequence degrades the original signal quality with unwanted information [43].

Interference usually occurs when two or more entities communicate independently on the same channel or on adjacent frequencies [43, 51]. 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 [43, 51]. Interference that occurs when two signals operate on the same channel can be seen in figure 4.1 and interference that occurs as a consequence of two signals operating on adjacent frequencies can be seen in figure 4.2.

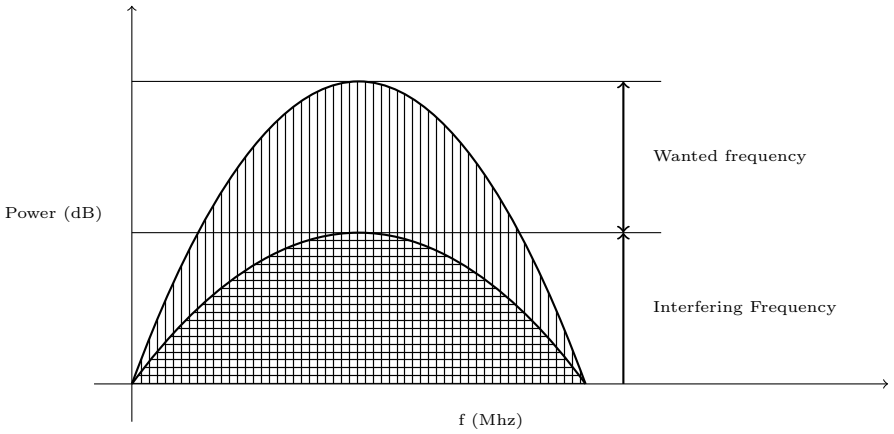

Figure 4.1: Co-channel interference

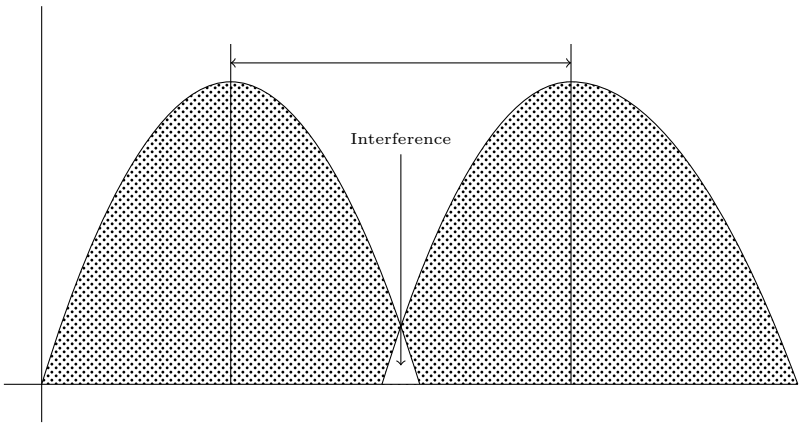

Figure 4.2: Adjacent channel interference

The magnitude of the interference experienced on a communicational link is at its maximum when the devices involved in the link are geographical close to each other. As the geographical distance between these devices increases, the magnitude of interference experienced decreases. Therefore, to minimise the impact interference will have on communication links a *separation* is defined [51]. More specifically, this separation is known as the channel reuse or frequency reuse distance within wireless networks [51].

This separation is defined as the minimum number of cells (which must all use different frequencies) that are allowed to be between two cells before either one of the cells are allowed to use the same frequency [51, 117]. The separation can be depicted visually as in figure 4.3 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 three cells (shaded in grey) because the separation for this network was set to three.

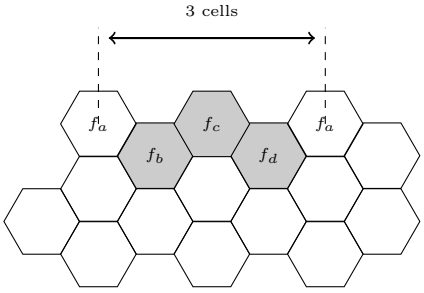

Figure 4.3: Frequency Separation

As discussed earlier, cellular networks are forced to reuse their licensed 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 [39,117]. Smaller cells can lead to a larger separation distance compared with when cells are larger [43, 51].

Cellular networks use the amount of interference on their networks as a qualitative measure for their quality of service (QoS) [51]. 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 [43, 51].

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 [88]. A situation where a call can be dropped is when a handover procedure occurs between two cells and one cell receiving the call is at full utilisation of its allocated frequencies [43, 88]. Another situation can be due to weather conditions. Weather brings forth natural interference or the caller entering a building, which drastically reduces cellular reception [43].

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 [2].

The SIR equation is actually based on the signal to interference noise power ratio (SINR) but since cellular networks are interference limited, the noise (denoted as N_0) is not considered in the final interference calculation [51]. Noise can be disregarded since the power of interference is much larger than the power of noise, which is why in the following formulation of SINR and SIR, the SIR formulation omits N_0 as part of the calculation [43]. A formulation of the SINR and SIR is as follows:

$$SINR = \frac{P_r}{N_0 + P_I} \tag{4.1}$$

$$SIR = \frac{P_r}{P_I} \tag{4.2}$$

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) [51]. 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 with the standard deviation for noise represented by the N_0 [2].

Using the SIR formula cellular networks are able to determine the bit error rate (BER) users on the network will experience on their connections [43]. The BER is defined as the probability that a received bit on the connection will be incorrect [119].

As the BER increases voice quality on the connection decreases since more bits that are used to describe the voice information are incorrectly received [43]. 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 [43]. 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* [82].

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 [2]. Each cell pair in the matrix has two corresponding values that indicate the level of interference if the *electromagnetic constraints* are violated [38, 82]. 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 [88, 127]. 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 frequencies, 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 [88,117]. 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 [2, 38, 39]. 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 [40,150]. In the benchmarks, which are discussed in section 4.8 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 three but can be any number greater than zero that the network operator deems necessary to avoid unwanted interference [2, 38, 39].

Handover — This constraint means that frequencies must differ by a predefined margin, i.e. two or three, 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 [2,38,39].

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 frequencies*. Frequencies that are in the set of globally blocked frequencies are usually frequencies that have been licensed to other networks [2, 117].

The second set of hard constraint frequencies is known as *locally blocked frequencies*. These frequencies are not allowed at certain geographic areas but are free for use at any other area [117]. A typical area where certain frequencies will be forbidden to be used is near a country border [117]. 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. The next section presents an overview of the various different sub problems in the FAP domain.

4.6 Frequency Assignment Problem types

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

4.6.1 Minimum Order Frequency Assignment Problem

The minimum-order frequency assignment problem (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 used [2].

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 of commercial cellular networks frequencies were not cheap since they were sold per unit [97].

Over the years as the law governing the wireless spectrum changed and new technology as well as standards emerged, MO-FAP lost its relevancy [2, 97]. Companies are no longer billed according to the different frequencies

they use, but they purchase licences from a regulatory body [2, 97]. This licence usually stipulates what channel band the network is allowed to use.

In some instances a piece of open frequency spectrum is put up for auction by a regulatory body, on which interested parties can bid to own the specified spectrum [2,97]. Open frequency spectrum is a series of frequencies which might be continuous or disjointed that is not owned nor used by any particular body. 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 used [2, 97].

4.6.2 Minimum Span Frequency Assignment Problem

The minimum span frequency assignment problem (MS-FAP) is a problem that is very relevant today, especially when network operators want to deploy a new network in a region [2]. The MS-FAP is concerned with keeping the interference below a certain level during assignment as well as minimising the span [115]. The interference threshold used is specified by the network designer as the minimum allowable interference on the network [115].

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 [2,115]. 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 [2, 115].

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 [2]. The Philadelphia benchmark is usually used to gauge how well the algorithm performs.

4.6.3 Minimum Interference Frequency Assignment Problem

The minimum interference frequency assignment problem (MI-FAP) or FS-FAP is encountered after the network operator has obtained a frequency band from a regulatory body [2]. 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 [55, 97]. The interference value does not play a large role in their respective objective functions [39, 55]. 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 [2, 39].

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 [2]. 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 [39, 97]. The quality and efficiency of the solution in this research is benchmarked against the COST 259 benchmark, which is discussed in section 4.8.

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

4.7 FS-FAP Mathematical Formulation

A mathematical definition of the FAP is given in this section. The mathematical definition is 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 and 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.

$$G = (V, E) \tag{4.3}$$

$$V = \{v_0, v_1, \ldots, v_i\} | i \in \mathbb{N} \tag{4.4}$$

$$E = \{\{v_0, v_1\}, \{v_0, v_2\}, \ldots, \{v_i, v_j\}\} | v \in V, \forall ij \in \mathbb{N}, i \neq j \tag{4.5}$$

$$D = \{d_{01}, d_{02}, \ldots, d_{ij}\} | \forall \{i, j\} \in E, \exists d_{ij} \in \mathbb{N}^+ \tag{4.6}$$

$$P = \{\{p_{\bar{0}0}, p_{\bar{0}1}^{=}\}, \{p_{\bar{1}0}, p_{\bar{1}1}^{=}\}, \ldots, \{p_{\bar{i}0}, p_{\bar{i}1}^{=}\}\} | \forall \{i, j\} \in E, \exists p_{ij} \in \mathbb{N}^+ \tag{4.7}$$

$$F = \{0, 1, 2, 3, \ldots, k\} | \forall k \in \mathbb{N}, \forall v \in V \exists f \in F \tag{4.8}$$

$$d_{ij} < |f(i) - f(j)|, \forall ij \in \mathbb{N}, i \neq j \tag{4.9}$$

Let G (see equation 4.3) be a weighted undirected graph, where V (see equation 4.4) is a set of vertices [97]. Each $v \in V(G)$ represents a transceiver in the FAP [97].

The variable E in equation 4.5 represents a set of edges [97]. 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 [97]. The constraints which govern the edge is represented two associated labels d_{ij} and p_{ij} that are defined on each edge [17, 98].

The label d_{ij} that is part of the set D (see equation 4.6) 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 4.9 the amount of interference that is generated between transmitters v_i and v_j can be determined [17, 98].

The other label, p_{ij}, forms part of the set P (see equation 4.7), which is referred to as the interference matrix (discussed in section 4.5) [39]. Each label p_{ij} contains two values, which represent interference:

- $\bar{p_{i0}}$ represents the value for co-channel interference [17, 98].
- $\bar{\bar{p_{i1}}}$ represents the value for adjacent channel interference [17, 98].

Finally the set F (see equation 4.8) denotes a set of consecutive frequencies for every transmitter in V [17,98]. Formally the FS-FAP can now be defined FS-FAP $= (V, E, D, P, F)$ with a required mapping of $f : V \rightarrow F$ [98]. FS-FAP is a permutation problem where an assignment of frequencies to transmitters needs to be found in order to minimise the sum of total interference (see equation 4.11).

The interference value for a single transmitter is represented by $c(p_i)$ which is based on equation 4.10. In equation 4.10 the co-channel interference value is represented by $\bar{p_{i0}}$ and the adjacent-channel interference value is represented by $\bar{\bar{p_{i1}}}$.

$$c(p_i) = \begin{cases} \bar{p_{i0}} & \text{, if } |f(i) - f(j)| = 0 \\ \bar{\bar{p_{i1}}} & \text{, if } |f(i) - f(j)| \leqslant d_{ij} \\ 0 & \text{, if } |f(i) - f(j)| > d_{ij} \end{cases} \tag{4.10}$$

$$TotalInterference = \sum_{i=0}^{|P|} c(p_i), p \in P \tag{4.11}$$

The following section provides a brief discussion on the different FAP benchmarks that exist. The section also defines benchmark against which the algorithm developed in this research is evaluated.

4.8 FAP Benchmarks

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

4.8.1 Philadelphia Benchmarks

The Philadelphia benchmarks are derived from an instance that was introduced in 1973 by Anderson [6]. 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 [2, 85].

In this benchmark interference is measured by a co-channel reuse distance [2]. 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 [85]. 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.

4.8.2 CELAR

In 1994 EUropean COopération on the Long term in the Defense (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 [2]. The project was granted to six research

groups. Within the project 36 instances were made available by the *Centre d'ELectronique de l'ARmement* (CELAR) for radio link frequency assignment [2, 35].

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 [2]. 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 [2]. This constraint is a hard constraint and may not be violated [35].

These instances were initially not available to the general public as they were contained to be within the CALMA project [1]. 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 [1]. The instances made available had been modified to take polarisations and controlled relaxations of certain EMC constraints [61].

4.8.3 COST 259

The COopération européene dans le domaine de la recherche Scientifique et Technique (COST) 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 [2,39]. The site contains 5 different sets of problems namely Tiny, Bradford, Swisscom, K and Siemens. Out of the 5 sets siemens has garnered the most attention which is due to the instances resembling real world instances. Consequently this is also why the siemens instances were chosen for the research presented in this dissertation.

The instances are difficult due to the large number of transmitters (900 - 4 000) that need to be assigned frequencies, with a 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' difficulty along with their 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 siemens 1, siemens 2, siemens 3 and

siemens 4. In the paper by Montemanni and Smith [98] the same subset of problems was used and to date their algorithm has produced some of the best results.

As discussed in section 4.3 constraint handling mechanisms need to be used for problems like the FAP. Each of the Siemens benchmarks define a set of globally or locally blocked frequencies, which are hard constraints. Due to this constraint one of the mechanisms that will be used in the algorithm to handle solutions violating this constraint is *to reject any solutions that is found to violate this hard constraint*. The characteristics of each instance will now be discussed.

Siemens 1

The siemens 1 instance resembles a GSM network that follows the GSM900 standard. This particular network has been allocated a spectrum set of frequencies $F = \{16, 17, 18, \ldots, 90\}$ 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. According to the problem instance frequencies ranging from $F = \{36, 37, 38, \ldots, 67\}$ are globally blocked. Only frequencies ranging from $F = \{16, 17, \ldots, 35\}$ and $F = \{68, 69, 70, \ldots, 90\}$ therefore 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 two and the co-cell separation is stated to be three.

In this problem a total of 931 transceivers needs to be assigned a frequency out of the 74 available frequencies. The higher the number of transceivers and the lower the number of available frequencies, the more difficult the problem is.

Siemens 2

The siemens 2 problem instance describes a GSM network based on GSM900 and has 86 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 four frequencies ranging from $F = \{42, 43, 44, 45, 46\}$ and a second block of frequencies ranging from $F = \{53, 54, 55, \ldots, 124\}$. The frequencies allocated to the network have been split into two blocks because frequencies ranging from $F = \{47, 48, 49, 50, 51, 52\}$ are globally blocked. Finally the problem specifies that the co-site separation must be set to two and the co-cell separation must be set to three.

This problem is more difficult than Siemens 1, as there are 977 transceivers that need to be assigned frequencies out of a pool of 75 frequencies. Contributing to the difficulty of Siemens 2 is due to the problem defining that there are 40 more transceivers for assignment and almost the same amount of frequencies to be used assignment as siemens 1.

Siemens 3

The siemens 3 problem describes a network based on GSM900. This network has been allocated a continuous set of frequencies $F = \{681, 682, \ldots, 735\}$. Thus the network has 55 frequencies, which can be allocated to transceivers in its network.

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

With siemens 3 the FAP becomes a lot more difficult as there are 1627 transceivers which needs to be assign frequencies out of a pool 55. The more transceivers that are defined which need assignment, the more frequencies have to be reused.

Siemens 4

The siemens 4 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 $F = \{56, 57, 58, \ldots, 94\}$. No frequencies are said to be globally or locally blocked in this network.

According to this problem instance this network has 276 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 two and the co-cell separation must be three. Out of the 4 benchmarks problems, siemens 4 is the most difficult as

it defines that the total number of transceivers that require assignment as 2781 with only 39 frequencies.

The next section presents a general overview on the different industries where the FAP is encountered. The purpose of this discussion is to get a better understanding about how far reaching the problem is and the different forms the problem is encountered in.

4.9 FAP in the Industry

FAP is a real-world problem that is encountered in industries that make use of wireless technology for wireless communication [2]. For each industry listed, a brief overview is given of how the problem differs compared with other industries.

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

Wireless mesh networks and WLANs 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 [128]. To be able to provide backbone support for these networks, a primary design goal when designing and deploying these networks is capacity [128]. 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 [128]. Multihop refers to the fact that messages are sent following different paths towards the recipient, the message is said to "hop" from one BTS to another until its recipient is reached or until a maximum number of hops. Due to messages that can get lost due to interference, multihop helps to establish some reliability [43].

Typical approaches allocating frequencies include using DFA and FFA (discussed in section 4.4). DFA is not very popular because the dynamic switching of frequencies lowers the response time on commodity hardware since there is a delay in milliseconds when switching frequencies. Typical packet transmission times are in microseconds. To guarantee uptime and high responsiveness, FFA is the preferred approach [128].

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* [128]. 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 [2].

4.9.2 Military Field Communication

In a military context the FAP is a very difficult problem to be solved due to its dynamic nature [1]. 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 [1, 35].

Due to the nature of the problem the DFA scheme is used to allocate frequencies to nodes. The military FAP has the property that any of the nodes are mobile and can move at any moment to a new location, potentially interfering with another connection [1, 35]. This property differs from the traditional FAP where the nodes that are allocated frequencies are stationary [1]. 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 [1, 35].

A lot of literature can be found on Military field communication. This is due to two organisations CELAR and EUCLID making data available to various research groups and allowing them to develop algorithms for frequency assignment [1, 35].

4.9.3 Television and Radio Broadcasting

The FAP encountered in broadcasting very closely resembles the problem domain found in cellular networks [2]. 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 [2].

Since the problem resembles the problem found in cellular networks, there are few articles that specifically discuss frequency assignment in broad-

casting as a main topic. Research that specifically examines FAP in broadcasting has been conducted by Idoumghar and Schott [64].

4.9.4 Cellular Communication

Cellular communication (see chapter 2 for a discussion) 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 a FAP still needs to be solved since these newer technologies still use GSM as their backbone architecture, as discussed in section 3.2.1. 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 [39].

Because the FAP problem is NP-Complete 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.

4.10 Summary

In this chapter a discussion was presented on the problem this dissertation is based upon. The problem was defined as being the 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 4.4.

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 4.5.

The FAP is not just one problem but consists of various sub problems 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. The various FAP sub problems were outlined and discussed in section 4.6.

A formal mathematical definition of the MI-FAP was also presented. Finally, this chapter concluded with a discussion on the different industries where the FAP is encountered and how the problem is handled.

Chapter 5

Metaheuristic Algorithms

5.1 Introduction

Metaheuristics is a subdomain of the AI domain [116]. 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 search space for the most optimal solution [116, 149]. Some problems explicitly define constraints to which the produced candidate solution must adhere to. Problems that define such constraints are referred to as constraint optimisation problems [41]. Problems where there are either no constraints or just a boundary constraint defined are referred to as unconstrained problems [41].

Constraints can be either hard or soft. A hard constraint is defined as a certain condition an algorithm or potential solution is not allowed to violate [2, 39, 116, 149]. A soft constraint is allowed to be violated but there is some sort of penalty or cost involved which is imposed on the potential solution, which lowers its desirability [2, 39, 116, 149].

An optimal solution would therefore be any solution that violates no hard constraints and violates no or a minimum number of soft constraints [2, 39, 116, 149]. Algorithms that are classified as being part of the collective body of algorithms known as metaheuristic algorithms are Tabu Search [93, 104], Simulated Annealing [129, 138] and Genetic Algorithm [110, 143].

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 produce good results [90]. The main

focus of this chapter is on 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.

AI search algorithms operate in search spaces, where they occupy various states while they are searching for a solution or goal state in a search space. Before these algorithms can be properly introduced, the concept of search spaces and states needs to be introduced.

5.2 Search Spaces and States

A search space is defined as a set of candidate goal states, which may or may not be a solution to a problem [116]. The candidate states are defined by the problem definition [116]. The possible states can be given by a successor function, which generates new states adhering to the problem-defined constraints if any [116]. However, this is not true for all problem domains. In some problem domains the generated states violate the problem constraints. The aim of the search in these problem domains is to then identify a state which satisfies all problem constraints.

The search space can also be explicitly defined with constraints to which a solution must adhere to [116]. When constraints are defined it is up to the algorithm how it moves from one state to another in the search space [116]. Each state the algorithm moves to is then checked whether it adheres to the constraints [116].

A state is a position in the search space [116]. This position represents a configuration that describes a possible candidate solution to the defined problem [116]. When an algorithm is said to be moving to a new state, the algorithm moves to a new position in the search space [116]. This new position represents a different candidate solution than the previous position that was occupied [116].

When dealing with an optimisation problem there can be more than one candidate solution [116]. Which is why with optimisation problems not only does an algorithm have to find a solution, but also the most optimal solution among the candidate solutions [116]. This subset of solutions in the search space is referred to as the *solution* space.

Both concepts of search spaces and states have now been defined. All search algorithms operate in search spaces and occupy states. The domain

of search contains a wide variety of algorithms. The following section discusses metaheuristics algorithms as this body of algorithms are used to solve optimisation problems.

5.3 Metaheuristics Algorithms

NP-Complete[1] problems have been proven not to be solvable in polynomial time by traditional uninformed search algorithms such as Breath-first Search and Depth-first Search [116]. Uninformed algorithms are not able to distinguish whether a particular goal state is more "correct" than any other goal state [116]. Uninformed search algorithms search spaces are represented by a tree structure [116].

A search is performed in the search space by starting at a root node, the algorithm expands successive nodes based on a strategy [116]. Bread-first search continuously expands all successive nodes and Depth-first search expands the deepest node first [116].

The path taken to a final goal state node represents a candidate solution [116]. In a worst-case scenario, an uninformed search will expand every single node in its search space, thus each and every single possible solution is evaluated [116].

It is not always viable to test every possible candidate solution in a given problem search space, especially in NP-Complete problems, since their search spaces are huge or infinite. This is why traditional algorithms are not able to produce optimal solutions in polynomial time [116].

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 be made to the algorithm model [83]. Metaheuristic algorithms do not dictate all aspects of the search procedure but define guidelines and aspects which directs the search [106]. Therefore a metaheuristic can be defined as follows: *the algorithm selects candidate solutions from a neighbourhood with respect to one or more current solutions of which the candidates are either rejected or accepted* [106].

As can be gathered from the name, a metaheuristic algorithm uses some sort of heuristic. A heuristic is a decision rule. Algorithms use heuristics

[1] A discussion of NP-Complete problems appears in section 4.2

to make decisions by applying the heuristic to the data the algorithm is currently working on [116, 149].

When an algorithm utilises a heuristic its forms part of a group known as informed search algorithms. As the name implies the algorithm is more "informed" using the heuristic the algorithm gains additional information about its search space. With this additional information the algorithm is able to perform a better search for a possible candidate solution [116].

Heuristics are strategies that direct the search based on the information available, to move towards areas where there is a higher probability of obtaining high quality candidate solutions [116]. 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 search space, generate new data, or select the current data as the most optimal solution [116, 149].

In general "meta" means *beyond* or *higher level* [116,149]. 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 [116, 149]. With a standard heuristic only the current state is considered [116]. A metaheuristic takes additional information into account when considering a current state [149]. This additional information can be previous states, or states that are considered to be close in the search space [116, 149].

Metaheuristic search algorithms are not guaranteed to find the most optimal solutions in the search space; instead these algorithms are 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 candidate solution will be obtained in difficult combinatorial problems [10].

Similarly, metaheuristics are not guaranteed to find suitable candidate 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 [147]. Besides the algorithm designer modifying the algorithm; metaheuristic algorithms that are inherently population-based, hybrid and/or distributed, use the concept of social and self-organisation to better exploit the solution space [89].

In this section the characteristics of metaheuristics that set these algorithms apart from the conventional algorithms used on difficult problems

was introduced. The next section of this chapter presents the tabu search algorithm.

5.4 Tabu Search

5.4.1 Introduction

Tabu search (TS) was first proposed by Glover [49] as a new searching technique to help algorithms avoid getting trapped in local optima in combinatorial and optimisation problems [112]. 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 [93], FAP [98], capacitated-lot sizing problem [54], Nurse Scheduling [34] and the Resource Constrained Assignment Problem [112].

Even though the problems mentioned differ by a large margin, the algorithm has been successful in most optimisation problems it has been applied to. If one observes the results obtained in research [98, 126], it can be inferred that tabu search has on average obtained the best results compared with previous attempts with other algorithms.

TS resembles in its most basic form the hill-climbing search algorithm [130]. The hill-climbing search algorithm starts from an initial candidate solution and then iteratively moves from the current solution to a neighbouring solution [116]. Each neighbour is rated based on its attractiveness as a possible optimal solution that the algorithm is being applied to [116].

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 [116]. The TS algorithm addresses this shortcoming by introducing the concept of memory [130]. The memory of the algorithm is actually a history of previous candidate solutions that the algorithm has moved to in its search for the search space for a candidate solution [130]. The memory used by the TS algorithm is known as the *tabu list*.

General search algorithms like hill-climbing, random-restart[2] or local

[2]Random-restart is a search algorithm where once a certain trend of repeated moves is noticed, the algorithm restarts by generating a new initial candidate solution [116].

beam search[3] tend to get trapped in local optima [116]. The local optima might be a very attractive candidate 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 candidate solution.

In actual fact the candidate solution that was found is the best solution in the *local* search space, but not in the *global* search space [42,116]. Therefore an important characteristic of algorithms being applied to optimisation problems is breaking out of local optima [42,116]. The next section discusses some of the characteristics that make the TS algorithm unique.

5.4.2 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 candidate solution.

Initial Solution Generation

The core feature of the TS algorithm is to sequentially improve an initial candidate solution [152]. An initial possible solution is a point in the search space where the TS algorithm will *start* exploring in search of a more optimal candidate solution [116, 152]. An important consideration one has to make is how initial candidate solutions are generated for the TS algorithm to start on [116, 152].

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 candidate solution [152]. Hence the generation of starting solutions must be controlled to limit the infeasibility of potential solutions [152].

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

[3]Local beam search keeps track of k states. The algorithm generates all successor states for each k state. If the goal is among the generated states. The algorithm stops. Otherwise the best successor amongst the generated successors is selected and the algorithm repeats [116].

infinity and negative infinity, the random number generator is constrained to only generate numbers between 5 and -5.

Neighbourhood Search

The following discussion on neighbourhood search is not meant to be an exhaustive survey on the different methods and how they differ under different problems. Instead the discussion is meant as a general overview to get an idea of neighbourhood generation in the TS algorithm context. The following neighbourhood discussion is based on the assumption that the underlying problem the TS is applied to, has a search space with defined boundaries that is suitable for neighbourhood generation.

TS uses a neighbourhood local search process to explore the solution space. There is no set process of how neighbourhood candidate solutions are selected as it is problem dependant. The overall quality of the solution produced by TS is also dependent on the neighbourhood search strategy used [152].

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 candidate 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 candidate solutions must be presented for the TS heuristic to allow the algorithm to decide to which solution it must move next.

The new possible candidate 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. This different means of generating neighbours is referred to as a *neighbourhood search strategy*.

The TS algorithm is not limited to just one neighbourhood search strategy. In the research by Gopalakrishnan *et al.* [54] 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 candidate solution; TS opts to stay on the current solution, as no neighbouring solution is better than the current one.

Another neighbourhood strategy that TS can use is the node exchange strategy. In research conducted by Wasan the node exchange strategy is utilised on the vehicle routing problem. Each candidate solution represents a vehicle route and each node is destination on the vehicles route. A series of connected nodes represents a route. Node exchange actually consists of two methods called *1-exchange* and *2-exchange* [142].

- 1-exchange — Moves a node on one route and places it on another route and then swapping out two nodes between two given routes [142]
- 2-exchange — Extends 1-exchange by moving two consecutive nodes from one route and places it on another route. Additionally it also swaps out four nodes between two given routes by taking two consecutive nodes from each route [142].

Depending on how nodes are connected many different unique routes can be created. A single node being connected to a different node represents a different candidate solution. Therefore, node exchange enables the TS algorithm to search much more broadly due to the constant supply of different solutions. Since candidate solutions are constantly modified, it enables the TS procedure to be a very fined-grained process, because often a small change in a potential candidate solution can have a big impact on the overall proposed solution by the TS algorithm.

In the research done by Zhang *et al.* [152] a neighbourhood selection scheme called *dynamic penalty* is developed. When the algorithm moves to 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 becomes trapped at local optima, the penalty is reduced, which allows the algorithm to consider moving onto infeasible solutions thus escaping local optima.

TS is an iterative algorithm, executing a set of operations sequentially until a stopping criterion is met [9, 93]. At each iteration the algorithm has to determine feasibility of the immediate neighbourhood candidate solutions [9, 93].

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 [9, 93]. 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 [9, 93].

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* [54, 152]. The algorithm is not allowed to move to any solution that is in the tabu list unless a candidate solution that is *tabu* is better than any current moves available in the immediate search neighbourhood [54, 142]. The process of overriding a solution's tabu status in the tabu tenure is called the *aspiration criterion* [54, 152]. 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, 105].

Research done by Sureka and 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* [130, 131]. Of the two memory schemes the explicit memory scheme is the most used in the literature [93].

With explicit memory the algorithm stores a complete candidate solution in the tabu tenure; hence the algorithm is prohibited from moving to that position in the search for as long as the solution is in the tabu tenure [130, 131]. With attribute-based memory the algorithm stores the

operation used to move from the previous solution to the current candidate solution [130, 131]. 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 [130, 131].

In research conducted by Clarkson *et al.* [66], the authors add two additional memory structures called medium term memory (MTM) and long term memory (LTM) besides the standard short term memory (STM), referred to as the tabu list [66]. Each additional structure remembers a different set of candidate solutions for use by the diversification and intensification phases in the algorithm. Both of these phases are discussed in depth in the next section.

STM is similar to the traditional tabu list: to store the most recent candidate solutions produced by the algorithm. MTM is designed to remember optimal or near-optimal candidate solutions. These solutions are therefore used later in the intensification phase. Finally, the LTM structure stores all the solutions that the algorithm has already explored and is thus used in the diversification phase of the algorithm [66].

Search Phases

As TS searches through the search space, it goes through two cycles of search phases called *diversification* and *intensification* [21, 44, 59, 69]. The diversification phase in the TS algorithm is the phase where the algorithm is directed to areas in the search space that has not yet been explored [44, 142].

Research by Fescioglu-Unver and Kokar [44] provides a strategy that consists of two components namely the *observer* and the *diversifier*. The goal of the observer is to continually monitor the best candidate 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 and the algorithm has not moved to a new candidate solution [44].

As soon as the current solution 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 [44].

The specific mechanism used to define a new position where the algorithm can continue to search, should ideally select areas in the search space that have not been explored yet [44,59]. Therefore, the diversification phase makes extensive use of the knowledge present in the long-term memory structures as an indication of what areas of the search space have been previously explored and which areas have not [44,59].

Intensification is the first phase of the TS algorithm and is where the algorithm explores significantly less to focus more its search more on a specific region of the search space. The intensification phase is also 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 [44]. 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 the algorithm can occupy the same solution within a span of iterations. The following section presents an overview of the flow the TS algorithm along with pseudo code describing the TS algorithm.

5.4.3 Flow of the algorithm

In this section the general flow of the TS algorithm is described using algorithm 1 as a reference point.

Algorithm 1 Basic Tabu Search Algorithm [98,112]

1: Initialize parameters
2: $\hat{x_0} \leftarrow$ Initialize starting solution
3: **while** stopping criteria not met **do**
4: $\quad \hat{y_i} \leftarrow$ Determine $\hat{x_i}$ neighbourhood solutions
5: $\quad$ Evaluate neighbouring solutions with fitness function $f(\hat{y_i})$
6: $\quad \hat{z_i} \leftarrow$Select best neighbour from $\hat{y_i}$
7: $\quad$ **if** Move to $\hat{z_i}$ is Tabu **then**
8: $\quad\quad$ **if** $\hat{z_i}$ meets Aspiration Criterion **then**

Algorithm 2 Basic Tabu Search Algorithm (continued) [98, 112]

```
9:              x̂_i ← ẑ_i
10:         end if
11:     else
12:         Add x̂_i to Tabu List
13:         x̂_i ← ẑ_i
14:         if x̂_i repeated ≥ max repeats then
15:             diversify()
16:         else
17:             intensify()
18:         end if
19:     end if
20: end while
21: Return x̂_i as best found solution
```

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 they are the means by which the algorithm is able to move from one possible solution to the next in the search space.

After all the solutions that are in the neighbourhood of the current possible solution have been generated, the algorithm needs to decide which of the possible neighbours is the most rewarding. 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.

If the neighbour z_i is in the tabu list, then another check is performed. The next check that is performed determines the probability that the algorithm can move to a neighbourhood solution even though it is tabu. The higher the probability the more likely the algorithm will move to a neighbourhood solution even though it is tabu. This probability is known as the aspiration criterion and is calculated on lines 8 – 10. If the aspiration criterion has been met, the algorithm makes neighbour z_i its current solution x_i.

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. The section that follows presents an overview of the TS applied to the FAP.

5.4.4 Tabu Search on the FAP

In a study conducted by Montemanni and Smith [98] the TS algorithm is used on the FS-FAP. 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. The TS algorithm uses the multistart TS algorithm, which randomly starts on different initial solutions [98].

The authors developed a technique called heuristic manipulation tech-

nique (HMT). HMT first monitors an underlying heuristic algorithm being used on the problem, in the case of the research presented the TS algorithm was used [98]. It then identifies characteristics that favourable solutions exhibit which is for instance frequencies that get assigned to transceivers which results in an overall lower interference value [98].

The HMT then uses the identified characteristics to add *additional* constraints to the problem [98]. By adding constraints, the search space is reduced. However, by reducing the search space, other near-optimal solutions, which might be far better are excluded [98]. It is for this reason that that Montemanni and Smith opted not to add the constraints permanently.

Montemanni and Smith applied their TS algorithm together with HMT to the COST 259 family of benchmarks, specifically the Siemens 1, Siemens 2, Siemens 3 and Siemens 4 problems. The results are presented in table 5.1. The values presented are scalar and indicate the total interference of the frequency plan. The lower the interference value is, the better the frequency plan.

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.

Bouju et al. [18] presents a TS algorithm that is applied to the MI-FAP. In the algorithm the fitness function counts the number of violations. The algorithm restricts the neighborhood by only selecting n cells that have large local constraint violations. n is gradually increased by the algorithm as the search progresses. Once all the instances are selected the algorithm performs arc consistency on them. Arc consistency is best explained with a formal example. Consider a cell c with neighbors given by the function $N(c)$. As frequencies get assigned to the neighbours of c the assignment will cause some frequencies on c to be blocked(violating constraints). If there are still free frequencies left to assign to c after the maximum number of blocked frequencies has been reached, then c can be removed from the selected instances.

In research by Montemanni et al. discuss a TS algorithm which utilizes dynamic tabu [96]. By using dynamic tabu the algorithm gradually reduces the length of the tabu list with each iteration. After a pre-defined number of iterations the algorithm applies cell re-optimisation. Cell re-optimisation fixes the assignment of frequencies to all cells except one. The frequencies

for this one cell can then be changed and optimised to provide the lowest interference [96]. The algorithm is applied to the COST 259 family of benchmarks and the results obtained are in the column "Dynamic Tabu" in table 5.1. Note that the authors did not apply their algorithm to Siemens 1, no reason was given why. The best results for COST 259 were obtained by the k-Thin algorithm which is discussed in section 5.5.4.

Problem instance	TS with HMT	Dynamic Tabu	Best COST 259
Siemens 1	2.769	—	2.200
Siemens 2	14.936	14.275	14.271
Siemens 3	6.650	5.186	5.129
Siemens 4	110.973	81.876	77.246

Table 5.1: Results of applying TS with HMT [98], and Dynamic Tabu [96] on COST 259

Adjakplé and Jaumard present a TS algorithm that utilizes block assignment [3]. Block assignment refers to a small set of frequencies which are assigned to the cell as a whole and not to a TRX individually. This set or block of frequencies is generated to minimise interference, maximise variance and to not use forbidden channels or violate constraints with respect to the particular cell. Frequencies are then assigned to TRXs individually from this block of frequencies, instead of the entire spectrum. The algorithm presented by the authors use block assignment for the initial solution generation. Neighbourhood moves are performed by changing the block assigned to a cell with another available block. The neighbourhood is restricted by only allowing blocks for selection where the number of local constraint violations they'll impose on a cell is minimal. The algorithm was tested on real life instances provided by Bell Mobility.

Finally, in work done by Galinier and Hao [45] a general TS algorithm is applied to various CSP problems with the FAP being one of them. The authors model the co-channel and adj-channel constraints into proprietary model readable by their algorithm. The algorithm performs moves by using the 1-exchange move strategy, where the nodes that are exchanged originated from different frequency plans [45].

When critically reasoning about the TS algorithm with regard to applying it to the FAP, the following disadvantages in theory 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 candidate solution for possible neighbours that might be the current solution for the next iteration. FAPs have huge search spaces due to their NP-Complete nature. Therefore, only searching for possible rewarding 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 more rewarding neighbours are searched.

Neighbourhood generation — The TS algorithm defines no set process for generating a neighbouring solution given a starting candidate 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 candidate 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. A possible option to allow the TS to reconsider the tabu solutions is to increase the aspiration criterion. Increasing the aspiration criterion does have its risks. 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. The following section discusses the simulated annealing algorithm.

5.5 Simulated Annealing

5.5.1 Introduction

Simulated annealing (SA) is a metaheuristic 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 [76,129]. Kirkpatrick was the first to use SA to solve optimisation problems but Metropolis *et al.* defined the basic algorithm structure in 1953 [101,138].

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 [5,91]. The crystallised form of the solid is known to be the global minimum of the solids internal energy state.

When the solid is rapidly cooled from a high temperature, the molecules have no time to reach a thermodynamic equilibrium stage [5,79]. 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 [79,91,138]. When the solid is slowly cooled in a controlled manner, the molecules are able to reach a thermal equilibrium at each temperature [91,103,138]. 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*.

$$M_{AC} = \begin{cases} 1, & \text{if } f(y) \leq f(x) \\ e^{-\frac{\Delta E}{T_k}}, & \text{otherwise} \end{cases} \tag{5.1}$$

The function f is the objective function or a function that determines the state of a given position in solution space [146]. The parameter T_k is the temperature of the algorithm at iteration k [146]. Finally, ΔE is the change in "energy" between two solutions x and y [146].

The main purpose of the SA algorithm (like most optimisation algorithms) is to minimise or maximise the cost function [129]. This cost function evaluates a candidate solution desirability compared with other solutions in the immediate *neighbourhood* of the algorithm's current position [30].

The immediate neighbourhood of solutions is generated based on a heuristic implemented by the algorithm designer [116]. This heuristic, as with the TS algorithm, can be simple or complex.

A neighbouring solution is only selected as the new best state if its desirability ranks higher than the current candidate solution.

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 candidate solution. A worse solution is only selected based on a probability, which is controlled by the *annealing temperature* of the algorithm [30].

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 [138]. 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 [129].

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 search space [111]. The following section presents a discussion on various characteristics of the SA algorithm.

5.5.2 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 search space.

After each iteration, whether the algorithm has selected a new best candidate 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 [84].

Thus the algorithm moves freely in the search space with little constraint. As the temperature decreases, the probability that the algorithm will select a bad or just any solution decreases [84]. When the temperature is very low, the SA algorithm is similar to a greedy algorithm in the sense that it only accepts downhill movements [84].

The cooling schedule provides the SA algorithm with the critical ability to control the rate the algorithm transitions from the diversification phase (high temperature) to the intensification phase (low temperature) [84]. By controlling this rate, one is able to direct the algorithm to explore more early on to locate the more promising areas for possible solutions. These promising areas can then be used by the algorithm in its intensification phase to find more promising solutions.

In the literature there are three annealing schedules in common use, namely *the logarithmic schedule*, the *geometric schedule* and the *Cauchy schedule* [101, 129]. The standard and most commonly used schedule is known as the logarithmic schedule and is based on Boltzmann annealing [101]. The main disadvantage of this schedule is that it is slow due to its logarithmic nature [101]. It also requires moves to be generated from a Gaussian distribution for it to be able to reach the global minimum [129]. The logarithmic annealing function has the following form:

$$T_k = \frac{T_0}{ln(k)}, \text{where k is the iteration value and } k \neq 0 \tag{5.2}$$

Where T_k is the temperature at iteration k. The next section describes how T_0, which is the initial temperature, is determined.

The Cauchy schedule is faster than the logarithmic schedule, which is to say with the Cauchy schedule the temperature decreases at a faster rate than with 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 [101, 129]. The Cauchy schedule is also referred to as fast annealing [101]. The schedule has the following form:

$$T_k = \frac{T_0}{k}\ , k \neq 0 \tag{5.3}$$

Finally, the fastest annealing schedule is known as the geometric or exponential annealing schedule [129]. By using the geometric schedule the SA

temperatures are rescaled which is called *re-annealing* [101]. The geometric schedule has the following form:

$$T(k) = T_0 e^{-C_k}, \text{where C is a constant} \tag{5.4}$$

Move generation which is used by the annealing schedules is discussed in more detail in the section that follows the initial temperature discussion.

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 [111]. Therefore, depending on what the initial value of the temperature is, the final result that the algorithm will produce can be influenced [124, 144].

When the initial temperature is set to a very high value, the algorithm takes a long time to reach a result since the search space is being explored more [111,144]. More exploration is favourable for SA as it lets the algorithm be less susceptible to local minimum.

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 minimum [111, 124, 144]. 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 [111, 144].

A low initial temperature together with a low cooling factor makes the time window for the algorithm to leave a local optimum very small [144]. 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 [144].

When the algorithm is near a global optimum, a low initial temperature and low cooling factor will allow the algorithm to reach the optimum in a fewer amount of iterations in the search space [144]. 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 [144].

Research by Suman and Kumar [129] present equation 5.5 which can be used to determine the best suited initial temperature after experimentation:

$$T_0 = -\frac{\delta f_0}{ln(\xi_0)} \tag{5.5}$$

where δf_0 is the average increase in the objective function. ξ_0 is the number of accepted bad moves divided by the number of attempted bad moves [129].

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 candidate solution is generated and then small changes are made to the solution to represent a new solution. The candidate solution is said to be perturbed to the next solution.

Move generation is the phase where neighbouring solutions to the current candidate solution are generated. It is the ideal section for an algorithm designer to embed domain-specific knowledge that will allow the algorithm to generate better possible solutions.

In research done by Tseung and Lin [138] uses a move generation technique known as *pattern* search is used. Pattern search has two forms of movement, namely the exploratory move and the pattern move. The exploratory move continually changes the certain variables of a candidate solution [138]. The variables define a potential solution and is specific towards the domain to which the solution is applicable. The explorative move, modifies a single varaible in the solution and then uses the objective function to check the fitness. If the variable change results in a lower fitness the change is considered to be a "downhill" move and the variable constitutes a pattern which is a downhill direction. 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 [138].

Algorithm Efficiency

The algorithm is also efficient with regard to CPU cycles when compared with the genetic algorithm. SA only has to evaluate a certain number of moves each iteration, instead of a whole population of individuals each iteration. The genetic algorithm is discussed in section 5.6.

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 may 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 may cycle. In the next section the flow of the SA algorithm is discussed and pseudo code for the SA algorithm is presented.

5.5.3 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 3 as a reference point.

Algorithm 3 Basic Simulated Annealing Algorithm [101, 103]

1: Initialize parameters
2: Set starting temperature $T(0)$
3: $\hat{x_0} \leftarrow$ Generate initial starting solution
4: **while** Stopping criterion not met **do**
5: $\quad \hat{y_i} \leftarrow$ Generate neighbouring solutions to $\hat{x_i}$
6: $\quad$ Evaluate $\hat{y_i}$ neighbours with fitness function $f(\hat{y_i})$
7: $\quad$ Calculate probability $\hat{p_i}$ of $\hat{y_i}$ neighbours with equation 5.1
8: $\quad \hat{x_i} \leftarrow$ Select $\hat{y_i}$ neighbour based on probability p_i
9: $\quad$ Reduce temperature $T(i)$ based on cooling schedule
10: **end while**
11: Return best solution $\hat{x_i}$

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 candidate 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 5.1 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 candidate solution, as observed in line 9.

Before the algorithm advances to the next iteration the temperature needs to be lowered. As discussed, the temperature is lowered according to a particular cooling schedule. In the algorithm the process of lowering the temperature occurs in line 10.

This concludes the discussion on the flow of the SA algorithm. The section that follows discusses the theoretical implications if the SA is applied to the FAP.

5.5.4 Simulated Annealing on the FAP

The SA algorithm, as with the TS algorithm, has achieved good results in other optimisation problems, as mentioned in section 5.5.1. Due to its success on other NP-Complete optimisation problems, the SA algorithm has also been applied to the FAP.

Research conducted by Duque-Anton et al. [36] utilise the SA on the FAP. The authors introduce the concept of a *dummy* frequency. The frequency is used to indicate (partial) unsatisfied demand. Where demand is defined as the amount of transceivers installed at a cell to handle expected traffic. By swapping out a real frequency with the dummy frequency or vice versa the algorithm is able to increase of decrease violation of the traffic demand. The particular cell and frequency that is used for this swapping that occurs is chosen at random. The authors opted to use a custom cooling schedule. In their schedule the cooling rate is calculated based on the average cost taken over two L-loops at temperature $t1$ and $t2$ is less than the standard deviation of solutions at temperature $t1$. Where L-loop refers to the SA algorithm inner loop where the temperature does not change.

A unconventional SA algorithm is presented by Zerovnik [151]. The defining characteristic that sets this algorithm apart from conventional SA algorithms is that the initial temperature T is never changed. The algorithm selects a cell based on the number of constraints it has violated. Using this cell frequencies are assigned to it based on a probability. This probability is formulated as follows. For every new frequency f the probability that is assigned to the cell is given by $e^{S_f/T}$, where S_f represents the number of violated constraints if f is assigned to the particular TRX at the cell.

Hellebrandt and Heller [60] present an SA algorithm which is applied to the COST 259 set of benchmark instances. In their algorithm, the initial solution is generated in such a manner to ensure it fulfills all hard constraints. The authors opted to set the initial temperature acceptance rate to between 0.8 and 0.9. Movement occurs through the 1-exchange movement strategy. The algorithm restricts the neighbourhood by only selecting potential moves which will not violate any constraints. The defining attribute of the algorithm presented by the authors is due to their *one-cell optimisation* method that is applied at the end of each iteration. With this method a cell is randomly selected and by using a simple dynamic program the authors show the cell can be efficiently optimised. The optimisation occurs by allowing that for the particular cell all frequencies can be changed simultaneously [60].

In literature by Beckmann and Killat [12] the SA algorithm is applied to the FAP. The resulting SA algorithm was benchmarked on the COST 259 Siemens benchmark instances. The results obtained by the authors are presented in table 5.2 in column "(BK) SA". Values represent the total interference generated by the frequency assignment.

Research presented by Mannino et al. [86] discusses an SA algorithm that utilizes *interval graphs*. A graph $G(V, E)$ is an interval graph if the graph contains a ordering of vertices $v_1, \ldots, v_n$ such that it can be shown that for any triple (r, s, t) that $r < s < t$, if $v_1v_2 \in E$, then $v_1v_s \in E$. If such an ordering exists within the graph, then a maximum weighted stable set can be found by a dynamic programming algorithm in $O(|V|)$-time. The authors generalised the graph interval to define when a graph is k-thin. Namely a graph $G(V, E)$ is k-thin when one can define a ordering of vertices $v_1, \ldots, v_n$ for V and partition the ordering into k classes $V^1, \ldots, V^k$, such that for any triple (r, s, t) that $r < s < t$, if $v_r, v_s \in E$ belong to the same k class and $v_t, v_r \in E$ then $v_t, v_s \in E$. If a graph is found to satisfy these conditions defined by the authors then the graph is an interval graph if and only if it is 1-thin. When the ordering and partition is given from the beginning, then the authors are able to find a maximum stable set S^* on a k-thin graph in $O(\frac{|V|}{k})^k$. Using this k-thin graph approach together with SA algorithm allowed the authors to out perform all the other results presented in the COST 259 benchmark.

In table 5.2 results are presented from research presented by two authors. As mentioned previously, the results in column "(BK) SA" are from the

research presented by Beckmann and Killat [12]. The results depicted in column "Best COST 259" are from the research presented by Mannino et al. [86] obtained by their k-Thin SA algorithm.

Problem instance	(BK)SA	k-Thin SA - Best COST 259
siemens 1	2.78	2.20
siemens 2	15.46	14.27
siemens 3	6.75	5.12
siemens 4	89.15	77.24

Table 5.2: (BK) SA [12] and k-Thin SA [86] on COST 259 Benchmark

An overview of the SA algorithm along with its unique characteristics have now been given. Utilising the knowledge that was gained from understanding how the SA algorithm operates a critical evaluation can be presented. In the following paragraphs, a theoretical critical evaluation is presented that lists the various characteristics, which would be problematic if the SA algorithm would be applied to the FAP.

Cooling Schedule — Depending on the cooling schedule selected, the algorithm might converge too quickly. As discussed previously the cooling schedule reduces the temperature. 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 (diversification) and later on will exploit more (intensification). 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 becomes 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 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. A desirable neighbourhood generation scheme would be one that keeps track of where the algorithm has been previously. By keeping history, the algorithm will be able to avoid previously explored areas in the search space.

Single solution based search — The SA algorithm is similar to the TS algorithm in the sense that it only searches from one candidate solution per iteration. It searches by generating neighbours around the current candidate 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.

5.6 Genetic Algorithm

5.6.1 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" [47,57,75,139]. The genetic algorithm (GA) was first proposed by Fraser, but it was not till the research presented by Holland that GA's became popular [42]. Holland initially applied the algorithm to adaptive systems but the algorithm has been widely used in optimisation due to its success on multidimensional problems [47, 139, 141]. The GA developed by Holland is referred to in the literature as the Canonical GA (CGA) [42].

The wide use of the GA can also be attributed to its generic algorithm structure as well as the ease of implementation [75,139]. The GA consists of three main operations. An initial population is created upon which the GA operates. Using a selection method (discussed in section 5.6.2) parents are chosen. The GA uses genetic operators like crossover and mutation (both

discussed in sections 5.6.2 and 5.6.2) on the selected parents to create offspring [50]. It should be noted that initially the mutation operator was not required to be part of the GA. Only after successive implementations of the GA did it showcase the explorative power that the mutation operator brings to the search capability of the GA. Hence, the importance of the mutation operator was recognised [42].

The GA search procedure involves searching the solution space through artificial evolution and natural selection [68, 121, 139]. An individual or point in the search space is known as a *chromosome* [22]. An initial set of chromosomes (referred to in the research as the *population*) is randomly generated to form the initial population [57, 68, 121, 139].

A chromosome consists out of a sequence of smaller parts called *genes* [42]. The sequence upon which these genes appear in the chromosome determines the characteristics of an individual [42]. In the GA a single gene represents a single variable of a candidate solution. Depending on the problem domain a variable can represent a different characteristic, for instance in the Traveling Salesman Problem variable that forms part of a candidate solution could be a node on a route [57,139]. A whole chromosome therefore represents a candidate solution [57, 139]. Exactly how best to represent a chromosome as a solution is problem dependent [42].

According to the evolution theory proposed by Darwin individuals of a population with the best genes have the best chance to survive and to reproduce [42]. These individuals are referred to as the fittest individuals of the population. In a GA population each individual is "rated" to determine how good the solution its genes represents [42]. The rating of an individual is referred to as its fitness value and is also problem dependent [42].

Determining the fitness value of a chromosome is achieved by means of a *fitness function*. The fitness function is a mathematical function, which maps the chromosome representation to a scalar value [42]. An optimisation problem has an objective function, which calculates how good a candidate solution is, therefore in the GA the fitness function represents the objective function [42].

When the fitness of individuals in a population has been determined the GA probabilistically selects individuals for the next generation [42]. The selection probability is referred to as the *selective pressure* [42]. Individuals are selected using a selection method (discussed in section 5.6.2) for the

reproduction phase of the GA where offspring are created [42].

Offspring are created by combining parts of one or more parent chromosomes to form a new chromosome. This procedure is called the *crossover* [42]. The chromosomes used in the production of the offspring are referred to as the parent chromosomes [42].

With regard to how offspring and parents are handled, there are two forms of the GA [139]. One form is called the *generational* GA and the other form is known as the *steady-state* GA [139, 143].

With the generational GA offspring is created from the current generations parents. The newly generated offspring replaces the old population and becomes the new population for the next generation [94, 123, 139]. In the steady-state GA only a certain portion of the population is replaced each generation. Depending on the selection pressure the least fit individuals are chosen for replacement [94, 123, 139, 143]. Steady-state GA is more favourable for systems where incremental learning is key and knowledge needs to be retained more aggressively [94, 123]. Due to individuals not being replaced at every generational step the knowledge of each individual "lives" longer [94, 123].

The GA search process moves around in the search space using genetic recombination operators i.e., crossover and mutation which are applied probabilistically instead of deterministically [139]. The operators aid the algorithm to avoid local optima regions in the search space [68]. Note that by using the operators there is no guarantee that the GA will completely avoid local optima [42].

The GA makes no assumptions about the search space and primarily works on the information provided by the chromosomes and the representation of solutions used [42, 68, 114]. Unlike the SA and TS algorithms, the GA if properly initialized, starts with a number of search points that cover a wide area of the search space. Depending on the operators used, diversity can be quickly lost and the population can become homogeneous very quickly [47, 68, 139].

The different operators used by the GA along with their probabilities are not specific. Each operator and probability can be changed to suit the problem domain the GA is being applied to. Therefore the particular probability and operators that the GA uses is problem dependent.

As discussed, the algorithm consists of two key parts. The first part, which is the selection method is used to select individuals upon, which the operators function. The second part is where the operators get applied. In the next section, the selection method, crossover and mutation operators are each individually discussed.

5.6.2 Important Genetic Algorithm Characteristics

The various operators used by the GA makes the procedure unique and is one of GA's defining characteristics. One of the GA's defining characteristic is discussed below, namely the selection operator.

Selection Method

The selection method is applied at first chance to the population after each generation. The method determines which individuals will be used to for the next generation and if need be create offspring [100, 114, 118].

Individuals that are selected by the selection method are used by the crossover and mutation operators (in the reproduction phase), to generate offspring which will form the next population [57, 75].

By favouring high fitness individuals above low fitness individuals the selection method is said to have a high selective pressure [42]. Care must be taken if the method uses a high selective pressure. Since with high selective pressure low fitness individuals are preferred, the algorithm effectively explores less and diversity among the individuals in the population deteriorates and thus result in premature convergence [42, 114].

There are a variety of selection methods available for use by the GA. Each with their respective strengths in certain problem domains. The following list is only a brief summary of the methods available.

- Fitness-Proportionate Selection — For each individual a calculation is made to determine the amount of times the individual will be used to reproduce. This "value" of an individual is determined by taking the individuals fitness and dividing it by the average fitness of the population. Roulette wheel is a commonly used in this selection method. Each individual is assigned a section of the wheel. The size of the section is proportionate to the calculated "value". At each generation,

the wheel is spun N times where N is the size of the population. The individual under the wheels marker after each spin is selected to be in the pool parents for the next generation [94].

- Sigma Scaling — The selection pressure in sigma scaling is aimed to be more constant over a full algorithm run and is based on a individuals expected value. High expected values means that the individual will be used more in the production of offspring. The expected value of a individual is a function of its fitness, the population mean, and the populations standard deviation [94].

- Elitism — Selects a few of the best individuals to be retained each generation, which would ordinarily be lost through crossovers and/or mutation.

- Boltzmann Selection — Similar to simulated annealing Boltzmann selection is based on a temperature that gets gradually changed. A High temperature means there is a low selection temperature and therefore all individuals a reasonable probability of reproducing. As the temperature gets gradually lowered according to schedule the selection pressure increases. As the selection pressure increases, the less likely it becomes that less fit individuals will be selected to produce offspring [94].

- Rank Selection — Individuals are ranked based on their fitness. Individuals are then selected based on their rank in the population rather than their fitness. Ranked based selection reduces dominance of high fit individuals [94].

- Tournament selection — Two individuals are randomly chosen from the population. The fittest individual between the two is selected with a defined probability to be a parent, if not the less individual is chosen [94].

- Steady-State selection — As discussed in the previous section, only a certain portion of the population is replaced each iteration.

With respect to numerical optimisation problems, some problems might be classified as *unimodal* or *multi-modal* [42,48]. The mathematical analysis on the search spaces of these numerical optimisation problems, the amount

of optimums a particular problem contains can be determined [42,48]. When a numerical problem is known to have only one global optimum, the problem is referred to as being unimodal [42,48]. Problems with more than one global optimum are known as multi-modal problems [42, 48].

With regard to unimodal problems and GA, a high selection pressure is beneficial [114]. With high selection pressure the selection method directs the search into a gradient-based direction that converges on a single optimum solution [114]. The high selection pressure forces the algorithm to explore less and exploit more, therefore converging in less generations to a single optimum. When the GA is applied to multi-modal problems a low selection pressure is more beneficial to the GA [114]. Low selection pressure allows the population of the GA to explore the search space vigorously [114].

Depending on the complexity of the objective function and the size of the population, the selection phase may be the most computationally expensive as well as time consuming phase [57].

These replacement policies define which individuals of the current population should be replaced by the newly generated offspring [42]. For instance, the policy can define that the entire current population should be replaced by the offspring [42]. Using this policy is not ideal as some generated offspring could have a much worse fitness than their parents but due to the replacement the solutions represented by the good parents are lost [42].

Finally, the selection scheme used should also incorporate a test to not allow duplicate individuals to be used for crossover since applying a crossover to two parents will effectively result in a copy of the parent [42].

Other replacement policies include replacing the worst individual in the current population with offspring on the premise that the offspring has a better fitness [42]. By replacing older chromosomes with better performing chromosomes in the population replacement strategy, the GA achieves a hill-climbing ability. The reader that is interested in more information on different selection operators is directed to the survey by Engelbrecht [42].

Crossover Operator

The crossover operator is the first operator applied to the population in the reproduction phase. The crossover operates exclusively on the chromosomes in the mating pool. Crossover works by interchanging genes from two or

more parent chromosomes. The parent chromosomes are selected from the current population using one of the selection methods discussed previously to form the mating pool from which offspring is produced [22, 118, 139].

Offspring is formed by applying the crossover operator with a defined probability to two or more parents. The probability is known as the *crossover probability* and is problem dependent [42]. By defining a high crossover probability that decreases over time, the good genes that form the current population are retained in the next population [42]. With good genes being retained more historical information is passed onto the next generation's population [139].

There are a variety of ways in which genes are interchanged between chromosomes in the crossover operation i.e. fixed point crossover, two point crossover, uniform crossover and Gaussian crossover [42]. All of these crossovers operate on the premise that a byte representation is used for the chromosomes.

With the byte representation, each chromosome is a byte. Each byte is made up of sequence of bits. As discussed previously each chromosome is made out of genes and therefore each bit is a gene [42].

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 [22]. Two-point crossover generates two random indices, which dictate a certain segment in the one parent to be interchanged with the other parent [114]. Where a segment consists of a subset of genes from the chromosome.

The 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 [141, 143]. Finally, the Gaussian crossover also uses the byte representation. The Gaussian crossover interchanges bits between parents based on a Gaussian distribution [141, 143].

Caution should be exercised by the crossover operator when selecting chromosomes for reproduction. It is possible for the operator to select the same chromosome twice to be the parent of a single offspring [42]. The operator is also at risk of selecting the same chromosome multiple times for reproduction [42]. The operator must therefore incorporate a test to detect unnecessary repeated usage of a chromosome [42]. Finally, it is important

to note that there exist strategies for both Gaussian crossover and mutation based operators for use with continuous-valued representations [41].

Mutation Operator

The mutation operator is a probabilistic operator and is applied to individuals in the offspring population with a probability referred to as the mutation rate [42]. The purpose of the mutation operator is to increase the diversity of the genes of an individual's chromosome [42]. By introducing new genes into an individual the diversity of the populations characteristics are increased [57, 114, 118].

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 [57]. 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 [75, 114, 139]. It is for this reason that it is recommended that the mutation rate be set to a low value to ensure good solutions are not distorted too much [42].

In a survey done by Engelbrecht [42] 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 uses any of the previously discussed crossover operators (see page 88).

The mutation operator is not always a basic random replacement of genes operation. In research done by Jeong and Lee [75], a mutation operator is presented that uses the SA algorithm to determine the genes that need to be replaced. The SA mutation operator generates a new chromosome from which genes are used to replace in the chromosome being mutated [75]. Addition mutation operators are discussed in Engelbrecht [42].

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. The initial population is generated by means of randomisation [136]. Since the algorithm searches multiple points simultaneously in the search space, it is desirable that the ini-

tial population have a wide diversity with regard to the problem search space [47, 100]. By controlling the initial population generation the amount of exploration the algorithm does initially can be controlled to a small degree [100]. Therefore care must be taken in the selection of the particular randomisation scheme that will be used to generate chromosomes for individuals.

In research by E. Cantu-Paz, the Mersenne Twister (MT) random number generator (considered to be the best pseudo random number generator) is compared against a severely limited MT random number generator [20]. The second MT generator was limited by the authors through reseeding the generator every 1000 calls with the original seed, which leads to the numbers being generated being a lot less random and repeating after every thousand calls [20, 29]. By comparing these two random number generators with respect to the performance of the GA, the authors came to the following conclusion that the initial population generation is the most sensitive to the quality of the random number generator used [20]. Other parts of the GA like crossover and mutation were found to not be impacted by the quality of the random number generator used [20, 29].

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 [4, 75, 109, 125]. 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 search spaces in order to get good chromosome diversity among individuals. Hence, the population size must be fine-tuned to achieve optimal performance in large search spaces while keeping diversity among the individuals of the population [42, 75]. Note, an increase in population does not guarantee good chromosome diversity among the population. As discussed in initial population generation it is also dependant on the random generation scheme used as well. Using a good random number generator like MT can help reduce the probability that in a larger population the there exists a higher percentage of duplicate individuals structurally, which is not ideal since the initial population should

have high diversity. High diversity correlates to a good initial exploration footprint in the search space.

The following section presents the pseudo code for the GA algorithm and also the flow of the GA algorithm is discussed

5.6.3 Flow of the Algorithm

The core concepts of the genetic algorithm were introduced in the previous section. To better understand the algorithm, a general overview of the algorithm is presented in this section using algorithm 4 as a reference point.

Algorithm 4 Basic Genetic Algorithm [42, 50]

pop_n ←Initialize population
while Stopping criteria is not met **do**
    Evaluate individuals of population with fitness function $f(\hat{x_i})$
    $\hat{y_k}$ ← Select parent individuals from population using selection operator
    repeat
        for Each chromosome $\hat{g_i}$ in $\hat{y_{k-1}}$ **do**
            $\hat{c_i}$ ← calculate crossover probability for $\hat{g_i}$
            if $\hat{c_i} \geq$ Crossover threshold **then**
                $\hat{g_i}$ ← Apply crossover operator to $\hat{g_i}$
            end if
            $\hat{o_i} \leftarrow \hat{g_i}$
            $\hat{m_i}$ ← Calculate Mutation probability for $\hat{o_i}$
            if $\hat{m_i} \geq$ Mutation threshold **then**
                Apply mutation operator to $\hat{o_i}$
            end if
            Add offspring chromosome $\hat{o_i}$ to new_{pop}
        end for
    until $size(new_{pop}) = size(pop_n)$
    pop_n ← select new population from pop_n and new_{pop}
end while
$\hat{x_i}$ ← Determine best chromosome in pop_n
Return best solution $\hat{x_i}$

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 of algorithm 4 on page 92.

The amount of individuals in a population to be generated is predefined and is known as the population size. In algorithm 4 on page 92 the population size is represented by the subscript n in pop_n, where $n > 0$.

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 $f(\hat{x}_i)$. After each individual within the initial population has been evaluated, the algorithm is able to start its searching process, which starts at line 3 and ends at line 17 of algorithm 4 on page 92.

Since each individual has a fitness value after being evaluated, the selection operator is applied. The selection operator used on line 4 determines which individuals will form part of the parents used to generate offspring for the next population. Once the selection operator has selected the parent individuals needed for the next population, the algorithm is ready to enter the reproduction phase, which ranges from lines 5 – 17 of algorithm 4 on page 92.

In the reproduction phase the crossover and mutation operators are applied probabilistically. For the crossover operator a crossover probability is calculated on line 7. The probability defines how much of the new population should consist of offspring created through the crossover. Thus the crossover value is a percentage value at its most basic but could also be made more intelligent and only consider high value parents.

Depending if the calculated probability satisfies the crossover probability the crossover operator is applied on line 9. Depending on the crossover used, offspring are generated using one or more individuals as parents. The resulting offspring from the crossover operation is assigned to $\hat{g}_i$.

On line 11 the value of $\hat{g}_i$ is assigned to the variable $\hat{o}_i$, which represents the offspring. From lines 7 – 11 it can be observed that a particular individual does not have to take part in a crossover operation to be carried over to the new population. Thus knowledge gained from the current population is persisted based on the crossover probability.

Note that if there is only one parent, crossover won't have an effect on the produced offspring [42].

The second step of the reproduction phase is where the mutation operator is applied. For each of the offspring 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 an individual is selected and the mutation operator is applied. Application of the mutation operator can be seen to occur in lines 13 – 15 of algorithm 4 on page 92.

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

After the algorithm has completed the reproduction phase the algorithm selects the new population to be used in the next generation. The new population is selected from the current population and the created offspring.

The algorithm continually generates a new population for each generation until a predetermined stopping criterion has been met. Once the criterion has been met, the algorithm selects the individual with the best fitness in the current population as its most optimal solution. The following section discusses literature where the GA has been applied to the FAP.

5.6.4 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 problems. These problems include: solving nonconvex nonlinear programming problems [8], data mining [125] and auto configuring metaheuristic algorithms for complex combinatorial problems [147].

In research by Cuppini [28] a GA is presented where the fitness function is the sum of two terms. The first term is the global interference and the second the amount of different frequencies used in the candidate solution. Within each chromosome C, a gene represents a subset of transceivers which are assigned a particular frequency f, formally stated $C = f_1, f_2, ..., f_{max}$ and $f_n = T_1, T_2, ...T_k$ where n denotes the frequency and T_k denotes the particular transceiver assigned. To generate a new generation the algorithm uses asexual crossover which operators on a single chromosome. The algorithm performs asexual crossover by choosing two genes. For each chosen

gene two crossover points are selected. The selected points are the same for both genes. By breaking the genes at the crossover points new genes are formed. The genes are formed by taking the first part of the one gene and completing it with the second part of the second gene and vice versa. The chromosome used for crossover is selected with a probability that is proportional to its fitness value.

Ngo and Li [102] is the same chromosome representation as Cuppini [28]. In their algorithm, two point crossover is used, where two genes would be selected. Due to each gene containing a group of transceivers the crossover only swaps out a subset from each gene between the two genes. The algorithm also applies mutation to the generated offspring by randomly selecting a single TRX and assigning it a different frequency. The algorithm aims to diversify its search by randomly applying a local search procedure the current best candidate solution. The procedure inspects the candidate solution and searches for the TRXs whose assigned frequencies caused the most interference. Once these TRXs have been located they are randomly assigned new frequencies.

A different chromosome representation scheme is used by Crisan and Mühlenbein [27]. The authors represent each chromosome as a n-dimensional vector v where each element in the vector represents a TRX and n the amount of TRXs that need to be assigned frequencies. A frequency $f_i \in [f_{min}, f_{max}]$ is assigned to a element (TRX) $v_j, j \leq n$ where $[f_{min}, f_{max}]$ represents the frequency spectrum. In their application of the GA on the FAP the mutation operator randomly randomly chooses a vector element v_j and assigns a new frequency f_i which originates from a candidate set and adheres to co-cell constraints. Crossover in their algorithm consists of first selecting a cell c where $c = v_1, v_2, \ldots, v_n$ thus a cell is a subset of v. The selected cell has no local constraint violations. The algorithm then determines all the cells that are involved with interference with this cell. All the involved cells are then moved as is into the new offspring chromosome. The rest of the cells are then taken from the other parent and moved to the offspring chromosome to complete the crossover.

Jaimes-Romero et al. [67] uses a two phase genetic algorithm approach to find candidate solutions for the FAP. During the first phase the main concern of the algorithm is to find a candidate solution where the probability that calls will be blocked is 0. Once the algorithm has found such a candidate

solution it enters the second phase. In the second phase the algorithm tries to minimise the interference.

Colombo and Allen [25] have developed a GA to be applied to FAP. The authors decomposed the FAP into smaller sub problems. On average the solution quality is improved by using the technique but at the expense of more complex and taxing evaluations that have to be performed [25].

In table 5.3 the results obtained by authors on the COST 259 benchmarks are compared. The listed values are scalars representing the total interference generated by the frequency plan. The results presented in this table are rounded to two decimals places due to the authors reporting their results with two decimals as well. The Best COST 259 values were obtained by the k-Thin SA algorithm discussed in section 5.5.4.

Problem instance	GA	Best COST 259
siemens 1	2.96	2.20
siemens 2	17.83	14.27
siemens 3	6.08	5.13
siemens 4	96.84	77.25

Table 5.3: GA [25] on COST 259 Benchmark

As per the results in table 5.3 the GA produces good results coming close to the best-obtained results in the benchmark. By critically reasoning about the GA if it were applied to the FAP in theory, the following disadvantages can be identified:

Diversity — The GA continually operates on a set population that is randomly initialised at the beginning of the algorithm. Each individual in the population is a chromosome which is composed of genes. It therefore only has this set of generated genes per individual in the initialised population to 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. As discussed in initial population generation, the diversity of the population is dependant on the random number generator used. A good random number generator for popu-

lation generation, maximises diversity by ensuring that structural similarity between individuals is kept to minimum.

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 a means by which more diversity is introduced into the chromosomes of the individuals. As discussed, the mutation operator introduces new genes to existing chromosome that can lead to an excellent candidate solution being distorted and becoming 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. Each chromosome would therefore be required to keep history of changes made to it. History of a chromosome can include by keeping track of the values genes are changing from and to what values genes are changed to. Additionally, the fitness of the old genes should be kept as well as the fitness of the new changes. Another means of applying the mutation operator is to set the mutation rate

proportional to an individual's fitness.

5.7 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 as well as the characteristics these algorithms exhibit was identified.

Three metaheuristic algorithms were discussed in this chapter. For every algorithm discussed 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. The next chapter presents a discussion on a class of algorithm that is new to optimisation research, namely swarm intelligence.

Chapter 6

Metaheuristic Algorithms: Swarm Intelligence

6.1 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 evolution, which can be seen being applied in practice in the GA (see section 5.6).

There are other approaches in artificial intelligence, which also have their routes in nature, for instance animal learning or the study of how dogs learn [15]. These approaches only look at a single agent's thought process when agent's percepts (senses) are mapped to actions in an agent's particular environment [15].

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 [15, 116]. In the field of Swarm Intelligence, animals and insects are represented by agents, which are simple stimulus-response agents and can only perceive changes in their local environments as well as react and interact with other agents. A group of agents is referred to as a *swarm*.

Swarm intelligence works on a key aspect observed in nature, the notion of emergent behaviour [19, 41]. Emergent behaviour is when a collective, swarm or group of simple agents operating in an environment give rise to more complex behaviors as a collective, swarm or group. [19, 41]. These

behaviours are not caused by a coordinated system, instead these behaviours stem from the interactions of individuals between themselves and with their environment [41].

In a swarm, the individuals communicate with each other about knowledge gained as a consequence of the changes in their local environments [19, 81]. Changes can be caused by emergent behaviour as exhibited by the swarm or by the environment the swarm is operating in [41, 42]. Communication among agents facilitates *knowledge sharing* [19, 41].

Social interaction among agents is certainly not the only means of interacting, but with regard to biological inspired systems it is the most prominent [42]. Using this social interaction between the agents of the swarm can influence their own local environment to move towards more promising space in the search space [42, 81]. Thus each agents of a swarm contributes to the swarm as a whole to locate and produce better solutions [24]. Due to the social interaction between agents the swarm is also referred to as a *social swarm* [41, 81].

As discussed the behaviour propagates from one agent to another through social interaction, which brings forth information exchange [19]. Social interaction is but one component of self-organisation. Other components that form part of self-organisation are [132]:

- Positive and negative feedback [132]
- Increased fluctuations of random events [132]

The means by which agents facilitate indirect communication with each other is known as *stigmergy*. Stigmergy as well as the different forms of stigmergy are discussed in 6.2.

Swarm intelligence algorithms are also meta-heuristic algorithms, with the distinction being made that swarm intelligence algorithms use multiple individuals who through simple actions and social interaction are able to work together as a collective entity to search the problem space [19, 24, 41, 42, 81].

The initial algorithms developed with regard to swarm intelligence, are derived from the coordination and behaviour exhibited by schools of fish and flocks of birds. The newer generation of algorithms include [19, 24, 81]:

- ACO [19].

- artificial bee colony (ABC) [24].
- PSO [81].
- bacterial foraging optimisation [42].
- Firefly optimisation [42].
- Fish school optimisation [42].

Swarm intelligence based algorithms are able to achieve good results since they have simple individuals searching in their own local environments for optimal solutions [41, 42]. A direct consequence of multiple individuals in a swarm searching is that the algorithms are able to explore multiple locations within the defined search space [41, 42]. Traditional single agent based metaheuristic algorithms like TS (section 5.4) and SA (section 5.5) only have in essence one "individual" searching for a solution [41, 42].

NP-Complete optimisation problems are but one of the fields where swarm intelligence algorithms have been adapted to. Other fields where swarm intelligence has been applied include neural network training [42], vehicle routing [31], clustering [58], search engines and electrical power systems [11].

This chapter is organised as follows. Before the algorithms are discussed an overview of stigmergy is presented in section 6.2. Starting with section 6.3 the first swarm intelligence algorithm is discussed namely, Ant Colony Optimisation. Artificial bee colony is discussed in section 6.4. Section 6.5 is a discussion about the Particle Swarm Optimisation algorithm. This chapter concludes with section 6.6 that summarises the chapter.

6.2 Stigmergy

Stigmergy is defined as the method used by animals and insects to facilitate indirect communication [14, 42]. Through the use of stigmergy animals or insects are able to socially interact with their own species to convey information to each other [33, 41].

Interaction occurs through signals that the individuals receive which might require them to perform a specific action [14, 33, 42]. 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 [42].

Examples of this type of stigmergy are nest building and brood sorting by ants [42]. 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 [19].

The other form, *sign-based stigmergy*, is an indirect form of interaction, where communication occurs through some sort of signal mechanism [42]. Ants use sign-based stigmergy to communicate with each other. More on how ants communicate with this type of stigmergy is discussed in section 6.3.1.

Other species that use sign-based stigmergy are bees [56]. 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 [56]. This pheromone alerts nearby other bees from the hive of the presence of an entity that is a potential danger to the hive [56].

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 [56]. As more bees sting the entity, more bee stingers emit the danger pheromone identifying the entity [56]. Hence the pheromone is reinforced and becomes stronger, which persuades more bees into action [56].

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 [14, 33, 42]. 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 [14, 33, 42].

The forth-coming sections discusses three swarm intelligence algorithms. Each section is divided into four subsections. First, an overview of the algorithm is given, where basic concepts about the algorithm are introduced as well as a general outline given of the search process the algorithm uses. The second subsection will give an in-depth discussion of some of the core characteristics that make the algorithm unique. The third subsection will provide a step-by-step discussion of the algorithm using pseudo code as a

reference. 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.

6.3 Ant Colony Optimisation (ACO)

6.3.1 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 [33, 42]. 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 [33, 42].

The bridge experiment outlined by Deneubourg placed a food source a certain distance away from the nest [33, 42]. Two bridges of equal length were established towards the food resource. The ants initially, randomly selected a path with no clear distinction of the more dominant path to take to retrieve food from the food source [33, 42]. After a finite amount of time, one of the paths to the food source became the preferred route for the ants even though both paths were of equal length.

Deneubourg concluded that ants utilise pheromones to communicate to the rest of the foraging ants the shortest path towards a food source. [41] By using pheromones to communicate with other ants it can be concluded that ants use sign-based stigmergy (discussed in section 6.2) when they retrieve food [14, 33, 42]. 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 [42]. The chemical signal that ants use to indicate optimal paths is called *pheromones* [33, 42].

The bridge experiment was extended to have two bridges that differ in length. The extended bridge experiment is presented in figure 6.1 and is known as the shortest path bridge experiment [41]. In the experiment the ants started to prefer the shortest bridge [41]. The conclusion was made that the ants preferred the shorter bridge because ants return to the nest quicker and therefore the path is reinforced with pheromones faster than on

the longer path [41].

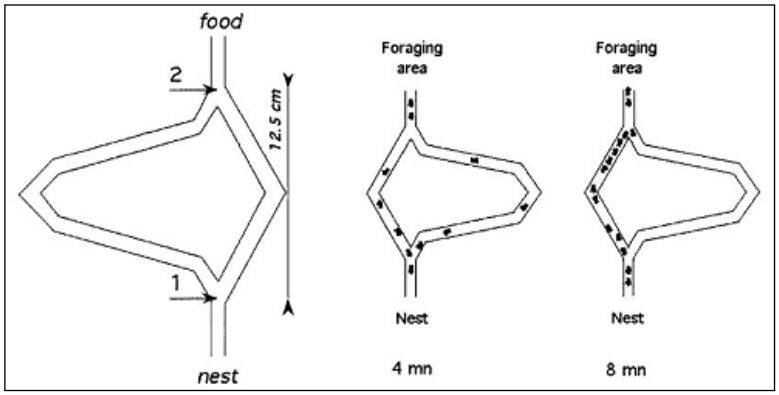

Figure 6.1: The shortest path bridge experiment [33]

Pheromones used by the ants in ACO go through two phases. The first phase is where pheromones are deposited on a particular path by an ant whether there exists previous pheromones or not. The second phase is where pheromones evaporate. Pheromones are not permanent and deteriorate over time [41]. By letting pheromones evaporate, ants can "forget" previous decisions made by the colony [41]. It can be concluded that the more pheromones evaporate the less influence the ants will have on their path selection, therefore promoting exploration [41].

The concept of pheromones and how the ACO proceeds in updating the pheromones is a critical concept of ACO. Hence, an in-depth discussion on pheromones is provided in subsection 6.3.2.

The ACO class of algorithms have a *core requirement* about the problem they are applied to [41]. The problem must be able to be modelled as a graph. The reason behind this requirement is that each individual ant in the ACO algorithm constructs a path through the graph [41]. The path constructed represents a solution.

A path through a graph is made up of a series of links between nodes [53,116]. A link between two nodes represents a movement from one node to the other [53,116]. Therefore, a path can be considered as the traversal of the interlinked nodes, from a starting node to some final node [53, 116]. A path differs from another path by the order in which the nodes are interlinked between a start and end node [53, 116].

The ACO class of algorithms has been applied to a wide range of problems that include single machine scheduling [62],weapon target assignment [77], flow shop scheduling [23] and image thresholding [153]. Variants of the standard algorithm have been developed, but all of the algorithms still follow the core structure of the ACO algorithm [41, 42]. The first algorithm developed based on the foraging behaviour of ants is known as the simple ant colony optimisation (SACO) and was proposed by Dorigo in 1992 [42]. The algorithm provided the basis for how pheromones are used and updated. The SACO is an algorithmic implementation of the double bridge experiment.

The first algorithm to improve upon the SACO is the ant system (AS) [14, 42]. The AS included heuristic information into the probability that an ant chooses to move towards a node. The AS also added memory to the AS by using a tabu list and also incorporated pheromone evaporation. The improvements made enabled the AS to better explore the search space and produce better results [14, 42].

The AS algorithm has achieved good success in the problems it has been applied to, but it does have some disadvantages [31, 148]. One of the primary disadvantages of AS is that it tends to converge prematurely to local optima [41, 148]. The premature convergence can be attributed to the ants exploiting the high concentration of pheromones on good solution paths too quickly [41]. With the ants focusing only on the good solutions less exploration occurs in the search space leading to local optima being produced as the best solution [41].

Subsequently, various algorithms have been developed that improve on the AS algorithm. These improved algorithms include the ant colony system (ACS), min-max ant system (MMAS), Ant-Q, fast ant system, AntTabu, AS-rank and approximate non-deterministic tree search (ANTS) [14, 42]. The discussion in this section is focused on providing an introduction to the general ACO algorithm concepts and not to discuss various improvements made by variants of the core algorithm. In the forth-coming sections reference is made to these algorithms and their respective improvements.

The AS algorithm is the base algorithm upon which all other ACO class algorithms are based. Therefore in the forth-coming sections when a reference is made to the ACO algorithm, it is directed at the base AS algorithm.

In this section the concepts upon which the core of ACO is based upon

are briefly introduced. The next section discusses each of these concepts.

6.3.2 ACO Characteristics

In this section characteristics that are important and unique to the ACO class of algorithms are discussed. The discussion is focused upon three core characteristics namely pheromone trials, pheromone evaporation, pheromone updates and state transition rules.

Pheromone Trail

The pheromone technique used by ants' forms part of the core methodology used by the ACO algorithm [46]. As an ant moves it lays down pheromones to mark the path it is walking.

With the use of pheromones ants are able to communicate the best and shortest link between nodes [33, 42, 46]. The more ants following a preferred link the more pheromones would be deposited on that specific link. This increases the strength of the pheromones [148]. The increase in strength of the pheromones on a link would thus let ants more clearly distinguish between links they should and should not take [148]. Therefore, a pheromone provides positive feedback to the colony [33, 42, 46].

Initially, all the ants will choose a random link to a node [33,42,46]. After all the ants have completed their paths, each path is evaluated using a cost function defined by the problem domain [42]. The amount of pheromones marking the links contained in a path in the standard ACO are related to the cost function [33, 42, 46]. Therefore, a low cost function value will have a high pheromone dosage and a high cost function value will have a low dosage [42].

By incorporating the cost of a particular path into the amount of pheromone deposited the colony is able to influence future decisions [42]. In terms of minimisation a path with a low cost will have a high pheromone value making the links of path more likely to be selected by future ants [42].

In the iterations following the initial one, the ants will at each node decide based on a probability whether it should move to a particular neighbouring node. The higher the pheromone intensity is at a neighbouring node, the higher the probability that the ant will choose to move towards that node

[33, 42, 46]. The probability with which ants choose links to neighbouring nodes are defined and discussed in the next section.

Due to ants choosing links to node based on a probability, it is still possible for the ant to choose a random link towards another node. Thus the ACO algorithms are considered stochastic search procedures due to the ants' ability to choose links randomly when exploring the search space [31, 148].

The pheromone trail was initially developed with only one colony in mind [42]. In research done by Tiwari *et al.* [135] 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 [33, 42, 46]. The ant will not follow or even consider the pheromone trail since any pheromone encountered from other colonies repulses the ant [135]. 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 [135]. This repulsion strategy promotes exploration among the multiple colonies [135]. The probability with which a link towards a particular node is chosen by an ant forms part of the *state transition rules*.

Pheromone Evaporation

Initially when the pheromone concept was first implemented the ants of the colony rapidly converged on a solution [42]. The search space was not adequately explored and the produced solution was a local optimum [33]. To combat this premature convergence and force the ants to explore the search space more, the concept of *pheromone evaporation* was introduced [14, 32].

Real pheromones used by ants to mark a particular link to a node is not permanent and over time the strength of the pheromone deteriorates until it eventually disappears [42]. Pheromone deteriorating is known in the literature as pheromone evaporation [42]. The pheromone will not completely evaporate as long there are ants traversing the defined link reinforcing the pheromone. The evaporation of pheromones is modelled in the ACO by equation 6.1 [14, 32]:

$$\tau_{ij}(t) = (1 - p)\tau_{ij}(t), p \in [0, 1] \tag{6.1}$$

The constant p defines the rate at which the pheromone evaporates. If $p = 1$ the pheromone completely evaporates every iteration. With no pheromone

on a link towards a node the ants take no knowledge gained from the previous iteration into account and therefore select a link randomly [33, 42]. Thus, the amount of exploration done by the algorithm can be controlled by the constant p [33, 42].

Equation 6.1 was first introduced in the AS [32, 42]. 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 [33, 42].

A more aggressive form of pheromone evaporation is added to the AS discussed in the research done by Gambardella *et al.* [46]. The more aggressive form works beside the already present pheromone evaporation, but this form seeks to add an additional search phase called *diversification* [46]. The aim of the diversification phase is to lead the algorithm into another direction of the search space [46]. This is done in an attempt to avoid local minima and stagnation [46].

In the ant system developed by Gambardella *et al.* the algorithm continually monitors the current best solution and keeps a history of recent best solutions [46]. 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 [46]. In this phase 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 [46].

State Transition Rules

The intention of this section is 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 be considered derivatives of the first.

As discussed previously, the ants select which link to follow towards a node based on a probability. This probability is also known as the *transition probability* and is formulated by equation 6.2.

$$p_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}}{\sum_{u \in N_i^k(t)} \tau_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)}, & \text{if } j \in N_i^k(t) \\ 0, & \text{if } j \notin N_i^k(t) \end{cases} \tag{6.2}$$

The transition probability is used by individual ants of the AS algorithm [41, 46]. An ant k uses this equation to decide with what probability it will move from node i to node j [42, 135]. τ_{ij} is the amount of pheromone on the link between nodes i and j [33,135]. Fitness information is incorporated into the equation through the symbol η_{ij}, which is the desirability of the link from node i to node j as evaluated by a heuristic function [33, 135]. Each ant starting at a source node moves from one node to another based on the defined transition probability until the ant reaches the final node.

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 $\alpha = \beta$ [46, 135]. When $\alpha = 0$ no pheromone is taken into account; hence, any history that the algorithm has on the link 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 link 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 interested reader that requires more information about state transition rules is directed to the survey by Engelbrecht [41].

Pheromone Update

Pheromones start to evaporate over time, and so the link marked by a pheromone trail becomes less attractive to the ants. Therefore, a path that represents a good solution needs its pheromone trail to be continuously reinforced. Certain rules govern when and by how much pheromones are reinforced.

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 [42]. One group is called the global update rule. The other group is called the iteration-based or local update rule [42].

The first local pheromone update rule was introduced in the AS algorithm [32, 33, 42]. 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:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{6.3}$$
$$\text{where } \Delta\tau_{ij} = \sum_{k=1}^{n_k} \Delta\tau_{ij}^{k}(t)$$

In equation 6.3 $\tau_{ij}(t+1)$ 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 since the first iteration to be updated [42]. Thus the global rule favours intensification where the algorithm exploits the global knowledge gained by the ants to find a better solution. By updating a pheromone the concentration of the particular pheromone is reinforced.

ACS was the first to use both the global update rule and local update rule together [42]. By using both types of rules the algorithm is able to efficiently exploit the history provided by the pheromones [42]. The global update rule used by the ACS is formulated in the following equation [42]:

$$\tau_{ij}(t+1) = (1-p_1)\tau_{ij}(t) + p_1\Delta\tau_{ij}(t), \tag{6.4}$$
$$\text{where } \Delta\tau_{ij} = \begin{cases} \frac{1}{f(x^+(t))} & \text{if } (i,j) \in x+(t) \\ 0 & \text{otherwise} \end{cases}$$

The parameter $f(x^+(t))$ represents the best/shortest path found so far by the algorithm [42]. p_1 is the variable that controls the 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 search space more. Exploitation is achieved since the best path is continually used in the update of the pheromone as can be observed in equation 6.4 [41,42]. As can be seen in the following equation, the ACS uses a slight variant of the local update rule first used in AS [42]:

$$\tau_{ij}(t) = (1-p_2)\tau_{ij} + p_2\tau_0 \tag{6.5}$$

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 [42]. With the local update rule, the algorithm is able to explore more. The path constructed by the individual ant is used to update the pheromone and no information from the best path found in the colony is incorporated, as with equation 6.4 [41, 42].

The MMAS algorithm as discussed also improves on the AS. The global update rule used by AS (see equation 6.4) has a disadvantage in the sense that the search might concentrate too quickly on a particular good solution (the global best path) [41]. MMAS addresses this disadvantage by using the global update rule on an iteration basis [41].

With the MMAS algorithm the best path found differs from one iteration to the next [41]. When the MMAS enters the pheromone update phase a different path will be updated every iteration [41]. This is due to the algorithm providing no guarantee that the best found path in one iteration will be the best path in the next iteration [41]. There is thus a possibility that the algorithm will deposit on various different paths as it progresses through iterations [41]. Due to different paths being deposited with global pheromones the colony considers a wider range of possible routes and thus explores the search space more [41].

Another shortcoming of the AS is that pheromone concentrations can become extremely high leading to less exploration by the algorithm [41]. The MMAS algorithm addresses this shortcoming by enforcing a maximum and minimum amount of pheromone that can exist on a path [41]. By defining a maximum the algorithm is prevents the algorithm from settling on one particular solution i.e. stagnation [41]. On the other hand, defining a minimum on all possible links between nodes ensures that links will be continuously considered for possible inclusion into a solution [41]. In addition to providing boundaries for the pheromones MMAS also uses a smoothing strategy to even out the difference between high and low pheromones [41].

6.3.3 Flow of the Algorithm

In this section the process the AS algorithm uses to explore the search space is described. Algorithm 5 is used as a reference point.

Algorithm 5 Ant System Algorithm [42]

1: Initialize τ_{ij} with small starting values
2: $t \leftarrow 0$
3: Place n_k ants on starting node
4: **while** stopping condition not reached **do**
5: **for** each ant $k \leftarrow 0$ to number of ants n_k **do**
6: $p^k(t) \leftarrow$ Initialize path $p^k$for time step t
7: **repeat**
8: Select next node based on probability equation 6.2
9: Add link (i,j) to path $p^k(t)$
10: **until** Final node reached
11: $x^k(t) \leftarrow$ Remove loops from path $p^k(t)$
12: Calculate length of path $f(p^k(t))$
13: **end for**
14: **for** each link (i, j) in graph **do**
15: $\tau_{ij} =$ Reduce pheromone of link (i, j) with equation 6.1
16: **end for**
17: **for** each ant $k = 0$ to number of ants n_k **do**
18: **for** each link (i, j) in $p^k(t)$ **do**
19: $\triangle\tau_{ij} = \frac{1}{f(p^k(t))}$
20: Update the pheromone τ_{ij} with equation 6.4
21: **end for**
22: **end for**
23: $t \leftarrow t + 1$
24: **end while**
25: **return** path $x^k(t)$ with the smallest $f(x^k(t))$ as the solution

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 5, lines 1 – 3. The main purpose of the ant is to explore the search space and to ultimately produce a solution that might be optimal. The ant explores the search 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 [41, 42]. 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 search space [41, 42]. 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 6.1.

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 5.

After all the ants have deposited pheromones on all the links represented by 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.

6.3.4 ACO on the FAP

ACO has been applied to a wide number of problems like weapon targeting [77], flow shop scheduling [23] and quadratic assignment [46] where it has achieved good results. As discussed in chapter 4, the FAP can be modelled as a graph and therefore the ACO has also been applied to it [82].

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 5 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 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.* [82] 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 [82]. 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 [82].

With regard to the heuristic updating of the pheromone trails, the authors opted to increase the pheromone by some magnitude [82]. This magnitude was hand tuned 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 [82]. Thus, the heuristic aims to amplify good choices made previously by the algorithm for the next iteration of the algorithm.

In table 6.1 the results obtained by Lune *et al.* [82] are presented. By evaluating the results obtained, one can clearly see that the ACO version that incorporates heuristic information to reinforce pheromone trails outperforms the version that does not [82].

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

Table 6.1: ACO and ACO* on custom GSM FAP benchmark [82]

The values depicted in the table represent the amount of interference that will result if the plan is used in the network [82]. The following section discusses the Artificial Bee Algorithm.

6.4 Artificial Bee Colony (ABC) Algorithm

6.4.1 Introduction

The ABC algorithm is the most recently presented algorithm in the literature discussed in this chapter [72, 73, 122]. The algorithm was first proposed by Karaboga in 2005 who wanted to mimic the foraging behaviour exhibited by bees [72, 73, 122]. 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 [72].

In a bee colony there are numerous bees, each with a specific role that dictates what actions a bee can perform. 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 [72].

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 [72]. If a scout bee has found a food source it needs to return to the colony to share the information with the worker bees [72]. When the bee enters the colony it needs to communicate to the other bees by using some form of stigmergy (see section 6.2) [72].

The scout bee accomplishes this communication by performing a dance known as the *waggle dance* in the colony for all the bees to see [72]. 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 [72]:

- 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 6.2). This is deduced from the dance, which is a physical form of communication.

The dance is observed by *onlooker* worker bees [13, 72]. These onlooker bees are initially *unemployed* in the colony [13, 72]. Once the information of the scout has been transferred to the onlooker bees, the onlooker bees become *employed* bees [13, 72]. They become employed bees when they operate on a particular food source to gather food [13,72]. Thus it is the job of the worker bees to *exploit* the information provided by the *exploration* done by the scout bees [72, 73].

Worker bees gather food from the designated food source, until the food source reaches a certain quantity with regard to nectar content [72,73]. Each time the bee returns to the colony it evaluates the current food source versus other food sources discovered [72, 73]. If a better food source is found, the bee abandons the previous source and starts gathering food from the new source [72, 73]. 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" [72, 73].

In the ABC algorithm, possible solutions are considered to be food sources [72, 73]. 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 [72, 73].

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 [72, 73]. 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 [72, 73].

As with real honeybees, 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 [13, 72, 80]. The onlooker bees choose food

sources depending on the nectar amount [13, 72, 80]; therefore as the nectar amount of a food source increases, the probability that more onlooker bees will choose the source increases [13, 72, 80]. How and what affects the probability is discussed in the next subsection.

Bees can transition to different roles depending on their situation [72,73]. 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 [13,72,80]. 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 [13,72,80]. Scout bees are sent to randomly generated food sources [13, 72, 80].

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 [72,73]. Thus, this can be considered exploitation and the algorithm is therefore performing a local search [72,73,80]. 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 [72,73,80]. 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 [72, 73, 80].

When a food source is abandoned, the previous bee that occupied the food source transitions to a scout bee [72, 73]. 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 [13, 72, 73]. The generation of food sources is 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 optimisation (BSO) [73, 87, 133].

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 [73,87]. 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 [73, 87].

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 [73]. Instead the VBA is a variant of the standard ABC algorithm, which is designed for numerical function optimisation [73]. In VBA bees move around in the search space communicating to each other any target nectar food sources that are found [73]. 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 [73].

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 [87], neural network training and reconfiguration of distribution networks [80]. 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 of other of problems.

6.4.2 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 [52, 72]. 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 [72, 122].

Employed bees evaluate these food sources by determining their nectar amount [72, 122]. The nectar amount is directly related to the fitness value calculated using a domain specific cost function [72, 80]. After the amount is determined the employed bee advertises the food source to the colony by performing the waggle dance.

Onlooker bees witness a number of dances from a variety of employed bees [24, 52]. 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 6.6 [72]:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{6.6}$$

The parameter p_i is the ith food source under consideration by the onlooker bee. 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 sources and hence the maximum employed onlooker bees [72].

In algorithm 6 the waggle dance can be seen being applied in line 11 where the probability of all the employed bee' solutions are calculated using equation 6.6. 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 [72, 73]. Instead the bees explore the immediate neighbourhood of the food source that was communicated to them [24, 52, 80]. They seek to find a food source that improves on the previous one [24, 73]. Equation 6.7 is used by the bees to generate new food sources in the neighbourhood of food source x_{ij} [52, 72].

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{6.7}$$

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 6.7 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 [13, 72, 73].

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 [72, 80]. Otherwise, the bee abandons the new solution and keeps the old solution in memory [72, 73]. Thus, the bee seeks to always move towards a better solution and therefore uses a greedy selection process [80, 122].

One of the problems with the above approach is that little information about the food source is used in generating a neighbouring food source. Research by Singh [122] proposed a slight variation 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 [122]. Solutions generated by scout and onlooker bees are not checked whether they collide with other potential solutions already present in the system [24, 72]. This is due to the purpose of scout and onlooker bees is to explore the search space by generating solutions and not to exploit the search space [24,72]. Exploitation of the search space is the sole purpose of employed bees [24, 72].

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 [13, 24, 73]:

- 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 [13, 72, 73].
- The bee cannot improve the search represented by the food source any further [13, 72, 73].

When a food source in the algorithm is abandoned, it needs to be replaced by a new food source [13,24,72]. Note that a food source is not abandoned when it represents the globally best-found solution. An employed bee transitions to a new role when it abandons a food source from an employed bee to a scout bee [13, 72, 73].

It is the responsibility of the scout bee to replace the abandoned food source with a new randomly generated one [13, 24, 72]. The scout bee uses equation 6.8 to produce a new food source that will replace food source x_i.

$$x_{ij} = x_{yj} + rand[0, 1](x_{zj} - x_{yj}) \tag{6.8}$$

The subscript y represents the minimum value of i and the subscript z represents the maximum value of i. In research done by Gómez-Iglesias *et al.* [52] an extension is made to the scout bees. The scout bee individuals are divided into two types of bees, namely *rovers* and *cubs'* bees [52].

- Rover bees are similar to traditional scout bees and hence use diversification strategies to explore the search space.
- Cub bees explore the search 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 search space in the beginning where diversity is preferred and late in the algorithm where intensification is preferred [52].

The following subsection describes the general flow of the ABC algorithm. The discussion of the algorithm aids in the understanding of how the algorithm searches a particular problem space.

6.4.3 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 6 as a reference point.

Algorithm 6 Basic Artificial Bee Colony Algorithm [72]

1: $b_n \leftarrow$ Initialize bees
2: $s_n \leftarrow$ Initialize starting solutions
3: Evaluate starting solutions with fitness function $f(s_n)$
4: $t \leftarrow 0$
5: **while** stopping criteria not met **do**
6: **for** each employed bee $eb_i = 0$ to max bees b_n **do**
7: $\hat{v_i} \leftarrow$ Generate new solution with equation 6.7
8: Evaluate with fitness function $f(\hat{v_i})$
9: Apply greedy selection between $\hat{v_i}$ and the current solution $\hat{s_i}$ of bee eb_i
10: **end for**
11: Calculate probability p_i for solutions $\hat{s_i}$ in $\hat{s_n}$ using equation 6.6
12: **for** each onlooker bee $ob_i \leftarrow 0$ to max bees b_n **do**
13: $\hat{x_i} \leftarrow$ Select solution $\hat{s_i}$ based on p_i
14: $\hat{v_i} \leftarrow$ Generate new solution with $\hat{x_i}$ and p_i
15: Evaluate $\hat{v_i}$ with fitness function $f(\hat{v_i})$
16: Apply greedy selection between $\hat{v_i}$ and bee ob_i current solution
17: **end for**
18: **if** there is an abandoned solution for a scout bee **then**
19: Replace with solution generated with equation 6.8
20: **end if**
21: $t \leftarrow t + 1$
22: **end while**

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. The operations that perform these functions can be observed to occur 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 search 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 6 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 6.6 which is discussed in section 6.4.2.

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 6.7 which is discussed in section 6.4.2. This procedure of generating neighbouring food sources by an onlooker bee can be observed in line 14 in algorithm 6.

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 6 this occurs in lines 18 – 22. The scout bee

uses equation 6.8 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.

6.4.4 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. The following critique is based on a theoretical implementation of the ABC algorithm on the FAP. Based on this evaluation 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 were 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, more onlooker bees will select it. By selecting the food

source the onlooker bee will then proceed to search in its neighbourhood for 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. 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 nectar amount becomes lower. As a consequence of the low nectar amount by the food source the bees 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. The algorithm first needs to be applied to a wider set of problems. Once the algorithm has matured in the research behind it the algorithm can be revisited and applied 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.

6.5 Particle Swarm Optimisation (PSO)

6.5.1 Introduction

PSO is population-based stochastic search technique that was developed by Kennedy and Ebenhart in 1995 [120]. The basic model of the algorithm is based on simulations done to recreate the natural behaviour of a flock of birds [145].

In the early stages of the particle swarm development, simulations were developed to closely model the stigmergy (see page 101 for a discussion on stigmergy) exhibited when a 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 [70].

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 [70].

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 101).

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 [42].

In the algorithm a particle is an individual [41]. A group of particles, referred to in the literature as a swarm, are moved through the search space of the problem the algorithm is being applied to [41]. Each particle changes its movement based on information shared with it by neighbouring particles in the swarm [41, 42].

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 [41, 42].

Two variants of the initial PSO algorithm were developed namely the 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 are discussed in the PSO characteristics subsection [107].

In the swarm, each particle is a potential solution that is represented by a D-dimensional vector [70, 113]. As a particle moves through the search space, it continually evaluates its current position and adjusts it accordingly to move in the general direction of its own personal best position and the position of the best particle in its neighbourhood.

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 [42].

As a particle moves through the search 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* [107].

Most of the research done on PSO has focused on the convergence of the algorithm as well as improving the diversity [41]. Some of these improvements and modifications are discussed in the subsection on PSO characteristics.

6.5.2 PSO Characteristics

The defining characteristics of the PSO algorithm are discussed in this section. The characteristics include *neighbourhood topology*, *the particle swarm*, *movement of particles* and *keeping particle velocities in check*.

The Particle Swarm

The initialising of particles in a swarm are the same as used by traditional population-based evolutionary algorithm to initialise their respective populations [150]. At the start of the algorithm the swarm is initialised by randomly generating possible solutions that will represent the position of particles [42].

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. By utilising good mutation operators the GA is able to insert new genetic material into the population. At least in the initial search phase of the GA, this increases diversity [42]. The PSO does not continually generate new solutions to be reinserted into the swarm to increase diversity [134]. 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 [41, 42].

The diversity of the swarm is not only dependant on the sharing model used but also on the parameters used for velocity updating of particles. Most important of these parameters are the inertia weight and acceleration coefficients. Depending on the values used the diversity of the swarm decreases as the swarm moves because, with each iteration, the swarm converges towards the neighbourhood best position [41, 42, 70]. This occurs due to the velocity equation directing the general movement of a particle in the direction of the

neighbourhood best [41, 42, 70]. If the best position in the neighbourhood does not change as the algorithm processes more iterations, a larger portion of the swarm will soon occupy a position which is a weighted average between the neighbourhood and each individual particles personal best [41, 42, 70]. On the other hand, if the wrong values are used, the diversity of the swarm will increase, but there is also the possibility that the swarm will diverge never to converge to a single solution but have increased diversity [41].

Diversity among the particles in the swarm is important and at least initially must be maintained for the search space to be explored adequately. As discussed previously, depending on the neighbourhood topology diversity can be increased. The next section discusses neighbourhood topologies.

Social structures

The social structure (also called neighbourhood topologies) used by a PSO algorithm dictates how information is shared among the particles of the swarm. According to Engelbrecht [41] there are 6 neighbourhood topologies. Each topology will now be listed and a short description will also be given.

star — With the star topology all particles within a swarm are interconnected with each other. Any one particle can communicate with any other particle in the swarm [41]. Using this topology each particle is attracted towards the best solution found globally by the swarm [41].

ring — Using the ring topology each particle communicates with its immediate adjacent neighbour [41]. Each particle aims to mimic the best solution found within its neighbourhood [41]. Neighbourhoods are allowed to overlap in the ring topology [41]. Overlapping allows for information sharing among neighbourhoods and facilitates the swarm in converging to a single solution [41].

wheel — With the wheel topology there is a central particle with which all other particles in the swarm are connected to [41]. The other particles are isolated and can only share information with the central particle [41]. The central particle utilises information about all the particles in its neighbourhood and adjusts its own best position accordingly [41]. Depending on whether the new best position represents a better solution than previously, the central particle shares the information to

its neighbours [41]. A consequence of the limited sharing that occurs in the wheel topology is that the propagation of good solutions through the swarm is slowed [41].

pyramid — The pyramid topology interconnects particles so that their connected structure resembles a three-dimensional pyramid [41].

four clusters — With this topology four clusters are formed. Each cluster containing interconnected particles [41]. Between each cluster there exists only two connections to other clusters [41].

von Neumon — Using this topology all particles are connected in a structure that resembles a grid [41]. This structure has been shown to enable the swarm to produce better results than other neighbourhood topologies [41]

Based on the neighbourhood topologies defined above the global PSO uses a star neighbourhood to allow for information to be shared among all particles in the swarm. The particle whose position in the search space indicates the best solution found by the swarm is denoted as *gbest* [41, 42, 107].

In contrast, the local PSO follows the process of natural birds more closely and uses the ring neighbourhood for information sharing [41,42,107]. Hence, particles only share information with their immediate neighbourhood and not with the whole swarm. The best particle in the local PSO is denoted as *lbest* [41, 42, 107].

Movement of particles

A particle moves with a certain velocity through the search 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 search 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 [41, 42, 107, 113].

The amount of global knowledge is referred to as the *social* component [41, 42, 107, 113]. The amount of personal information used by a particle is referred to as the *cognitive* component [41, 42, 107, 113].

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 search space and particles are moved [42].

The velocity update is where the personal experience of a particle and the knowledge gained through social sharing are incorporated. By updating the velocity of a particle, the particle is steered into a more promising direction. The velocity of a particle is updated based upon on equation 6.9.

$$\hat{v}_i(t+1) = \hat{v}_i(t) + c_1\hat{\phi}_1(t)[\hat{pbest}_i - \hat{x}_i(t)] + c_2\hat{\phi}_2(t)[\hat{gbest}_i - \hat{x}_i(t)] \quad (6.9)$$

$$\hat{x}_i(t+1) = \hat{x}_i(t) + \hat{v}_i(t+1) \quad (6.10)$$

where $\hat{v}_i(t+1)$ is the new velocity of particle i for the next time step $t+1$. The cognitive component is represented by the term $c_1\hat{\phi}_1(t)[\hat{pbest}_i - \hat{x}_i(t)]$ where c_1 is the cognitive coefficient. The social component is represented by the term $c_2\hat{\phi}_2(t)[\hat{gbest}_i - \hat{x}_i(t)]$ where c_2 is the social coefficient (discussed on page 131) [41, 42]. Each of the respective components c_1 and c_2 controls how much neighbourhood information is used in the calculation of the new velocity.

The variables $\hat{\phi}_i$ and $\hat{\phi}_2$ are vectors containing random scalar values in the range $[0, 1]$. The current position of a particle in the search space at time step t is represented by parameter $\hat{x}_i(t)$ [41, 42]. After the new velocity is calculated the position of the particle is updated for time step $t+1$ using equation 6.10 [41, 42]. The velocity update can be visually depicted as shown in figure 6.2.

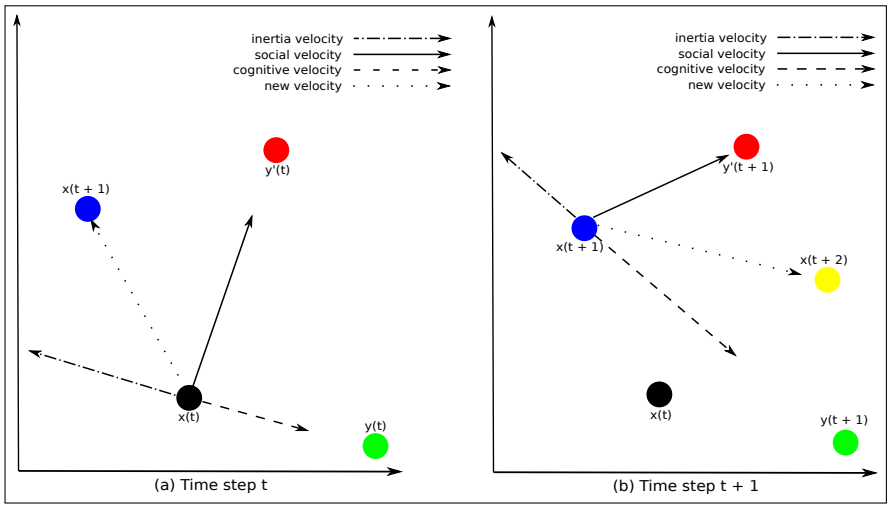

Figure 6.2: Visual particle velocity update [41, 42, 107, 113]

As discussed earlier gbest is the best position the swarm has occupied since the start of the algorithm. The literature defines that 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 [41,42]. 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 [41, 42].

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.* [113] particle positions in the solution space are continually monitored. If the particle appears to be stagnant in the search 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 [113].

In research done by Kalivarapu *et al.* [71] 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 algo-

rithm developed by the authors achieved promising results, with it finding solutions faster and also better solutions than other PSO algorithms [71].

Other research done by Monson and Seppi [95] 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 search space and another particle at a certain time step occupies that same space, the particles are said to collide. As one would expect, when a collision occurs both particles are deflected into random directions [95]. At a greater expense of computational time due to constant collision detection, the PSO gains greater exploration in the search space.

Finally, in the research by Lenin and Monan a PSO algorithm is developed that is called the attract repulse particle swarm optimisation (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 [78].

Keeping velocity in check

As can be observed in equation 6.9 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.

Velocities should be kept in check since if a particles velocity becomes too large, it can overstep the search spaces boundaries and produce infeasible solutions [41]. Thus the velocity of a particle needs to be bounded to ensure that its movement within the search space stays within acceptable bounds. One of the means to bound the velocity is to clamp it. Clamping of velocity is achieved by applying equation 6.11. The equation is applied on the velocity

before its position is updated [41].

$$\hat{v_i}(t+1) = \begin{cases} \hat{v_i'}(t+1), & \text{if } \hat{v_i'}(t+1) < \hat{V_{max}} \\ \hat{V_{max}}, & \text{if } \hat{v_i'}(t+1) \geq \hat{V_{max}} \end{cases} \tag{6.11}$$

$$\hat{V_{max}} = \delta(\hat{x_{max}} - \hat{x_{min}}) \tag{6.12}$$

In equation 6.11 $\hat{v_i'}$ is calculated using equation 6.9 for the global PSO. Where $\hat{V_{max}}$ is the maximum allowed velocity and $\delta \in (0, 1]$. The values $\hat{x_{max}}$ and $\hat{x_{min}}$ are the respective maximum and minimum position vectors in the domain the algorithm is being applied to [41]. The value of δ is very problem dependent and must be carefully chosen to maximise the exploration-exploitation trade-off [41]. The use of velocity clamping is not mandatory and should be considered only if the problem requires it [41]. Finally there is no guarantee that velocity clamping will prohibit velocities becoming too large [41]. There is still a chance, just to a lesser extent [41].

Velocity clamping is not the only developed means by which the exploration-exploitation trade-off of the PSO can be controlled. Consider the case when 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 object in the previous direction. Inertia weight seeks to add this type of behaviour to the particles of the PSO algorithm. The velocity update equation with added inertia is formulated in equation 6.13 [41].

$$\hat{v_i}(t+1) = w\hat{v_i}(t) + c_1\hat{\phi}_1(t)[\hat{pbest_i} - \hat{x}_i(t)] + c_2\hat{\phi}_2(t)[\hat{gbest_i} - \hat{x}_i(t)] \tag{6.13}$$

Inertia (w in equation 6.13) was added to the general velocity update equation in an attempt to control the exploration and exploitation of the PSO as well as eliminate the need for velocity clamping [41]. Although the inertia component did succeed in enabling the control of the PSO's exploration-exploitation in the search space, the need for velocity clamping could not be eliminated [41].

For values of $w > 1$, the inertia of the particle is increased. With increased inertia the particle will explore more but it is also more likely to leave the boundaries of the search space leading to infeasible solutions [41].

When $w < 1$ and depending on the values of c_1 and c_2 each time a particles velocity is updated a certain amount of momentum is lost. The particle seems to slow down, allowing it to exploit the current solution space

in finer detail [41]. This is not always the case, as $w < 1$ can also lead to the particles in the swarm diverging never to converge on an optimal solution.

To allow for a greater trade-off between exploration and exploitation, the inertia value can be made dynamic. Exploration is favoured early on in an optimisation algorithm and exploitation later on the algorithm when it is near an optimum. 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 search space [41, 42].

Finally, a similar inertia type component was developed from the analysis of particle dynamics [41]. This new component is called the *constriction coefficient* and, like the inertia above, also modifies the velocity update equation slightly [41, 42, 95].

This modification can be observed in equation 6.14, which is the standard velocity equation with the constriction coefficient. The constriction coefficient is formulated in equation 6.15 [41, 42, 95].

$$v_i(t+1) = \chi[v_i(t) + c_1\phi_1(t)[pbest - x_i(t)] + c_2\phi_2(t)[gbest - x_i(t)]] \quad (6.14)$$

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \quad (6.15)$$

The search constriction coefficient is represented by the value ϕ. 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 search space and converge quickly [41, 42, 95].

Discrete Value representation

The PSO was developed for continuous valued problem spaces in mind [41, 42]. Fortunately the algorithm has been adapted to a range of problems known as discrete valued problems. These type of problems have the characteristic that the variables contained in the problem domain are finite [41, 42]. Problems that are classified as being of the discrete variant include the n-queens problem, the traveling salesman problem and the FAP [41, 42]. One of the means by which the PSO can be modified to operate in these discrete valued spaces is by changing the representation of position vectors or

redefining the operations used for arithmetic such as addition, subtraction and multiplication [41, 42].

One of the PSOs developed that changes the representation of a position vector is the *Binary* PSO [41, 42]. Although with the Binary PSO, the particles of the swarm operate in binary space, the algorithm can also be applied to real-valued problems since their values can be transformed to fit into the binary space [41,42]. A consequence of operating in the binary space is that each particles position is represented by $x_{ij} \in 0, 1$ [41,42]. A particle is moved in the binary space by flipping bits in its position vector [41, 42].

Due to the change in how positions are represented in the binary PSO the velocity and trajectory of particles need to be interpreted differently [41,42]. Velocity is interpreted to represent a probability. A velocity $v_{ij}(t) = 0.4$ defines that the probability that the bit will be 1 is 40% and the probability that the bit will be 0 is 60% [41, 42].

A different and simpler method of applying the PSO is by rounding the position to the nearest discrete value [41, 42]. Using this method requires that the position vectors of particles not be outside the bounds of the problem space [41, 42].

A more complex approach, as mentioned previously, is to redefine the arithmetic operations used when calculating new velocities of particles. In research conducted by Clerc a discrete PSO is applied to the traveling salesman problem, where all the arithmetic operations have been redefined [41, 42]. The traveling salesman problem is defined as follows. Given an $n_x \times n_x$ distance matrix D, find a permutation, π, that minimises the objective function:

$$\sum_{j=1}^{n_x-1} D_{\pi_j \pi_{j+1}} + D_{\pi_{n_x} \pi_1} \tag{6.16}$$

Clerc redefined the operators as follows:

- Velocity of vector length — The number of changes to the permutation represented by a tour is defined as the length of the vector [41]. Formally the length of the velocity is defined as $|v_i(t)| = J$ where J is the number of elements.
- Addition of velocity to a position — When velocity is added to a position, a new position is formed [41]. A new position represents a new permutation of discrete values and is created by performing swaps

on the position and velocity being added [41]. Let x_i denote a position and let v_i denote the velocity [41]. Then

$$x_i \oplus v_i = p_i \tag{6.17}$$

where p_i is the new position found by applying the swap operation $v_{i1} = (\pi_{a1}, \pi_{b1})$, then for the second swap $v_{i2} = (\pi_{a2}, \pi_{b2})$ up and till the last swap [41].

- Subtracting positions from each other — The result of subtracting two positions from each other is defined as being a velocity [41]. If x_1 and x_2 are positions and v is the resultant velocity, then

$$x_1 \ominus x_2 = v \tag{6.18}$$

v is defined as such that by applying v to x_1 gives x_2 [41].

- Adding two velocities — Adding one velocity v_1 to another velocity v_2 results in a new velocity v_3 [41]. The new velocity v_3 is a concatenation of the swaps from v_1 and v_2 and duplicate swaps are ignored in the new velocity vector [41].

- Multiplication of a coefficient to a velocity — By multiplying a velocity v_1 with a constant c and new velocity is formed. Formally, $v_2 = c \otimes v_1$, where $|v_2| = \lceil c|v_1| \rceil$ [41]

For a more thorough discussion as well as applications of discrete PSOs the reader is directed to the survey by Engelbrecht [41]. Note that the FAP is also a discrete valued problem, the algorithm presented in this research will also need to redefine the standard operators as was done by Cerc for the traveling salesman problem. This redefinition of operators is presented in chapter 7.

6.5.3 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 7 as a reference point. Note that for the algorithm presented a star neighbourhood topology is used and the algorithm is applied on a minimisation problem.

Algorithm 7 Basic Global Particle Swarm Optimisation Algorithm [42]

1: Initialize s_n swarm
2: **while** Stopping condition not met **do**
3: **for** each particle $\hat{p_i} \leftarrow 0$ in s_n **do**
4: Evaluate particle with fitness function $f(\hat{p_i})$
5: **if** $f(\hat{p_i}) \leq pbest(\hat{p_i})$ **then**
6: personal best of $\hat{p_i}$ to $f(\hat{p_i})$
7: **end if**
8: **if** $f(\hat{p_i}) \leq f(\hat{gbest})$ **then**
9: $\hat{gbest} \leftarrow f(\hat{p_i})$
10: **end if**
11: **end for**
12: **for** each particle $\hat{p_i} \leftarrow 0$ in s_n **do**
13: update velocity of $\hat{p_i}$ with equation 6.9
14: update position of $\hat{p_i}$ with equation 6.10
15: **end for**
16: **end while**

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 $f(\hat{p_i})$ 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 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 stored 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 7 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 6.9.

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 6.10 and occurs on line 14 of algorithm 7.

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.

6.5.4 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 is 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.* [40] 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 [40].

The FEA method then iterates over the calls in the list and assigns the lowest possible frequencies to the calls without violating interference constraints [40]. 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 [40]. Due to 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.

The algorithm also makes extensive use of knowledge gained by the various particles as they search the problem space. Depending on the values used for w, c_1 and c_2, a particle does not only keep personal history (with pbest), but the swarm as a whole keeps a history of the best particle (neighbourhood best). 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 research the PSO was applied to the FS-FAP on the COST 259 benchmark problems. 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 which are mentioned in section 4.8.3 page 51. In contrast with MS-FAP, the performance measure is explicitly the number of constraints violated not interference. Applying the PSO to FS-FAP has not been done before and considering the success of the PSO on other NP-Problems together with its successful application on the other variants of the FAP, the PSO seems like a good candidate to be applied to the FS-FAP.

6.6 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 pseudo code 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 described in algorithm 7. 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

Discussion and results of implementing a PSO algorithm on the FAP

Chapter 7

Applying the PSO to the FAP

7.1 Introduction

PSO, as discussed previously (see page 126), 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 6.5.2 on page 131).

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. This constant dimensionality introduces an intriguing problem if one wants to apply the PSO to an inherent multidimensional 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 the problem space.

As was discussed in section 6.5.4, applying the PSO to the FAP introduces a variety of challenges. One of the challenges is how exactly one moves 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.

As mentioned in the previous chapter the FAP is a discrete valued problem and for the PSO to operate in the FAP space custom operators (called velocity functions in this discussion) had to be developed to enable the particles of the swarm to move. These velocity functions are discussed 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

This section presents a description on 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 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 4.8. For cellular networks, the frequency plan outlines which frequency 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 frequencies that

can be used or the number of frequencies available for assignment is more than the number of transceivers in the network.

The reality is that there are only a finite number of frequencies available for cellphone transmissions, as was discussed in chapters 3 and 4. 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 frequency it wants, it is bound to interfere with someone else also utilising the same frequency.

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 4.8) for instance Siemens 1, the allotted spectrum is from frequency 16 to 90. Which gives the network operator 74 frequencies 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 4.5. As discussed in chapters 3 and 4, 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. 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 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

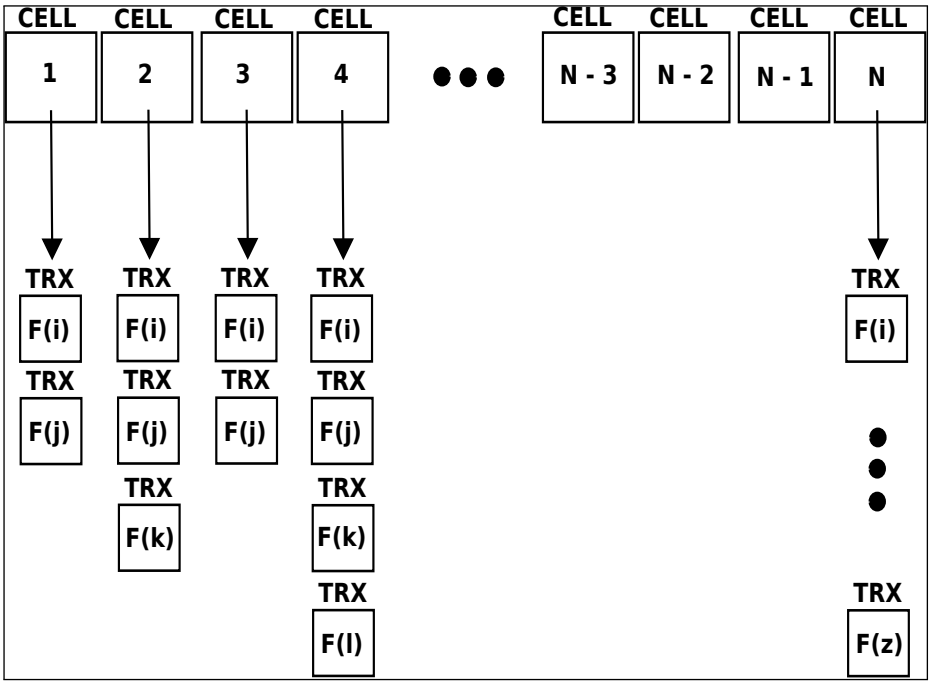

Figure 7.1: The structure of a frequency plan

networks have a large number of cells that range from 500 to more than 1000.

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 frequency 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 amount of interference that is experienced. 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 4.5. 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.

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 10 and 11.

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 8 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.

Algorithm 8 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
```

The position is purely random in that the only considerations made by the position generator are that valid frequencies are assigned to transceivers installed at cells. Thus the generator does not check whether a frequency 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.

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 is provided in section 7.4. The next section presents an explanation 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 4.5) 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 frequency 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 frequency of a transceiver, in one cell, differs by 1 from another frequency 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 particles in the developed PSO share information based on the star social network structure. 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 50 for the formal equation) and algorithm 9 is the pseudo code of the implemented equation used by the PSO .

Algorithm 9 FAP Cost Function

Require: normalCell
Require: interferingCell
1: $totalInterference \leftarrow 0$
2: **for** Each TRX trx_i in interferingCell **do**
3: **for** Each TRX trx_j in normalCell **do**
4: $interference \leftarrow 0$
5: $difference \leftarrow |trx_i - trx_j|$
6: **if** difference is 0 **then**
7: **if** *coChannelInterference* $\leq$ *minInterferenceThershold* **then**
8: $interference \leftarrow interference + 0$
9: **else**
10: *interference* $\leftarrow$ *interference* + *coChannelInterference*
11: **end if**
12: **else**
13: **if** difference is 1 **then**
14: **if** *adjChannelInterference* $\leq$ *minInterferenceThershold* **then**
15: *interference* $\leftarrow$ *interference* + 0
16: **else**
17: *interference* $\leftarrow$ *interference* + *adjChannelInterference*
18: **end if**
19: **end if**
20: **end if**
21: *totalInterference* $\leftarrow$ *totalInterference* + *interference*
22: **end for**
23: **end for**

As can be seen in algorithm 9 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 notice-

able impact on the communication link. The following section discusses 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 velocity 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 6.5.2.

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. 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 frequencies that each individual transceiver installed at a cell must use for communication. Thus a frequency plan can be 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 frequency. This frequency is used for communication.

As discussed previously, a position is a frequency plan and each particle has a current position and a best position. A visual depiction of a particle position is presented in figure 7.2. The global best particle is also depicted in the figure.In the figure, position 1 represents a particle. The current compartment in the particle represents the current position and hence a frequency plan. The best compartment represents the best position found and hence the best frequency plan found thus far by the particle. When providing examples on how any of the following velocity methods operate, figure 7.2 is used as reference point.

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 frequency allocated to a *transceiver* in a particular *cell* of one frequency plan to another frequency 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 frequencies that 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 frequencies each individual transceiver uses for communication differ.

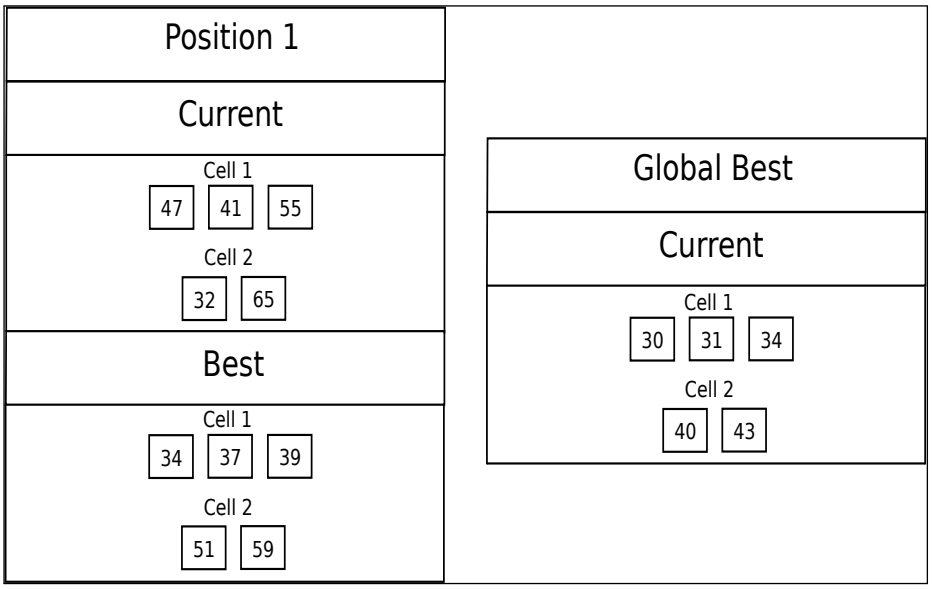

Figure 7.2: FAP PSO Particle Position and Global Best Position

Using this realisation, the conclusion can be made that the velocity equation can only work with the frequencies assigned to transceivers. Therefore a potential velocity equation mechanism needs to operate on the finest granularity of a frequency plan, i.e. the frequencies.

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 frequencies. 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 6.5.2 can be broken up into the following parts:

- **Subtraction**
  SubtractionResultPbest: $pbest - x_i(t)$

SubtractionResultGbest: $gbest - x_i(t)$

- **Multiplication**
  MultiPbestResult: $c_1\phi_1 * SubtractionResultPbest$
  MultiGBestResult: $c_2\phi_2 * SubtractionResultGbest$

- **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 frequencies. These frequencies 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.

The first velocity method is listed in algorithm 10. 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 frequencies that have been allocated to each transceiver within a cell differ. Thus in velocity method 1, each cell has an array of transceivers. The array of transceivers contains the individual frequency numbers that have been allocated to a cell.

Before the algorithm is presented, the operations used for addition, subtraction, multiplication and multiplication with a scalar needs to be defined. The following equations formulate all of these operations and are used by Velocity method 1.

$$\Delta c_{ij} = f(c^1_{ij}) + f(c^2_{ij}) \tag{7.1}$$

$$\Delta c_{ij} = f(c^1_{ij}) - f(c^2_{ij}) \tag{7.2}$$

$$\Delta c_{ij} = f(c^1_{ij}) * f(c^2_{ij}) \tag{7.3}$$

$$\Delta c_{ij} = f(c_{ij}) * s \tag{7.4}$$

where, $c \in \{c_{00}, c_{01}, c_{10}, \ldots, c_{ij}\}, \forall i, j, s \in \mathbb{N}, i \leq \text{MaxCells}, j \leq t(c_i)$

$$p_i \in \{\{c_{01}, c_{02}, \ldots, c_{ij}\}_1, \{c_{01}, c_{02}, \ldots, c_{ij}\}_2, \ldots, \{c_{01}, c_{02}, \ldots, c_{ij}\}_k\}$$

In the above equations c^1 and c^2 represent different frequencies plan positions. The function f retrieves the frequency assigned to a TRX c_{ij}. MaxCells is the maximum amount of cells in the frequency plan. The function t retrieves the total amount of TRXs installed at a particular cell c_i. The variable s in equation 7.4 is any scalar value. The variable p_i represents a particle in the swarm and k is the max swarm size. Now that all the equations have been defined, the algorithm can be represented.

Velocity method 1 is depicted in algorithm 10. The algorithm moves one array of transceivers to another array of transceivers. The variable r in algorithm 10 below is a random scalar value. In each of the equations the particular arithmetic operation is applied to the same TRX found in both frequency plans. 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, and utilises equation 7.2.

Algorithm 10 Velocity Method 1

Require: currentParticle – The particle that needs to move (*fromPosition*)
Require: globalBestParticle – The particle to move towards (*toPosition*)

1: $pbest \leftarrow$Particle best position
2: $gbest \leftarrow$globalBestParticle position
3: $pBestSubtractResult \leftarrow currentParticle - pbest$ with equation 7.2
4: $gBestSubtractResult \leftarrow currentParticle - gbest$ with equation 7.2
5: $a \leftarrow localCoeff \times pBestSubtractResult \times r$ using equation 7.3 and 7.4
6: $b \leftarrow globalCoeff \times gBestSubtractResult \times r$ using equation 7.3 and 7.4
7: **if** first time velocity is calculated for current Particle **then**
8: $v \leftarrow a + b$ using equation 7.1
9: **else**
10: $abAdditionResult \leftarrow a + b$ using equation 7.4
11: $v' \leftarrow abAdditionResult + v$ using equation 7.1
12: $v \leftarrow w \times v'$ using equation 7.4
13: **end if**
14: $currentParticle \leftarrow v + currentParticle$ using equation 7.1
15: SanatizePosition(currentParticle)

To better understand how velocity method 1 operates an example will now be presented using figure 7.2 as the example position and example global best. Note that in following example, only $cell_1$ is considered for the current,

personal and global best position.

$$
\begin{aligned}
pbest_1 &= \{34, 37, 39\} \\
gbest_1 &= \{30, 31, 34\} \\
cell_1 &= \{47, 41, 55\} \\
pbestSubtractResult_{11} &= cell_{11} - pbest_{11} \\
&= 47 - 34 \\
gbestSubtractResult_{11} &= cell_{11} - pbest_{11} \\
&= 47 - 30 \\
pbestSubtractResult_{12} &= cell_{12} - pbest_{12} \\
&= 41 - 37 \\
gbestSubtractResult_{12} &= cell_{12} - pbest_{12} \\
&= 41 - 31 \\
pbestSubtractResult_{13} &= cell_{13} - pbest_{13} \\
&= 55 - 39 \\
gbestSubtractResult_{13} &= cell_{13} - pbest_{13} \\
&= 55 - 34 \\
r &= 0.725 \\
pbestSubstractResult &= \{13, 4, 16\} \\
a &= 0.5 \times pbestSubtractresult \times r \\
&= \{4, 1, 5\} \\
gbestSubtractResult &= \{17, 10, 21\} \\
r &= 0.654 \\
b &= 0.4 \times gBestSubtractResult \times r \\
&= \{4, 2, 5\} \\
abAdditionResult &= \{4, 1, 5\} + \{4, 2, 5\} \\
&= \{8, 3, 10\} \\
newVelocity &= abAdditionResult \times 0.5 \\
&= \{4, 1, 5\}
\end{aligned}
$$

$$\begin{aligned} cell_1 &= cell_1 + newVelocity \\ &= \{47, 41, 55\} + \{4, 1, 5\} \\ &= \{51, 42, 60\} \end{aligned} \tag{7.5}$$

In the following discussion the *fromPosition* is the position of the currentParticle. The *toPosition* is the position the currentParticle must move towards. The toPosition in this discussion is the position of the *globalBestParticle*. The algorithm operates on every cell in the frequency plan. The algorithm starts off by selecting the first cell in the fromPosition and the same for the toPosition.

Once the cell has been selected the algorithm obtains the exact same transceiver in the toPosition frequency plan. This operation is quick, as the two plans are identical except for the frequencies 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. After all the cells in the frequency plan has been processed the algorithm proceeds to the next phase.

After the subtraction, a position is returned which is the result of subtracting the fromPosition from the toPosition. Subtraction of two frequency plan positions occurs on lines 4 – 5. Using the subtraction result the velocity method 1 algorithm is now ready to apply the next operation in the velocity update equation, namely multiplication by using equations 7.3 and 7.4 which are applied on lines 5 – 6.

In the multiplication stage of the velocity update operation, the local and global coefficients defined for the PSO are multiplied into a position, i.e. frequency plan. The coefficient are scalar values and therefore a different method needs to be used to multiply a scalar value with a frequency plan. Equation 7.4 is used to multiply scalar values with frequency plans. Multiplying a frequency plan results in another frequency plan. Scalar values need to be multiplied in for the following cases in the velocity update.

- The inertia case: $w \times v(t+1)$, where w is the inertia value and $v(t+1)$ is velocity that has already been calculated for a particular particle t. Note, a velocity that has been calculated is still a frequency plan. In the algorithm, inertia is applied in line 12.

- Standard velocity calculation randomisation case: As can be observed from the velocity equation 6.9 and also from the multiplication bullet point on page 152, 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. Frequencies are integers so the loss of the fractional component is warranted as it is of no use. Note that it is highly probable that the frequencies depicted in the plan are out of the defined spectrum boundary. As per the last bullet point on page 152 the last basic operation that occurs in the velocity equation is the addition of frequency plans. Note that the calculated velocity is still considered to be a frequency plan on its own.

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 6.10. The addition is performed by the algorithm in line 14. Since the addition operation is also used in the final position update of the particle, its last opportunity where frequencies can be bounded. The purpose of the bound operation is to keep the frequencies within valid value ranges. In equation 7.1 the frequencies are bounded using the function b and is defined in the following section.

7.4.2 Keeping Frequencies Bounded

The previous section discussed the first velocity method that was developed. 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 that are imposed inherently by the domain as well as the network for which a frequency plan is being created.

Due to the swarm not having a concept of the constraints a constraint handling mechanism needs to be used. Each of the methods discussed in this

section fall under the category of using the *repair method* which is discussed in section 4.3. The first method that was developed was the BoundValue method and is formulated in equation 7.6.

$$\Delta b(c_{ij}) = \begin{cases} F_{min} + f(c_{ij}) \bmod F_{max} & \text{, if } f(c_{ij}) \leq F_{max} \\ b(f(c_{ij}) + F_{min}) & \text{, if } f(c_{ij}) \leq F_{min} \end{cases} \tag{7.6}$$

Where F_{min} and F_{max} are the minimum and maximum frequencies of the defined spectrum. The function b is the BoundValue function. The constraints that are imposed are part of the spectrum that may under no circumstances be used anywhere in the network, which as discussed in chapter 4 is known as globally blocked frequencies. Some frequencies that are only allowed to be used in certain parts of the network are known as locally blocked frequencies. Then of course the swarm also needs to take into account the electromagnetic constraints.

The velocity function used by the PSO needs to be altered to make the swarm 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 frequency 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 frequencies be assigned and thus does not penalise invalid assignments.

To keep assigned frequency 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 frequency boundary check is only applied when one of the following conditions are met after the calculated velocity has been applied to the current position:

- If a frequency allocated to a transceiver is above the maximum allowable frequency (upper bound) given to the network.

- If a frequency allocated to a transceiver is below the minimum allowable frequency (lower bound) given to the network.

As can be seen from equation 7.6, 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 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.

$$\begin{aligned} 10 \bmod 50 &= 10 \\ 60 \bmod 50 &= 10 \\ 50 \bmod 50 &= 0 \\ 100 \bmod 50 &= 0 \\ 35 \bmod 50 &= 35 \end{aligned}$$

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 frequencies, the following occurs: If, for instance, the maximum allowable frequency is 50, and the transceiver has a frequency 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 frequency. In essence, the value is wrapped around to always be within acceptable range.

The difficult case is when the frequency value is lower than the minimum frequency given to the network. This is because modding the frequency value has no effect. For example, if the lowest allowable frequency 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 frequency. Then add the result to the minimum allowable frequency. The resultant value is checked again as to whether it oversteps the bounds of the maximum allowable frequency and is bounded accordingly.

2. Add the lower value to the minimum allowable frequency. The resultant value is checked as to whether it oversteps the bounds of the maximum allowable frequency and is bounded accordingly.

3. Repeatedly subtract the lower value from the maximum allowable frequency until the resultant frequency is within the acceptable frequency range.

An important notion to consider is that, based on the velocity equation, it is entirely within the realm of possibility that a frequency value after movement might contain a negative value. As can be seen in the second case of equation 7.6, if the assigned frequency is less than F_{min} then F_{min} is added to the frequency and the boundary function is applied again. Therefore, the boundary function is repeatedly called until the frequency to be bounded comes into a valid range.

Using this bound method to keep frequencies within the acceptable range of frequencies add unnecessary complexity. This complexity can be completely avoided by using frequency index values rather than actual frequency values for frequency plans. How indices are used instead of raw frequency values is discussed in the following section.

7.4.3 Using Indices instead of Frequencies

As discussed in section 7.4 and as can be observed from algorithm 10, the first velocity method that was developed for the PSO worked with raw frequency values. This is not ideal since upon closer inspection the frequency range that the swarm used to move around was indeed incorrect. The bound value function only keeps frequencies within a minimum and maximum allowable frequency range, but globally blocked frequencies and locally blocked frequencies can be in between this minimum and maximum frequency range.

With velocity method 1 i.e algorithm 10 and the PSO using raw frequency values, the swarm increasingly moved towards allocating these blocked frequencies to transceivers since the fitness function does not penalise the use of blocked or invalid frequency 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 frequencies are invalid and cannot be used. If a network were to use a plan that uses blocked frequencies, it could cause unexpected interference to other services and the governing body that controls the spectrum could fine the network. 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 frequencies or locally blocked frequencies.

2. Instead of letting the swarm work with raw frequency 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* frequencies. Thus the swarm then moves around in a range from 0 to F, where F is the size of the frequency 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.

The second method is also a method known in constraint handling as *preserving feasibility* (see section 4.3 for a discussion on constraint handling methods). By using only valid frequencies the search space is even more constricted to only feasible solutions.

Working with index values rather than with raw frequency values led to a second velocity method. Algorithm 11 is the pseudo code 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.

$$\Delta c_{ij} = f'(c^1_{ij}) + f'(c^2_{ij}) \tag{7.7}$$

$$\Delta c_{ij} = f'(c^1_{ij}) - f'(c^2_{ij}) \tag{7.8}$$

$$\Delta c_{ij} = f'(c^1_{ij}) * f'(c^2_{ij}) \tag{7.9}$$

$$\Delta c_{ij} = f'(c_{ij}) * s \tag{7.10}$$

where, $c \in \{c_{00}, c_{01}, c_{10}, \ldots, c_{ij}\}, \forall i, j, s \in \mathbb{N}, i \leq \text{MaxCells}, j \leq t(c_i)$

In the above equations, the function f' retrieves the index value for a particular TRX c_{ij}. The rest of the symbols represent the exact same as they were defined for in equations 7.1 - 7.3. The pseudo code of the second velocity method will now be presented.

Algorithm 11 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: $v'_{ij} \leftarrow \delta_1 \times r_1 \times (c_{ij}^{pbest} - c_{ij}) + \delta_2 \times r_2 \times (c_{ij}^{pbest} - c_{ij}^{gbest})$
6: **if** First time velocity is calculated **then**
7: $c_{ij} \leftarrow c_{ij} + v'_{ij}$
8: **else**
9: $v'_{ij} \leftarrow w \times v'_{ij}$
10: $c_{ij} \leftarrow c_{ij} + v'_{ij}$
11: **end if**
12: **end for**
13: SanatizePosition(currPos)

In the algorithm v' represents the newly calculated velocity and w the inertia. The variables c^{pbest} and c^{gbest} each represent the personal best position and the global best position for a particle. The variable r is a random scalar value where $r \in \mathbb{N}$. Variables δ_1 and δ_2 represent the local and global coefficients.

Velocity method 2 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.

The algorithm applies the standard velocity equation 6.9 formulated in chapter 6 as is to the raw indices.

An important note, is that if one analyses velocity method 1, the switch to using index values instead of raw frequencies does not affect the method's operation. Since the method only requires explicit knowledge on the data

to operate on in the BoundValue function, 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 frequencies. The BoundValue function needs to be updated since it is the primary means by which velocity method 1 ensures valid positions.

As with velocity method 1, a more practical example will now be presented using figure 7.2 as reference point. The example will aid in understanding the operation of velocity method 2 better. Note that with each variable the subscript 1 indicates that the values for cell 1 are used.

$$\begin{aligned}
currPos_1 &= \{47, 41, 55\} \\
pbestPos_1 &= \{34, 37, 39\} \\
gbestPos_1 &= \{30, 31, 34\} \\
r_1 &= 0.435 \\
r_2 &= 0.288 \\
newVelocity_{11} &= 0.5 \times r_1 \times (34 - 47) + 0.4 \times r_2 \times (34 - 30) \\
newVelocity_{12} &= 0.5 \times r_1 \times (37 - 41) + 0.4 \times r_2 \times (37 - 31) \\
newVelocity_{13} &= 0.5 \times r_1 \times (39 - 50) + 0.4 \times r_2 \times (39 - 34) \\
w &= 0.5 \\
newVelocity &= |0.5 \times newVelocity_1| \\
currPos_1 &= currPos_1 + newVelocity_1
\end{aligned} \tag{7.11}$$

Both the velocity methods that are utilised by the developed FAP PSO algorithm have now been discussed. Why the developed PSO was modified to operate on frequency index values rather than frequency values was also discussed. The PSO is 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 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.

Algorithm 12 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 forreturn** 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 12.

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 12 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 5 and 6, some of the algorithms had a problem with regard to some cells or even transceivers overshadowing better cells or transceivers. In this research, 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 frequencies assigned to their respective transceivers, and other might have the best. Now the few cells with the worst frequencies generate a great deal of interference, whereas the cells with the best frequencies 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 frequencies 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 frequency 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 frequency 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 frequencies allocated to its installed transceivers.

2. Besides the particle storing the total fitness, it also needed to store the interference generated by a frequency 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 12, 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 frequency allocated to a particular transceiver and also compare the allocation with other similar transceivers in other frequency plans.

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 13 uses interference information of cells to build a gbest. Algorithm 14 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 13.

Algorithm 13 Building Global Best with Cells

Require: gbest

Require: swarm

1: **for** Each particle p_i in swarm **do**

$$gbest_i = \begin{cases} gbest_i, \text{if } cost(gbest_i) \geq cost(c_i) \\ c_i, \text{if } cost(gbest_i) \leq cost(c_i) \end{cases} \tag{7.12}$$

2: **end for**

Algorithm 14 Building Global Best with Transceivers

Require: gbest

Require: swarm

1: **for** Each particle p_i in swarm **do**

$$gbest_{ij} = \begin{cases} gbest_{ij}, \text{if } cost(gbest_{ij}) \geq cost(c_{ij}) \\ c_{ij}, \text{if } cost(gbest_{ij}) \leq cost(c_{ij}) \end{cases} \tag{7.13}$$

2: **end for**

In the above algorithms, the cost function utilises equation 4.11 to determine the interference. Each algorithm will now be discussed since the difference between them is subtle. Algorithm 13 was the first global best building scheme which was developed and is discussed first.

In algorithm 13, the interference generated by a cell i is retrieved for the current best particle in the swarm and is represented by $gbest_i$ and for the same cell in the current particle c_i under consideration by algorithm. 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.

Algorithm 14 is a more finer grained algorithm than algorithm 13. After analysing the algorithm using cells, it was concluded that it is possible that a single bad frequency allocation to a transceiver within a cell can overshadow other potentially good frequency allocations to other cells within the cell. Algorithm 14 obtains both transceivers to determine the interference (cost) of their respective frequency allocations generated. The TRX in the global best particle is represented by $gbest_{ij}$ where i is the cell and j is the TRX within cell i. Similarly, c_{ij} represents the TRX j within cell i. For both TRXs the interference generated by their frequency allocations is given by applying the *cost* function as can be seen in algorithm 14. Using the cost values the algorithm determines whether the current transceiver frequency allocation generates less interference than the frequency allocated to the same transceiver in the global best frequency plan.

If the current transceiver frequency generates less interference, the algorithm then proceeds to replace the transceiver frequency in the global best with the current transceiver frequency. Thus it can be seen that algorithm 14 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 becomes progressively worse as the swarm progresses 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. By building a gbest the algorithm resembles the ACO algorithm in a sense that a solution is progressively built.

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 frequency 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 are incorporated by adding to each cell a list, which keeps track of each frequency 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 10 and 11 the last step that occurs in both algorithms is that the SanitizePosition method is called.

The SanitizePosition method is listed in algorithm 15. 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.

Algorithm 15 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**

In the FAP PSO the tabu list check works slightly differently from what one would expect. As can be observed in algorithm 15, 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 16.

Algorithm 16 ResolveCollision

Require: cell
Require: tabuList
1: EnsureUniqueFrequenciesWithinCell(cell)
2: **for** Each trx_i in cell **do**
3: **while** trx_i exists in TabuList **do**
4: trx_i = Generate random frequency
5: **if** Collision not resolved after 20 attempts **then**
6: Break out of while loop
7: **end if**
8: **end while**
9: **end for**

As can be observed in algorithm 16, when a frequency value is found to exist either in the tabu list or assigned to a different transceiver within the same cell, a collision is said to occur. In the case where the same frequency is assigned to two different transceivers within the same cell, the method randomly assigned a new frequency to the offending transceiver. The randomly assigned frequency is also subject to check whether it is Tabu. In the FAP PSO a collision means that the specific frequency 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 frequency that can be assigned to the transceiver as can be seen to occur within the while-loop in lines 3 – 4.

The algorithm generates a new random frequency 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 frequency. As long as a collision occurs the algorithm will continually generate a new random frequency until it has attempted 20 random frequencies with no frequency colliding.

After 20 attempts the algorithm just accepts the last generated frequency as the new frequency. The maximum number of 20 attempts was selected through testing and can be increased at the expense of more computational time.

In addition to checking if a frequency is tabu, resolve collisions also handles the case where if a transceiver within a cell has the same frequency

as a different transceiver within the same cell. This process occurs on line

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 frequency assignments to transceivers tabu the algorithm is forced to try new frequency 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 frequencies 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 frequency 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[1].

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. 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 pseudo code 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 pseudo code. The chapter concluded with small additions made to the algorithm to improve performance and, most important of all, improve solution quality.

[1]This is the default behaviour of the .Net 4.0 random number generator

Chapter 8

Results

8.1 Introduction

The previous chapter presented a discussion on the FAP PSO algorithm . 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 6 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 frequencies 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.

The previous chapter discussed all the modifications that were made to the standard PSO. The modifications were made to enable 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 in the previous chapter was benchmarked against the COST 259 benchmark Siemens instances.

This chapter presents the results 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 affect the PSO performance as well as how the global selection schemes have an influence on the final results.

8.2 PSO COST 259 siemens Results

The FAP PSO is applied to the benchmarks on a machine with the following specifications and frameworks:

- 4 GB RAM
- Windows 7
- Intel Quad Core CPU
- C# using .Net 4 Framework with Parallel Extensions

As discussed, the FAP PSO is applied to Siemens 1, Siemens 2, Siemens 3 and Siemens 4 of the COST 259 benchmark suite. For more information about the nature of the benchmarks, the reader is directed to section 4.8.3.

For each benchmark, 12 results are presented. The following changes were made to the FAP PSO to obtain 12 different results, one for each variant of the algorithm.

- The two velocity methods that were developed for the PSO are tested.
- Each velocity method is also paired with one of the three global selection schemes namely, standard (traditional PSO global best selection), building Gbest with Cells and building a Gbest with TRXs.

The following values are used for the FAP PSO algorithm regardless of the variant being benchmarked. These values were chosen after a series of trail runs showed that these values produce the best results.

- The swarm size was set to 20, 50 and 100.
- Inertia was set to 0.5.
- Cognitive coefficient was set to 0.4.

- Social coefficient was set to 0.5.
- The stopping criteria was set to max iterations 50.

The velocity methods and global selection schemes were discussed in the previous chapter. To obtain the best representation for the quality of results produced and performance of the different methods used by the algorithm, the benchmark is executed 20 times for every variant of the algorithm and different population size.

The developed variants of the FAP PSO are compared against the algorithms that have obtained the best published results on the COST 259 benchmarks as discussed in chapter 5. The sections these algorithms were discussed in are as follows: **Dynamic Tabu** section 5.4.4, **k-thin FAP** section 5.5.4 and **GA** 5.6.4. Note the name this dissertation will use to refer to these algorithms in this chapter, is in bold face font.

Each section in this chapter presents results on a COST 259 benchmark instance and uses the following layout.

- A table with overall results for velocity method 1 and 2. Each table presents the global best selection, population, total interference, average, standard deviation and variance. The average, standard deviation and variance values depicted in this table, were calculated based on the minimum interference achieved across the 20 different algorithm runs.
- Within each table the lowest interference obtained is indicated in **bold**.
- A table indicating interference statistics from the overall best frequency plan. The table has the following columns:
  - **co-channel max** — The max co-channel interference generated
  - **co-channel avg** — The average co-channel interference generated
  - **co-channel std** — The standard deviation of the generated co-channel interference
  - **adj-channel max** — The max adjacent-channel interference generated

  - **adj-channel avg** — The average adjacent-channel interference generated
  - **adj-channel std** — the standard deviation of the generated adjacent interference
  - **TRX max** — The max interference generated by a TRX
  - **TRX avg** — The avg interference generated by a TRX
  - **TRX std** — The standard deviation generated by a TRX

- A table indicating the amount of TRX pairs exceeding a certain interference threshold for the overall best frequency plan.
- A line graph indicating the progress of the algorithm from iteration 1 to 50 as it obtained the best overall frequency plan for velocity method 1 and method 2.

After the results are presented a discussion will follow about the results and what they represent and reveal about the performance of the FAP PSO. The next section presents the results obtained for the benchmark instance Siemens 1.

8.2.1 Siemens 1

GBest selection	Population	Interference	Average	Std. Deviation	Variance
Standard	100	100.23	103.62	9.44	4.05
Standard	50	100.90	105.69	10.44	4.19
Standard	20	101.15	106.70	12.23	5.54
GBestFromCells	20	36.41	39.60	9.09	3.18
GBestFromCells	50	**35.19**	38.72	7.84	2.36
GBestFromCells	100	35.27	38.82	8.80	3.52
GBestFromTrxs	20	106.23	113.91	16.82	10.48
GBestFromTrxs	50	107.33	114.42	16.15	10.03
GBestFromTrxs	100	109.80	114.94	14.61	9.70
Dynamic Tabu	—	N/A	—	—	—
k-Thin SA	—	2.20	—	—	
GA	—	2.96	—	—	

Table 8.1: Overall Siemens 1 results with velocity method 1

GBest selection	Population	Interference	Average	Std. Deviation	Variance
Standard	20	378.53	784.02	794.81	23397.43
Standard	50	353.06	652.72	728.49	20411.46
Standard	100	317.16	511.48	613.45	17105.75
GBestFromCells	20	325.69	553.86	561.71	11685.80
GBestFromCells	50	161.25	322.53	373.15	5355.40
GBestFromCells	100	209.01	249.04	149.56	1016.78
GBestFromTrxs	20	193.83	577.81	1111.87	45787.46
GBestFromTrxs	50	149.19	347.40	885.59	30164.26
GBestFromTrxs	100	**142.19**	244.64	417.93	7939.47
Dynamic Tabu	—	N/A	—	—	
k-Thin SA	—	2.20	—	—	
GA	—	2.96	—	—	

Table 8.2: Overall Siemens 1 results with velocity method 2

Algorithm	co-channel			adj-channel			TRX		
	max	avg	std	max	avg	std	max	avg	std
Best FAP PSO	0.31	0.02	0.03	0.19	0.02	0.02	0.31	0.03	0.03
Dynamic Tabu	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
k-Thin SA	0.03	0.00	0.00	0.03	0.00	0.00	0.05	0.01	0.01
GA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 8.3: co-,adj-channel and TRX interference statistics for best Siemens 1 frequency plan

Algorithm	TRX pairs exceeding interference								
	0.01	0.02	0.03	0.04	0.05	0.10	0.15	0.20	0.50
Best FAP PSO	286	131	98	64	147	31	13	8	0
Dynamic Tabu	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
k-Thin SA	33	4	1	0	0	0	0	0	0
GA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 8.4: TRX pair interference breakdown for best Siemens 1 frequency plan

8.2.2 Algorithm run graph for Siemens 1

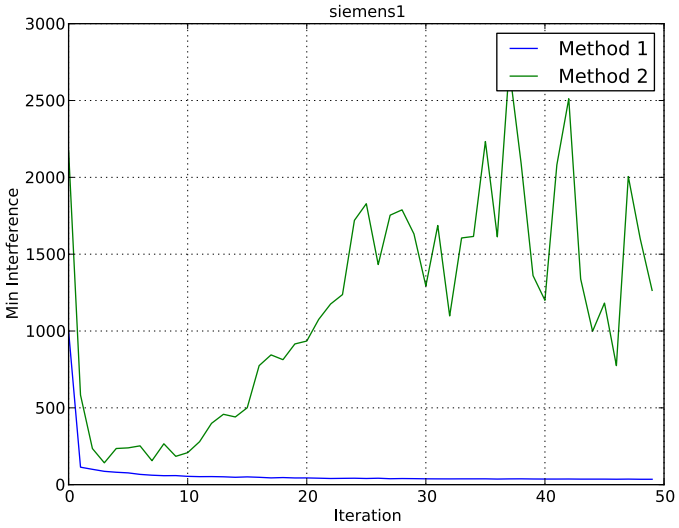

Figure 8.1: Algorithm run velocity method 1 versus method 2

8.2.3 Siemens 2

GBest selection	Population	Interference	Average	Std. Deviation	Variance
Standard	20	73.32	77.65	6.20	1.43
Standard	50	74.23	76.85	4.38	0.74
Standard	100	74.19	76.65	3.54	0.62
GBestFromCells	20	**52.63**	55.55	7.93	2.33
GBestFromCells	50	53.44	55.50	5.79	1.29
GBestFromCells	100	52.78	54.38	4.10	0.84
GBestFromTrxs	20	77.45	81.46	12.29	5.59
GBestFromTrxs	50	78.00	81.04	8.08	2.51
GBestFromTrxs	100	77.10	81.24	7.67	2.94
Dynamic Tabu	—	14.28	—	—	—
k-Thin SA	—	14.27	—	—	—
GA	—	17.83	—	—	—

Table 8.5: Overall Siemens 2 results with velocity method 1

GBest selection	Population	Interference	Average	Std. Deviation	Variance
Standard	20	174.21	206.56	82.50	252.09
Standard	50	161.40	207.63	90.62	315.83
Standard	100	158.31	199.62	77.74	302.17
GBestFromTrxs	20	106.79	121.19	43.35	69.61
GBestFromTrxs	50	**101.03**	112.28	36.08	50.07
GBestFromTrxs	100	102.06	108.32	16.95	14.36
GBestFromCells	20	250.15	461.26	524.44	10186.49
GBestFromCells	50	245.19	344.08	301.00	3484.60
GBestFromCells	100	173.36	255.28	210.97	2225.35
Dynamic Tabu	—	14.28	—	—	—
k-Thin SA	—	14.27	—	—	—
GA	—	17.83	—	—	—

Table 8.6: Overall Siemens 2 results with velocity method 2

Algorithm	co-channel			adj-channel			TRX		
	max	avg	std	max	avg	std	max	avg	std
Best FAP PSO	0.23	0.01	0.02	0.02	0.00	0.00	0.23	0.03	0.04
Dynamic Tabu	0.11	0.01	0.01	0.02	0.00	0.00	0.20	0.03	0.03
k-Thin SA	0.07	0.01	0.01	0.02	0.00	0.00	0.16	0.03	0.03
GA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 8.7: co-,adj-channel and TRX interference statistics for best Siemens 2 frequency plan

Algorithm	TRX pairs exceeding interference								
	0.01	0.02	0.03	0.04	0.05	0.10	0.15	0.20	0.50
Best FAP PSO	589	208	1400	92	193	33	8	3	0
Dynamic Tabu	343	89	24	18	9	1	0	0	0
k-Thin SA	359	71	27	17	13	0	0	0	0
GA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 8.8: TRX pair interference breakdown for best Siemens 2 frequency plan

8.2.4 Algorithm run graph for Siemens 2

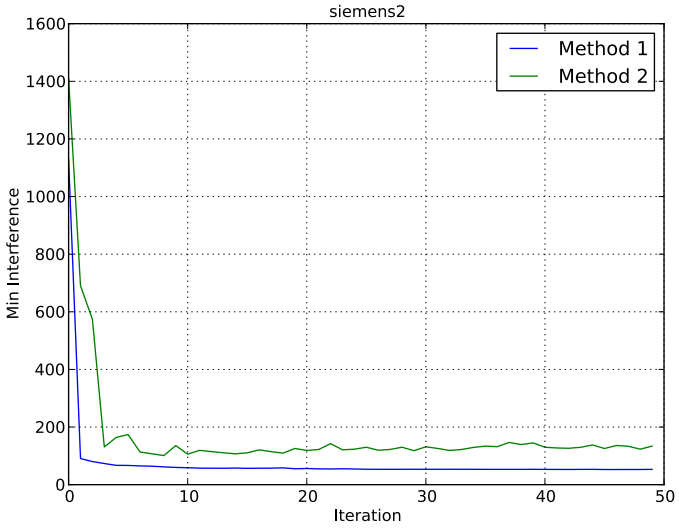

Figure 8.2: Algorithm run velocity method 1 versus method 2

8.2.5 Siemens 3

GBest selection	Population	Interference	Average	Std. Deviation	Variance
Standard	20	133.47	139.18	11.77	5.54
Standard	50	132.81	136.65	9.67	4.07
Standard	100	131.79	135.43	8.06	3.61
GBestFromCells	20	**44.64**	48.37	8.06	2.60
GBestFromCells	50	46.50	48.79	5.08	1.12
GBestFromCells	100	45.37	48.75	7.11	2.81
GBestFromTrxs	20	132.93	145.69	25.55	26.12
GBestFromTrxs	50	136.42	144.03	19.09	15.84
GBestFromTrxs	100	136.86	145.75	17.21	16.45
Dynamic Tabu	—	5.19	—	—	—
k-Thin SA	—	4.73	—	—	—
GA	—	6.08	—	—	—

Table 8.9: Overall Siemens 3 results with velocity method 1

GBest selection	Population	Interference	Average	Std. Deviation	Variance
Standard	20	834.88	1203.31	1037.77	43078.83
Standard	50	488.22	963.06	1439.96	90151.59
Standard	100	360.76	701.69	1064.67	62973.79
GBestFromCells	20	413.96	571.54	535.42	11467.16
GBestFromCells	50	269.23	392.32	319.11	4427.39
GBestFromCells	100	221.26	315.09	203.02	2289.75
GBestFromTrxs	20	229.05	619.35	1365.12	74541.80
GBestFromTrxs	50	222.19	370.61	646.37	18165.03
GBestFromTrxs	100	**210.01**	277.58	350.43	6822.20
Dynamic Tabu	—	5.19	—	—	—
k-Thin SA	—	4.73	—	—	—
GA	—	6.08	—	—	—

Table 8.10: Overall Siemens 3 results with velocity method 2

Algorithm	co-channel			adj-channel			TRX		
	max	avg	std	max	avg	std	max	avg	std
Best FAP PSO	0.22	0.01	0.02	0.14	0.00	0.01	0.22	0.03	0.02
Dynamic Tabu	0.04	0.00	0.00	0.03	0.00	0.00	0.07	0.01	0.01
k-Thin SA	0.03	0.00	0.00	0.02	0.00	0.00	0.08	0.01	0.01
GA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 8.11: co-,adj-channel and TRX interference statistics for best Siemens 3 frequency plan

Algorithm	TRX pairs exceeding interference								
	0.01	0.02	0.03	0.04	0.05	0.10	0.15	0.20	0.50
Best FAP PSO	494	228	150	91	146	27	6	4	0
Dynamic Tabu	88	14	3	0	0	0	0	0	0
k-Thin SA	80	6	4	0	0	0	0	0	0
GA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 8.12: TRX pair interference breakdown for best Siemens 3 frequency plan

8.2.6 Algorithm run graph for Siemens 3

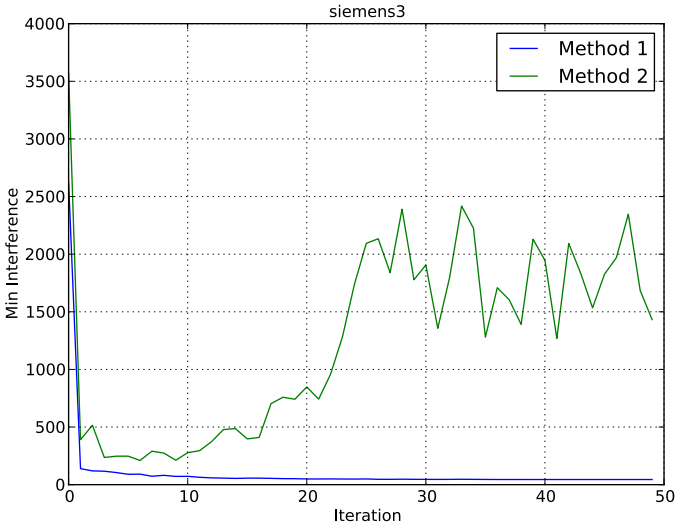

Figure 8.3: Algorithm run velocity method 1 versus method 2

8.2.7 Siemens 4

GBest selection	Population	Interference	Average	Std. Deviation	Variance
Standard	20	513.86	527.40	28.30	30.81
Standard	50	507.81	521.39	25.28	29.05
Standard	100	512.97	517.03	9.02	10.17
GBestFromCells	20	284.60	291.73	26.44	26.89
GBestFromCells	50	**277.36**	288.84	22.52	23.06
GBestFromCells	100	282.36	288.14	15.41	33.94
GBestFromTrxs	20	526.47	535.88	31.79	38.87
GBestFromTrxs	50	518.61	533.59	35.25	56.49
GBestFromTrxs	100	523.24	533.67	13.59	23.07
Dynamic Tabu	—	81.88	—	—	—
k-Thin SA	—	77.25	—	—	—
GA	—	96.84	—	—	—

Table 8.13: Overall Siemens 4 results with velocity method 1

GBest selection	Population	Interference	Average	Std. Deviation	Variance
Standard	20	1482.01	2184.35	1232.51	58426.63
Standard	50	1646.26	2058.09	987.90	44361.28
Standard	100	1746.22	2036.60	366.09	16752.78
GBestFromCells	20	1176.55	1572.82	1208.47	56169.28
GBestFromCells	50	862.67	1078.10	702.90	22457.42
GBestFromCells	100	834.08	1007.29	286.25	10242.22
GBestFromTrxs	20	**826.82**	1761.64	2234.28	192000.61
GBestFromTrxs	50	942.28	1261.93	903.42	37098.69
GBestFromTrxs	100	844.45	1092.90	363.64	16529.03
Dynamic Tabu	—	81.88	—	—	—
k-Thin SA	—	77.25	—	—	—
GA	—	96.84	—	—	—

Table 8.14: Overall Siemens 4 results with velocity method 2

Algorithm	co-channel			adj-channel			TRX		
	max	avg	std	max	avg	std	max	avg	std
Best FAP PSO	0.57	0.01	0.03	0.08	0.00	0.00	0.57	0.08	0.07
Dynamic Tabu	0.20	0.01	0.01	0.05	0.00	0.00	0.43	0.06	0.05
k-Thin SA	0.19	0.01	0.01	0.05	0.00	0.00	0.36	0.06	0.05
GA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 8.15: co-,adj-channel and TRX interference statistics for best Siemens 4 frequency plan

Algorithm	TRX pairs exceeding interference								
	0.01	0.02	0.03	0.04	0.05	0.10	0.15	0.20	0.50
Best FAP PSO	1970	889	452	286	666	253	127	247	5
Tabu	2161	971	547	344	209	12	2	0	0
SA	2053	871	445	282	163	11	2	0	0
GA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Table 8.16: TRX pair interference breakdown for best Siemens 4 frequency plan

8.2.8 Algorithm run graph for Siemens 4

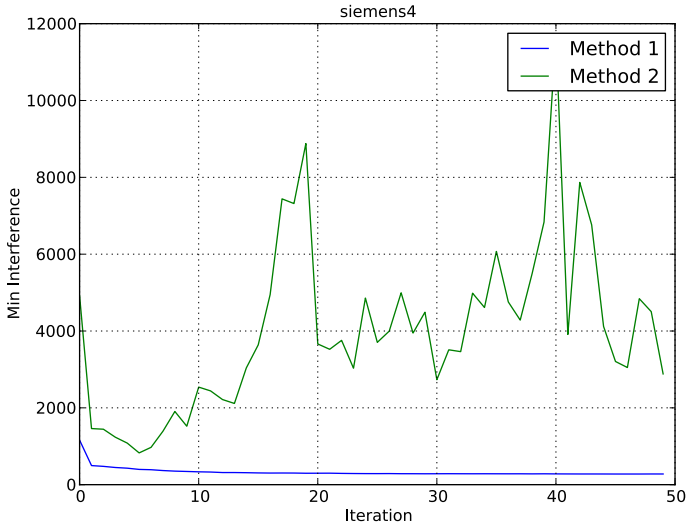

Figure 8.4: Algorithm run velocity method 1 versus method 2

8.2.9 Comparison with best results in COST 259

Below the best results achieved by the FAP PSO are compared to the best results obtained with each Siemens problem as published on the FAP website [63].

Algorithm	Siemens 1	Siemens 2	Siemens 3	Siemens 4
FAP PSO	35.19	52.63	44.64	277.36
k-thin FAP	**2.20**	**14.27**	**4.73**	**77.25**
Dynamic Tabu Search	N/A	14.28	5.19	81.88
GA	2.96	17.83	6.08	96.84

Table 8.17: FAP PSO results compared to best obtained results

All the results and relevant statistics have now been presented. The section that follows, presents a discussion and analysis of the results.

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 are discussed. First, the effect of the two developed velocity methods in terms of the results are described.

In section 8.3.2 the three global schemes that are used by the algorithm are discussed. This section concludes with a discussion on the interference statistics and what it means for the peformance of the FAP PSO.

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 problem 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 supposed to be operated on. Thus when method 2 moves a particle the frequency plan is moved piece by piece to a destination in the problem space. How method 2 accomplishes this movement was discussed in chapter 7 section 7.4.3.

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 that 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 minor check that can be done to ensure that all the frequency 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 frequency 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 frequency allocations at each stage of the velocity equation.

Taking into accounts how the two methods move particles around the solution space as well as the calculated standard deviation and variance a defining characteristic can be derived for method 1 and for method 2. Based on the standard deviation and variance, velocity method 1 is consistent and steadily moves to better positions. Velocity method 2 has high deviation and variance, which means that the algorithm explores the search space a lot more and is less directed towards a particular section of the solution space where good solutions might be.

Velocity method 1 has the key characteristic of performing an intensification search which is similar to the intensification phase in TS discussed in 5. In contrast velocity method 2 has the key characteristic of diversifying and exploration, similar to the diversification phase of the TS discussed in chapter 5. These characteristics become even more evident when analysing each graph presented. In all the graphs, velocity method 1 starts of at a initial point and then steadily moves downward to better positions. Where as, velocity method 2 is very sporadic, moving up and down.

Except in siemens 2 where the algorithm was a lot less sporadic considering how sporadic the search is when compared to other instances of the method operating on benchmark instances. When only comparing the search process of velocity method 2 to 1 on the graph and taking into account the variance as well as the standard deviation, the characteristic of method 2 still holds. The search seems less sporadic, but based on the variance, which is a lot more than method ones variance, the algorithm with

method 2 explores a lot more comparatively.

Based on the graphs, velocity method 2 can also perform the function of a disruptor and break the algorithm out of local minima.

8.3.2 Different Global Schemes

The previous chapter identified and discussed three global selection schemes. 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 frequency allocations due to overshadowing of frequencies[1]. Even with overshadowing the standard global scheme does not produce bad results, which seems to indicate that overshadowing of frequency 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 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 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 section 7.5 the scheme only considers the interference generated by a single frequency 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 siemens 4 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 frequencies allocated to transceivers, the scheme is discarding all other possible interferences.

[1] Overshadowing is discussed in chapter 7

It might select a frequency 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 frequency is too close on the spectrum to another frequency 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 frequency 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 frequencies 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 frequencies that have been allocated to this particular cell are the best combination that 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 The average, standard deviation and variance

For each of the presented algorithm variants that are presented the average, standard deviation and variance was calculated. As mentioned in the introduction of this chapter, each algorithm variant was executed 20 times. The statistics presented are thus not based on a single run but on this set of 20 results for each variant. The minimum achieved is the lowest minimum among the 20 result set.

By analysing the statistics presented for each variant of the algorithm it can be concluded that by using velocity method 1, the swarm search

process is more directed and focused. The deviation obtained while using velocity method 1 is low meaning that the gbest value of the swarm is near the average value of the gbest obtained across the swarm. Thus by using velocity method 1 the search process of the swarm is more focused and does not explore as much.

On the other hand, with the high deviation and high average and especially the high variance, it can be concluded that velocity method 2 pushes the swarm to explore the problem space a lot more. Even though the search space is explored more, with velocity method 2 the algorithm fails to intensify and converge on a good solution. This lack of convergence is evident if one analyses the algorithm progress graphs presented in fig. 8.1, 8.2, 8.3 and 8.4. Note that these graphs are based on a single best produced result out of all the produced results by all the variants of the algorithm. The graphs establishes the notion clearly that velocity method 1 is more focused and directed and velocity method 2 explores, disrupts or diversifies.

Unfortunately the algorithms to which the FAP PSO is compared against namely, Dynamic Tabu, k-Thin SA and GA only published their respective minimums interference for their generated frequency plans along with the plans interference statistics. Therefore, in the next section, the interference statistics for best generated frequency plans by the FAP PSO for each benchmark is discussed.

8.3.4 Interference statistics

In the results sections two tables were presented in addition to the table depicting the overall algorithm performance. As discussed in the introduction, each algorithm variant was executed 20 times. The two additional tables, depicts the interference statistics of the single best result out of all the produced different algorithm results.

Based on the co-channel interference statistics for all of the benchmarks, the FAP PSO is prone to assign frequencies to TRXs which induce co-channel interference. This can be seen due to the average and standard deviation being higher than the adjacent-channel interference.

Potential co-channel interference is difficult to detect because in the algorithm, at the time of frequency assignment, each cell is not aware with which other cells it interferes. It is possible that a cell could interfere with

all of the other cells in a network. If only a single frequency plan was being considered making the cell aware of the cells it interferes with would be optional but in the FAP PSO it isn't. Due to the FAP PSO using a population to explore a search space, making a cell aware would entail making it aware of cells in each and every single individual of the swarm. Even if one were to implement such awareness on a cell, it will complicate new frequency assignment and prolong processing by a huge amount since the whole populations cells need to be checked for co-channel violations.

It should also be noted, that the FAP PSO algorithm performs poorly in minimising the effect a co-channel assignment will have. This is evident since in all of the presented interference statistics, the max co-channel interference is also the highest TRX interference. Compared to the Dynamic Tabu and k-Thin the FAP PSO max co-channel violation is an order of magnitude more.

In contrast the adj-channel interference statistics compare favourably to the Dynamic Tabu and k-Thin averages and standard deviation. Again, as noted previously, the FAP PSO algorithm does not minimise the effect of a co- or adjacent channel interference. The max adjacent-channel interference is marginally higher than those reported by Dynamic Tabu and k-Thin.

When only considering the interference statistics of max, average and standard deviation, it is not clear as to why the FAP PSO algorithm generate frequency plans with such high interference when compared to the best frequency plans. Although when considering the interference statistics relating to the amount of TRXs exceeding a interference threshold, it becomes clear. The FAP PSO algorithm simply has far too many TRX pairs in the 0.10, 0.15, 0.20 and 0.50 thresholds. Comparatively the Dynamic Tabu and k-Thin typically have close to 10, but mostly 0 in these high interference thresholds. The FAP PSO on the other hand, has in just about every case above 100 TRX pairs that exceed the threshold.

The TRX interference statistics makes it abundantly clear where the FAP PSO needs to improve. For the algorithm to make any progress towards generated frequency plans with lower interference, the amount of TRXs that exceed the higher thresholds from 0.10 needs to be reduced drastically.

Further improvements that will require future work is discussed in the next chapter.

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 Siemens 1, Siemens 2, Siemens 3 and Siemens 4. These four benchmarks were discussed in detail in chapter 4. 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. For each benchmark three tables were presented. The first table presented the results for the overall performance of the different algorithm variants on the benchmark. The second and third table presented the interference statistics for the best generated frequency plan by the FAP PSO. Finally each section concluded with a graph depicting the performance of the best FAP PSO algorithm run for velocity method 1 and velocity method 2.

The chapter concluded with critical analysis of each of the different algorithms developed in this study to enable the PSO to operate in the FAP space as well as a discussion on the interferences statistics and what it reveals about the FAP PSO performance.

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.

Seven 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 and what components are involved when devices communicate in the network ?* 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 factors influence the quality of communication and what is interference ?* 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 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 5** — *With regard to particle swarms, how can a particle best be represented as a frequency plan ?* Chapter 6 described in detail how a frequency plan is represented as a particle in the algorithm.
- **Question 6** — *With regard to particle swarms, how can one frequency plan be moved towards another frequency plan ?* Chapter 6 also describes the two methods that were developed to facilitate moving one frequency plan to another.
- **Question 7** — *With regard to particle swarms, how can particles be prevented from using forbidden frequencies when they move towards a particular plan ?* Finally, chapter 6 also provided detail on exactly how when moving particles the algorithm avoids assigning invalid or forbidden frequencies.

9.3 Proving the hypothesis

In chapter 2, the hypothesis which this research aims to prove was stated. The hypothesis consisted of two key factors which needed to be proven.

The first factor, was to discern whether it is possible to apply the PSO to the FAP. Based on the results presented in chapter 8 it has been proven that it is indeed possible to apply the PSO to the FAP.

The second factor was to gauge the quality of the solutions the a PSO operating on the FAP produces. As discussed in chapter 8 the FAP PSO algorithm does not produce high quality results when compared to the best results obtained on the COST 259 benchmark. This is understanable due to applying the PSO with the specific FAP variant has never been attempted before. The basis for a PSO algorithm operating the FAP has now been created. Therefore, further refinement on the algorithm is now possible and allows for experimentation with other methods. Possible methods are discussed in section 9.5.

The next section discusses the difficulties that were faced during the design and implementation of the FAP PSO algorithm.

9.4 Difficulties faced

In the development of the PSO FAP algorithm a few challenges were faced. First and foremost there is no formal definition of the COST 259 fitness function. By just blindly summing all the interference values the calculated value will be wrong. One way to verify whether the interference calculation is corrected is to download one of the assignment plans that have been submitted. Each plan defines the total interference that will be generated by applying the plan.

The key in calculating the right interference value with the COST 259 benchmark is to notice that a minimum tolerable interference value is defined in each and every problem. Therefore, if the generated interference is either less than or equal to this minimum value, the interference is tolerable and can be safely ignored.

Another difficulty that was faced was improving the efficiency of the algorithm. By using the parallel extensions framework in .Net 4 the algorithm uses all cores available, but at the expense of potential expensive context switching. Context switching especially becomes expensive when large populations are used as values need to be moved in and out of a cores cache as the threads move from core to core based on how they are scheduled.

In this section an overview of the difficulties that were encountered during the development of the FAP were discussed. In the following section an overview of the potential future work based on this study is presented.

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 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.

Bibliography

[1] 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.

[2] 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.

[3] P. M. Adjakplé and B. Jaumard. Greedy and tabu search heuristics for channel block assignment in cellular systems. Technical Report G-97–45, École Polytechnique de Montréal, July 1997.

[4] 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.

[5] Mahmoud H. Alrefaei and Sigrún Andradóttir. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science*, 45:748–764, 1999.

[6] L. G. Anderson. A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system. *IEEE Transactions on Communications*, 21:1294–1301, 1973.

[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] M.B. Aryanezhad and Mohammad Hemati. A new genetic algorithm for solving nonconvex nonlinear programming problems. *Applied Mathematics and Computation*, 199(1):186 – 194, 2008.

[9] 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.

[10] 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.

[11] 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.

[12] D. Beckmann and U. Killat. Frequency planning with respect to interference minimization in cellular radio networks. Technical Report TD(99) 032, COST 259, Vienna, Austria, January 1999.

[13] 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.

[14] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353 – 373, 2005.

[15] 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.

[16] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin. Frequency assignment in cellular phone networks. *Annals of Operations Research*, (76):73–93, 1998.

[17] Ralf Borndörfer, Andreas Eisenblätter, Martin Grötschel, and Alexander Martin. The orientation model for frequency assignment problems. Technical report, Konrad-Zuse-Zentrum Berlin, 1998.

[18] A. Bouju, J.F. Boyce, Dr. A. Bouju, Dr. J. F. Boyce, J.G. Taylor, C.H.D. Dimitropoulos, G. Vom Scheidt, and Prof J. G. Taylor. Tabu search for the radio links frequency assignment problem. In *Proceedings of the International Conference on Digital Signal Processing*, 1995.

[19] 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.

[20] Erick Cantu-Paz. On random numbers and the performance of genetic algorithms, 2002.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[27] Christine Crisan and Heinz Mühlenbein. The breeder genetic algorithm for frequency assignment. In AgostonE. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature*, volume 1498 of *Lecture Notes in Computer Science*, pages 897–906. Springer Berlin Heidelberg, 1998.

[28] Monica Cuppini. A genetic algorithm for channel assignment problems. *European Transactions on Telecommunications*, 5(2):285–294, 1994.

[29] Miguel Crdenas-Montes, MiguelA. Vega-Rodrguez, and Antonio Gmez-Iglesias. Real-world problem for checking the sensitiveness of evolutionary algorithms to the choice of the random number generator. In Emilio Corchado, Vaclav Snasel, Ajith Abraham, Michal Wozniak, Manuel Grana, and Sung-Bae Cho, editors, *Hybrid Artificial Intelligent Systems*, volume 7208 of *Lecture Notes in Computer Science*, pages 385–396. Springer Berlin Heidelberg, 2012.

[30] 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.

[31] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243 – 278, 2005.

[32] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243 – 278, 2005.

[33] Marco Dorigo, Eric Bonabeau, and Guy Theraulaz. Ant algorithms and stigmergy. *Future Gener. Comput. Syst.*, 16(9):851–871, 2000.

[34] 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.

[35] 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.

[36] M. Duque-Anton, D. Kunz, and B. Ruber. Channel assignment for cellular radio using simulated annealing. *Vehicular Technology, IEEE Transactions on*, 42(1):14–21, 1993.

[37] A. Eisenblätter, M. Grötschel, and A. Martin. Frequency planning and rammifications of colouring. Technical report, Konrad-Zuse-Zentrum fu r Informationstechnik Berlin, 2000.

[38] Andreas Eisenblätter. Assigning frequencies in gsm networks. Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), 2001.

[39] Andreas Eisenblätter. *Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2001.

[40] 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.

[41] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006.

[42] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, 2007.

[43] Dr. Kamilo Feher. *Wireless Digital Communications: Modulation & Spread Sprectrum Applications*. Prentice Hall, 1995.

[44] 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.

[45] Philippe Galinier and Jin-Kao Hao. A general approach for constraint solving by local search. *Journal of Mathematical Modelling and Algorithms*, 3(1):73–88, 2004.

[46] 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.

[47] Gautam Garai and B. B. Chaudhuri. A distributed hierarchical genetic algorithm for efficient optimization and pattern matching. *Pattern Recogn.*, 40(1):212–228, 2007.

[48] F.R. Giordano. *A First Course in Mathematical Modeling*. Brooks/Cole, 2013.

[49] Fred Glover. Tabu search - part i. *INFORMS Journal on Computing*, 1(3):190–206, 1989.

[50] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[51] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[52] 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.

[53] Michael T. Goodrich. *Data Structures and Algorithms in Java*. John Wiley & Sons, 2005.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] Julia Handl and Bernd Meyer. Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113, December 2007.

[59] 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.

[60] M. Hellebrandt and H. Heller. A new heuristic method for frequency assignment. Technical Report TD(00) 003, COST 259, Valencia, Spain, January 2000.

[61] 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.

[62] 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.

[63] http://fap.zib.de/. Fap web, 2010.

[64] 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.

[65] C. Bettstetter J. Eberspächer, H.-J. Vögel and C. Hartmann. *GSM - Architecture, Protocols and Services*. Wiley Publishing, third edition, 2009.

[66] 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.

[67] F.J. Jaimes-Romero, D. Munoz-Rodriguez, and S. Tekinay. Channel assignment in cellular systems using genetic algorithms. In *Vehicular Technology Conference, 1996. Mobile Technology for the Human Race., IEEE 46th*, volume 2, pages 741–745 vol.2, Apr 1996.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108 – 132, 2009.

[73] 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.

[74] C.R. Kothari. *Research Methodology: Methods and Techniques*. New Age International (P) Limited, 2004.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] Hongbo Liu, Ajith Abraham, and Maurice Clerc. Chaotic dynamic characteristics in swarm intelligence. *Applied Soft Computing*, 7(3):1019 – 1026, 2007.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] C. Mannino, G. Oriolo, and F. Ricci. Solving stability problems on a superclass of interval graphs. Technical Report 26–02, Universitá di Roma "La Sapienza", 2002.

[87] 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.

[88] Asha Mehrotra. *GSM System Engineering*. Arteh House, Inc, 1997.

[89] 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.

[90] 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.

[91] 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.

[92] Marie-Bernadette Pautet Michel Mouly. *The GSM System for Mobile Communications*. Cell & Sys, 1992.

[93] 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.

[94] M. Mitchell. *An Introduction to Genetic Algorithms*. A Bradford book. MIT Press, 1998.

[95] 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.

[96] R. Montemanni, J.N.J. Moon, and D.H. Smith. An improved tabu search algorithm for the fixed-spectrum frequency-assignment problem. *Vehicular Technology, IEEE Transactions on*, 52(4):891–901, 2003.

[97] Roberto Montemanni. *Upper and Lower bounds for the fixed spectrum frequency assignment problem*. PhD thesis, School of Tecnology, University of Glamorgan, 2001.

[98] Roberto Montemanni and Derek H. Smith. Heuristic manipulation, tabu search and frequency assignment. *Comput. Oper. Res.*, 37(3):543–551, 2008.

[99] Garry Mullet. *Wireless telecommunications systems and networks*. Thomsan Delmar Learning, 2006.

[100] 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.

[101] 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.

[102] C.Y. Ngo and V.O.-K. Li. Fixed channel assignment in cellular radio networks using a modified genetic algorithm. *Vehicular Technology, IEEE Transactions on*, 47(1):163–172, Feb 1998.

[103] 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.

[104] 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.

[105] 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.

[106] Sigurdur Ólafsson. Chapter 21 metaheuristics. In Shane G. Henderson and Barry L. Nelson, editors, *Simulation*, volume 13 of *Handbooks in Operations Research and Management Science*, pages 633 – 654. Elsevier, 2006.

[107] Michael O’Neill and Anthony Brabazon. Self-organising swarm (soswarm). *Soft Comput.*, 12(11):1073–1080, 2008.

[108] Thomas La Porta Patrick Traynor, Patrick McDaniel. *Security for Telecommunications Networks*, volume 40 of *Advances in Information Security*. Springer US, 2008.

[109] 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.

[110] Jean-Yves Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63:337–370, 1996.

[111] 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.

[112] 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.

[113] A. Ratnaweera, S.K. Halgamuge, and H.C. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *Evolutionary Computation, IEEE Transactions on*, 8(3):240 – 255, june 2004.

[114] D. J. Reid. Genetic algorithms in constrained optimization. *Mathematical and Computer Modelling*, 23(5):87 – 111, 1996.

[115] 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.

[116] Stuart Russel and Peter Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall, second edition, 2003.

[117] 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.

[118] 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.

[119] Mischa Schwartz. *Mobile Wireless Communications*. Cambridge University Press, 2005.

[120] 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.

[121] Patrick Siarry, Alain Pétrowski, and Mourad Bessaou. A multipopulation genetic algorithm aimed at multimodal optimization. *Adv. Eng. Softw.*, 33(4):207–213, 2002.

[122] Alok Singh. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing*, 9(2):625 – 631, 2009.

[123] S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[124] 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.

[125] 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.

[126] 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.

[127] Gordon L. Stuüber. *Principles of Mobile Communication*. Kluwer Academic Publishers, 1996.

[128] 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.

[129] 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.

[130] 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.

[131] 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.

[132] Peter Tarasewich and Patrick R. McMullen. Swarm intelligence: power in numbers. *Commun. ACM*, 45(8):62–67, August 2002.

[133] 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.

[134] 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.

[135] 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.

[136] 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.

[137] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[138] 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.

[139] Roger L. Wainwright. A family of genetic algorithm packages on a workstation for solving combinatorial optimization problems. *SIGICE Bull.*, 19(3):30–36, 1994.

[140] Haohong Wang, Lisimachos Kondi, Ajay Luthra, and Song Ci. *4G Wireless Video Communications*. Wiley Publishing, 2009.

[141] 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.

[142] 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.

[143] 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.

[144] 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.

[145] 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.

[146] Lihua Wu and Yuyun Wang. An introduction to simulated annealing algorithms for the computation of economic equilibrium. *Computational Economics*, 12:151–169, 1998.

[147] 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.

[148] 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.

[149] Xin-She Yang. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[150] 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.

[151] Janez Zerovnik. Experiments with a randomized algorithm for a frequency assignment problem. Technical report, ECOLE NORMALE SUP'ERIEURE DE LYON, 1997.

[152] 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.

[153] 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.

Part III

Appendix

FAP PSO Code

.1 Introduction

In this chapter the C# code for the FAP PSO algorithm is presented. Note that only the relevant code for the PSO algorithm is presented and not the code that parses the benchmark scenario files.

.2 Source Code

.2.1 PSO Particle

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.PSO.Interfaces;

namespace PSOFAPConsole.PSO
{
    public class Particle<T> : ICloneable
    {
        public T Position { get; set; }
        public T Velocity { get; set; }
        public double Fitness { get; set; }
        public ParticleBest<T> PersonalBest { get; set; }

        public Particle()
        {
            Fitness = -1;
        }

        public Particle(T position)
        {
            Position = position;
            Fitness = -1;
        }

        public double Evaluate(IFitnessFunction<T> function)
        {
            double evalFitness = function.Evaluate(Position);
            if (IsPersonalBest(evalFitness))
            {
                SavePersonalBest(evalFitness);
            }
            Fitness = evalFitness;
```

```
            return evalFitness;
        }

        public void MoveTowards(Particle<T> TargetPosition, IMoveFunction<Particle<T>> function)
        {
            function.MoveTowards(this, TargetPosition);
        }

        private bool IsPersonalBest(double fitness)
        {
            if (PersonalBest == null)
            {
                return true;
            }
            else if (fitness <= PersonalBest.Fitness || PersonalBest.Fitness < 0)
            {
                return true;
            }
            return false;
        }

        private void SavePersonalBest(double cost)
        {
            PersonalBest = new ParticleBest<T>(Position, cost);
        }


        #region ICloneable Members

        public object Clone()
        {
            Particle<T> clone = new Particle<T>();
            clone.PersonalBest = (ParticleBest<T>)PersonalBest.Clone();
            clone.Position = (T)((ICloneable)Position).Clone();
            clone.Fitness = this.Fitness;
            return clone;

        }

        #endregion
    }
}
```

.2.2 PSO Particle Best

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace PSOFAPConsole.PSO
{
    public class ParticleBest<T> : ICloneable
    {
        public T Position { get; set; }
        public double Fitness { get; set; }

        public ParticleBest(T Pos, double fitness)
        {
            T clone = (T)((ICloneable)Pos).Clone();
            Position = Pos;
            Fitness = fitness;
        }



        public object Clone()
        {
```

```
            ParticleBest<T> clone = new ParticleBest<T>(Position, Fitness);
            return clone;
        }
    }
}
```

.2.3 PSO Algorithm Factory

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.FAP;
using PSOFAPConsole.PSO;
using PSOFAPConsole.PSO.Interfaces;
using PSOFAPConsole.FAP.Interfaces;

namespace PSOFAPConsole.FAPPSO
{
    using ParticleCellArray = Particle<ICell[]>;
    using FitnessFuncCellArray = IFitnessFunction<ICell[]>;
    using ParticleMoveFunction = IMoveFunction<Particle<ICell[]>>;
    using PositionGenCellArray = IPositionGenerator<ICell[]>;
    using PSOFAPConsole.FAPPSO.Functions;

    public class FAPPSOFactory
    {


        public FAPPSOFactory()
        {
        }

        public PSOAlgorithm<ICell[]> CreateFrequencyValueBased(int Population,
                        FAPModel Model,double localCoefficient, double globalCoefficient)
        {
            PositionGenCellArray generator = new FrequencyPositionGenerator(Model);
            FitnessFuncCellArray evalFunction = new FAPCostFunction(Model);
            ParticleMoveFunction moveFunction = new ParticlePerTrxFunction(Model,
                            Model.GeneralInformation.Spectrum[0], Model.GeneralInformation.Spectrum[1],
                 localCoefficient, globalCoefficient, CreateCollisionResolver(Model));
            ICellIntegrityChecker checker = new GBCViolationChecker(Model.GeneralInformation.GloballyBlockedChannels);
            String benchName = Model.GeneralInformation.ScenarioID;
            return new FAPPSOAlgorithm(benchName,Population, evalFunction,
                            moveFunction, generator, checker,GBestFactory.GetStandardSelector(),
                            new StdStatisticalAnalyser(Model));
        }

        public PSOAlgorithm<ICell[]> CreateFrequencyIndexBased(int Population,
                        FAPModel Model, double localCoefficient, double globalCoefficient)
        {
            PositionGenCellArray generator = new FrequencyIndexPositionGenerator(Model);
            FitnessFuncCellArray evalFunction = new FAPIndexCostFunction(Model);
            ParticleMoveFunction moveFunction = new ParticlePerTrxFunction(Model, 0,
                            Model.Channels.Length - 1, localCoefficient, globalCoefficient,
                            CreateCollisionResolver(Model));
            ICellIntegrityChecker checker = new GBCIndexBasedViolationChecker(Model);
            String benchName = Model.GeneralInformation.ScenarioID;
            return new FAPPSOAlgorithm(benchName, Population, evalFunction,
                            moveFunction, generator, checker,
                            GBestFactory.GetStandardSelector(), new IndexStatisticalAnalyser(Model));
        }

        public PSOAlgorithm<ICell[]> CreateIndexMovementBased(int Population,
                        FAPModel Model, double localCoefficient, double globalCoefficient)
        {
            PositionGenCellArray generator = new FrequencyIndexPositionGenerator(Model);
            FitnessFuncCellArray evalFunction = new FAPIndexCostFunction(Model);
```

```
            ParticleMoveFunction moveFunction = new PerTRXChannelIndexFunction(Model,
                                localCoefficient, globalCoefficient, CreateCollisionResolver(Model));
            ICellIntegrityChecker checker = new GBCIndexBasedViolationChecker(Model);
            String benchName = Model.GeneralInformation.ScenarioID;
            return new FAPPSOAlgorithm(benchName, Population, evalFunction,
                                moveFunction, generator, checker, GBestFactory.GetStandardSelector(),
                                new IndexStatisticalAnalyser(Model));
        }

        public PSOAlgorithm<ICell[]> CreateIndexBasedFAPPSOWithGlobalBestCellBuilder(int Population,
                        FAPModel Model, double localCoefficient, double globalCoefficient)
        {
            PositionGenCellArray generator = new FrequencyIndexPositionGenerator(Model);
            FitnessFuncCellArray evalFunction = new FAPIndexCostFunction(Model);
            ParticleMoveFunction moveFunction = new ParticlePerTrxFunction(Model, 0,
                                Model.Channels.Length - 1, localCoefficient, globalCoefficient, CreateCollisionResolver(Model));
            ICellIntegrityChecker checker = new GBCIndexBasedViolationChecker(Model);
            String benchName = Model.GeneralInformation.ScenarioID;
            return new FAPPSOAlgorithm(benchName, Population, evalFunction, moveFunction,
                                generator, checker, GBestFactory.GetGlobalBestCellBuilderSelector(),
                                new IndexStatisticalAnalyser(Model));
        }

        public PSOAlgorithm<ICell[]> CreateIndexMovementBasedWithGlobalBestCellBuilder(int Population, FAPModel Model,
                         double localCoefficient, double globalCoefficient)
        {
            PositionGenCellArray generator = new FrequencyIndexPositionGenerator(Model);
            FitnessFuncCellArray evalFunction = new FAPIndexCostFunction(Model);
            ParticleMoveFunction moveFunction = new PerTRXChannelIndexFunction(Model, localCoefficient,
                                globalCoefficient, CreateCollisionResolver(Model));
            ICellIntegrityChecker checker = new GBCIndexBasedViolationChecker(Model);
            String benchName = Model.GeneralInformation.ScenarioID;
            return new FAPPSOAlgorithm(benchName, Population, evalFunction, moveFunction, generator, checker,
                                GBestFactory.GetGlobalBestCellBuilderSelector(), new IndexStatisticalAnalyser(Model));
        }

        public PSOAlgorithm<ICell[]> CreateIndexBasedFAPPSOWithGlobalBestTRXBuilder(int Population, FAPModel Model,
                        double localCoefficient, double globalCoefficient)
        {
            PositionGenCellArray generator = new FrequencyIndexPositionGenerator(Model);
            FitnessFuncCellArray evalFunction = new FAPIndexCostFunction(Model);
            ParticleMoveFunction moveFunction = new ParticlePerTrxFunction(Model, 0, Model.Channels.Length - 1,
                                localCoefficient, globalCoefficient, CreateCollisionResolver(Model));
            ICellIntegrityChecker checker = new GBCIndexBasedViolationChecker(Model);
            String benchName = Model.GeneralInformation.ScenarioID;
            return new FAPPSOAlgorithm(benchName, Population, evalFunction, moveFunction, generator, checker,
                                GBestFactory.GetGlobalBestTRXBuilderSelector(), new IndexStatisticalAnalyser(Model));
        }

        public PSOAlgorithm<ICell[]> CreateIndexMovementBasedWithGlobalBestTRXBuilder(int Population, FAPModel Model,
                        double localCoefficient, double globalCoefficient)
        {
            PositionGenCellArray generator = new FrequencyIndexPositionGenerator(Model);
            FitnessFuncCellArray evalFunction = new FAPIndexCostFunction(Model);
            ParticleMoveFunction moveFunction = new PerTRXChannelIndexFunction(Model, localCoefficient,
                                globalCoefficient,CreateCollisionResolver(Model));
            ICellIntegrityChecker checker = new GBCIndexBasedViolationChecker(Model);
            String benchName = Model.GeneralInformation.ScenarioID;
            return new FAPPSOAlgorithm(benchName, Population, evalFunction, moveFunction, generator, checker,
                                GBestFactory.GetGlobalBestTRXBuilderSelector(), new IndexStatisticalAnalyser(Model));
        }

        protected AbstractCollisionResolver CreateCollisionResolver(FAPModel model)
        {
            return new RandomCollisionResolver(model.Channels);
        }

    }
}
```

.2.4 FAP PSO Algorithm

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.FAP.Interfaces;
using PSOFAPConsole.PSO;
using PSOFAPConsole.PSO.Interfaces;
using System.Diagnostics;
using PSOFAPConsole.FAPPSO;
using PSOFAPConsole.FAP;

namespace PSOFAPConsole.FAPPSO
{
    using ParticleCellArray = Particle<ICell[]>;
    using FitnessFuncCellArray = IFitnessFunction<ICell[]>;
    using ParticleMoveFunction = IMoveFunction<Particle<ICell[]>>;
    using PositionGenCellArray = IPositionGenerator<ICell[]>;
    using GlobalBestSelector = IGlobalBestSelector<Particle<ICell[]>>;
    using System.Threading.Tasks;
    using System.IO;

    public class FAPPSOAlgorithm : PSOAlgorithm<ICell[]>
    {
        public List<ParticleCellArray> Particles { get; set;}
        public ParticleCellArray GlobalBest { get; set; }
        public int Population { get; set; }
        public FitnessFuncCellArray EvaluationFunction { get; set; }
        public ParticleMoveFunction VelocityFunction { get; set; }
        public PositionGenCellArray Generator { get; set; }
        public GlobalBestSelector Selector { get; set; }
        public FAPCostFunction Analyser { get; set; }
        private ICellIntegrityChecker cellIntegrityChecker;
        public String BenchName { get; set; }
        private List<double> runFitness;

        public FAPPSOAlgorithm(string benchName,int population,FitnessFuncCellArray evalFunction,
                        ParticleMoveFunction moveFunction, PositionGenCellArray positionGenerator, ICellIntegrityChecker checker,
                        GlobalBestSelector selector, FAPCostFunction analyser)
        {
            BenchName = benchName;
            Particles = new List<ParticleCellArray>();
            Population = population;
            EvaluationFunction = evalFunction;
            VelocityFunction = moveFunction;
            Generator = positionGenerator;
            cellIntegrityChecker = checker;
            Selector = selector;
            Analyser = analyser;
            BenchName += "("+moveFunction.GetType().Name+")";
            BenchName += "("+selector.GetConstructionMethodName()+")";
            BenchName += "[" + evalFunction.GetType().Name + "]";
            Console.WriteLine(BenchName);
            runFitness = new List<double>();
            Initialize();
        }

        #region PSOAlgorithm<T> Members

        public void Initialize()
        {
            for (int i = 0; i < Population; i++)
            {
                Particles.Add(new Particle<ICell[]>(Generator.GeneratePosition()));
            }
        }

        public Particle<ICell[]> GetGlobalBest()
        {
```

```
            return GlobalBest;
        }

        public void Start()
        {
            DateTime start = DateTime.Now;
            for (int i = 0; i < 50; i++)
            {
                Parallel.ForEach(Particles, EvaluateParticle);
                UpdateGlobalBest();
                UpdateSwarmMovement();
                runFitness.Add(GlobalBest.Fitness);
            }
    }

        private void UpdateGlobalBest()
        {
            GlobalBest = Selector.FindGlobalBest(Particles);
            GlobalBest.Evaluate(EvaluationFunction);
        }

        //Applies the velocity function to each particle in the swarm
        public void UpdateSwarmMovement()
        {
            Parallel.ForEach(Particles, particle => particle.MoveTowards(GlobalBest,VelocityFunction));
        }

        public void EvaluateParticle(Particle<ICell[]> particle)
        {
            double cost = particle.Evaluate(EvaluationFunction);
        }

        #endregion
    }
}
```

.2.5 Frequency Position Generator

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.PSO.Interfaces;
using PSOFAPConsole.FAP.Interfaces;
using PSOFAPConsole.FAP;

namespace PSOFAPConsole.FAPPSO
{
    public class FrequencyPositionGenerator : IPositionGenerator<ICell[]>
    {
        private ICell[] Cells;
        private int[] Spectrum;
        private int[] Channels;

        public FrequencyPositionGenerator(FAPModel model)
        {
            Spectrum = model.GeneralInformation.Spectrum;
            Cells = model.Cells;
            Channels = model.Channels;
        }

        #region IPositionGenerator<ICell> Members

        public ICell[] GeneratePosition()
        {
            ICell[] position = new ICell[Cells.Length];
            for (int i = 0; i < Cells.Length; i++)
            {
```

```
            position[i] = new BasicCell(Cells[i]);
            InsertUniqueChannels(position[i].FrequencyHandler);
        }
        return position;
    }



    #endregion

    private void InsertUniqueChannels(FrequencyHandler frequencies)
    {
        Random random = new Random();
        int index = random.Next(0, Channels.Length);
        for (int i = 0; i < frequencies.Length; i++)
        {
            while (ContainsNumber(Channels[index], frequencies))
            {
                index = random.Next(0, Channels.Length);
            }
            frequencies[i] = Channels[index];
        }
    }

    private bool ContainsNumber(int number, FrequencyHandler array)
    {
        foreach (int i in array)
        {
            if (number == i)
                return true;
        }
        return false;
    }
}
}
```

.2.6 Index Based Frequency Position Generator

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.PSO.Interfaces;
using PSOFAPConsole.FAP;
using PSOFAPConsole.FAP.Interfaces;
using System.Threading.Tasks;

namespace PSOFAPConsole.FAPPSO
{
    public class FrequencyIndexPositionGenerator: IPositionGenerator<ICell[]>
    {
        private ICell[] Cells;
        private int[] Channels;

        public FrequencyIndexPositionGenerator(FAPModel model)
        {
            Cells = model.Cells;
            Channels = model.Channels;
        }

        public ICell[] GeneratePosition()
        {
            ICell[] position = new ICell[Cells.Length];
            for (int i = 0; i < position.Length; i++)
            {
                position[i] = new BasicCell(Cells[i]);
                GenerateUniqueFrequencyIndexes(position[i].FrequencyHandler);
            }
```

```
            return position;
        }

        private void GenerateUniqueFrequencyIndexes(FrequencyHandler FrequencyHandler)
        {
            Random random = new Random();
            for (int i = 0; i < FrequencyHandler.Length; i++)
            {
                int channelIndex = random.Next(Channels.Length);
                if (FrequencyHandler.Contains(channelIndex))
                {
                    i--;
                    continue;
                }
                else
                {
                    FrequencyHandler[i] = channelIndex;
                }
            }
            FrequencyHandler.MigrateFrequenciesToParent();
        }

    }
}
```

.2.7 Standard FAP Cost Function

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.PSO.Interfaces;
using PSOFAPConsole.FAP.Interfaces;
using PSOFAPConsole.FAP;

namespace PSOFAPConsole.FAPPSO
{
    public class FAPCostFunction : IFitnessFunction<ICell[]>
    {
        public ICellRelation[] InterferenceMatrix { get; private set; }
        public GeneralInformation GeneralInformation { get; private set; }

        public FAPCostFunction(FAPModel model)
        {
            InterferenceMatrix = model.InterferenceMatrix.ToArray();
            GeneralInformation = model.GeneralInformation;
        }

        #region IFitnessFunction<ICell[]> Members

        public virtual double[] Stats(ICell[] position)
        {
            return new double[19];
        }

        public virtual double Evaluate(ICell[] position)
        {
            return CalculateInterfernece(position);
        }

        protected virtual double CalculateInterfernece(ICell[] position)
        {
            double totalInterference = 0;

            foreach (ICellRelation cellRelation in InterferenceMatrix)
            {
                ICell cell = position[cellRelation.CellIndex[0]];
                ICell interferingCell = position[cellRelation.CellIndex[1]];
```

```
                double interference = 0;
                for (int i = 0; i < interferingCell.FrequencyHandler.Length; i++)
                {
                    for (int j = 0; j < cell.FrequencyHandler.Length; j++)
                    {
                        if (SameFrequency(interferingCell.FrequencyHandler[i],cell.FrequencyHandler[j]))
                        {
                            interference += ZeroIfOusideInterferneceThreshold(cellRelation.DA[0]);
                        }
                        else if (FrequenciesDifferByOne(interferingCell.FrequencyHandler[i],cell.FrequencyHandler[j]))
                        {
                            interference += ZeroIfOusideInterferneceThreshold(cellRelation.DA[1]);
                        }
                    }
                }
                totalInterference += interference;

            }
            return totalInterference;
        }

        #endregion

        protected double ZeroIfOusideInterferneceThreshold(double value)
        {
            if (value <= GeneralInformation.MinTolerableInterference)
            {
                return 0;
            }
            return value;
        }


        protected bool SameFrequency(int freqA, int freqB)
        {
            return freqA == freqB;
        }

        protected bool FrequenciesDifferByOne(int freqA, int freqB)
        {
            return Math.Abs(freqA - freqB) == 1;
        }
    }
}
```

.2.8 FAP Cost Function with Index Based Frequencies

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.PSO.Interfaces;
using PSOFAPConsole.FAP.Interfaces;
using PSOFAPConsole.FAP;
using System.Threading.Tasks;
using System.Threading;

namespace PSOFAPConsole.FAPPSO
{
    public class FAPIndexCostFunction : FAPCostFunction
    {
        private int[] channels;
        public FAPIndexCostFunction(FAPModel model)
            : base(model)
        {
            channels = model.Channels;
        }
```

```
        protected override double CalculateInterfernece(ICell[] position)
        {
            double totalInterference = 0;
            foreach (ICellRelation cellRelation in InterferenceMatrix)
            {
                ICell cell = position[cellRelation.CellIndex[0]];
                ICell interferingCell = position[cellRelation.CellIndex[1]];
                double interference = 0;
                for (int i = 0; i < interferingCell.Frequencies.Length; i++)
                {
                    for (int j = 0; j < cell.Frequencies.Length; j++)
                    {
                        int frequencyA = channels[interferingCell.Frequencies[i].Value];
                        int frequencyB = channels[cell.Frequencies[j].Value];
                        double trxInf = 0;
                        if (SameFrequency(frequencyA, frequencyB))
                        {
                            trxInf = ZeroIfOusideInterferneceThreshold(cellRelation.DA[0]);
                            interference += trxInf;
                        }
                        else if (base.FrequenciesDifferByOne(frequencyA, frequencyB))
                        {
                            trxInf = ZeroIfOusideInterferneceThreshold(cellRelation.DA[1]);
                            interference += trxInf;
                        }
                        cell.FrequencyHandler.SetSingleTrxInterference(j, trxInf);
                    }
                    //cell.Interference += interference;
                }
                cell.Interference += interference;
                totalInterference += interference;

            }
            return totalInterference;
        }
    }
}
```

.2.9 Velocity Method 1: Particle Per TRX

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.PSO.Interfaces;
using PSOFAPConsole.FAP;
using PSOFAPConsole.PSO;
using PSOFAPConsole.FAP.Interfaces;
using System.Threading.Tasks;

namespace PSOFAPConsole.FAPPSO
{
    public class ParticlePerTrxFunction : IMoveFunction<Particle<ICell[]>>
    {
        private int upperBound;
        private int lowerBound;
        private double localCoefficient;
        private double globalCoefficient;
        private AbstractCollisionResolver collisionResolver;
        private int coSite;
        private int coCell;

        public ParticlePerTrxFunction(FAPModel model,int lowerBound, int upperBound, double localCoef,
                        double globalCoef,AbstractCollisionResolver collisionResolver)
        {
            this.upperBound = upperBound;
            this.lowerBound = lowerBound;
            localCoefficient = localCoef;
```

```
        globalCoefficient = globalCoef;
        this.collisionResolver = collisionResolver;
        coSite = model.GeneralInformation.CoSiteSeperation;
        coCell = model.GeneralInformation.DefaultCoCellSeperation;

    }

    #region IMoveFunction<Particle<Cell[]>> Members

    public Particle<ICell[]> MoveTowards(Particle<ICell[]> from, Particle<ICell[]> to)
    {
        ICell[] pbest = from.PersonalBest.Position;

        ICell[] a = Multiply(localCoefficient,true,Subtract(pbest,from.Position));
        ICell[] b = Multiply(globalCoefficient,true,Subtract(to.Position, from.Position));
        if (from.Velocity != null)
        {
            from.Velocity = Multiply(0.5,false, Add(from.Velocity,Add(a, b)));
        }
        else
        {
            from.Velocity = Add(a, b);

        }
        //EnsureUnique(from.Velocity);
        EnsureUnique(Add(from.Position, from.Velocity));
        MigrateFrequencies(from.Position);
        return from;
    }

    private void MigrateFrequencies(ICell[] iCell)
    {
        Parallel.ForEach(iCell, cell =>
            {
                collisionResolver.ResolveCollisions(cell.FrequencyHandler);
                AdhereToSeperation(cell.FrequencyHandler);
                cell.FrequencyHandler.MigrateFrequenciesToParent();
            });
    }

    private void AdhereToSeperation(FrequencyHandler frequencyHandler)
    {
        for (int i = 0; i < frequencyHandler.Length; i++)
        {
            if (ViolatesSeperation(i, frequencyHandler, coCell))
            {
                frequencyHandler[i] = BoundValue(frequencyHandler[i] - coCell);
            }
        }
    }

    private void EnsureUnique(ICell[] iCells)
    {
        Parallel.ForEach(iCells, MakeUnique);
    }

    private void MakeUnique(ICell cell)
    {
        Random r = new Random();
        FrequencyHandler handler = cell.FrequencyHandler;
        for (int i = 0; i < handler.Length; i++)
        {
            if (hasValue(i, handler))
            {
                int newFrequency = r.Next(upperBound);
                newFrequency = BoundValue(newFrequency);
                handler[i] = newFrequency;
                i--;
            }
        }
```

```
}

private bool hasValue(int i, FrequencyHandler handler)
{
    for (int j = 0; j < handler.Length; j++)
    {
        if (handler[i] == handler[j] && i != j)
        {
            return true;
        }
    }
    return false;
}

private bool ViolatesSeperation(int i, FrequencyHandler handler,int seperation)
{
    for (int j = 0; j < handler.Length; j++)
    {
        if (i == j)
            continue;
        if (handler[i] == handler[j])
        {
            return true;
        }
        if (Math.Abs(handler[j] - handler[i]) < seperation)
        {
            return true;
        }
    }
    return false;
}

//Loops through each cell and then each trx that a cell contains.
//It takes each trx and subtracts it from the same cell trx found in the other cell array
private BasicCell[] Subtract(ICell[] a, ICell[] b)
{
    BasicCell[] answ = new BasicCell[a.Length];
    for(int i = 0; i < a.Length;i++)
    {
        answ[i] = a[i].Clone() as BasicCell;
        for (int j = 0; j < a[i].FrequencyHandler.Length; j++)
        {
            answ[i].FrequencyHandler[j] = (a[i].FrequencyHandler[j] - b[i].FrequencyHandler[j]);
        }
    }
    return answ;
}

//Loops through the given cell array and then it loops through each trx that a cell contains and multiplies it with x.
//If random is true, the product of the trx multiplied by x is also multiplied with a random value in range [0.0,1.0]
private ICell[] Multiply(double x,Boolean random, ICell[] a)
{
    Random r = new Random();
    for (int i = 0; i < a.Length; i++)
    {
        for (int j = 0; j < a[i].FrequencyHandler.Length; j++)
        {
            if (random)
            {
                a[i].FrequencyHandler[j] = (int)(a[i].FrequencyHandler[j] * x * r.NextDouble());
            }
            else
            {
                a[i].FrequencyHandler[j] = (int)(a[i].FrequencyHandler[j] * x);
            }
        }
    }
    return a;
}
```

```
        //Takes each corresponding trx found in a cell and adds it together.
        //The result of two trx's being added is then bounded to ensure it lies within the spectrum bounds
        private ICell[] Add(ICell[] a, ICell[] b)
        {
            for (int i = 0; i < a.Length; i++)
            {
                for (int j = 0; j < a[i].FrequencyHandler.Length; j++)
                {
                    a[i].FrequencyHandler[j] += b[i].FrequencyHandler[j];
                    a[i].FrequencyHandler[j] = BoundValue(a[i].FrequencyHandler[j]);
                }
            }
            return a;
        }

        private int BoundValue(int value)
        {
            if (value < 0)
                return BoundValue(Math.Abs(value));
            if(value > upperBound)
            {
                return lowerBound + (value % upperBound);
            }
            else if (value < lowerBound)
            {
                return BoundValue(value + lowerBound) ;
            }
            return value;
        }

        #endregion
    }
}
```

.2.10 Velocity Method 1: Per Index Based TRX

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.PSO.Interfaces;
using PSOFAPConsole.FAP.Interfaces;
using PSOFAPConsole.FAP;
using PSOFAPConsole.PSO;
using System.Threading.Tasks;

namespace PSOFAPConsole.FAPPSO.Functions
{
    public class PerTRXChannelIndexFunction : IMoveFunction<Particle<ICell[]>>
    {
        public AbstractCollisionResolver CollisionResolver { get; set; }
        public double LocalCoefficient { get; set; }
        public double GlobalCoefficient { get; set; }
        public int[] Channels { get; set; }
        private int coSite;
        private int coCell;

        public PerTRXChannelIndexFunction(FAPModel model,double localCoefficient, double globalCoefficient,
                        AbstractCollisionResolver collisionResolver)
        {
            CollisionResolver = collisionResolver;
            Channels = model.Channels;
            LocalCoefficient = localCoefficient;
            GlobalCoefficient = globalCoefficient;
            coSite = model.GeneralInformation.CoSiteSeperation;
            coCell = model.GeneralInformation.DefaultCoCellSeperation;
        }
```

```
public Particle<ICell[]> MoveTowards(Particle<ICell[]> from, Particle<ICell[]> to)
{
    ICell[] velocity = from.Velocity;
    for (int i = 0; i < from.Position.Length; i++)
    {
        FrequencyHandler pbest = from.PersonalBest.Position[i].FrequencyHandler;
        FrequencyHandler fromIndex = from.Position[i].FrequencyHandler;
        FrequencyHandler toIndex = to.Position[i].FrequencyHandler;
        FrequencyHandler movedIndexes = MoveIndexes(pbest, fromIndex, toIndex);
        if (velocity == null)
        {
            velocity = new ICell[from.Position.Length];
            from.Position.CopyTo(velocity, 0);
        }
        FrequencyHandler indexVelocity = CalculateIndexVelocity(velocity[i].FrequencyHandler, movedIndexes);
        ApplyVelocity(indexVelocity, fromIndex);
    }
    MigrateFrequencies(from.Position);
    return from;
}

private void MigrateFrequencies(ICell[] iCell)
{
    Parallel.ForEach(iCell, cell => {
        CollisionResolver.ResolveCollisions(cell.FrequencyHandler);
        AdhereToSeperation(cell.FrequencyHandler);
        cell.FrequencyHandler.MigrateFrequenciesToParent();
    });
}

private void AdhereToSeperation(FrequencyHandler frequencyHandler)
{
    for (int i = 0; i < frequencyHandler.Length; i++)
    {
        if (ViolatesSeperation(i, frequencyHandler, coCell))
        {
            if (frequencyHandler[i] - coCell < 0)
            {
                frequencyHandler[i] = Channels.Length - coCell;
            }
            else
            {
                frequencyHandler[i] = frequencyHandler[i] - coCell;
            }
        }
    }
}

private bool ViolatesSeperation(int i, FrequencyHandler handler, int seperation)
{
    for (int j = 0; j < handler.Length; j++)
    {
        if (handler[i] == handler[j] && i != j)
        {
            return true;
        }
        //handler[j] and [i] contain channel indexes
        if (Math.Abs(Channels[handler[j]] - Channels[handler[i]]) < seperation)
        {
            return true;
        }
    }
    return false;
}

private void ApplyVelocity(FrequencyHandler indexVelocity, FrequencyHandler fromIndex)
{
    for (int i = 0; i < indexVelocity.Length; i++)
    {
        fromIndex[i] = Math.Abs((fromIndex[i] + indexVelocity[i])) % Channels.Length;
```

```
            }

        }

        private bool IsDuplicate(int i, FrequencyHandler fromIndex)
        {
            for (int j = 0; j < fromIndex.Length; j++)
            {
                if (j == i)
                    continue;
                else if (fromIndex[j] == fromIndex[i])
                {
                    return true;
                }
            }
            return false;
        }


        private FrequencyHandler CalculateIndexVelocity(FrequencyHandler velocity, FrequencyHandler movedIndexes)
        {
            for (int i = 0; i < velocity.Length; i++)
            {
                velocity[i] = velocity[i] + (int)(0.5 * movedIndexes[i]);
            }
            return velocity;
        }


        private FrequencyHandler MoveIndexes(FrequencyHandler pbest, FrequencyHandler from, FrequencyHandler to)
        {
            Random random = new Random();
            for (int i = 0; i < from.Length; i++)
            {
                double r1 = random.NextDouble();
                double r2 = random.NextDouble();
                double a1 = (LocalCoefficient * r1 * (pbest[i] - from[i]));
                double a2 = (GlobalCoefficient * r2 * (pbest[i] - to[i]));
                from[i] = (int)(a1 + a2);
            }
            return from;
        }

        private bool hasFrequencyIndex(FrequencyHandler frequencies, int frequency)
        {
            return frequencies.Any(el => el == frequency);
        }


    }
}
```

.2.11 Global Best Builder

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOFAPConsole.PSO.Interfaces;
using PSOFAPConsole.PSO;
using PSOFAPConsole.FAP.Interfaces;
using PSOFAPConsole.FAP;

namespace PSOFAPConsole.FAPPSO
{

    public class GBestBuilder : IGlobalBestSelector<Particle<ICell[]>>
    {
```

```
        public delegate void GlobalBestConstructionMethod(Particle<ICell[]> particle,ref Particle<ICell[]> gbest);
        private Particle<ICell[]> globalBest;
        private GlobalBestConstructionMethod BuildGBest;


        public GBestBuilder(GlobalBestConstructionMethod globalBestBuildMethod)
        {
            BuildGBest = globalBestBuildMethod;
        }

        public Particle<ICell[]> FindGlobalBest(List<Particle<ICell[]>> population)
        {
            return BuildGlobalBest(population);
        }

        private Particle<ICell[]> BuildGlobalBest(List<Particle<ICell[]>> population)
        {
            foreach (Particle<ICell[]> particle in population)
            {
                if (globalBest == null)
                {
                    globalBest = (Particle<ICell[]>)particle.Clone();
                }
                else
                {
                    BuildGBest(particle,ref globalBest);
                }
            }
            return globalBest;

        }

        public string GetConstructionMethodName()
        {
            return BuildGBest.Method.Name;
        }
    }
}
```

Acronyms

ABC artificial bee colony. 92, 108–111, 114, 116–118, 132

ACO ant colony optimisation. x, 91, 94–99, 103–107, 111, 124, 132

ACS ant colony system. 96, 101, 102

AFP automatic frequency planning. 32

AGCH access grant channel. 26

AI artificial intelligence. 2, 55, 56

ANTS approximate non-deterministic tree search. 96

ARPSO attract repulse particle swarm optimisation. 124

AS ant system. 96, 97, 99–102

AUC authentication centre. 20

BCCH broadcast control channel. 25, 27

BCH broadcast channel. 25

BCO bee colony optimisation. 110

BER bit error rate. 41

BSC base station controller. 15, 23, 28–30

BSO bee swarm optimisation. 110

BSS base station subsystem. 15, 18, 23, 26, 27

BTS base transceiver stations. 15–17, 22, 23, 25, 27–30

CAP channel assignment problem. 32

CBCH cell broadcast control channel. 27

CCCH common control channel. 25, 26

CDMA code division multiple access. 12

COST COopération européene dans le domaine de la recherche Scientifique et Technique. x, 45, 48, 67, 68, 75, 86, 88, 132

D/ACCH dedicated/associated control channel. 26

DCCH dedicated control channel. 25

DCS1800 digital cellular system. 9

DFA dynamic frequency allocation. 36–38, 51, 52

DS-CDMA direct sequence collision detection multiple access. 11

EDGE enhanced data global evolution. 8

EIR equipment identity register. 20, 21

EUCLID EUropean COopération on the Long term in the Defense. 48, 52

FACCH fast associated control channel. 27

FAP frequency assignment problem. ii, v, vi, ix, x, 8, 11, 32, 34–36, 43–47, 49, 51–54, 59, 67, 68, 75, 76, 86–88, 94, 104–107, 116–118, 127, 129, 131, 132, 134–138, 142, 146, 147, 152–154, 156–160

FCCH frequency control channel. 25

FDM frequency division multiplexing. 18

FEA frequency exhaustive assignment. 131

FFA fixed frequency allocation. 36–38, 51, 52

FS-FAP fixed spectrum frequency assignment problem. ii, 3, 45, 47, 67, 131, 132, 139

GA genetic algorithm. 77–80, 83, 84, 86–88, 90, 120

GPRS general packet radio system. 8

GSM general system for mobile communication. 8–13, 15, 16, 18–25, 27, 28, 30, 31, 48–50, 53

HLR home location register. 19–21

HMT heuristic manipulation technique. 67

HSCSD high speed circuit switched data. 8

IMEI international mobile equipment identity. 13, 21

IMSI international mobile subscriber identity. 13

ISDN integrated service digital network. 9, 22

LA location area. 15

LAPDm link access protocol D channel modified. 22

LTE long term evolution. 12

LTM long term memory. 63

MI-FAP minimum interference frequency assignment problem. 45, 54

MMAS min-max ant system. 96, 102

MMS multimedia message service. 8

MO-FAP minimum-order frequency assignment problem. 44

MS mobile station. 13, 15, 18, 19, 21–31, 38

MS-FAP minimum span frequency assignment problem. 44, 45, 48, 131, 132

MSC mobile switching centre. 15, 18–20, 23, 27, 29, 30

MTM medium term memory. 63

NCH notification channel. 26

NMC network management centre. 21, 22

NSS network switching subsystem. 18

OMC operations and management centre. 21, 22

PCH paging channel. 26

POS point of sale. 13

PRNG pseudo random number generator. 83

PSO particle swarm optimisation. ii, ix, 3, 92, 119–122, 124–127, 129, 131, 132, 134–136, 138–142, 145, 147, 149, 152–154, 156–160, 172

PSTN public switching telephone network. 19

QoS quality of service. 40

QRNG quasi-random number generator. 83

RACH random access channel. 26

SA simulated annealing. 69–76, 79, 82, 86, 88, 92

SACCH slow associated control channel. 26

SACO simple ant colony optimisation. 96

SCH sychronisation channel. 25

SDCCH stand-alone dedicated control channel. 26, 27, 29

SIM subscriber identification module. 13

SINR signal to interference noise power ration. 41

SIR signal-to-noise ratio. 10, 11, 41, 42

SMS short message service. 8, 19

SRES signed response. 20

STM short term memory. 63

TCH traffic channel. 24, 26, 27, 29

TDM time division multiplexing. 18

TDMA time division multiple access. 9, 22, 24, 25

TRX transceiver. 18, 32

TS tabu search. x, 59–68, 70, 74–76, 79, 86, 88, 92, 157

UMTS universal mobile telecommunications system. 11, 12

VBA virtual bee algorithm. 110

VLR visitor location register. 19, 20

WCDMA wide band collision detection multiple access. 11

WiMAX worldwide interoperability for microwave access. 12