PARTICLE SWARMS AND THE FREQUENCY ASSIGNMENT PROBLEM

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

JUNE 2011

Contents

List of Figures

List of Algorithms

List of Tables

Part I

Background

Chapter 1

Introduction

1.1 Test

blah blah blah blah

Chapter 2

Cellular Technology

2.1 Introduction

In the information age we currently live in almost every device has a wireless technology of sort that it uses to provide a specific service. Radios for audio entertainment; television remotes to change channels; cellular phones for communication; wireless access points to create wireless LAN's [2]. Wireless technology is now part of our everyday life.

The popularity and rapid adoption of wireless technology has made it difficult to plan, manage and operate wireless networks [2, 28, 48, 75, 82]. Wireless technology facilitates communication between two entities by transmitting data or voice via a radio frequency [2, 28, 48, 75, 82].

As the popularity and use of services that use wireless technology increase, the greater the need for these services to use different radio frequencies to communicate [2, 28, 48, 75, 82]. This need arises due to an effect called *interference* that occurs when two or more connections between entities use the same radio frequency to facilitate communication [2, 28, 48, 75, 82]. This effect is discussed in more detail in Chapter 3.

Therefore the use of frequencies by services must be carefully considered to avoid interference when entities communicate with each other, which is a problem since the amount of entities that communicate far out strip the amount of frequencies available for communication [2, 28, 48, 75, 82]. Thus assigning frequencies to entities for communication is a difficult problem and referred to as the frequency assignment problem (FAP) [2, 28, 48, 75, 82].

Initially manual techniques were used to assign frequencies in an attempt to solve the FAP [2,28,48,75,82]. But as a result, the assignment of frequencies was either to complex or just too daunting because of shear amount of entities that need to be assigned frequencies [2, 28, 48, 75, 82]. Also, because of the rapid adoption of wireless technology the assignment of frequencies needed to be dynamic and hence, automated [2, 28, 48, 75, 82].

When a task needs to be automated an algorithm needs to be developed to tell the machine how it must go about to solve the problem it is destined to work on [2, 28, 48, 75, 82]. Early algorithms developed to solve the FAP utilized brute force[1] techniques to assign frequencies [2, 28, 48, 75, 82].

Since the FAP is proven to be NP-Complete problem, thus using an algorithm that tries to brute force a solution is futile as no solution can be found in a reasonable time frame in chapter 3 NP-Complete problem will defined as well as a overview of the FAP. Hence, algorithms based on heuristic techniques are utilized in an attempt to either solve the FAP or come close to a solution. Heuristic Algorithms will be discussed in Chapter 4.

In this chapter we start by discussing GSM as well as a brief history of GSM. The topology of GSM is then explained before moving onto the problems in that are present GSM network. One of the major problems being the FAP.

2.2 GSM Networks

The General System for Mobile Communications (GSM) is a system for multi-service cellular communication that is capable of providing voice as well as data services [2, 28, 48, 75, 82]. Most cellular networks in operation are GSM based [2, 28, 48, 75, 82]. 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 such as *general packet radio system* (GPRS), *enhanced data rate for global evolution* (EDGE) and *high speed circuit switched data* (HSCSD) are becoming more important [28, 48].

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

GSM is one of the most widely used radio communications technologies, which is why we need to look at the history behind it in order for us to understand the domain of radio communication better [48]. A brief history of the GSM network specification will now be presented in the next section.

2.2.1 A Brief History of GSM Networks

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

At the start of the GSM specification in the early 1980's it was initially thought that the system would be analogue based, but this soon changed with the *integrated service digital network* (ISDN) specification nearing completion. As such the GSM specification started following much of the same design principles and access protocols that ISDN exhibited [2, 48, 75].

After the completion of the ISDN specification and the advantages of switching to digital instead of analogue for communication became clear. One of the primary advantages of ISDN is that it is capable of transmitting data as higher speeds. GSM would therefore be based on digital transmission and speech would be represented by a digital stream of 16 Kbits/s [2,48,75].

The primary benefit that one can deduce from the use of ISDN over slower analogue connections is because data can be transmitted faster, more data can be sent for the same amount of time it would've taken on an analogue connection [2, 48, 75]. Hence, since more data can be sent it has the net effect of increasing the quality and efficiency of the connection between two entities, the network therefore does not need to remove as much information from a packet to be sent to fit within the data transmission constraints [2, 48, 75].

Before the switch to digital transmission was finalized the GSM first wanted to evaluate the spectral efficiency of analogue and digital based transmission [2,48,75]. Spectral efficiency plays an important part in wireless communication since the radio spectrum is a limited resource and whichever transmission technology is used, the utilization of the spectrum should be maximised [2, 48, 75].

[2]In 1990 the United Kingdom requested that 1800MHz band be added to the scope of the GSM standard group. This variant of the GSM specification became known as the *Digital Cellular System 1800* (DCS1800) [2, 75].

Maximum utilisation is an important problem that will discussed in detail in later sections of this chapter. After an evaluation of spectral efficiency it was finalized that the GSM system would be digital based using *time division multiple access* (TDMA) [28, 71, 75].

By the early 1990s GSM became an evolving standard and the first GSM based network was demonstrated in 1991[3] [2,28,48,75]. The following year a number of GSM networks were operating in Europe due to mobile terminals / equipment capable of operating on the networks becoming more widely available to the general public [2, 28, 48, 75]. In the same year an operator in Australia became the first non-European operator to implement a GSM based network [28].

The collective subscriber base of GSM networks surpassed the million-subscriber mark in 1993. Due to this phenomenal growth in GSM network use, numerous extensions were made to the GSM specification. Some of the extensions that were made are the following [2, 28, 48, 75]:

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

The specification with these extensions defined is known as the 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, 75].

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, 48]. Data transmission was improved by these technologies by enabling that more than one GSM slot be used for a terminal or service at a time [2, 48].

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

By enabling that more than one GSM slot be used by a terminal or service, transceivers are required to have a higher signal to noise ratio [48, 71]. This requirement has an impact that effects 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 [28, 71].

The actual signal to noise ratio at a receiver is dependent on a number of factors that include [2, 48]:

- 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 signal to noise ratio at a transceiver is not trivial. For a more in depth discussion on the calculation the reader is directed at the survey by [2].

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

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

UMTS is a large improvement of the GSM in two areas namely Data Transmission bandwidth and Frequency Planning due to UMTS utilising *DS-CDMA* (direct sequence code division multiple access) and *WCDMA* (Wide Band Code Division Multiple Access). The higher data transmission speed (2 Mb/s) can be attributed to UMTS using the DS-CDMA scheme. The scheme also allows more users to be served than previous generation of networks [28, 110].

A direct consequence of UMTS utilizing DS-CDMA and WCDMA, which sends data over a wide-band of 5 MHz, therefore no frequency planning problem comparable to GSM has to be solved [28, 110].

Code division multiple access (CDMA) is a technology primarily used in broadband systems. Users do not gain access to only a small portion of the bandwidth, but rather use the entire band for the duration of a connection. Users also do not gain exclusive access to the whole band, but instead share the usage of the bandwidth with other users simultaneously, hence the name *multiple access*. Users using a band simultaneously are separated using orthogonal codes [48].

With CDMA a users 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 [48]. The spreading factor fluctuates between values of 10 and 1000 [48]. Using these spreading factors less interference and disturbances are encountered because the broadband signal is generated from a narrowband signal [48]. UMTS may be a large improvement, but its adoption does not spread very far from busy city centres that contain a large concentration of clients in a small geographical area. The reason for this is that, as mentioned previously, UMTS caters for larger data usage and therefore more clients can be serviced simultaneously [48].

Most network operators do not implement entirely new backbone architecture for UMTS to operate on but instead, utilize the same backbone used for GSM and GPRS. This not only extends the lifetime of previous infrastructure investment by the operator but also builds upon the redundancy provided by the GSM network [48]. Thus even with new technological improvements such as UMTS, GSM as a wireless technology is still used for communication and is therefore still relevant today.

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

2.3 Topology of a GSM Network

A GSM networks consists of a variety of different subsystems to realise the goal of establishing a radio communication link between two parties.

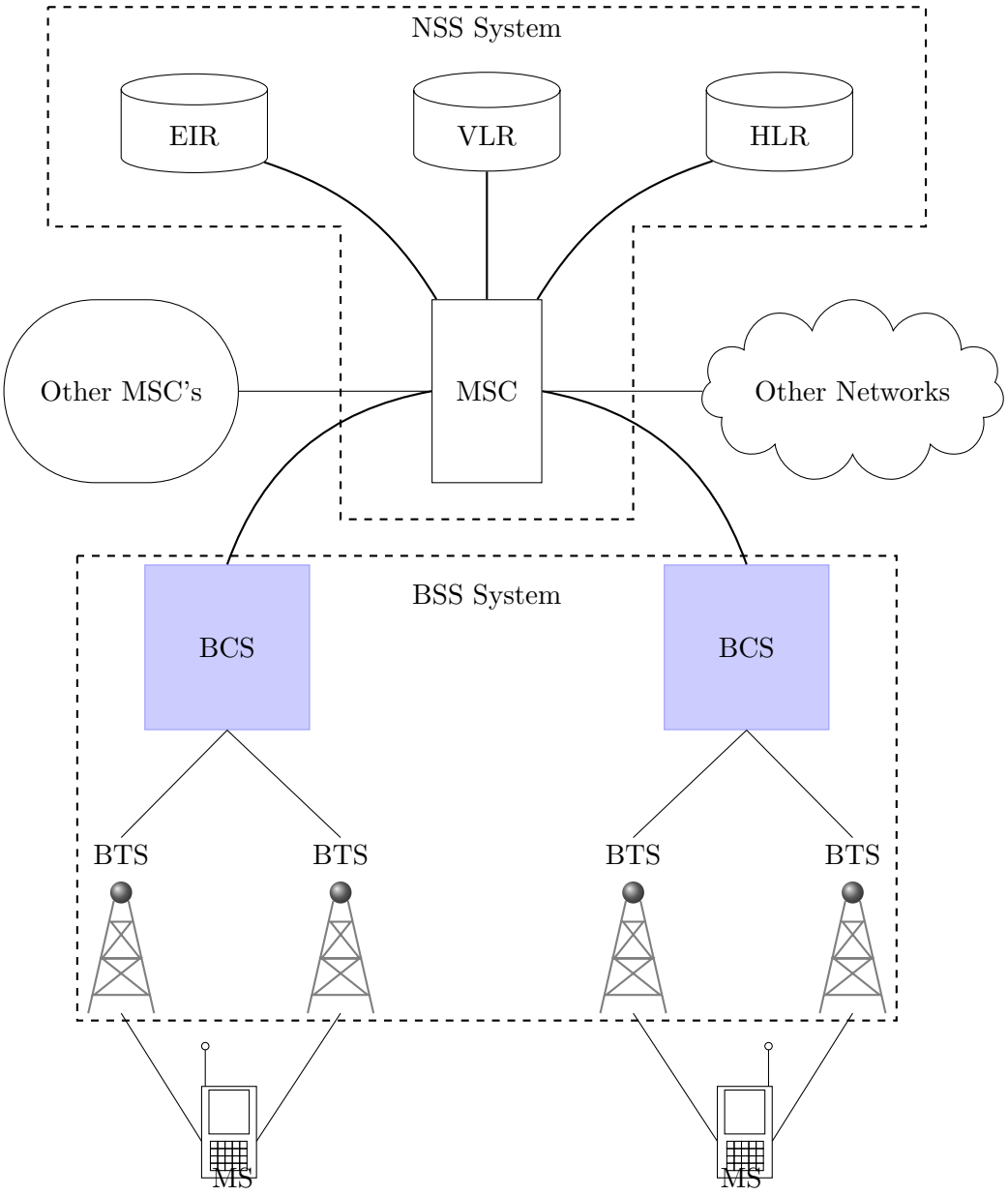

Figure 2.1: GSM Architecture

The hierarchy of systems and their respective connections to each other is illustrated in figure 2.1. Each subsystem will now be discussed.

Mobile Station (MS)

A Mobile station (MS) as it is defined in the GSM specification refers to any mobile device that is capable of making and receiving calls on a GSM network. The MS is the main gateway for a user to gain access to the GSM network [28, 48]. Typical devices that fall under the category of MS are Cellular Phones, Smart Phones and POS devices. The MS has two features that play an important role throughout the GSM Network, namely:

Subscriber Identification Module (SIM) — Usually inserted into a mobile devices. 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 [28].

International Mobile Equipment Identity(IMEI) — Used to identify mobile station equipment. Primarily used in the denial of service to equipment that has been blacklisted[4] and tries to gain access to the network [28].

The MS has the capability to change the transmission power it uses from its base value to a maximum value of 20 mW. The change in transmission power is automatically set to the lowest level by the base transceiver station (BTS) to ensure reliable communication after evaluating the signal strength as measured by the MS [48, 71]. The power adjustment also minimizes cochannel interference because the transmitted signal is less powerful therefore less likely to interfere with other signals [71].

2.3.1 Base Station Subsystem (BSS)

According to the GSM Phase 2+ specification this system is viewed by the *mobile switching centre* (MSC) through an Abis radio interface as the system responsible for communicating with mobile stations in a particular location area [28]. The BSS usually consists of one *base station controller* (BSC) with one or more *base transceiver stations* (BTS) that it controls [28]. The

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

communication link between the MSC and BSC is the called the A-interface and the communication link between the BSC and BTS is called the Abis interface [28]. A BTS has similar equipment to that of a Mobile Station [71]. Both have transceivers, antennas and the necessary functions to perform radio communication.

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

In a GSM network the Service Area (SA) is subdivided into Location Areas (LAs) which are then further divided into smaller radio zones called cells [90]. When a cellular network is modelled, cells are modelled as hexagonal shapes. Each cell in the modelled network is served by only one BTS[5] and is usually regarded to be in the centre of a cell as can be seen in figure 2.2 [48]. Even though cells are modelled as being hexagons (see figure 2.2) the actual coverage area of a cell has no predefined regular shape [48].

With the network modelled as a series of interconnecting hexagons it allows one to more easily take constraints into account. For a cell to serve its geographical area, it needs to be allocated frequencies to operate on. Therefore, for each cell i in the modelled network a subset S_i of frequencies from the total frequencies F allocated to the GSM network is assigned [48]. Neighbouring cells must at all cost 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 [48]. Since the amount of cells in a network greatly outnumber the available of subset frequencies available one is forced to start reusing frequency subsets in cells. To ensure the reused frequency subsets do not interfere with their neighbouring cells a reuse distance D is defined [48]. Which basically means that a certain amount of cells must be between the cell already assigned the frequency subset S_i and the cell to be assigned a frequency subset [48]. The amount of cells is the distance value D.

The size of a cell determines the amount of potential traffic that cell will be required to handle [28, 48, 75]. Therefore, if the size of a cell is chosen

[5]Also referred to as a Site

to be small, less channels will be required to be allocated to that cell as it would not be required to handle that much traffic as a larger cell.

By making the size of cells smaller, the network operator is required to invest more into its network infrastructure. Smaller cells has a direct consequence that more cells would be required to serve the same geographical area and more cells means more BTS's etc. need to be built and maintained, not to mention that more BTSs also means more locations need to be rented [48]. 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 regard to also be a cell just smaller in size and not omni-directional [48, 71, 75].

A cell is divided into 3 to 6 service sectors and each sector is allocated an antenna/transceiver [71]. Depending on how many sectors are at a cell, the operating angle of the antennae needs to be adjusted accordingly to ensure 360 degree service. If there is only one sector an omni-directional antenna is used, otherwise the antennae operating angle are adjusted to $\frac{360^\circ}{n}$ where n is the amount of antennae [28].

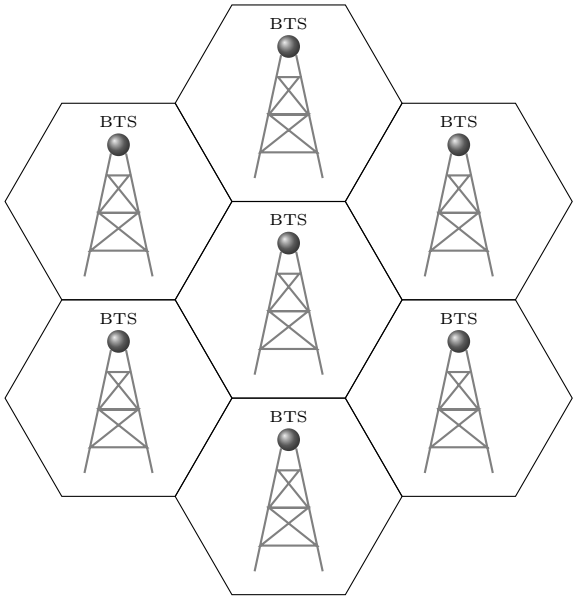

Figure 2.2: Cells with BTS's

By dividing the cell into sectors the amount of co-channel interference that would occur in a cell is greatly reduced [48]. For instance, if a cell using an omni-directional transceiver is assigned 6 channels. If the cell were to be divided into 3 sectors, where the sectors antennae are 120° apart, the amount of interfering co-channels shrink from 6 to 2 and from 6 to 1 in the case that the cell is divided into 6 sectors [48, 71, 75].

Each sector operates one or more elementary transceivers called TRXs. The amount of TRXs per sector is determined by the expected peak traffic demand that the cell must be able to handle. Each TRX can handle 7 to 8 communication links or calls in parallel except the first TRX, which handles fewer calls than normal due to it being responsible for transmitting cell organisation and protocol information [28]. TRXs are able to handle 7-8 calls in parallel due to the use of *frequency division multiplexing* (FDM) and *time division multiplexing* (TDM) schemes.

TRXs are assigned channels, which enable them to provide conversion between digital traffic data on the network side and the radio communication between Mobile Stations and the GSM network [13,65]. The various channels that are used by a cell for communication are discussed in section 2.5.

2.3.2 Mobile Switching Centre (MSC)

The MSC is at the heart of 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 [71, 75]. The MSC has interfaces on the one side to communicate with GSM subscribers (through the BSS) and on the other it has interfaces to communicate with external networks [75]. The MSC interfaces with external networks to utilise their superior capability in data transmission as well as for the signalling and communication with other GSM entities [75].

The most basic functions that an MSC is responsible for in a network are the following [82]:

- 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 [75, 82].

The VLR retrieves this information from the *Home Location Register (HLR)* that contains all the registered GSM subscriber information for the network. This information enables the MSC to quickly retrieve the necessary information to setup a call between two entities [71, 90].

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

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

2.3.3 Network Databases

The HLR, AUC (Authentication Centre) and EIR (Equipment Identity Register) are the 3 'back-end' databases, which stores and provides information for the rest of the GSM Network. We will now briefly discuss what the purpose of each database is and its core functions.

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

Subscriber parameters that are stored in the database include: Billing information, routing information, identification numbers, authentication parameters, subscribed services. The following information is also stored but the information is of a temporary nature and can change at any time: Current VLR and MSC the subscriber is registered with; whether the subscriber is roaming [71].

Authentication Centre (AUC) The Authentication center 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 [71, 75]. The information that is stored contains authentication information as well as keys that are used in ciphering of information [71, 75].

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 [71, 75]. 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 [71].

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 [71, 75]. The AUC is sometimes included with HLR to allow for fast communication between the two entities [71].

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

Typically there is only one EIR database per network and interfaces to the various HLR database contained in the network. The IMEI's are grouped into 3 categories: *White List*, *Black List* and the *Gray List*. The White list contains only the IMEI numbers of valid MS's; the Black List stores the IMEI numbers of equipment that has been reported stolen and the Gray List stores the IMEI numbers of equipment that has some fault (faulty software, wrong make of equipment).

2.3.4 GSM Network Management entities

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

For a network to function properly and efficiently network engineers need to be kept up to date on the state of the network and be alerted if *any* problems occur. For this purpose there exist two systems in the GSM network architecture that allows for this functionality required by network engineers.

The one system is called the Operations and Management centre which is responsible for centralized regional and local operational and maintenance activities. The other system called the Network Management System unlike the OMC provides global and centralized management for operations and maintenance of the network supposed by the OMCs [71].

We will discuss the OMC and NMC in a bit more detail where we'll define the most critical functions they perform.

Operational and Management Centre The OMC is capable of communicating with GSM entities using two protocols namely SS7 and X.25. The SS7 protocol is usually used when the OMC is communicating within the GSM network over short and medium distances. The X.25 protocol is used for large external data transfers. All communication where the OMC is involved typically occurs over fixed line networks and/or leased lines. The OMC is usually used for day to day operation of a network [71].

The OMC has support for alarm handling. An alarm in a GSM network goes of whenever a predefined expected condition does occurs. Engineers are able to define the severity of an alarm, which defines who and what is further, alerted when and if the alarm is escalated to a higher level [71].

The OMC is also capable of fault management in the GSM network. The OMC is able to activate, deactivate, remove and restore a service manually or automatically of network devices [75]. Various tests can be run as well as diagnostic information can be retrieved on the network devices to detect any current or future defects [71].

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

2.4 GSM Interfaces

In the GSM network all the various entities communicate with each other through various pre-defined interfaces. In this section an overview will be presented on these various interfaces that exist between the entities.

Um Interface — This interface is the link between a MS entity and a BTS and is also referred to as the *Air* interface since communication occurs wirelessly. The primary protocol used on this interface is the LAPDm, which is an extension to 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 [82, 90].

Abis Interface — Between the BTS and BSC the interface used for communication is known as the Abis interface. The only messages that the BTS are interested in are those that have to do with management of radio resources [82,90]. 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 to manage the mobility of MS in its administrative area [48, 82].

Other Interfaces — The MSC has various interfaces going from itself to the various databases and other external network. Each interface connects to a specific database,MSC or network and is therefore very specific as to what function it performs [48,82]. An interface going from one MSC to another MSC will typically convey information regarding handling the administration of handover a MS device leaving the one

administrative area to another MSC's administrative area. The handover procedure is a very delicate process which will be described in section 2.6.

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

2.5 GSM Channels

GSM defines a series of logical channels, which are used for communication over these interfaces. A distinction needs to be made between channels and frequencies. As discussed earlier, a network is licensed a certain section of the wireless spectrum for use for commercial communication. This piece of spectrum is referred to as bandwidth and is measured in Hz, therefore W Hz, where W denotes the allocated bandwidth [122].

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 [122]. This channel of W/N Hz width is also referred to as a frequency. The terms channel and frequency can therefore be used interchangeably. In this dissertation the authors will use the term channel.

Using TDMA the GSM system is able to provide additional transmission capacity by dividing the channel into eight equal time slots [82]. As can be

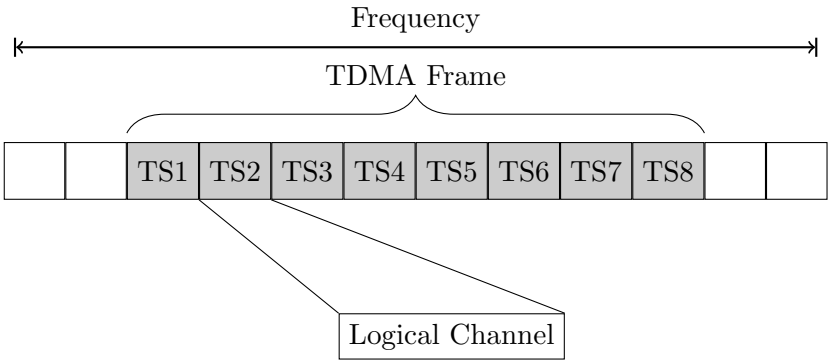

Figure 2.3: TDMA Frame and Logical channels [82]

observed from figure 2.3 each TDMA frame has a series of consecutive timeslots. Each timeslot can be used for both uplink and downlink transmission. A GSM *channel* is a logical channel and refers to a single timeslot within a TDMA frame [48, 82].

The GSM system is hence 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 channels than can be used for communication in GSM is increased eight fold [82].

Frequencies are assigned to the uplink and downlink portion of the connection with a certain duplex separation in the frequency band [6] to avoid interference between uplink and downlink. There are two types of channels, traffic channels and control channels. Traffic channels (TCHs) primary purpose is to enable communication of user speech and data and therefore carry no control information [48].

A TCH is assigned to a MS device when the device requests that it needs to communicate with another device either with speech or data. When a MS is done 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 over the control channels [48].

Control channels are much more active used in a GSM network since it is the primary means by which control and management of the network occurs [48]. These channels are used even when the MS has no active connection and is hence in idle mode. This constant activity on the control channels is to keep the network update with information such as the position of the MS (location updating) and signal strength [28, 48, 71].

The control channels are divided into three main channel groups namely Broadcast Channel (BCH), Common Control Channel (CCCH) and Dedicated Control Channel (DCCH) [48]. Each of these channel groups contains furthermore channels that aid in the control and management of the network. Each group along with the associated channels shall now be briefly discussed.

The first group, BCH consists of three channels:

Broadcast Control Channel (BCH) — This channel is broadcasted using the very first frequency assigned to a cell. Using this frequency,

[6] Separation is usually 45 MHz

the channel broadcasts information regarding the network. This information includes radio channel configuration of the current and neighbouring cells; synchronization information, registration identifiers and most importantly the format of the CCH used by the local BTS [48].

Frequency Correction Channel (FCCH) — Synchronization information is broadcasted to the MSs to enable them to perform frequency correction on the transmission. Typical synchronization information on this channel for instance, is the exact frequency the local BTS is using for transmission to enable the MSs attune itself to the same frequency [48].

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

The FCCH and SCH are always broadcasted with the BCH since these channels are needed for operation of the radio subsystem [48]. The CCCH is a point-to-point signalling channel that is used to localize a MS with the use of paging. The channel is also used to assign dedicated channels [48]. The CCCH is made up of the following channels:

Random Access Channel (RACH) — RACH forms the uplink portion of the CCH channel and is randomly accessed by the MSs to request a dedicated channel for a single signalling transaction [48].

Access Grant Channel (AGCH) — AGCH from the downlink part of the CCH channel. It is used by the radio subsystem to assign a SDCCH or TCH to a MS [48].

Paging Channel (PCH) — PCH also forms part of the downlink portion of the CCH channel. This channel is used by the radio subsystem to page specific MSs, which aids in the process of locating a MS [48].

Notification Channel (NCH) — This channel is used to inform an MS of any incoming group calls or calls that is being broadcasted [48].

The last group of signalling channels are referred to as Dedicated/Associated Control Channels (D/ACCH). These groups of channels have the characteristic that they are bidirectional point to point channels [48].

Stand-alone Dedicated Control Channel — This channel is used for communication between the BSS and MS even when there is no active connection. Hence the 'stand-alone' since it means that there need not be a TCH assigned for communication to occur between the BSS and MS [48].

Slow Associated Control Channel (SACCH) — When a TCH or SDCCH is assigned, there must be accompanied SACCH 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 synchronization information. Packets must be continuously sent over the SACCH as it is used as proof that there is still a physical radio connection [48]. When the MS is done using the SACCH channel, it transmits a report regard the current results of the radio signal level, which is continuously measured [48].

Fast Associated Control Channel (FACCH) — When more bandwidth is required for signalling purposed, the signal of the TCH is modified using dynamic pre-emptive multiplexing. The additional bandwidth comes at the expense of the user date transport. When a channel is created in this manner it is called a FACCH [48].

Finally, one last channel is defined named the Cell Broadcast Channel (CBCH), which shares the same physical channel that the SDCCH uses. On this channel messages of the Short Message Service Cell Broadcast are broadcasted [48].

We have now explained the *logical* channels used for communication between a 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. In the next section, we will discuss how the network still enables a MS to be connection and makes calls while it is moving around geographically within.

2.6 Handover

In this section a discussion will be given describing the process GSM network entities go through when a handover needs to occur. The Handover process

in a GSM network is initiated when a MS with an active call moves outside the coverage area of a cell, BSS or MSC [28, 48, 82]. Besides the MS moving outside the coverage area, a handover might also be initiated because of measurements indicating bad channel quality [48].

Various information needs to be migrated across and shared between the entities to ensure a smooth handover, which results in the MS active call not suddenly ending. It is just not the GSM architecture entities that need to continuously share information but also the MS. An MS is required to continuously observe and measure signal strength of up to 6 neighbouring cells. The MS does this by monitoring the BCCH [48,82]. This information is of course relayed to the MSC and BTS as it plays a critical role in the decision process as to which entity will be the best in taking over the administration of the active call [48, 82].

A MS can in some cases receive the same BCCH from different cells, which are most likely neighbours [48]. This problem of duplicate BCCH from different cells can be attributed to the frequency reuse in the cellular network as well as due to the smaller sector cells forming clusters and therefore overlapping coverage area [48]. As discussed in section 2.3.1 page 12, cells are typically divided into sectors, which lessen the problem of co-channel interference. This division into sectors can cause clustering of cells and overlapping of coverage area.

This forms a problem for the MS since the it needs to distinguish between the two cells measurements as it does not know which measurement belongs to what cell [48]. 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 reliably determined are included in the report sent to the BTS [28, 48, 82][7]. Using this report the handover algorithm is able to decide how to handle the handover, which cell needs to take over the call [28,48,82].

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 [28, 48, 82]. 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.

[7] Cells whose identity can’t be determined can be due to environmental factors like signal strength or to packets getting lost/dropped

disconnected [28].

There are three core handover procedures when a handover needs to occur in a GSM network. *Intra-BSC*, *Inter-BSC* and *Inter-MSC* each of which involves different GSM entities [82]. A brief discussion on each handover will now be presented.

Intra-BSC — Also known as *Intercell handover*, this handover is concerned with the transfer of a active call / connection from a MS to another cell which is controlled by the same BSC as the current cell. The current BTS that manages the active call of the MS constructs a report containing measurement information from the MS, as well as measurements the BTS has taken on signal strength and error bit ratio of the current connection [48, 82].

The constructed report is forwarded to the managing BSC who analyses the report on the necessity of handing over the active call to another BTS. If a handover is deemed necessary the BSC starts by initializing the BTS to prepare it to handle the new connection [48, 82].

The BSC then notifies the MS through the old BTS of the new BTS identity, and various properties of the new connection such a 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 [48,82].

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 is 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 [48,82].

Inter-BSC — In an Inter-BSC handover a call of a MS that is being managed by a BTS is transferred to another BTS who has a *different* controlling BSC. Thus, the call is essentially moved between two BSCs, its up to the new control BSC to select a suitable BTS who will actually handle the call of the MS being handed over [48, 82].

This handover occurs due to the MS 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

the new BSC is able to make suitable provision to assume control of the call within one of its BTSs [48, 82].

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 [48, 82].

The new BSC informs the MSC of the new connection details which the cell will use to handle the handover and maintain the active connection of the MS. The MSC forwards the connection information to the old BSC who forwards it to the MS. The MS makes the necessary adjustments for it and then moves on to the new connection. The MS informs the new BSC that the handover has been completed successfully. The new BSC informs the MSC, who then instructs the old BSC to relinquish the old connection used by the MS [48, 82].

Inter-MSC — With an Inter-MSC handover, control and management of an active call on a MS must be transferred to another BTS, which resides in another different area that is managed by a different MSC. The handover process follows the same basic formula as the Intra-BSC and Inter-BSC handover once the handover request is made [48, 82]. The BSC managing the BTS to which the MS is going to be handed over to, is to be determined by the MSC. The BSC is notified by the new managing MSC that certain BTS must bring a new connection online for the incoming MS; the entities upstream[8] are notified [48,82].

The new MSC informs the old MSC of the new connection details. The old MSC transfers the connection information to the MS in question who is going to be handover to another BTS. The MS makes the corresponding adjustments where it then starts operating on the new connection. The MS informs the BSC of the successful handover [48, 82].

The BSC in turn informs the new MSC, who in turn informs the old

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

MSC that the handover was successful. The MSC then instructs the BSC to ensure the old connection resources are relinquished by the BTS.

In this section a discussion was presented on the process the GSM network follows as a MS device moves around geographically in the network to keep a active call on the MS active and not get disconnected. An outline was on what effect this has on channel selected as well as the different handover procedures. In the next section a summary of this chapter will be presented.

2.7 Summary

In this chapter a broad discussion was given on modern cellular technology specifically the GSM cellular network technology. A brief history on how GSM was developed to be the most widely used cellular technology in use today was presented in section 2.2.1. A discussion on the various GSM architecture entities followed the history on GSM. In the GSM architecture a overview of all the entities present in a modern GSM network was presented.

Following the discussion on the GSM entities, a broad overview followed on the various communication interfaces used between the various GSM entities to communicate with each other in section 2.4. Since the communication interfaces were discussed a section on GSM Channels followed which defines the different channels which are used on the interfaces to communicate information.

The chapter ends off with a discussion on the handover process which is used to allow a MS device to move freely geographically within the network.

Chapter 3

The Frequency Assignment Problem

3.1 Introduction

The Frequency Assignment Problem (FAP)[1] is a generalisation of the graph-colouring problem and is subsequently an NP-Complete problem [26]. The FAP is a NP-Complete problem due to fact that one only has a finite amount of frequencies which can be assigned to antennae/transceivers (TRX's), where the amount of transceivers to be assigned frequencies greatly out weigh the amount of available frequencies [26].

In wireless communication a big concern is a notion known as interference which will be discussed in detail in section 3.4. Essentially for the FAP the primary concern is to develop an approximate plan on assigning frequencies in such a way that interference is kept to a minimum.

Using exact algorithms to find a solution is not practical since the time to find a solution will be polynomial. Generally Meta heuristic algorithms are used to find optimal solutions to NP-Complete problems [65]. We will discuss Meta heuristic algorithms that are generally used to find solutions for NP-Complete problems in Chapter 4. A definition on what it means for a problem to be NP-Complete will be given in section 3.2.

A wireless cellular operator is not allowed to just operate on any frequency. A governing body license a certain piece of the available wireless

[1] Also known as Automatic Frequency Planning (AFP) or Channel Assignment Problem (CAP) [65]

spectrum to the operator for use in their network [26]. These frequencies that need to be licensed for use are known as *commercial* frequencies and are very scarce as it is immensely expensive to license these frequencies [26].

Usually a licensed piece of spectrum contains a series of consecutive frequencies as well as gaps. Where the gap frequencies are barred from being used by any device within the network. These frequencies are barred from use as they may have been allocated already to another operator for use [12]. By barring frequencies, a scenario is avoided where the different networks equipment interfere with their respective operations [12]

Due to the whole spectrum not being available to network operators and only a subset is for commercial communication as per the frequencies allocated to them, networks opt to reuse their frequencies [12]. 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, it needs to be licensed [26].

It is not always possible for simply allocate more frequencies to a network even if the network pays the associated fees. The whole commercial spectrum could have already have been licensed to various different entities. Hence, licensed frequencies are a very valuable and scarce commodity [2, 12, 26, 28].

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

In this section we gave a brief introduction as to what the Frequency Assignment Problem is and why it occurs as well as why it is a problem. In the next section an overview of what it means when a problem is NP-Complete will be presented.

3.2 NP-Complete

The term NP stems from the field known as complexity analyses. Algorithms are typically measured for their worst running time using On notation. The field of complexity analyses is more interested in measuring complexity of a problem than the running time of an algorithm [100].

The field of complexity analyses makes a keen distinction between problems that can be solved by algorithms in polynomial time and problems that cannot be solved in polynomial time using any available algorithm [100]. Polynomial time refers to a timespan that is reasonably feasible like for instance 8 hours or 1 day.

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

Problems that can be solved in polynomial time usually have worst case running times of $O(n), O(\log n)$ and $O(n^2)$. These class of problems are relatively easy to solve using most algorithms available today. Problems that can be easily solved with this kind of running times are classified as being in the P range of complexity problems [100].

Another range of problems exist which consists mostly of problems that cannot be solved in polynomial time. Hence, if an algorithm were to try each and every possible solution, it would take an arbitrarily long time, which cannot be determined, which is why these problems also have the characteristic of being non-deterministic. Problems in this range are referred to as being in the NP range of complexity problems [100].

There is another range of NP problems that are a subset of NP problems, which are referred to as the "most extreme" problems within NP. These problems are collective known as NP-Complete problems and are the most difficult problems to determine feasible solutions for in the NP problem range [100].

The Frequency Assignment Problem (FAP) is one such problem, which has been proven to be within this difficult class of problems known as NP-Complete [2, 27, 28, 79, 137].

In this section a discussion was given on NP and NP-Complete problems. A definition was given on what it means for a problem to be classified as being in the NP range of problems. In the next section a discussion will be presented on the two different frequency allocation schemes that exist within the FAP problem domain.

3.3 Frequency Assignment Types

In this section a discussion will be presented on the different methods that exist to allocate frequencies to cells in a cellular network. Furthermore the method that relates to the specific FAP variant this dissertation is concentrated on will be stated and discussed.

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

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

- *Fixed Frequency/Channel Assignment (FFA/FCA)* is where channels assigned to cells are static; therefore they cannot be changed until a new assignment plan is calculated.
- *Dynamic Frequency/Channel Assignment* (DFA/DCA) is the process of allocating channels to cells as they require it to meet the current traffic demand imposed on them by clients.

A discussion on each of the above assignment schemes will be presented in the following subsection.

3.3.1 Fixed Frequency/Channel Assignment (FFA/FCA)

Fixed Frequency/Channel Assignment (FFA/FCA) is the process of permanently assigning frequencies to cells (cellular towers). The frequencies assigned are fixed and cannot be changed immediately changed 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 [111].

If a channel assigned to a specific cell is suddenly changed, the cell might start interfering with neighbouring cells communication links. Hence, if the cell is sectored[2], the cell can interfere with a minimum of three and up to a maximum of six neighbouring cells [111].

[2]link to discussion of sectorization of cells in chapter 2

When a FCA plan is created, cells are assigned frequencies based on the estimated traffic that cell will be expected to handle during peak network usage. FCA is ideally suited for macro cellular networks since the nature of the traffic encountered in such networks has the characteristic of being homogeneous, stationary and predictable [111]. Cellular networks can be classified as being in the macro cellular group of wireless networks.

With FCA networks are able to permanently allocate a certain subset of frequencies to cells since the nature of the traffic on their network allows them to predict with reasonable certainty the call blocking probability [111]. A Call is blocked on the network when a cell has no available channels to use when establishing a communication link [111].

In situations where the nature of the traffic is neither homogeneous nor stationary using the FCA allocation scheme is not feasible as its use of available frequencies is grossly inefficient [111]. With FCA, if there is a new call or hand off in a cell where all the permanently assigned frequencies are in use, the call will be blocked, even if adjacent cells have available frequencies, which can be used to handle the call [111].

In this sub section we described what FCA is, in which networks it is typically used and also outlined where its use is not feasible. In the next sub section an overview of the DCA allocation scheme will be presented.

3.3.2 Dynamic Frequency/Channel Assignment (DFA/DCA)

Dynamic Frequency/Channel Assignment (DFA/DCA) is a channel allocation scheme where frequencies assigned to cells are not permanent but rather assigned to cells as the need arises [111]. Therefore all the frequencies licenses by a particular network, is available to each and every cell to establish a communication link as long as the channel does not violate the co-channel reuse constraint [111].

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

The DCA allocation scheme is ideally suited for micro cellular wireless networks since the traffic on these networks have the characteristic of being immensely unpredictable as traffic demand varies constantly [**?**, 36, 111].

As the name dictates, micro cellular wireless networks have much smaller cell sizes than macro cellular networks. Thus a cell in a micro cellular network must handle a lot more hand off traffic than a cell in a macro cellular network, since a MS with an active connection is much more likely to move out of the coverage area of a micro cell than a macro cell [**?**,36,111].

Since a micro cellular network has increased hand off traffic compared to a macro cellular network, a DCA scheme must be fast with the allocation of frequencies to requesting cells that must handle the hand offs [**?**,36,111].

DCA is much more efficient than FCA when the amount of mobile traffic on the network is relatively low. On the other hand, when the network is under heavy mobile traffic load, the FCA scheme outperforms the DCA scheme, since the DCA usually allocates frequencies to cells in an inefficient arrangement that might the amount of interference encountered on the network [**?**,36,111].

Finally DCA has the inherent requirement that it requires a great deal more computational power that FCA, since the frequencies need to be selected and allocated with great speed otherwise the cell requesting a channel won't be able to handle the call and therefore block the call or drop the call [**?**,36,111].

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

In this sub section an overview of the DCA allocation scheme was presented. An outline was given on what the DCA scheme entails, what its advantages as well as disadvantages is. This concludes the discussion on the different allocation schemes that is used in modern cellular networks. In the next section a discussion will be presented on the different industries where FAP is encountered.

3.4 Interference

This section discussion will focus on Interference. A description will be given of what interference is and why it is important for cellular networks as well as an overview will be given of when interference occurs.

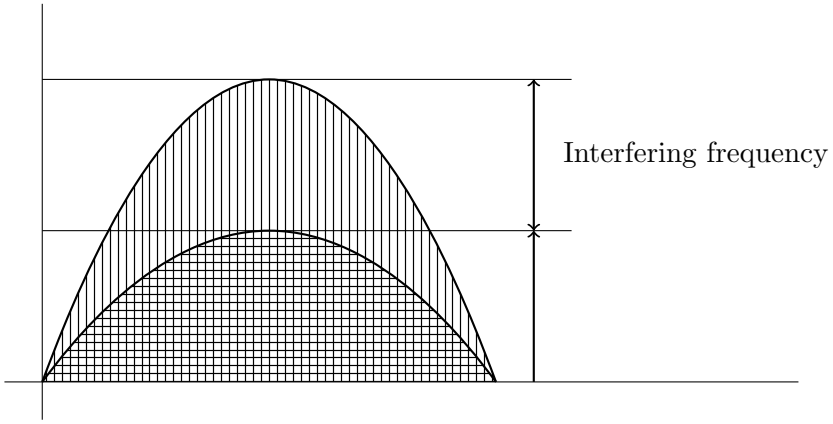

Figure 3.1: Co-Channel interference

Interference can be defined as any unwanted signal that is received along with a signal of interest. The unwanted signal is said to *interfere* with the original signal and as a consequence degrades the original signal quality with unwanted information [32].

Interference usually occurs when two or more entities communicate independently on the same channel or on adjacent channels [32,36]. Other external factors can also contribute to interference on a communication link, such as machines, which inherently produce some sort of electromagnetic distortion like for instance a cars ignition or a big turbine [32,36].

Interference that occurs when two signals operate on the same channel can bee seen in figure 3.4 and interference that occurs as a consequence of

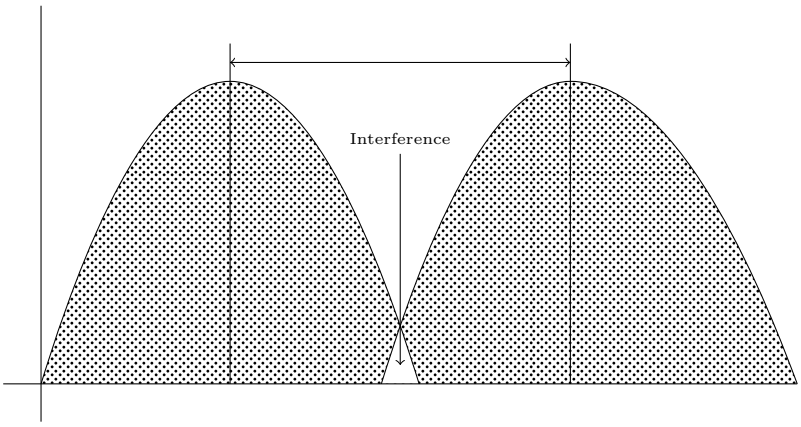

Figure 3.2: Adjacent Channel interference

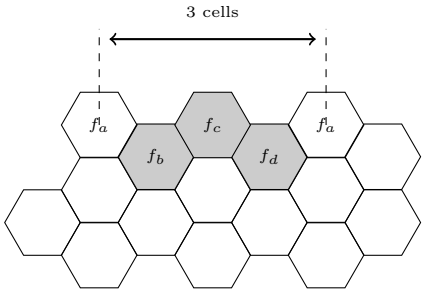

Figure 3.3: Frequency Separation

two signals operating on adjacent frequencies can be seen in figure 3.4.

The impact interference will have on entity, that has established a connection and that operates on the same channel or adjacent channel as another entity, decreases as the geographic distance between them increases [28, 32, 36, 101]. Therefore, to minimise the impact interference will have on communication links a *separation* is defined [28,32,36,101]. More specifically this separation is known as the channel reuse or frequency reuse distance within wireless networks [28, 32, 36, 101].

This separation is defined as the minimum amount of cells (which must all use different channels) one cell, which has been allocated a channel, and another cell must be apart before a cell is allowed to reuse the same channel another cell has been allocated [28, 32, 36, 101].

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

As discussed earlier, cellular networks are forced to reuse their licensed frequencies multiples times to keep cost to a minimum. Therefore, the design of a cellular network is limited to the defined separation distance between cells as it defines the size cells will be in the network. Smaller cells can lead to a larger separation distance compared to when cells are larger [28,32,36,101].

Cellular networks use the amount of interference on their networks as qualitative measure for their QoS. A network with high interference would experience a lot of dropped connections/calls, which occurs when the interference is too high to sustain a connection or call for communication;

consequently their QoS degrades as interference increases [32, 36].

Even though interference can cause a call or connection to be lost i.e. dropped, there are other situations where a call can be dropped due to other factors. For instance, a call can be dropped when a handover procedure occurs between two cells and the one cell receiving the call is at full utilization of its allocated frequencies [32, 36, 71].

In the literature a variety of methods are given to calculate the amount of interference in a network. The signal-to-noise (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-plus-noise power ratio (SINR) but since cellular networks are interference limited the noise is hence neglected in the interference calculation [36]. Noise can be neglected since the power of interference is much larger than the power of noise [32, 36].

A formulation of the SINR and SIR will now be presented.

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

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

Where P_r is the power of the received signal and P_I is the power associated with interference from within a cell (intra cell interference) and interference from outside a cell (inter cell interference) [36].

This calculation can be considered a best guess as it models the environment, weather and other factors which may influence the potential interference at a point with a Gaussian distribution for noise represented by the N_0 [2, 36].

Using the SIR formula cellular networks are able to determine the *bit-error rate* (BER) users on the network will experience on their connections [32, 36]. Where the BER is defined as the probability that a received bit on the connection will be incorrect [32, 36, 104].

As the BER increases voice quality on the connection decreases since more bits that are used to describe the voice information is incorrectly received. SIR and BER probability are interlinked. As SIR increases, which is to say, less interference is encountered on the communication link and

therefore the probability that bits will be received incorrectly decreases [32, 36, 104].

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* [65].

An Interference Matrix consists of a number of cell pairs (i,j), where i is the cell receiving interference and j the cell whose allocated channel is providing the interference. Each cell pair in the matrix has two corresponding values that indicate the level of interference if the *Electromagnetic constraints* are violated [2, 27, 28, 65].

Primarily Interference occurs when the Electromagnetic constraints are violated, which are defined as:

Co-Channel — As discussed earlier, when a cell i and a cell j operate on the same channel interference will occur [2, 28, 32, 36, 65, 71, 101, 103, 111]. When this type of interference occurs it is referred to as *co-channel* interference.

Adjacent Channel — When a cell i and a cell j operate on adjacent channels, their allocated frequencies differ by one i.e. cell i operates on channel f then if cell j operates on either channel f - *1* or f + *1* then interference will occur [2,28,32,36,65,71,101,103,111]. 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, 27, 28]. 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 [**?**, 2, 29, 137]. Where cell i and cellj serve different sectors.

Co-Cell — Channels used on the same antennae of a cell must differ by a certain number which is typically set to 3 but can be any number

larger than 0 the network operator deems necessary to avoid unwanted interference [**?**, 27, 28].

Handover — This constraint imposes that frequencies must differ by a predefined margin i.e. 2 or 3 when one cell hands over a call to another cell. If this constraint is violated a mobile subscriber will experience a dropped call since the handover between cells fails [**?**, 27, 28].

Within the licenses of wireless networks there are two hard-constraints set forth which forbid networks from using certain frequencies. Where hard-constraints means that under no circumstance are these constraints allowed to be violated.

The first set of hard-constraint frequencies are known as *Globally blocked channels*. Channels that are in the set of globally blocked channels are usually frequencies that have been licensed to other networks [2, 28, 101].

The second set of hard-constraint frequencies are known as *Locally blocked channels*. These channels are only not allowed at certain geographic areas but are free for use at any other area [2,28,101]. A typical area where certain frequencies will be forbidden to be used is near a country border [2, 28, 101]. The locally blocked frequencies are most likely in use by another network resident to the bordered country.

In this section we described what interference is and what the consequences are of too much interference in a network. We also laid out under which circumstances interference can occur in a wireless network. In the next section and overview will be given the various different sub problems that exist in the FAP problem domain.

3.5 Frequency Assignment Problem types

In this section a brief discussion will be presented on each of the various problem variants that exist for the FAP. The discussion will start on one of the first and oldest problems in the FAP problem domain. This section will conclude with a final discussion on the particular variant of FAP that this dissertation will concentrate on.

3.5.1 Minimum Order FAP

The Minimum Order FAP (MO-FAP) was the first FAP that emerged in the 70's. The MO-FAP is concerned with assigning frequencies to transmitters while interference is minimized as well as minimizing the amount of different frequencies that are used.

In MO-FAP channel re-use is prioritised and the usage of a channel has a certain cost associated with it. The reason for this is because when the wireless network industry started out, operator's were billed according to the amount of different frequencies they used. In the beginning frequencies weren't cheap since they were sold per unit [2, 79].

Over the years as the law governing the wireless spectrum changed and new technology as well standards emerged, thus MO-FAP has lost its relevancy. Companies aren't billed according to the different frequencies they use, but they purchase licenses from a regulatory body. This license usually stipulates what channel band the network is allowed to use.

In some instances a certain band of frequencies is put up for auction by a regulatory body, to which interested parties can bid to own the specified spectrum. Due to the shift in how frequencies are allocated to network, neither the regulatory bodies nor the network operators care about the amount of different frequencies are used. Thus MO-FAP has lost its relevancy in the modern wireless industry.

3.5.2 Minimum Span FAP

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

The span is defined as an interval on the frequency domain. This interval is calculated by taking the difference of the maximum and minimum frequencies used during assignment. 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, 79, 99].

The MS-FAP and MO-FAP are two very similar problems, the only difference is that MO-FAP focuses on minimizing different frequencies and MS-FAP focuses on minimizing the interval of frequencies used during assignment [2]. The Philadelphia benchmark is usually used to gauge how good the algorithm performs.

3.5.3 Minimum Interference FAP

The Minimum Interference FAP (MI-FAP) or Fixed Spectrum FAP (FS-FAP) is typically encountered after the network operator has obtained a frequency band from a regulatory body. Other problems use matrices to forbid certain frequencies from being with certain transmitter [2, 28, 39, 79].

Unlike previous discussed 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. The interference value doesn't play a large role in their respective objective functions. In MI-FAP the objective is to minimize the total amount of interference on the network. It is important to note that this amount of interference might not necessarily be zero [2, 28, 39, 79].

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

Since MI-FAP is very close to real world instance problems, authors tend to use real world instances or benchmarks that resemble real world instance to test the quality and efficiency of their algorithms [2, 28, 39, 79]. We'll benchmark the quality and efficiency of our solution with the COST 259 benchmark which is discussed in section 3.7.

In this section we laid out the different types of Frequency Assignment Problems there are in the literature. We also gave a brief discussion on some of the literature found on the individual problems. Finally we formally stated on which one of the frequency assignment problems we will be concentrating on, namely the Fixed Spectrum Minimum Interference Frequency Assignment Problem (MI-FAP).

In the next section we will give a Mathematical definition for the Fixed

Spectrum MI-FAP that will form the bases for the objective/cost function that we are going to minimize to find an optimal frequency plan.

3.6 Fixed Spectrum MI-FAP Mathematical Formulation

In this section we will give a Mathematical definition of the Frequency Assignment Problem that will form the core of the algorithm discussed in this dissertation will optimize. We'll start of by denoting the symbols we will use and then we will give the Mathematical definition of the cost function we will minimize.

The Frequency Assignment Problem can be represented as a graph colouring problem hence it is known to be NP-Complete. Before we can formally define the Frequency Assignment Problem we first need to introduce some symbol definitions.

$$G = (V, E) \tag{3.3}$$
$$V = \{v_0, v_1, ..., v_i\} | i \in \mathbb{N} \tag{3.4}$$
$$E = \{v_0v_1, v_0v_2, ..., v_iv_j\} | v \in V, \forall ij \in \mathbb{N}, i \neq j \tag{3.5}$$
$$D = \{d_{01}, d_{02}, ..., d_{ij}\} | \forall \{i, j\} \in E, \exists d_{ij} \in \mathbb{N}^+ \tag{3.6}$$
$$P = \{\{p_{\bar{0}0}, p_{\bar{=}01}\}, \{p_{\bar{1}0}, p_{\bar{=}11}\}, \ldots, p_{\bar{i}0}, p_{\bar{=}i1}\}\} | \forall \{i, j\} \in E, \exists p_{ij} \in \mathbb{N}^+ \tag{3.7}$$
$$F = \{0, 1, 2, 3, ..., k\} | \forall k \in \mathbb{N}, \forall v \in V \exists f \in F \tag{3.8}$$
$$d_{ij} < |f(i) - f(j)|, \forall ij \in \mathbb{N}, i \neq j \tag{3.9}$$

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

E (see 3.5) is a set of edges. An edge consists of two vertices v_i and v_j that are joined because there exists a constraint on the frequencies that can be assigned between the two vertices or transmitters. Each edge has two associated labels d_{ij} and p_{ij} [13, 80].

The label d_{ij} that is part of the set D (see 3.6) denotes the maximum separation that is required to exist between frequencies assigned to two transmitters v_i and v_j. Using $f(i)$ to denote the frequency assigned to i, we

can determine using equation 3.9 if the interference involving the transmitters v_i and v_j is acceptable [13, 80]

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

- $\bar{p_{i0}}$ represents the value for co-channel interference [13, 80].
- $\bar{\bar{p_{i1}}}$ represents the value for adjacent channel interference [13, 80].

Lastly we have the set F (see 3.8) that denotes a set of consecutive frequencies for every transmitter in V [13, 80].

Formally the Fixed Spectrum Frequency Assignment Problem (FS-FAP) can now be defined as a 5-tuple $FS-FAP = \{V, E, D, P, F\}$ with a required mapping of $f : V \rightarrow F$ [80]. The objective of the FS-FAP is to find an assignment of frequencies to transmitters that minimize the sum of total interference (see 3.11).

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

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

In this section we Mathematically defined the Frequency Assignment Problem using the symbols we defined. In the next section we will give a brief discussion on the different Frequency Assignment Benchmark Problems that exist and also define the benchmark we will be using in our implementation.

3.7 FAP Benchmarks

In this sections will discuss some of the most used benchmarks in the FAP domain. We will start of with the first benchmark that was introduced in the 70s and end of with a discussion on the benchmark we will be using to test our implementation.

[3] Discussed in section 3.4 page 32

[4] Interference values can be zero in some cases

3.7.1 Philedelphia Benchmark

The Philedelphia benchmarks are derived from an instance that was introduced in 1973 by Anderson. Each instance is a hexagonal grid of cells that overlaps the area of interest. At the centre of each cell there is a transmitter. Past approaches used these hexagonal systems to model modern cellular networks [2, 68].

In this benchmark interference is measured by a co-channel reuse distance. This distance stipulates that the difference of the frequencies assigned to two cells must greater of equal to a certain value d. A channel cannot be assigned to a cell if it violates this minimum distance [2, 68].

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

3.7.2 CELAR

In 1994 EUCLID introduced a project called CALMA, which was a combined effort by various European governments that were part of EUCLID to investigate algorithms for Military applications. The project was granted to six research groups. Within the project 36 instances were made available by CELAR for Radio Link Frequency Assignment [2, 25].

All the CELAR instances have the constraint that the difference between frequencies assigned to interfering radio links must be greater that a certain predefined distance in the frequency domain. This is a soft constraint and may be violated. Another constraint in the CELAR instances is that each pair of parallel links must differ by an exact predefined distance. This constraint is a hard constraint and may not be violated [25].

These instances were initially not available to the general public as it was contained to be within the CALMA project. In 2001 the CELAR launched the International ROADEF challenge, were certain instances from the CALMA project were made available for the research teams taking part in the challenge. The instances made available had been modified to take polarizations and controlled relaxations of certain EMC constraints [44].

3.7.3 COST 256

The COST (COoperation europene dans le domaine de la recherche Scientifique et Technique) 259 is a set of real world GSM instances made available by die 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, 28].

The instances are fairly difficult due to the large amount of transmitters (900 - 4000) that need to be assigned frequencies, with a relatively small amount spectrum of frequencies. The main important characteristic of this benchmark is that is resembles real world GSM network data, which is why we the authors have selected this as the primary benchmark we will be concentrating on [2, 45].

More specifically we will concentrate on a small subset of the instances that are available, namely Siemens1, Siemens2, Siemens3 and Siemens4. In the paper by Montemanni and Smith [80] the same subset of problems is used and to date their algorithm has produced some of the best results.

The characteristics of each instance will now be discussed.

Siemens1

The Siemens 1 instance resembles a GSM network that follows the GSM900 standard. This particular network has been allocated a spectrum set of frequencies $F = 16, 90$ 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. The allocated frequencies are split into two because according to the problem instance frequencies ranging from 36 to 67 are globally blocked, thus frequencies ranging from 16 - 35 and 68 - 90 are available for assignment.

This problem instance finally also defines that this network consists of a total of 506 cells where on average each cell has 1.84 transceivers that needs to be assigned a frequency. The Co-Site separation is stated to be 2 and the Co-Cell separation is stated to be 3.

Siemens2

The Siemens 2 problem instance describes a GSM network based on GSM900 and has 86 active sites. The problem specifies that the network consist of 254 cells where each cell has on average 3.85 tranceivers that need to be assigned frequencies.

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

Siemens3

The Siemens 3 problem describes a network based on GSM900. This network has been allocated a continous set of frequencies that starts at 681 and ends at 735. Thus the network has 55 frequencies, which can be allocated to tranceivers in its networks.

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

Siemens4

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 set of frequencies starting at 56, thus $F = (56, 94)$. No frequencies are said to be globally nor locally blocked in this network.

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

In this section a discussion was presented on the various benchmark functions that are available to researchers to test their frequency assignment algorithms. This section was concluded with a discussion on the benchmark the authors will be using the test the algorithm presented in this dissertation.

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

3.8 FAP in the industry

In this section we will list some of the industries where the FAP is encountered. We provide a brief overview how the problem differs compared to other industries. We will also give some references to literature where the FAP and the particular domain are discussed.

3.8.1 Satellite communication

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

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

A large distance in the frequency domain separates the frequencies used by the ground terminals for uplink and downlink communication. The typical distance is much larger than the bandwidth. When frequencies are assigned to transmitters, downlink transmitters are ignored and only uplink transmitters are considered [2].

A radical difference with regard to the use of frequencies compared to the standard FAP in Cellular Networks is that, frequencies are only allowed to be used once. This is specific to the satellite domain to avoid interference [2].

3.8.2 Wireless mesh networks and WLANs

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

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

Multiple WLANs are increasingly being used to provide backbone support for large fixed line networks, enterprise networks, campuses and metropolitan areas. To be able to provide backbone support for these networks, a primary design goal when designing and deploying these networks is capacity. A limiting factor for WLAN capacity is interference, which affects multi-hop hop settings. Thus the overall network interference needs to be minimized to increase the capacity of the network [112].

Typical approaches allocating frequencies include using DCA and FCA[6]. DCA isn't very popular since the dynamic switching of channels lowers the response time on commodity hardware since there is a delay in milliseconds when switching channels. Typical packet transmission times are in microseconds. To guarantee uptime and high responsiveness, FCA is the preferred approach [112].

The FAP in Wireless Mesh networks and WLANs differ to 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* [112]. Another aspect to consider is the placement of access points (AP) in the network, which is similar to the problem cellular networks face with regard to base station placement [2].

3.8.3 Military field communication

In a Military context the FAP is a very difficult problem to be solved due to its dynamic nature. During deployment connections need to be established rapidly between nodes with not guarantee that the nodes would stay static at locations. Usually nodes are military field phones can be any transceiver device [1, 25].

Due to the nature of the problem the DCA scheme is used to allocate frequencies to nodes. The Military FAP differs due to the property that any of the nodes are mobile and can move at any moment to a new location, potentially interfering with another connection [1,25]. 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

[6] Discussed in section 3.3 on page 30

communication interfering [1, 25].

A lot of literature can be found on Military field communication. This is due to two organizations CELAR[7] and EUCLID[8] making data available to various research groups and allowing them to develop algorithms for frequency assignment [1, 25].

3.8.4 Television and Radio Broadcasting

The FAP encountered in broadcasting very much resembles the problem domain found in Cellular networks. The only notable difference is the required distance allocated frequencies must differ in the frequency domain is 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 broadcasting as a main topic. Research that specifically discusses FAP in broadcasting is presented by Idoumghar and Schott [47]. The authors present a distributed hybrid genetic algorithm and a cooperative distributed tabu search algorithm. They compare these algorithms with their sequential counterparts of their algorithms and with an ANTS algorithm. The benchmark instances they used were provided by the TDF-C2R Broadcasting and Wireless research centre.

3.8.5 Cellular Communication

Cellular communication[9] can be considered the main driving force with regard to research in the Frequency Assignment domain. As new standards are developed and used in 3G networks, in general a frequency assignment problem still needs to be solved. Since these newer technologies still use GSM as their backbone architecture as discussed in section 2.2.1 on page 8.

There is a wealth of research that concentrate on the FAP within cellular networks. This is because cellular networks are used by millions of people around the world and as such presents an interesting notion to produce better results. Since viable solutions have the possibility to impact millions

[7] Centre Electronique de L'Armement

[8] European Cooperation on the Long Term Defense

[9] An overview of a Cellular Communication technology called GSM is presented in Chapter 2.

of people. As such, 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 [28].

Because the problem is NP-Hard most presented algorithms are either of the meta heuristic type or more recently of the swarm intelligence type. Both of these algorithmic types are discussed in Chapter 4 and 5 respectively.

In this section a discussion was presented on the different industries in which the Frequency Assignment Problem is encountered and also a brief description was given on how the problems differ with regard to what constraint each problem domain imposes on the frequencies for the particular industry. Brief references were given to some literature where the FAP is discussed with regard to the particular domain. In the next section a brief description will be given of the different types of techniques that are used when frequencies are assigned.

3.9 Summary

In this chapter has discussion was presented on the problem this dissertation will be based upon. The problem was defined as being the Frequency Assignment Problem (FAP). A brief discussion was given on the problem where the problem was categorized as being part of the set of NP-Complete problems. The NP-Complete nature of the problem is an important concept to understand which is why a section describes what NP-Complete means was presented.

Within the frequency assignment problem domain there exists two different techniques when assigning frequencies. The two different techniques was discussed in section 3.3.

An important concept that needs to be understood to understand why the frequency assignment problem exists is the concept of interference. An in depth discussion on interference was presented in section 3.4.

The frequency assignment problem 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 amount of frequencies used, other are more concerned with the amount of interference that is generated on the network because of the assignment. Section 3.5 is where each sub problem is discussed.

In this section about the various FAP sub problems, a formal definition was given on what sub problem this dissertation will be concentrating on, namely, the MI-FAP.

In section 3.6 a formal mathematical definition is presented on the MI-FAP. Finally, the chapter concludes with an overview on the various benchmarks problems which are used to test the viability of frequency assignment algorithms.

Chapter 4

Meta-heuristic Algorithms

4.1 Introduction

Meta-heuristics is a sub domain of the artificial intelligence domain. It evolved out of a need for more efficient search techniques with regard to hard problems.

Meta-heuristics forms part of a collective body of algorithms that use heuristics to search a particular domain's problem space, for the most optimal solution adhering to certain hard and soft constraints [100, 136].

A hard constraint is defined as a certain condition an algorithm or potential solution is not allowed to violate [2, 28, 100, 136]. A soft constraint is allowed to be violated but there is some sort of penalty or cost involved which is imposed onto the potential solution, which lowers its desirability [2, 28, 100, 136].

An optimal solution would therefore be any solution that violates no hard constraints and violates none or a minimum number of soft constraints [2, 28, 100, 136].

Some of the most algorithms that are classified as being part of the collective body of algorithms known as meta-heuristic algorithms are:

- Tabu Search
- Simulated Annealing
- Genetic Algorithm

The above mentioned algorithms are not the only algorithms to form part of this sub-domain, but they are the algorithms that have received the most attention in the literature and generally produce good results [73].

The main focus for this chapter will be to discuss each of the above listed algorithms in detail. Before each of the algorithms are discussed, a section will be presented that gives a brief overview on the various characteristics that meta-heuristic algorithms exhibit.

Following the discussion on the characteristics there will be three sections that will discuss the above mentioned algorithms along with a brief literature study for each algorithm.

4.2 Characteristics of Meta-heuristics

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

These traditional search algorithms are concerned with the path taken from a starting solution to a final solution. Hence, the algorithms test each and every possible solution and store any alternatives in memory since a final solution constitutes the path taken from a start node to a final node [100].

It is not always viable to test each and every possible solution in a given problem search space, especially in NP-Complete problems since their search spaces are usually huge or infinite which is why traditional algorithms are not able to produce optimal solutions in polynomial time [100].

With NP-Complete problems, the path taken from an initial possible solution to a final optimal solution is irrelevant [100]. Only the final optimal solution is needed. Algorithms that disregard the path taken to a solution and only care about the final solution are classified as being *Local search* algorithms [100].

Local search algorithms tend to use a small amount of memory since the algorithm is only concerned with the current state and only moves to a neighbouring state when searching for a potential solution. This makes Local search algorithms a good fit for optimization problems as a optimized

[1] A discussion of NP-Complete problems is presented in section 3.2 page 28

solution is just a solution which constitutes to the current best state found by the algorithm [43, 100].

At its most basic form, a local search algorithm is just an algorithm that moves form one state to another. How it generates neighbouring states to move to and how why it moves to a certain state as well as how it moves towards that state, makes the local search algorithm unique [31, 43, 100].

Meta-heuristic algorithms are considered to be *general-purpose* algorithms and can thus be applied to a wide variety of optimization problems with only small modifications that need to made to the algorithm model [66].

As can be gathered from the name, a meta-heuristic algorithm uses some sort of heuristic. A heuristic at its most basic form is a decision rule. Algorithms use heuristics to make decisions by applying the heuristic to the data the algorithm is currently working on [100, 136].

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. should it move to a different point in the solution space, generate new data, or select the current data as the most optimal solution [100, 136].

A Meta-heuristic differs slightly from a normal heuristic. In general the "meta" means *beyond* or *higher level* [100, 136]. A meta-heuristic therefore refers to a heuristic that is more complex with regard to the decisions it is able to make compared to a standard heuristic [100, 136].

In the literature algorithms are generally classified as being of the meta-heuristic variant when they satisfy the following criteria [100, 136]:

- Uses randomization
- Uses a local search
- Algorithm is stochastic

Meta-heuristic Algorithms do not search the solution space statically by testing and evaluating every possible permutation in the solution space, as it is inefficient and a normal path based algorithm can be used instead [8].

Meta-heuristic algorithms make use of certain strategies and heuristics (specific to the problem domain) to search the solution space intelligently through trail and error [8]. Intelligent searching is accomplished by combining the local search strategy along with a heuristic [31, 100, 136].

These algorithms iteratively move through the solution space, using a heuristic to guide the search to move to more desirable regions in the solution space where there is a high probability of obtaining high quality candidate solutions [73, 80].

Meta-heuristic based search methods are not guaranteed to find the most optimal solutions in the solution space, instead these methods are usually used to find near-optimal solutions. Thus most algorithmic development in the meta-heuristic domain focus developing new techniques that will increase the probability that a good solution will be obtained in difficult combinatorial problems [8].

Similarly, Meta-heuristics are not guaranteed to find suitable solutions or perform well in each problem domain it is applied to. The quality of the solution and performance of the meta-heuristic is very much dependant upon on the expertise of the algorithm designer [134].

The standard meta-heuristic algorithms will not take advantage of specific domain knowledge as the algorithm is designed to be general purpose. Therefore for the algorithm to exploit the search domain the algorithm designer needs to embed domain specific knowledge [100, 134]. Otherwise the algorithm will search the solution space generally and will produce relatively poor results [100, 134].

Although heuristics play a key role in the performance of meta-heuristic algorithms, it is not the only factor that has an impact on performance and results. Algorithms also use techniques and concepts from other system paradigms like multi-agent systems [72].

In multi-agent systems, multiple agents have to communicate with each other and the system as a whole has to perform some sort of autonomous self-organization [72].

This social and self-organization concepts enable these systems to be distributed, robust and flexible. Which is why in meta-heuristic algorithms that are population-based, hybrid and/or distributed these same concepts are used to better exploit the solution space [72].

In this section the characteristics of meta-heuristics that sets these algorithms apart from the conventional algorithms used on difficult problems was introduced. A general overview was given on how solutions are obtained as well as the quality of solutions. This section concluded with a brief discus-

sion on why for each problem domain the algorithm used, must be changed to fit the domain.

In the next section of this chapter a discussion will be presented on the Tabu Search algorithm.

4.3 Tabu Search

Algorithm 1 Basic Tabu Search Algorithm

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

4.3.1 Introduction

Tabu Search (TS) was first proposed by Glover as a new searching technique to help algorithms avoid getting stuck in local optima present in combinatorial and optimization problems [94]. Since Glover introduced the algorithm

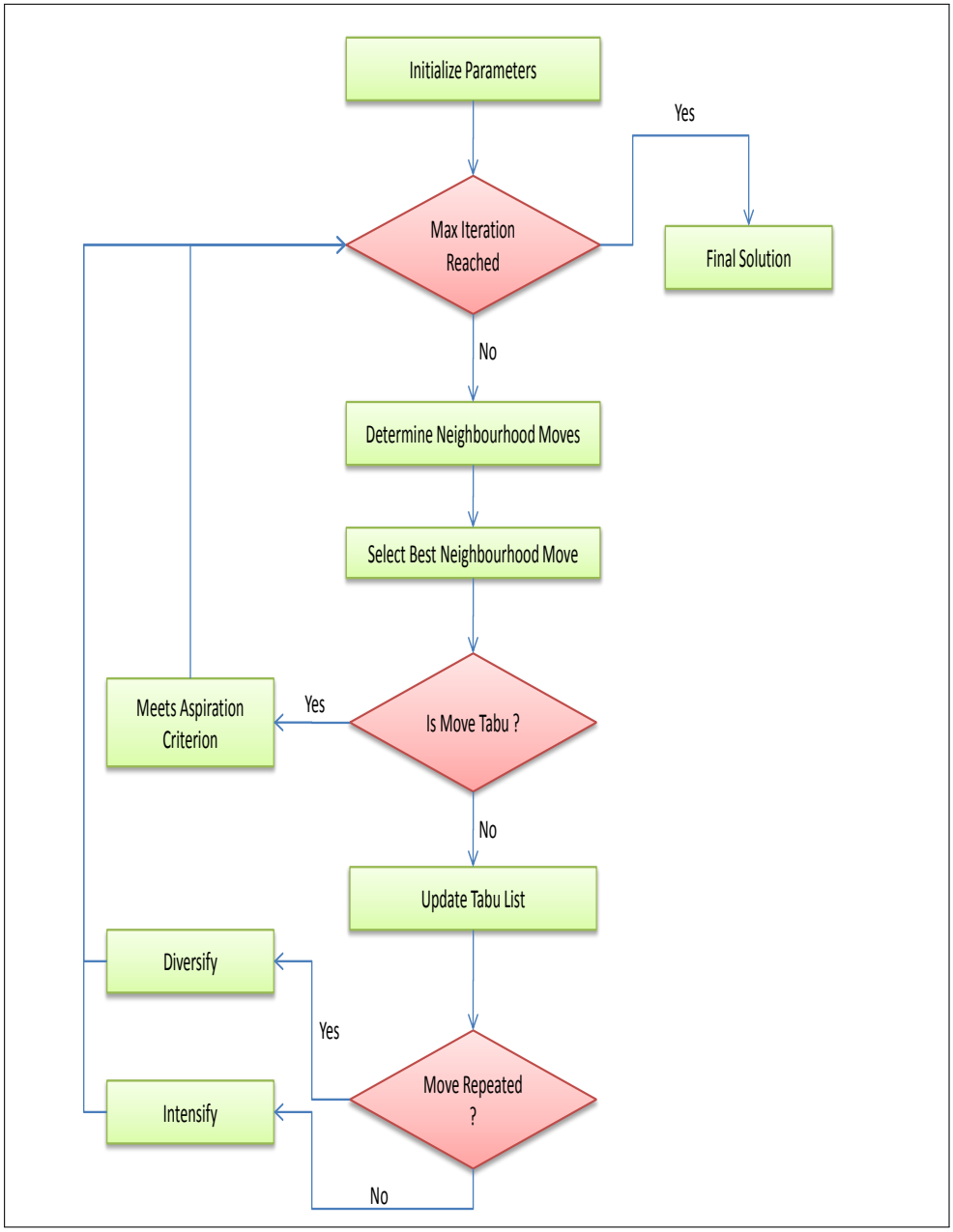

Figure 4.1: Flow Chart for Tabu Search Algorithm

in the 1980's, Tabu Search has been applied to a wide range of problems such as the Vehicle Routing Problem [76], Frequency assignment problem [80], Capacitated-Lot Sizing Problem [38], Nurse Scheduling [24] and the Resource Constrained Assignment Problem [94].

Even though the problems mentioned differ by a large margin, the algorithm has been relatively successful in most of optimization problems it has been applied to. If we observe the results presented in the following research [15, 38, 76, 80, 86, 87, 94, 110, 114, 127] we can deduce that Tabu Search has on average obtained the best results compared to previous attempts with other algorithms.

Tabu search resembles in its most basic form the Hill-climbing search algorithm [114]. The Hill-climbing search algorithm starts from an initial solution and then iteratively moves from the current solution to a neighbouring solution [100]. Each neighbour is rated based on its attractiveness as a possible optimal solution the algorithm is being applied to [100].

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 [100].

The Tabu search algorithm addresses this short coming of the hill-climbing by introducing the concept of memory [114]. Where the memory of the algorithm is actually a history of previous solutions the algorithm has moved to in its searching of the problem space for a solution [114].

General search algorithms like Hill-climbing, Random-restart[2] or Scatter search tend to get stuck on local optima [100]. The local optima might be a very attractive solution and thus general search algorithms will not move to better solutions since according the algorithms built in strategy it has found the best solution.

In actual fact the solution that was found, is the best solution in the *local* search space but not in the *global* search space [31, 100]. Therefore an important characteristic that algorithms being applied on optimisation problems need to posses is breaking out of local optima [31, 100].

[2]Random-restart is a search algorithm where once a certain trend of repeated moves are noticed, the algorithm restarts by generating a new initial solution to start from and then continues its search process from that generated solution [100].

In this section the TS algorithm introduced. Various characteristics and concepts that brought about the formation of the TS algorithm as well as what makes it unique was mentioned. In attempt to better understand the algorithm in the next section a discussion will be presented on the flow of the TS algorithm.

4.3.2 Flow of the algorithm

In this section as discussion will be presented on the general flow of the TS algorithm using algorithm 1 as reference point.

Before the algorithm can actually start searching it first needs to initialize various parameters. These parameters include but are not limited to, the Tabu List size, the Aspiration criterion and the starting solution. The initialization can be observed to occur from line 1 - 2.

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

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

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

Once all the neighbours have been evaluated, the algorithm selects the best neighbour who 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 on line 6.

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

The first check that needs to be performed is whether the neighbour z_i is in the Tabu List and occurs on line 7. A solution is only in the Tabu list

if the algorithm has in previous iteration had the particular solution as its current solution. Tabu lists will be discussed in section 4.3.3.

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

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

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

In the algorithm, if a neighbour z_i is found not to be in the Tabu list, the algorithm then adds the current 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 line 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 line 15 – 19, the algorithm calculates whether the new selected solution has been repeated for a certain amount of iterations.

If the solution has been indeed been repeated for a predetermined amount of iterations, the algorithm activates its diversification strategy otherwise, the algorithm intensifies its search. In the next section 4.3.3 a sub section will be presented on these two strategies.

A general overview of the TS algorithm has now been given as well as a brief explanation on the flow of the algorithm. During the discussion of the flow of the algorithm various characteristics that make the TS algorithm unique like Tabu List and aspiration criterion was mention. In the next section these characteristics will be discussed in much more detail.

4.3.3 Important Tabu Search characteristics

In this section various characteristics that are important to the tabu search (TS) algorithm will be discussed. The first characteristic that will be discussed is exactly how the TS algorithm iteratively improves upon the initial start solution. After initial solution generation a discussion will be presented on research done with regard to different neighbourhood strategies for TS. One of the most important features of TS will be discussed in the memory structures section. Finally, this section on tabu search characteristics will conclude with a section that discusses the two search phases present in TS.

Initial Solution Generation

The core feature of the TS algorithm is to sequentially improve an initial solution [139]. An initial possible solution is a point in the problem space where the TS algorithm will *start* exploring in search of a more optimal solution [100, 139].

Therefore an important consideration one has to make is how initial solutions are generated for the TS algorithm to start on [100, 139].

Random initial solutions might seem to be a good starting point, but by introducing randomization it becomes hard to control the quality of the end solution [139]. Hence the generation of starting solutions must be controlled to limit the infeasibility of potential solutions [139].

Control of the randomly generation solutions can be achieved by simply constraining the random solution generator to only generate initial starting points in a bounded subset of the entire search space.

For example, instead of letting the random initial starting point be any number between positive infinity and negative infinity, the random number generator is constraint to only generate numbers between 5 and -5.

Neighbourhood search

Tabu Search uses a neighbourhood local search process to explore the solution space. There is no set process of how neighbourhood candidate solutions are selected. Depending on the problem the TS is applied different neighbourhood solution selection strategies are needed. The overall quality of

the solution produced by TS is also dependent on the neighbourhood search strategy used [139].

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

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

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

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

The TS algorithm isn't limited to just one neighbourhood search strategy. In the paper by Gopalakrishnan et al. [38] five neighbourhood move strategies are developed and are used interchangeably, in some cases a strategy is used three times in a row due to stagnation in the search space.

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

Other neighbourhood strategies developed is one developed by N. A. Wassan [127]. In the authors paper a neighbourhood selection strategy is used that exchanges route nodes from initial vehicle routes for the Vehicle Routing Problem. This route exchange enables the TS algorithm to search much more broadly due to the constant supply of different solutions.

Since initial solutions are constantly modified it enables the TS procedure to be a very fined grained process, because often a small changes in a

potential solution can have a big impact on the overall proposed solution by the TS algorithm.

In the research done by Zhang et. al [139] an interesting neighbourhood selection scheme called *dynamic penalty* is discussed. When the algorithm moves onto an infeasible solution a penalty is imposed. By dynamically changing the penalty that is imposed the "feasibility" of solutions produced is influenced.

Therefore, when and if the algorithm continually produces infeasible solutions, the penalty imposed is increased as to guide to algorithm to produce more feasible solutions. Finally, in the case when the algorithm is stuck on local optima, the penalty is reduced, which allows the algorithm to consider moving onto infeasible solutions thus escaping local optima.

Considering all the research done to develop new neighbourhood selection strategies that improve Tabu Search to search the solution space more efficiently and produce better faster solutions, Tabu Search still has some draw-backs, especially with problems that have very large solution spaces [7].

Tabu Search is an iterative algorithm, executing a set of operation sequentially until a stopping criterion is met as can be seen in the flow-diagram presented earlier. At each iteration the algorithm has to determine feasibility of the immediate neighbourhood candidate solutions [7, 76].

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. Hence, 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 [7, 76].

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. Tabu Search addresses the short coming of these algorithms by incorporating an important concept the notion of memory.

In its most basic form Tabu Search 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* [38,51,127,139].

The algorithm is not allowed to move to any solution that is in the Tabu list unless a solution that is *Tabu* is better than any current moves available in the immediate search neighbourhood [38, 51, 127, 139]. The process of overriding a solutions Tabu status in the Tabu tenure is called the *aspiration criterion* [38,51,127,139]. 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 [6, 87].

Research done by Ashish Sureka and Peter R. Wurman makes an important distinction with regard to the memory scheme that is used in the TS algorithm. Two memory schemes are discussed; *explicit* memory and *attribute-based memory* [114, 115]. Between the two memory schemes the explicit memory scheme is the most used in the literature [76].

With explicit memory the algorithm stores a complete solution in the Tabu tenure, hence the algorithm is prohibited to move to that position in the solution for as long as the solution is in the Tabu tenure [114,115]. With attribute-based memory the algorithm stores the *operation* the is used to move from the previous solution, to the current solution [114,115]. Therefore with attribute-based memory, the Tabu tenure intended function is changed from prohibiting certain solutions already encountered, to rather prohibit making changes to the current solution that would lead to solutions already present in the Tabu tenure [114, 115].

In research conducted by D.M. Jaeggi and G.T. Parks and T. Kipouros and P.J. Clarkson [49], the authors add two additional memory structures called *Medium Term Memory* (MTM) and *Long Term Memory* (LTM) besides the standard *Short Term Memory* (STM), typically referred to as the Tabu List [49]. Each additional structure remembers a different set of solutions for use by the diversification and intensification phases in the algorithm. These two phases will be discussed in the next section.

STM purpose is similar to the traditional Tabu list, to store the most recent solutions produced by the algorithm. MTM is designed to remember optimal or near optimal solutions. These solutions are therefore used later in the intensification phase. Finally, the LTM structure stores all the regions that the algorithm has already explored and is thus used in the diversification phase of the algorithm [49].

Search phases

As Tabu Search searches through the solution space, it goes through two cycles of search phases called *diversification* and *intensification* [15, 33, 43, 51].

The diversification phase in the TS algorithm is the phase where the algorithm is directed to areas in the solution space that hasn't been explored yet. The algorithm usually applies diversification as mechanisms monitoring the memory, notice that solutions being produced are being repeated [33, 127].

In Research done by Fescioglu-Unver and Kokar [33] a strategy is presented that consists of two components namely the *Observer* and the *Diversifier*. The goal of the Observer is to continually monitor the best solution obtained by the algorithm whether it violates the *stagnation period*. The stagnation period is defined as the amount of iterations where the current best obtained solution hasn't changed [33].

As soon as the algorithm exceeds the stagnation period the Observer component activates and transfers the necessary information needed by the Diversifier component. The Diversifier component dynamically changes the size of the Tabu tenure based on the information the Observer gathered. The diversifier mainly targets older moves to diversify, but for short burst of time it would decrease the Tabu list size to a very small value in an attempt to combine new and old moves [33].

The specific mechanism used to define a new position where the algorithm can continue search, should ideally select areas in the solution space that have not been explored yet. Therefore, the diversification phase typically makes extensive use of the knowledge present in the long term memory structures as an indication to what areas of the solution space have been previously explored and which areas have not [15, 33, 43, 51].

Intensification is usually the first phase of the Tabu Search algorithm, since it is responsible to build up a history in memory for which the diversification phase can act upon. Fescioglu-Unver and Kokar also presented an intensification strategy based on control theory in their research [33]. The authors identified the repetition length as a critical value for their intensification strategy to be based upon. The repetition length is a control measure that defines how many times a solution may be repeated.

In this section a discussion was presented on the various characteristics that enable the TS algorithm to produce near optimal solutions. For each core characteristic a general overview was given on its operation and how it relates to the TS flow diagram.

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

4.3.4 Tabu Search on the FAP

As discussed, the TS algorithm is an optimization algorithm and has been applied to a wide variety of optimization problems (see page 55). In the literature the TS algorithm has also achieved relatively good results.

In a study conducted by Robert Montemanni and Derek Smith [80] the TS algorithm is used on the FS-FAP problem. Since the tabu search algorithm is generic, the authors had to make some alterations to the algorithm to suite their needs as well as to make the algorithm more efficient exploring in the FAP solution space.

The TS algorithm used by the authors is the Multi-start tabu search algorithm, which randomly starts on different initial solutions [80].

The authors developed a technique called Heuristic Manipulation Technique (HMT). HMT first monitors an underlying heuristic being used on the problem by the algorithm [80]. It then identifies certain characteristic good solutions exhibit. In the FAP, it is transmitters that get assigned different frequencies which results in an overall lower interference value [80].

The HMT then uses the identified characteristics to add *additional* constraints to the problem [80]. By adding constraints, the search space reduced which therefore makes it easier for the algorithm to find an optimal solution, as there are fewer solutions to consider.But by reducing the search space, other near optimal solutions which might be far better are excluded [80]. It is for this reason that the authors opted not to add the constraints permanently. Instead the constraints are replaced by other constraints [80].

The authors applied their TS algorithm together with HMT on the COST 259 family of benchmarks. Specifically the Siemens 1, Siemens 2, Siemens 3 and Siemens 4 problems. The results will now be presented. As can be observed from the above results obtained by the authors, the TS algorithm with HMT produces results that rank very favourably against other

Problem instance	TS with HMT	Best Cost 259
Siemens 1	2.7692	2.200
Siemens 2	14.9360	14.280
Siemens 3	6.6496	5.19
Siemens 4	110.9725	81.89

algorithms also applied to the COST 259.

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

Search based on a single solution The Tabu search algorithm at any moment in time only searches in the vicinity of *one* current solution for possible neighbours that might be the current solution for the next iteration. FAP problems typically have huge search spaces due to their NP-Hard nature. Therefore, only searching for possible lucrative neighbours from only potential solution seems to be terrible inefficient. A better strategy would be to use the notion of population based algorithms and have multiple solutions from which lucrative neighbours are searched.

Neighbourhood generation The Tabu search algorithm, defines no set process for generating a neighbouring solution given a starting solution. Generating neighbours from a solution is a critical process in the TS algorithm, for it is the only means by which the algorithm considers other solutions i.e. it is the mechanism by which the algorithm searches. Generating a new neighbour can be as simple as changing only one value from the current solution or it can be very complex and incorporate other algorithms together with mathematics formulas. 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 also is 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 on the algorithm search process. A large majority of

the neighbours of this solution is vastly superior solutions compared to any of the current solutions produced by the algorithm. Due to the solution with these neighbours being in the Tabu list, these neighbours won't be reconsidered until much later on when the solution is removed from the list. The only other option is if the aspiration criterion is met, which is another parameter which needs to by fine tuned. A high aspiration criterion and the algorithm might be too eager to just select any solution even though it's Tabu. A low aspiration criterion and the algorithm will be too strict in selecting a tabu solution.

In this section a literature review was presented on the TS algorithm being applied to the same problem this dissertation addresses. In the next section a discussion will be presented on the Simulated Annealing algorithm.

4.4 Simulated Annealing

Algorithm 2 Basic Simulated Annealing Algorithm

1: Initialize parameters
2: Set starting temperature $T(0)$
3: $x_0 \leftarrow$ Generate initial starting solution
4: **while** Stopping criteria not met **do**
5: $y_i \leftarrow$ Generate neighbouring solutions to x_i
6: Evaluate y_i neighbours with fitness function $f(y_i)$
7: Calculate probability p_i of y_i neighbours with equation 4.1
8: $x_i \leftarrow$ Select y_i neighbour based on probability p_i
9: Reduce temperature $T(i)$ based on cooling schedule
10: **end while**
11: Return best solution x_i

4.4.1 Introduction

Simulated Annealing (SA) is a heuristic search technique proposed in the 80's by Kirkpatrick to solve combinatorial optimisation problems. The technique is based on a natural process which is known in metallurgical as Annealing [60, 85, 113, 123]. Kirkpatrick was the first to use the simulated annealing to solve optimisations problems but the basic algorithm structure was defined in by Metropolis et. al. in 1953 [84, 123]

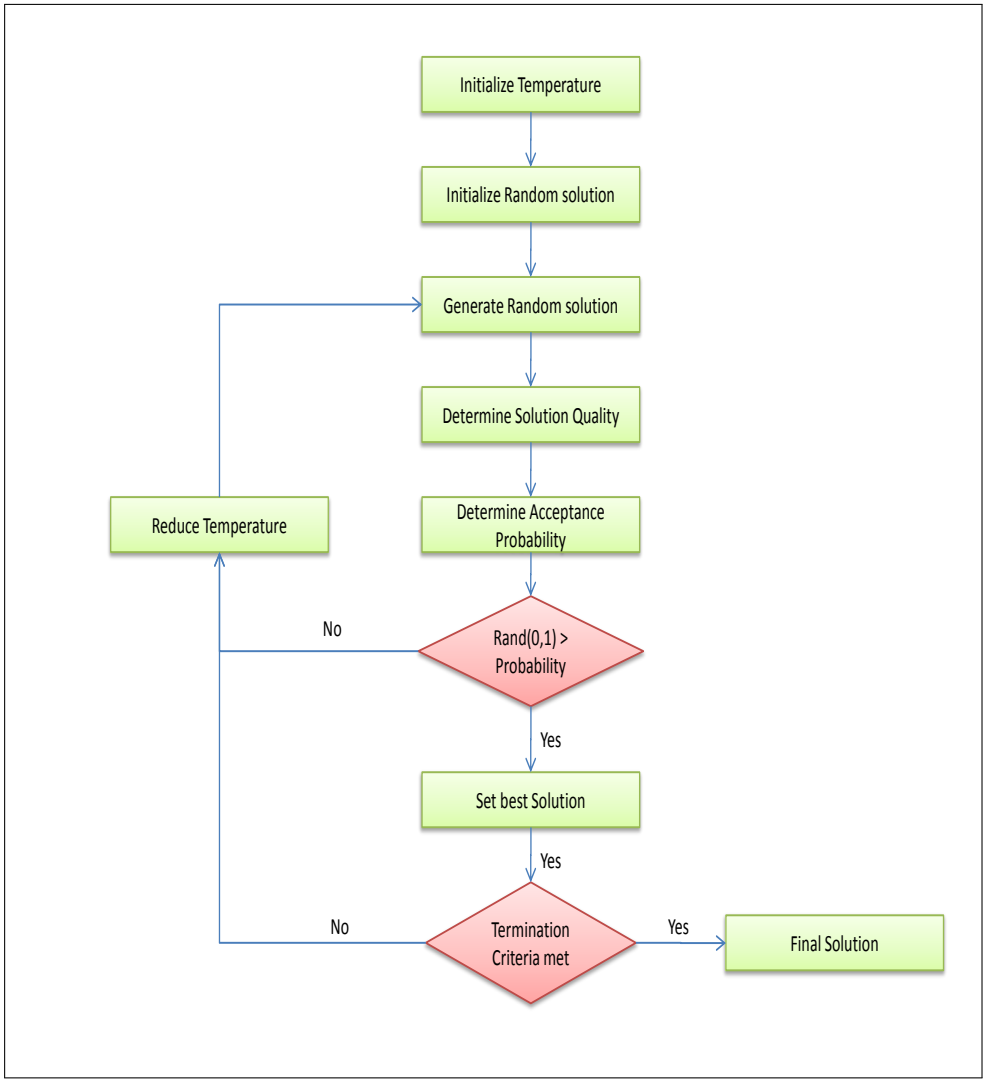

Figure 4.2: Flow Chart for Simulated Annealing Algorithm

Annealing is the natural process of crystallization when a solid is heated to a high temperature and then systematically cooled to a lower temperature to reach a crystallized form [5,62,74,123]. This crystallized 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, 62, 74, 123]. 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 minima [62,74,123]. When the solid is slowly cooled in a controlled manner, the molecules are able to reach a thermal equilibrium at each temperature [5,62,74,85,123].

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 performed by the algorithm. This function is known in the literature as the *Metropolis Criterion*.

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

The function f is the objective function or a function that determines the state of a given position in solution space [133]. The parameter T_k is the temperature of the algorithm at iteration k [133]. Finally, ΔE is the change in "energy" between two solutions x and y [133].

The main purpose of the SA algorithm (like most optimization algorithms) is to minimize or maximise the cost function [113]. This cost function typically evaluates a solution desirability compared to other solutions in the immediate *neighbourhood* of the algorithms current position [20].

The immediate neighbourhood of solutions are generated based on some heuristic implemented by the algorithm designer [100]. This heuristic, as with the TS algorithm, can be simple or complex []. The neighbourhood generated is the first step right after the starting solution has been generated by the algorithm as can be observed from figure **??**.

Typically a neighbouring solution is only selected as the new best state if its desirability ranks higher than the current solution. When the algorithm

moves to a better solution from the previous solution, the move is typically referred in the literature as a *downhill* move [123].

The best state isn't always selected, in some case the algorithm is also able to move to solutions that are worse than the current solution. A worse solution is only selected based on some probability, which is controlled by the *annealing temperature* of the algorithm [20].

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 [123]. When the algorithm moves to a worse solution, the move is typically in the literature referred to as an *uphill* move [123]. Uphill moves allows 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 [113].

The SA algorithm is also very popular due to the basic structure of the algorithm being generic [93]. As with the TS algorithm, the standard SA algorithm does not define a set neighbourhood generation mechanism, instead it is up to the algorithm designer to implement a suitable generation mechanism that will allow the algorithm to adequately explore the problem space [93].

Therefore, applying the algorithm to other problems requires slight changes. These changes usually need to be applied to the *Neighbourhood selection* scheme and the *Cooling Schedule* [93, 121]. Both of these concepts will be discussed in the next sub section.

In this sub section we gave a brief overview of the Simulated Annealing (SA) algorithm. We briefly discussed what the algorithm is based on and how the algorithm goes about searching the solution space. We also introduced some concepts like move probability selection and annealing temperature, which forms part of the core the algorithm. In the next section we will explain some of the concepts we've touched upon in this section as well as more advanced concepts that makes the algorithm unique.

4.4.2 Flow of the algorithm

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

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

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

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

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

The general search flow of the SA algorithm has now been explained. In the next section, characteristics that were only mention in the introduction section and in this section will be discussed in much more detail.

4.4.3 Important Simulated Annealing characteristics

In this section four brief discussions will be presented on the characteristics of the Simulated Annealing Algorithm that make the algorithm unique. The first discussion will be on one of the most important defining characteristic, namely the cooling schedule.

Furthermore a discussion on the importance of the initial temperature generation and what impact it has on the overall algorithm. Finally, this section will concluded with an overview on the efficiency of the algorithm.

Cooling Schedule

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

As can be observed from the figure 4.2 after each iteration, whether the algorithm has selected a new best solution or not, the temperature is reduced by a certain amount. This amount of 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 [67]. Thus the algorithm moves freely in the solution space with little constraints. As the temperature decreases the probability that the algorithm will select a bad or just any solution gets lower. When the temperature is very low, the SA algorithm is similar to a greedy algorithm in a sense that it only accepts downhill movements [67].

In the literature there are three annealing schedules in common use namely *the logarithmic schedule*, the *geometric schedule* and the *Cauchy schedule* [84, 113].

The standard and most common used schedule is commonly known as the logarithmic schedule and is based on Boltzmann annealing [84]. The main disadvantage of this schedule is that is slow due to its logarithmic nature [84]. It also requires moves to be generated from a Gaussian distribution for it to be able to reach the global minimum [113]. The logarithmic annealing function has the following form:

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

Where T_k is the temperature at iteration k.

The Cauchy schedule is faster than the logarithmic schedule. Similar to the logarithmic, this schedule also has a movement requirement. Moves must be generated from a Cauchy distribution for the algorithm to be able to reach the global minimum [84,113]. The Cauchy schedule is also typically referred to as Fast annealing [84]. The schedule has the following form:

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

Finally, the fastest annealing schedule is known as the geometric or exponential annealing schedule [113]. This schedule introduces the concept of *re-annealing*. Re-annealing is a procedure by which all SA temperatures are rescaled [84]. This schedule has no move generation requirement to reach the global minimum, since there is no regious proof in the literature to prove it [113]. The geometric schedule has the following form:

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

Initial temperature

The initial temperature is a very important parameter to define in the SA algorithm, since it defines a point from which the cooling schedule will start. Therefore, depending on what the initial value of the temperature is the final result that the algorithm will produce can be influenced [93, 108, 129].

When the initial temperature is set to a very high value the algorithm takes a long time reach a result. On the other hand if the initial temperature is set to a very low temperature the algorithm might converge too quickly and thus produce a result which may be the local minima [93, 108, 129].

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 [93, 129].

A low initial temperature together with a small cooling factor makes the time window for the algorithm to leave a local optimum very small [129]. 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 [129].

When the algorithm is near a global optimum, a low initial temperature and low cooling factor will allow the algorithm to reach the optimum faster in the solution space. In contrast, if a high temperature and a very low cooling factor is used the algorithm will take longer to reach the optimum even though it is near the global optimum [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 solution is generated and then small changes are made to the solution to represent a new solution. The solution is said to be perturbed to the next solution.

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

In research done by Tseung and Lin [123] an initial solution isn't modified but a move generation technique known as *Pattern* search is used. Pattern search has two forms of movement namely the exploratory move and the patter move. The exploratory move continually changes the certain variables of a solution [123]. This is done so that it 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 [123].

Algorithm efficiency

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

4.4.4 Simulated Annealing on the FAP

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

In the literature presented by Carlo Mannino and Gianpaolo Oriolo [69] the SA algorithm is applied to the FAP. The authors utilized *dynamic programming* together with the SA algorithm to alter the SA algorithm in such a way, to better explore the FAP solution domain.

Dynamic programming is a computer science technique that is usually applied to problems with big search spaces and thus makes it difficult to solve [100]. The technique decomposes the problem into smaller problems that are easier to solve since the smaller problems have smaller solutions spaces to explore [100, 132].

Utilizing the dynamic programming, the authors decompose the FAP problem into smaller interval graphs. The authors are able to do this since the FAP is similar to the graph problem, which means a FAP problem can be modelled as a graph [19]. The smaller interval graphs are graphs of vertices that are closely related and form a clique [19].

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

Cooling Schedule Depending on the cooling schedule selected the algorithm might converge too quickly. As discussed the cooling schedule reduces the temperature and the temperature plays a large part in the determination of whether a particular solution will be moved to or not in an iteration. Thus early on the algorithm will explore a lot more and later on exploit more. In the FAP, the algorithm must not only be able to explore and exploit but also, be able to return to an exploration phase if need be. With the SA algorithm, as the temperature gets colder the algorithm exploits more and therefore won't easily move to a worse of solution. In the FAP, it might be desirable to later on rather move a worse off solution, as the particular current solution is a local minima and yields bad neighbours as potential next solutions. With the cooling schedule it is simply not possible, unless the temperature and schedule is reset. Resetting the temperature and schedule is not ideal, since the algorithm has 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 pace in search of an optimal solution. Therefore applying the SA algorithm would require a custom neighbourhood

Problem instance	SA	Cost 259 (old)	Cost 259 (new)
Siemens 1	22.96	23.00	2.200
Siemens 2	14.72	14.75	14.280
Siemens 3	52.43	52.55	5.19
Siemens 4	80.96	80.80	81.89

Table 4.1: SA on Cost 259 Benchmark

generation scheme, which would be difficult to control as the algorithm keeps no history and also doesn't really expose the temperature for use with neighbour generation. This control and history as the FAP search space is huge and a desirable quality would be to know if the algorithm is moving towards an area that has already been explored.

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

4.5 Genetic Algorithm

4.5.1 Introduction

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" [35, 41, 59, 124]. GA was initially developed for adaptive systems by Holland but has then, been widely used in the optimization field of study due to its effective exploration of the solution space as well as its relative success in multi-dimensional problems [35, 124, 125].

The wide use of the GA algorithm can also be attributed to its generic algorithm structure as well as the ease of implementation of the algorithm [59, 124]. GA is application dependent and thus the designer needs to tailor the algorithm to his needs to obtain good results [41].

Algorithm 3 Basic Genetic Algorithm Algorithm

$pop_n \leftarrow$Initialize population
Evaluate population with fitness function $f(i)$
while Stopping criteria is not met **do**
  $y_k \leftarrow$ Select best performing genes from population
  Remove other genes from population
  repeat
    for Each gene g_i in y_{k-1} **do**
      $z_i \leftarrow$ Perform crossover with g_i and g_{i+1}
      $m_i \leftarrow$ Calculate Mutation probability for z_i
      if $m_i \geq$ Mutation threshold **then**
        Perform mutation on gene z_i
      end if
      Add gene z_i to new_pop
    end for
  until $size(new_pop) = size(pop_n)$
  $pop_n \leftarrow new_pop$
end while
$x_i \leftarrow$ Determine best gene in pop_n
Return best solution x_i

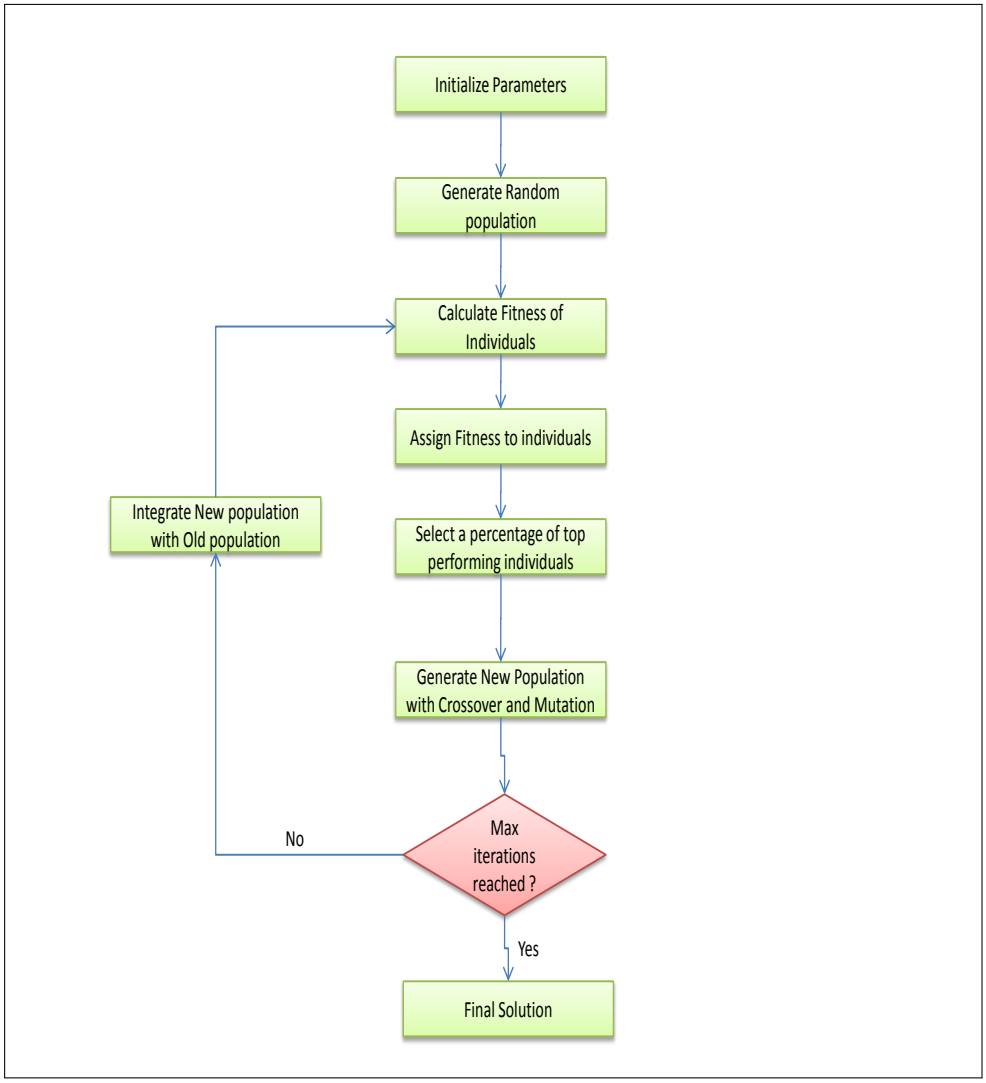

Figure 4.3: Flow Chart for Genetic Algorithm

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

The GA search procedure involves searching the solution space through artificial evolution and natural selection [50, 106, 124]. An individual or point in the solution space is known as *chromosome* in the literature [16]. An initial set of chromosomes (referred to in the research as the *population*), are randomly generated [41, 50, 106, 124].

Unlike other algorithms, GA does not concentrate on one point when searching the solution space, but concentrates on a wide range of points represented by the population [35, 50, 124]. Each chromosome in the solution space represents a string encoding of the problem parameters [124]. Encoding problem parameters also attributes to the wide use of the GA since difficult mathematical problems can now be easily modelled [41].

The population is artificially evolved each iteration by using a set of stochastic operators [120]. This set consists of a selection operator, crossover operator and mutation operator [106,120]. Each operator plays an important role in emulating the evolutionary process.

The selection operator is in charge of applying the *objective function* to each chromosome in the population [16, 59]. Depending on if the selection operator is setup for maximization of minimization, each individual is ranked based on its "fitness" or objective function value.

In the figure 4.3 the selection operator is the first operation applied after a random starting population has been generated. It is also the first operation applied, when the algorithm starts a new generation.

In accordance with the Darwinian theory, only the fittest individuals are selected from the population [16]. Based on the algorithm flow presented in figure 4.3, the fittest individuals are only selected once the whole population has been assigned their respective fitness.

The fittest individuals are copied and sent to the next phase of the algorithm, which is known as the *Reproduction* phase [16]. The reproduction phase can be observed in 4.3 as the operation where the new and old population are integrated.

Depending on how complicated the objective function is and how large the population is, the selection phase may be the most computationally expensive as well as time consuming [41]. The Reproduction phase mostly

consists of string manipulations on the chromosome, to drive to search forward and it thus a phase which is completed quickly, especially if the string encoding is of a binary nature [41, 59]. These string manipulations occur through the application of the crossover and mutation operators [97].

The reproduction phase is where a new population is generated for the next generation to be evaluated by the selection operator. The reproduction is said to generate "offspring" from the selected fittest population who are in turn known as the "parents" of the offspring [16, 97]. Reproduction will occur until a certain number of predefined generations are reached or a suitable solution is found to be greater/less than a certain fitness threshold that would indicate a good chromosome [92].

There are two basic forms of the GA where both forms differ in the way that offspring and parents are handled [124]. The one form is called the *generational* GA and the other form is known as the *steady-state* GA [124, 128].

With the generational GA the offspring are not immediately used in the next generation, instead they are kept in a pool until the pool reaches a certain size [124]. The offspring are then used to replace the parents entirely in the next generation [124]. In the steady-state GA the offspring are continually integrated with the population, thus offspring and parents occupy the same population pool every generation [124, 128].

The GA search process moves around in the search space using probabilistic rules rather than deterministic rules [124]. The probabilistic transition rules aid the algorithm to avoid local optima regions in the solution space [50].

Some sequential search algorithms require that the objective function be differentiable [97]. These sequential algorithms use the derivative to obtain gradient information so that they can move in the solution space [97, 120]. The derivative of the objective function may be used in to increase the efficiency of the GA, but is by no means a core requirement of the algorithm [50, 97, 120]. GA makes no assumptions about the solution space and primarily works on the information provided by the objective function [50, 97].

In this section an overview of the Genetic Algorithm was presented. The core concepts on which the algorithm is based upon was introduced as well

as defined how the algorithm searches for solutions in a given domain. In the next section a general outline of the flow of the algorithm will be given.

4.5.2 Flow of the algorithm

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

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

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

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

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

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

The first step in the reproduction phase is where various individuals of the subset are mated together to produce a new offspring individual. Mating occurs through the use of the crossover operator. The purpose of the crossover operator is to take certain characteristics of the two mating individuals and combine them to form a new individual referred to as the offspring. Generation of new individuals with the crossover operator can be observed from line 7 – 8

The second step of the reproduction phase is where mutation occurs. For each of the offspring generated by the crossover a mutation probability is calculated. If the calculated probability for a particular offspring is high

enough, the algorithm enters the mutation phase. In the mutation phase, the offspring is mutated, which is to say, the solution represented by the individual is altered slightly to be different. Mutation can be seen to occur on line 9 – 10.

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

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

In section a general overview of genetic algorithm flow was presented. In the next section various characteristics that make up the genetic algorithm and have up and till now only been mentioned will be explained in much more detail.

4.5.3 Important Genetic Algorithm characteristics

In this section an overview on characteristics that make the GA search procedure unique will be presented. The first discussion will be on the initial population generation. Furthermore, for each operator namely, selection, mutation and elitism operators a brief overview will be given.

This section will conclude with a discussion on the efficiency of the GA algorithm.

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 the desirability. Generally the initial population is generated by means of randomization [120]. Since the algorithm searches multiple points simultaneously in the solution space, it is desirable that the initial population have a wide diversity with regard to the solution they represent [35, 83]. By controlling the initial population generation we can control, to a small degree, the amount of exploration the algorithm does

initially as well as avoid premature convergence [83]. Therefore, care must be taken in the selection of the particular randomization scheme that will be used to generate solutions.

In a survey done by Andrea Reese [96], two randomization schemes are defined namely pseudo-random number generators (PRNGs) and quasi-random number generators (QRNGs). PRNGs where found to be heavily problem dependant, improving the search efficiency in some instances and in other instances having no considerable impact. QRNGs on the other hand, were shown to significantly improve the final solution produced by the GA as well as lowering the number of generations for the solution to be obtained [96].

Not all GA algorithm use randomization entirely for their initial population generation. In research done by Amit Nagar, Sunderesh S. Heragu and Jorge Haddock [83] an algorithm is presented that generates an initial population through some aid of a branch-and-bound algorithm. The branch-and-bound algorithm provides the GA with an upper bound of acceptable solution in the solution space. The initial population is then randomly generated in the constrained space defined by the upper bound [83].

Selection Operator

The Selection operator is the first operator to be applied to the population after each generation. This operator is in charge of evaluating the current population to determine which individuals will survive and which will be terminated [83,97,102]. The individuals who survive and thus have the highest fitness are moved to a "mating pool" [16]. Individuals from this mating pool will be used in the reproduction phase to generate a new population [41,59].

By favouring high fitness individuals above low fitness individuals the operator guides the search towards better high quality solutions [97]. Care must be taken if the operator is too eager on high quality individuals since it may eliminate diversity in the population and thus result in premature convergence for certain problems [97]. If the solution space is known to have only one optimum, then a strict selection policy may be used, therefore directing the search into a gradient based direction [97]. In contrast, with a solution space that is known to have multiple optima, a forgiving selection policy might be more favourable since it allows the solution space to be more widely explored [97].

The most widely adopted selection scheme is known as the *Roulette Wheel* selection scheme [97, 102, 128, 138]. With this scheme an individual is selected based on a probability defined by the fitness of the individual divided by the collective fitness of the population [128].

Crossover Operator

The Crossover operator is usually the first operator applied to the population in the reproduction phase. The crossover operates exclusively on the chromosomes in the mating pool. This operator is the main process by which by the GA algorithm is able to diversify as well as exploit certain optimal regions [83, 102].

Crossover works by interchanging and matching two parent chromosomes randomly selected from the mating pool to produce a single chromosome known as the offspring [16, 102, 124]. Since two chromosomes are combined or partially changed, some historical information is retained in the new chromosome [124].

In some algorithms like for instance the one presented in Nagar et. al., before the crossover operator is applied to the two parent chromosomes, the parents are first evaluated to determine if they represent suboptimal regions [83].

If either of the parents is from a suboptimal region, a disruption operator is applied that interchanges certain domain specific information between the parents. After the disruption operator is applied the crossover operator is applied [83].

There are a variety of ways with which values are interchanged between chromosomes in the crossover operation i.e. Fixed point crossover, Two Point Crossover, Uniform Crossover and Gaussian Crossover. Fixed point crossover operates on binary parents where by a point is selected in one parent and then all other bits are replaced by the other parents bits [16].

Two point crossover generates two random indices' which dictates a certain segment in the one parent to be interchanged with the other parent [97].

Uniform crossover is the most basic of all crossovers since it randomly selects bits from one parent to be replaced by another parents bits and is usually used when a large solution space must be search [125, 128].

Finally, the Gaussian crossover, interchanges bits between parents based on a Gaussian distribution [125, 128]. Depending on the state of the algorithm, crossover operators can also be interchanged or even paired if the algorithm needs better search performance for large or small solution spaces [4, 125].

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

Mutation Operator

The mutation operator is a probabilistic operator, which means it is applied infrequently and thus only with a certain probability will it be applied to parent solution. The operator typically changes some small in the solution regardless of the fitness of the chromosome [16, 138].

Given enough time, the mutation operator enables the algorithm to search the entire search space [124]. The operator also aids the algorithm with regard to escaping local optima in the solution space [124].

Due to the way the crossover works, some information may be lost when it is replaced by another chromosomes bits [41, 97]. Mutation is a source of new information that is continuously inserted into the algorithm; hence it works against information loss [41, 97, 102].

Usually, the mutation operator has no previous information on the chromosome it is mutating, thus it is entirely possible that the mutation may modify the chromosome for the worse [41]. 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 isn't always the case and thus in the literature the probability of the mutation operator is set to be very low [59, 97, 124].

In a survey done by Engelbrecht [31] 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 using any of the previous discussed crossover operators (see page 84).

The mutation operator isn't always simple random operations. In research done by Il-kwon Jeong and Ju-jang Lee, a mutation operator is

presented that incorporates the Simulated Annealing algorithm. Th SA mutation operator generates a new chromosome whose fitness is also calculated. If the new chromosome fitness is worse than the chromosome to be mutated, then depending on the SA mutation temperature as well cooling schedule, the newly generated chromosome might replace the chromosome to be mutated. Otherwise, if the new chromosome has a better fitness than the chromosome to be mutated, then it is replaced [59].

Elitism Operator

The elitist operator differs from the crossover and mutation operator in a sense that, it doesn't modify the chromosomes in any way. Instead, the operator works on the population [91]. The elitist operator ensures that the best chromosomes do not get lost from one population to the next, when the crossover and mutation operators are applied [4]. Thus the elitist operator is only applied after the crossover and mutation operators have generated a new population.

The elitist selects a certain number of high quality chromosomes from the parent population and transfers them in the new population without any modification from the other two operators [91].

The operator does not just move parents into the new population. The operator typically replaces sub par chromosomes in the new population with the higher quality parents [46]. Therefore, the elitist operator helps the algorithm retain knowledge gained from previous generations and prevents the best solution from extinction [89].

Finally, the retainment of knowledge and best solution aids the algorithm with regard to global convergence [109].

Algorithm Efficiency

The GA is a powerful, yet simple algorithm and tends to find good solutions given enough time. But the algorithm 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. If the population is too small the algorithm won't have enough diversity to search and tend to premature converge.

A large population is preferred in large problem spaces, but then the algorithm is very computationally expensive since more time is spent evaluating than evolving new populations and the speed of the algorithm convergence decreases drastically. Hence, the population size must be fine tuned to achieve optimal performance in large problem spaces

Another disadvantage of GA is that it is memory intensive, most sequential algorithms search on a single point bases through the solution space. As discussed above (see page 79), searches multiple points simultaneously and therefore requires more memory the keep track of all the possible solutions.

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

4.5.4 Genetic Algorithm on the FAP

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

In the literature by Colombo and Allen [19] the authors developed a GA algorithm to be applied on the FAP. The authors also utilize the notion of dynamic programming, by decomposing the FAP problem into smaller sub problems. On average the solution quality is improved by using the technique but at the expensive of more complex and taxing evaluations that have to be performed [19].

The evaluations are more complex, since in most instances the sub problems (which are sub graphs) overlap with other sub problems, therefore more complex evaluations need to be performed taking into account the overlapping [19]. The authors report that the technique is viable to improve solution quality as long as the connectivity between the sub problem graphs is not too high [19].

The results the authors obtained will now be represented.

By critically evaluating the genetic algorithm if it would be applied to the FAP, the following disadvantages can be identified.

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

Table 4.2: GA on Cost 259 Benchmark

Diversity The GA algorithm continually operates on a set population that was randomly initialized in the beginning of the algorithm. The algorithm hence only has this set of generated genes in the initialized population to therefore evolve successive populations from. If one does not regard mutation, the GA algorithm is a process by which the optimal combinations of the starting genes are found. Thus, the GA algorithm purifies the starting population genes in an attempt to find those individual genes, which if combined into a single individual, will produce an optimal individual i.e. solution. Therefore, the genetic algorithm is very much reliant on the quality of the random generator used. Hence, the probability of the algorithm finding a particular desirable gene that is exhibited by the starting population is directly related to the rate of mutation, as mutation is the only means by which more diversity gets introduced in the GA algorithm.

Crossover The Crossover operation in the GA algorithm is the only means by which successive populations are generated and can therefore be regard 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 large interference that 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 utilizing a crossover operation which works on individual channels assigned rather than cells. Crossover operation is also a memory and computational expensive operation since individuals need to be constantly created and values need to be copied to these new individuals from the respective parents.

Mutation Mutation is the primary means by which more diversity is introduced in the GA population. Typically the mutation value is set to a low probability due to mutation actually *modifying* individuals and not introducing *new* individuals. Therefore, there is a possibility that a mutation might alter an excellent solution in such a way, that the solution is suddenly one of the worst solutions. With regard to the FAP, the low probability of mutation is not desirable, as the FAP search space is huge and therefore requires constant diversity to be introduced to accurately explore the FAP search space. A possible good mutation would be one that is slightly more intelligent than the standard mutation operator, which is 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 gather historic knowledge alter the individual.

4.6 Summary

In this chapter a presentation was given meta-heuristic algorithms. A definition was given on what it means for an algorithm to be classified as being of a meta-heuristic nature and also what characteristic these algorithms must exhibit.

Three meta-heuristic algorithms were discussed in this chapter. For each algorithm as flow chart was presented which visually depicts the general flow of the algorithm along with a discussion on how the algorithm works as well as the various characteristics that makes the algorithm unique.

For each algorithm, a brief overview of literature using the particular algorithm was given as well as a series of brief discussions on some of the disadvantages or challenges that will be faced when applying the particular algorithm on the FAP.

The first algorithm discussed was the Tabu search algorithm and the second algorithm was the simulated annealing algorithm. The chapter concluded with a discussion on the Genetic algorithm. In the next chapter a discussion will be given on a class of algorithm that is relatively new to optimization research, namely the Swarm Intelligence class of algorithms.

Chapter 5

Swarm Intelligence

5.1 Introduction

The research field of Artificial Intelligence stands a lot to gain by studying the underlying inner workings of nature itself; this is why there is a branch of Artificial Intelligence that incorporates some of natures processes, like genetics, which can be seen being applied in practice in the genetic algorithm (see section 4.5 page 76).

There are other approaches in Artificial Intelligence which also have their routes in nature, like for instance animal learning or the study of how dogs learn. These approaches only look at a single agent thought process when it maps percepts (senses) to actions in its respective environment [11].

A percept can be said to be a process that is specifically designed to take data from the surrounding environment, process it and present it as information upon which decisions can be made [11, 100]. For instance, eyes are percepts, which take visual data presented by the environment. The visual data is processed by the brain into information to enable decisions to be made on navigating the environment.

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. This study of animals or insects in groups has already contributed to Artificial Intelligence as a whole [18, 64]. Each individual in the swarm might not be able to solve the problem on his own, but by interacting with other

individuals and performing primitive actions, the individual can contribute to solving the problem as a whole entity [18].

New algorithms that incorporate the lessons learned from the study of the collective intelligence are now being used and form part of the Meta-heuristic algorithm group. A discussion on some of the more popular meta-heuristic algorithms was presented in chapter 4 (see page 51.

Swarm intelligence algorithms are also meta-heuristic algorithms, with the distinction being made that swarm intelligence algorithms use multiple agents as a collective entity of knowledge to search the problem space [14, 18, 30, 31, 64].

The initial algorithms developed with regard to swarm intelligence, where based on the coordination and behaviour exhibited by schools of fish and flocks of birds. The newer generation of algorithms include [14, 18, 64]:

- Ant Colony Optimization
- Artificial Bee Colony Optimization.
- Particle Swarm Optimization

These newer generations of algorithms are primarily being used in combinatorial NP-Complete problems, where there is no defined solution, only an optimization that comes close to an optimal solution. Swarm intelligence works on a key feature observed in nature, the notion of emergent behaviour [14, 30, 31]. Whenever an entity does something unconventional and gains success, it can be considered as exhibiting emergent behaviour as if it is emerging out of the masses way of doing things [14, 30, 31].

Typical emergent properties exhibited by Swarms are self-organization and synchronization [14]. In Swarm intelligence, when an agent exhibits emergent behaviour, the behaviour of the agent needs to be shared to the whole swarm, in order for the swarm to adapt and use the newly gained knowledge [14, 30, 31, 64].

The information that is shared among the individuals of the swarm can be anything that contributes to the swarm as a whole, like for instance the information might be about a new solution that is better than all previous solutions [14, 30, 31, 64].

The behaviour propagates from one agent to another through social interaction, which brings forth information exchange [14]. Social interaction is

but one component of self-organization. Other components that form part of self-organization are [14]:

- Positive and negative feedback
- Increasing the magnitude of fluctuations

Individuals within swarm are able to accomplish social interaction with each other to achieve information sharing by using a concept referred to as *stigmergy*. A discussion on stigmergy as well as the different forms of stigmergy that exist will be discussed in section 5.2.

Utilizing stimergy along with self-organization swarm intelligence algorithms like Ant Colony Optimization, Particle Swarm Optimization and Artificial Bee Colony Optimization are able to produce relatively good results for NP-Complete problems [14, 30, 31].

Swarm intelligence based algorithm are able to achieve this since they have small individuals searching for more optimal values for a potential optimal solution on multiple fronts. As one individual makes progress on a certain front, its knowledge is shared [30, 31].

With traditional single agent based meta-heuristic algorithms like Tabu Search (section **??** page **??**) and Simulated Annealing (section **??** page **??**) only have in essence one "individual" searching for a solution [30, 31].

With algorithms like TS and SA, the information is not shared since there is nothing to share it with; rather the information is kept to influence future decisions [76, 80, 100, 123, 133]. Swarm Intelligence algorithms also store information for use by the various individuals that make up the swarm to make more informed decisions as the solution space is explored [30, 31].

NP-Complete[1] optimization problems is but only one of the fields of applicability where swarm intelligence algorithm have been applied. Other fields where swarm intelligence has been applied include neural network training, network routing, clustering [42], search engines and load balancing []. Swarm Intelligence thus seems more suited towards problems with combinatorial complexity (B. Denby, 2003).

In this chapter the first discussion will be on the Ant Colony Optimization in section 5.3. Particle Swarm Optimization will be discussed in

[1] Discussion on NP-Complete problems is presented in section 3.2 page 28

section 5.5. This chapter will conclude with section 5.4 where a discussion on the Bee Colony Algorithm will be presented.

Before the algorithms are discussed. A small overview on stigmergy will be given, as it forms a core concept upon which swarm intelligence algorithms are based upon.

5.2 Stigmergy

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

Interaction occurs through signals that the individuals receive which might require them to perform a specific action [10, 23, 31].

Two forms of stigmergy can be observed in nature. The one form, *Sematectonic stigmergy*, is a direct and physical form of interaction since it relies on altering the environment or by using some for of physical action to communicate [31].

Examples of this type of stigmergy being used include nest building and brood sorting by ants [31]. 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 and alert each other [14].

The other form, *Sign-based stigmergy* is an indirect form of interaction, where communication occurs through some sort signal mechanism [31]. Ants use sign-based stigmergy to communicate with each other. More on how ants communicate with this type of stigmergy will be discussed in section 5.3.1.

Other species that use sign-based stigmergy are bees [40]. When a bee determines that entity poses a threat to the hive, the bee might decide to sting the entity. The sting of a bee not only injects a toxin into the entity, but also releases a pheromone [40]. This pheromone alerts nearby other bees from the hive of the presence of entity that is a potential danger to the hive [40].

The other bees of the hive pick up this pheromone that is released by the initial bees stinger and therefore attacks the entity by also stinging it [40].

As more bees sting the entity, more bee stingers emit the danger pheromone identifying the entity [40]. Hence the pheromone gets reinforced and thus stronger which persuades more bees into action [40].

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 where stigmergy is used in nature. 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.

In the coming sections three swarm intelligence algorithms will be discussed. Each section is divided into 3 subsections.

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

The second sub section will give an in depth discussion on some of the core characteristics that make the algorithm unique.

For each algorithm discussed, the stigmergy used by the algorithm will be explicitly mentioned. Each discussion on a particular algorithm will conclude with a literature review of the algorithm being applied on the FAP.

5.3 Ant Colony Optimization (ACO)

5.3.1 Overview

Ant Colony Optimization (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. 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 [23, 31].

Deneubourg seeked to know how ants where able to find the shortest path towards a food resource and communicate it to the whole nest [23]. Thus an experiment was setup up to study the ants' behaviour. In the

Algorithm 4 Basic Ant Colony Optimization Algorithm [31]

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

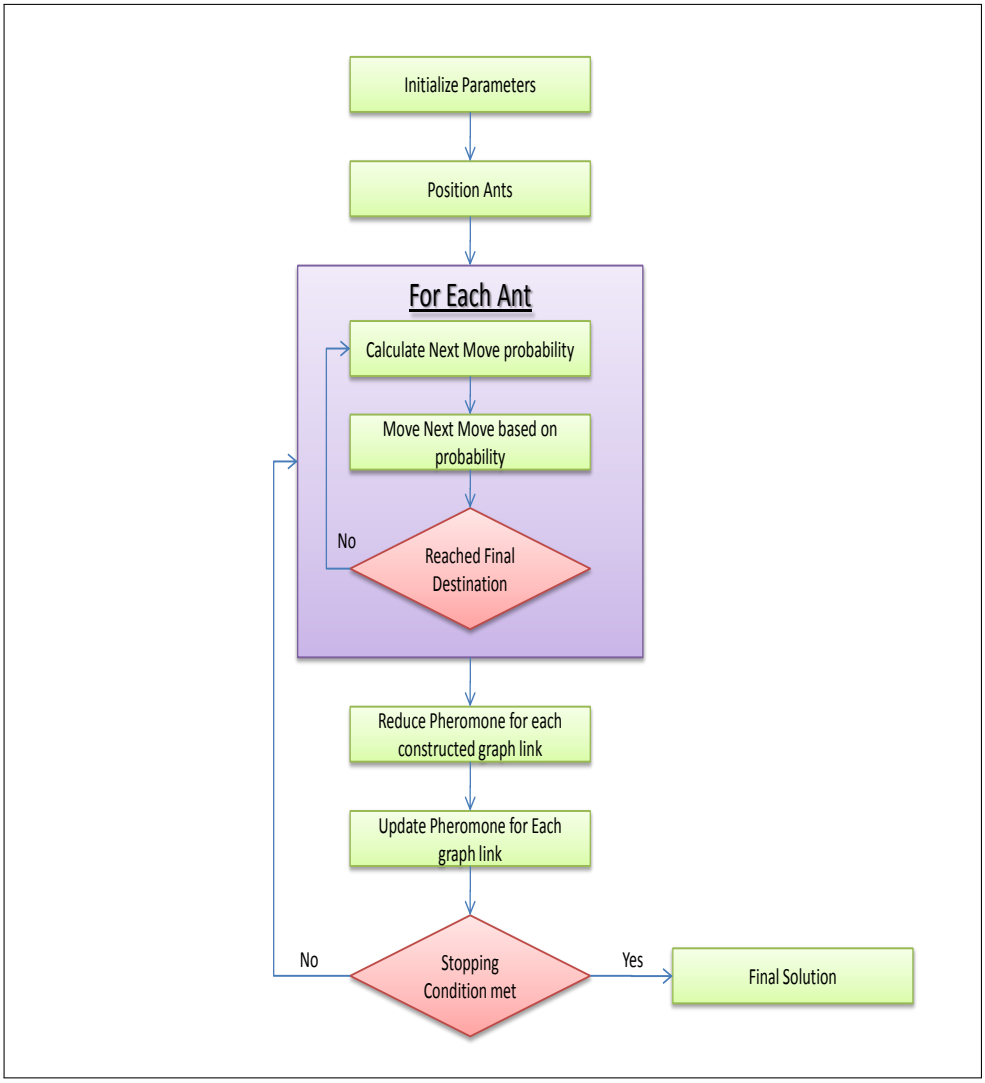

Figure 5.1: Flow Chart for ACO Algorithm

experiment a food source was placed a certain distance away from the nest. Two paths were setup towards the food resource, one path was purposely made longer than the other path as can bee seen in figure 5.2 [23].

Ants would exit the nest in order to forage food. The ants first started exploring the area outside the nest in an attempt to locate a suitable food source. As the ants in the experiment explored the immediate space outside the nest to the food source, they had to take one of the provided paths to the food source [23].

The ants initially oscillated between the paths with no clear distinction of the more dominant route to take to retrieve food from the food source. After a certain amount of time the one path to the food source became the preferred route for the ants as the specific path was indeed the shortest path towards the food source [23].

Naturally the questioned that was asked was how were the ants able to communicate to each other that the one food source was closer than the other? The answer lies in the use of *stigmergy* by ants. Ants use sign-based stigmergy (discussed in section 5.2) when they are retrieving food. 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 [31]. The chemical signal that ants use to indicate optimal paths is called *pheromones*.

A path is made up of a series of links between nodes. A link between two nodes represents a movement from the one node to the other. Therefore, a path can be considered as the traversal of the interlinked nodes, from a starting node to some final node. A path differs from another path by the

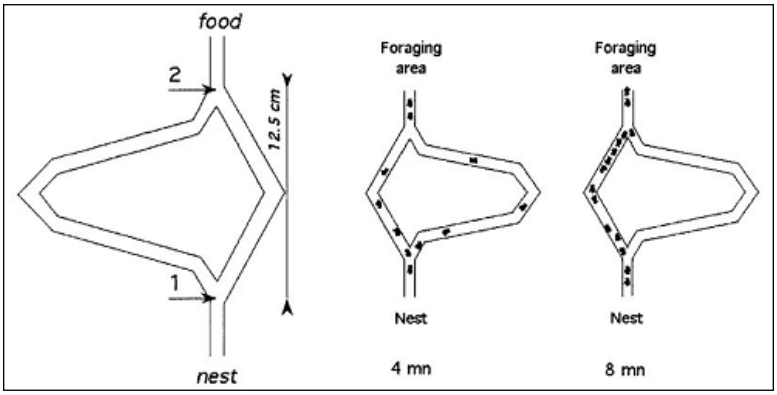

Figure 5.2: The Ant bridge experiment [23]

order in which the nodes are interlinked between a start and end node.

In the algorithm all ants in are initially placed on some random starting node. The main purpose of the ant is to explore the solution space and to ultimately produce a solution that might be optimal. The ant explores the solution space by performing a series of moves from one node to another node. Each move from one node to another is a link that is added to the path. This process can be seen in algorithm 4 lines 7–10.

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 link representing the movement from the current node to the next node [30, 31].

As the ant moves it records each link between the nodes it traverses until the ant reaches the final node. All the links the ant has traversed represents a path taken through the solution space [30, 31]. Thus, as the ant is moving it is actively building an optimal solution.

After a path has been constructed, the ant needs to inform the rest of the ants on 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 what movements worked well and what movements did not. Which is to say, the ant needs to signal the other ants, which is accomplished with pheromones. Therefore, in the next phase of the algorithm, pheromones get updated which occurs from line 14 – 22 in algorithm 4.

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

The ACO algorithms are considered stochastic search procedures due to their innate use of randomness when exploring the solution space [21, 135]. The ACO class of algorithms has been applied to a wide range of problems that include — INSERT EXAMPLES. The algorithm does have some disadvantages. One of the primary disadvantages of ACO is that it tends to get stuck on local minima in the solution space therefore, the solution space isn't adequately explored and the algorithm prematurely converges to a local optima [135].

To combat the shortcoming of getting stuck in local optima, the MAX-MIN Ant Optimization algorithm was developed. The MAX-MIN algorithm is based on the original Ant system algorithm. MAX-MIN addresses the

local minima problem by imposing dynamically changing bounds on the pheromones of the ants such that it is always falls within the limit of the heuristic and best current path of the colony (Gang Wang, 2005).

The first algorithm to be based on the behaviour of ants was called the Ants System (AS) [10, 31]. The AS was initially applied to the Traveling Salesman problem (TSP) as a proof of concept but its performance on TSP was lacklustre compared to other algorithms applied to the same problem [10, 31].

Subsequently, various algorithms have been developed that improve on the AS algorithm. These improved algorithms include the Ant Colony System (ACS), The ANTS system, Ant-Q and AntTabu [10, 31]. Where applicable reference to these algorithms will be made to indicate how the have improved the AS system.

In this sub section an overview of the ACO algorithm was given. A discussion was presented on how the algorithm was developed from observing ants. Furthermore a discussion was also presented on how the algorithm utilizes the core communication concept used by ants, called pheromones, is used to search the solution space. In conclusion a small indication of the typical problems at which ACO excels at as well as some of the disadvantages the algorithm has in its standard form was given.

In the next section a more in depth discussion on the core characteristics of the ACO will be given as well as additional characteristics that were developed in response to improve the efficiency of the algorithm.

5.3.2 Flow of the algorithm

5.3.3 ACO characteristics

In this section various characteristics that are important and unique to the ACO class of algorithms will be discussed. A discussion on each of the following characteristics will be discussed (in order of discussion): Pheromones, State Transition Rules and Pheromone evaporation.

Pheromones trail

The pheromone technique used by ants' forms part of the core methodology used by the Ant Colony Optimization (ACO) algorithm [34]. As an ant

moves it lays down pheromones to mark the path it is walking. This step can be observed in algorithm 4 on lines 17–21.

Pheromones decay over time. As the ant follows the pheromone trail it reinforces the pheromone that is already on a path [34]. Thus the more ants following a path the stronger the pheromone trail for that path and the stronger the pheromone the higher probability is that a ant will follow the path [34]. Pheromone decay can be seen in algorithm 4 on lines 14–16.

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

In the bridge experiment the ants started following the shorter path, because an ant following the shorter path would reinforce its pheromone trail faster than on the longer path. Initially, the ants oscillated between the two paths since there was no clear indication to the colony what the shortest path was, therefore the ants initially randomly selected the paths they would follow [34].

Once a path as been marked with pheromones the ant no longer selects a path based solely on randomness. Instead the ant selects a path based on a probability transition rule [23]. Transition rules will be discussed in the next sub section.

With the use of pheromones ants are able to communicate the best and shortest path. The more ants following a preferred path the more pheromones would be laid on that specific path and in return increase the strength of the pheromones [135]. The increase in strength of the pheromones on a path would thus let ants more clearly distinguish between paths it should and should not take [135]. Therefore, a pheromone provides positive feedback to the colony.

When the algorithm starts there are no pheromones to indicate path and the ants are placed at certain positions, which are problem dependent. In the general ACO algorithm presented in algorithm 4, the ants are placed on the first node.

Initially, all the ants will choose random paths. After all the ants have completed their paths, each path is evaluated [31]. The amount of pheromone marking a path is in the standard ACO related to the cost function. Therefore, a low cost function value will have a high pheromone dosage indicating the path the ant took from each node and a high cost function value will have a low dosage [31].

The pheromone of each path, therefore allows the colony to remember good and bad decisions from previous iterations [31]. This is a form of local pheromone updating which will be discussed in the pheromone update section.

In the iterations following the initial one, the ants will at each node decide based on a probability function whether it should follow the pheromone trail laid down in previous iterations or choose a new path to another node. Thus as the ACO iterates through more iterations the stronger particular path pheromone trail will become, hence signalling the path out as the best found [31].

The pheromone trail was initially developed with only one colony in mind [31]. In the research done by Tiwari et al. 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.

The ant will not follow or even consider the pheromone trail since any pheromone encountered from other colonies repulse the ant. 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. This repulsion strategy promotes exploration among the multiple colonies [119].

In this section we gave an in depth discussion on what pheromones are as well as how they are applied in the most basic of cases. In the next sub section we will discuss the different transition rules that each algorithm in the class of ACO algorithms use.

State transition rules

As discussed in the previous sub section the ants select which path next to follow based on a probability. This probability is also known as the *transition probability*. In this sub section a discussion will be provided on some of transition probabilities in use. Pheromone trails is the core concept upon which the Ant System is based and was the first algorithm developed

based on ants that used the pheromones concept [31].

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

The first transition probability is formulated in equation (5.1) and is used by individual ants of the AS algorithm [22, 31]. An ant k uses this equation to decide with what probability it will move from a node i to a node j [31]. τ_{ij} is the amount pheromone that exists on the link between node i and j [23, 31]. Heuristic information is incorporated into the equation through the symbol η_{ij}, which is the desirability of the path from node i to node j as evaluated by a heuristic function [23, 31].

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

Through the use of parameters α to represent pheromone intensity and β to represent heuristic information the algorithm is able to achieve a good balance between exploration and exploitation when $\alpha = \beta$ [31]. When $\alpha = 0$ no pheromone is taken into account, hence, any history that the algorithm has on the path between node i and node j is neglected and the algorithm degrades to a stochastic greedy search procedure. If $\beta = 0$ then the algorithm does not take into account the amount of desirability the path between node i and node j as dictated by the problem specific heuristic function.

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

Various other algorithms have been developed that improve on the basic AS. Each algorithm uses a different transition probability equation that might be very specific to the domain or a slight variant of the above equation as with the Ant Colony System.

This section is not intended to give an exhaustive survey of different transition rules that exist in the literate. Therefore, only the first transition rule that was developed is discussed, since most of the other rules can simply be considered derivatives of the first transition rule.

Pheromone update

As discussed in the pheromone section, pheromones start to evaporate [2]over time hence, the path marked by the pheromone trail becomes less attractive to the ants. Therefore, a path that represents a good solution needs its pheromone trail to be continuously updated. In this sub section a discussion will be provided on the rules that govern when and by how much pheromones are updated.

Most of the variants that have been developed differ in the sense of what pheromone update rules they employ. In the literature pheromone update rules are classified into two groups [31]. The one group is called the global update rule. The other group is called the iteration based or local update rule of which an example has been given on page 102 [31].

The first local pheromone rule was presented in the AS algorithm [31]. The ants would retrace their path after each iteration, depositing pheromones on each link that makes the complete path. The following equation was used to update the pheromone:

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

Pheromone update rules that are in the global update group, only allow the pheromone trail of the path representing the best found solution by the algorithm since the first iteration, to be updated [31]. Thus the global rule favours intensification where the algorithm exploits the global knowledge gained by the ants to find a better solution. By updating we mean the pheromone is reinforced.

The Ant Colony System (ACS) was the first to use the global update rule together with the local update rule [31]. By using both types of rules the algorithm is able to efficiently exploit the history provided by the pheromones [31]. The global update rule used by the ACS is formulated

[2]Pheromone evaporation is discussed on in section 5.3.3 page 105

in the following equation [31]:

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

The parameter $x + (t)$ represents the best / shortest path so far found by the algorithm [31]. The rest of the parameters represents the same concepts as discussed on page 103

As can been seen in the following equation, the ACS uses a slight variant of the local update rule first used in AS [31].

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

In the above equation τ_0 is a small constant and $p_2 \in [0,1]$ is the constant that defines the rate of evaporation [31].

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

Pheromone evaporation

Initially when the pheromone concept was first implemented the ants of the colony rapidly converged on a solution and did not adequately search the solution space for alternate path that might lead to better solutions. To combat this premature convergence and force the ants to explore the solution space more, the concept of *pheromone evaporation* was introduced [10, 22, 23, 31].

As discussed in the pheromone sub section (see 100) the pheromone marking a trail evaporates over time until an ant reinforces it. The evaporation of pheromones is governed by equation 5.5 [10, 22, 23, 31]:

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

The constant p defines the rate at which pheromone evaporates. If $p = 1$ the pheromone completely evaporates every iteration and the ants take no history into account with regard to there path selection and the search is completely random [23, 31]. Thus, the amount of exploration done by the algorithm can be controlled with the constant p [23, 31].

The above equation (5.5) was first introduced in the Ant System [10, 22, 23, 31]. Most subsequent algorithms that are 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 [10, 22, 23, 31].

A more aggressive form of Pheromone evaporation is added to the Ant system discussed in the research done by [34]. The more aggressive form works beside the already present pheromone evaporation, instead this form seeks to add an additional search phase called *diversification*.

In the ant system developed by the authors a mechanism is added that monitors a certain number of solutions that have been recently produced. If the solution hasn't changed a certain number of iterations the mechanism removes all pheromone trails currently in the system.

Therefore, the algorithm is forced to re-search the search space to create new solutions, as it cannot rely on previous historic information provided by the pheromone trails. This mechanism forces the algorithm to diversify [34].

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

5.3.4 ACO on the FAP

As discussed in the overview section 5.3.1 the ACO has been applied to a wide number of problems and has produced good results. Thus, the ACO has also been applied to the FAP instance of problems.

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

As an ant moves in the frequency planning domain, the ant 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, which is to say the ant 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 high dosage of pheromone would therefore be the frequency plan with the lowest interference found by the algorithm.

When analysing the basic ACO algorithm 4 one can identify the following possible disadvantages if the algorithm would be 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 cell, it might not select the previous cell (due to probability) to move to, but rather generate an entire 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 As discussed in section **??** 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 of with a slightly worse decision. In the FAP, a cell can have multiple interfering cells, but a particular ant only knows about the 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, which is to say, optimize the channels allocated to these cells so that interference is low. The first interfering link is now optimized, and subsequent ants will now reinforce this channel allocation since the interference is low. When the ants later on reach the other cells that also interfere with the first cell that has been optimized, the ants

Time limit	ACO*	ACO
120s	104719.72	91140.04
600s	103752.12	89703.44
1800s	103781.86	88345.94

Table 5.1: ACO and ACO* on custom GSM FAP benchmark

will have difficulty changing the assignments that has 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 presented by Luna et. al [65] an ACO algorithm is presented and applied to a custom cellular network instance. The custom cellular network instance the authors use has 711 sectors with 2,612 transceivers, which need to be assigned frequencies. For their particular network, only 18 channels are available for assignment. The channels start at 134 and end at 151 [65].

The authors presented two versions of the algorithm. The one version uses no heuristic information (hence forth referred to as ACO*) and the other version uses heuristic information to update the pheromone laid done by the artificial ants [65].

With regard to the heuristic updating the pheromone trails, the authors opted to increase the pheromone by some magnitude [65]. This magnitude was hand tuned to be 100. The heuristic only increases a certain paths pheromone if the frequencies assigned to the transceivers this path represents differ enough as to not cause significant interference [65]. Thus, the heuristic aims to amplify good choices made previously by the algorithm for the next iteration of the algorithm.

The results the authors obtained will now be presented for both versions of the algorithm that was developed [65].

The authors note, that if one observed the results they obtained, one can clearly see that the ACO version that incorporates heuristic information to reinforce pheromone trails outperforms the version that doesn't do it [65].

The above values represent the amount of interference that will result if the plan is used in the network [65].

5.4 Artificial Bee Colony Algorithm

Algorithm 5 Basic Artificial Bee Colony Algorithm

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: $v_i \leftarrow$ Generate new solution with equation 5.7
8: Evaluate with fitness function $f(v_i)$
9: Apply greedy selection between v_i and the current solution s_i of bee eb_i
10: **end for**
11: Calculate probability p_i for solutions s_i in s_n using equation 5.6
12: **for** each onlooker bee $ob_i \leftarrow 0$ to max bees b_n **do**
13: $x_i \leftarrow$ Select solution s_i based on p_i
14: $v_i \leftarrow$ Generate new solution with x_i and p_i
15: Evaluate v_i with fitness function $f(v_i)$
16: Apply greedy selection between 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 5.8
20: **end if**
21: $t \leftarrow t + 1$
22: **end while**

5.4.1 Overview

The Artificial Bee Colony (ABC) algorithm is the youngest algorithm discussed in this chapter. It was first proposed by Karaboga in 2005 and seeked to mimic the foraging behaviour exhibited by bees [55, 56, 107]. Like ants, bees need to gather food to support the colony. To understand how the

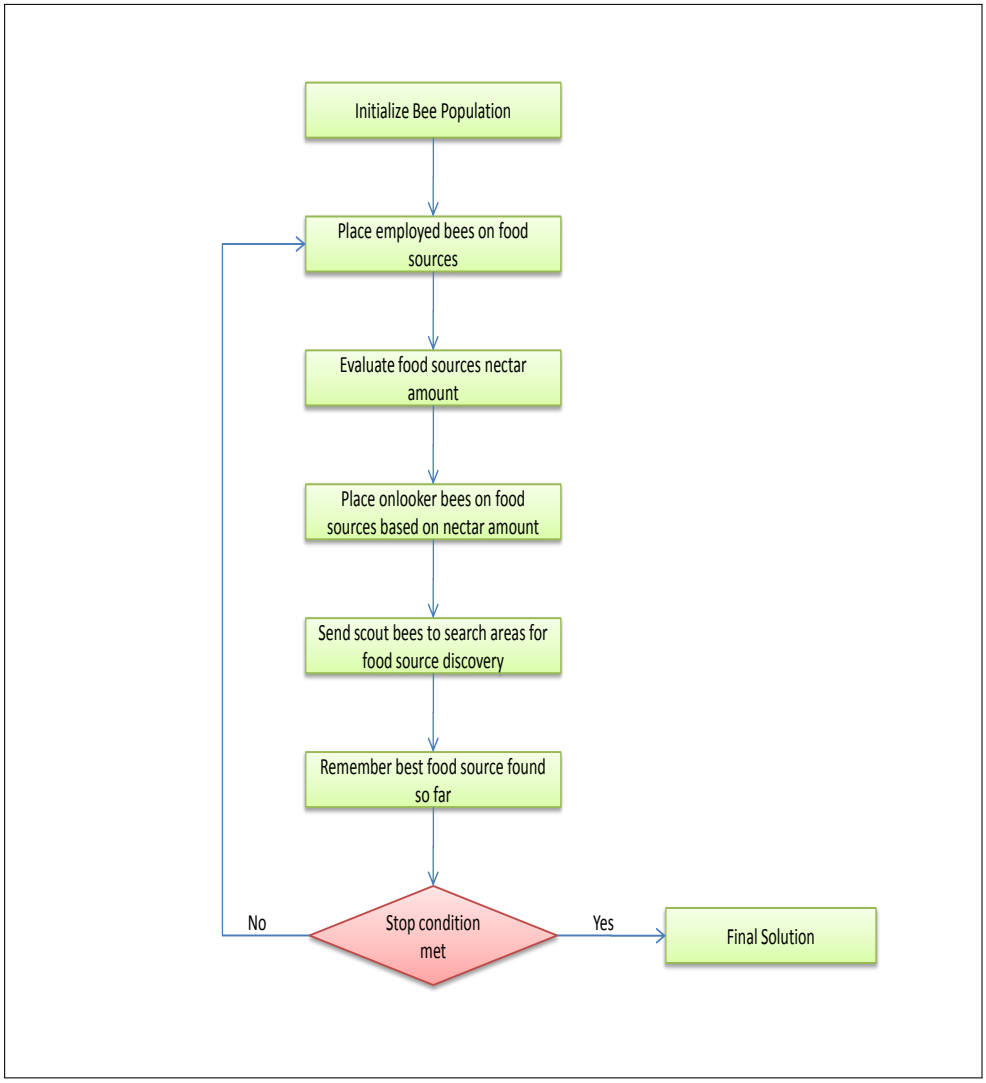

Figure 5.3: Flow Chart for the BEE Algorithm

ABC algorithm tries to mimic the foraging behaviour of bees, we first need to explain how real bees forage [55].

In a bee colony there are a numerous number of bees, each with a specific role that performs certain actions for the colony. There are bees that protect the queen, maintain the colony, scout for resources and finally bees that gather food i.e the worker bees. The most important bees for foraging are those that scout and gather food [55].

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 [55]. If a scout bee has found a food source, then it needs to return to the colony to share the information with the worker bees [55]. When the bee enters the colony it needs to communicate to the other bees by using some form of stigmergy (see section 5.2 page **??**) [55].

The scout bee accomplishes this communication by performing a dance known as the *waggle dance* in the colony for all the bees to see [55]. This isn't a dance like in the traditional sense, since through certain movements the bee is able to communicate a variety of characteristics about the food source that include [55]:

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

Therefore, it can be concluded that with regard to foraging bees use *Sematectonic* stigmergy (discussed in section 5.2 page 94) as the dance is a physical form of communication.

The dance is observed by *onlooker* worker bee [9, 55]. These onlooker bees are initially *unemployed* in the colony [9, 55]. After the information of the scout be has been transferred the onlooker bees become *employed* bees since it is moving to the food source to start gather food [9, 55]. Thus it is the job of the worker bees to *exploit* the information provided by the *exploration* done by the scout bees [55, 56].

Worker bees gather food from the designated food source, until the food source reaches a certain quantity with regard to nectar content [55,56]. Each time the bee returns to the colony it evaluates the current food source versus other food sources discovered [55, 56]. Hence, if there is better food source

found, the bee abandons the previous and starts gathering food from the new source [55,56]. On the other hand, if the food source has been exhausted meaning there is no more nectar content to gather, the bee returns to the colony and becomes "unemployed" again [55, 56].

In the ABC algorithm, possible solutions are considered to be food sources [55, 56]. 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 an other more attractive food source [55, 56].

A food source might be more attractive to a bee because it defined nectar content is more than the current food source the bee is operating on [55, 56]. 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 [55, 56].

As with real honey bees, a *waggle dance* is performed to all the onlooker bees by employed bees that provide information on the nectar amount (fitness value) that they represent [9, 55, 63]. The onlooker bees choose food sources depending on the nectar amount [9, 55, 63]. Therefore as the nectar amount of a food source increases the probability that the more onlooker bees will choose the source increases [9, 55, 63]. How and what effects the probability will be discussed in the next sub section.

Bees can transition to different roles depending on their situation [55,56]. An onlooker bee becomes employed when assigned to a food source and an employed bee can become a scout if his initial food source becomes exhausted [9,55,63]. 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 [9,55,63]. Scout bees are sent to randomly generated food sources [9, 55, 63].

The more onlooker bees a food source attracts, the more the neighbourhood will get explored since the onlooker bees move to the food source and choose an immediate neighbouring food source to be employed upon [55,56]. Thus, this can be considered exploitation and the algorithm is therefore performing a local search [55, 56, 63]. Finally, the number of onlooker bees a food source has also indicates its desirability, hence, a very good solution will have the majority of onlooker bees choosing it and search for nearby better food sources [55, 56, 63]. 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 [55, 56, 63].

When a food source is abandoned, the previous bee that occupied the food source transitions to a scout bee [55, 56]. The scout bee is responsible for replacing the abandoned food source by finding a new one, therefore, a new food source is generated and communicated back to the colony [9,55,56]. The generation of food sources will be discussed in the next sub section.

Karaboga was not the first to base an algorithm on the above foraging behaviour. In the literature other bee foraging inspired algorithms have been developed such as the BeeHive Algorithm, Bee Colony Optimisation (BCO) and Bee Swarm Optimization (BSO) [56, 70, 117].

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 [56, 70]. The solutions are then progressively modified using certain strategies. Finally BSO solutions are iteratively constructed by forager (worker) bees and the best solution is communicated to the rest of the colony by performing a dance [56, 70]. All of the above mentioned algorithms were developed to be primarily used on combinatorial problems [55].

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 the sense that it isn't designed for combinatorial problems [56]. Instead, VBA is designed for numerical function optimization [56]. In VBA bees would move around in the search space communicating to each other any target nectar food sources that were found [56].

Karaboga developed the ABC algorithm based on the previous research done on bee colony optimization and the above algorithms. The ABC algorithm is designed to be a multi-variable optimisation algorithm and has to date been applied to the Job Scheduling Problem, Clustering [70], Neural Network training and Reconfiguration of Distribution Networks [63]. Due to the nature of the algorithm being similar to that of the ACO, we can expect the ABC algorithm to be applied to a whole host of problems present in the literature.

In this sub section we gave a brief overview of the Artificial Bee Colony Algorithm. We discussed the real bee behaviour the algorithm tries to recreate and gave other algorithms that are also based on the same process. Finally, we defined the primary design goal of the algorithm and presented some of the problem on which the algorithm has been applied.

In the next sub section, we will discuss some of the core characteristics of the algorithm.

5.4.2 Flow of the algorithm

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

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

Each bee is initially assigned to one of the initial generated solutions (food sources). Hence, the bees start of 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 initialized, therefore the algorithm enters the next phase, which is where the actual optimization and search procedure occurs. This phase stretches from line 5 – 22.

From line 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, which is to say, the fitness of the generated solution is calculated. The employed bee then compares the newly generated nectar amount with the nectar amount of the solution in the bees' memory. If the newly generated solution has a better nectar amount, the bee replaces the current solution in its memory with the new generated solution.

After all the employed bees have finished determining whether to keep the newly generated solution or keep the one in memory, the employed bees 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 4 from line 12 – 17.

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

Once an onlooker bee has selected a food source based on the calculated probability, the bee then starts search the immediate neighbourhood of the selected food source, for other food sources. The neighbouring food sources are generated using equation 5.7 which will be discussed in section 5.4.3. This procedure of generating neighbouring food sources by an onlooker bee can be observed on line 14 in algorithm 5.

The onlooker bee then applies the same procedure as an employed bee with regard to determining if the newly generated food source should be remembered or discarded. The bee does this by evaluating each generated neighbouring food source to determine its nectar amount, which is then compared to the nectar amount of the food source in the bees' 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 amount has not increased. When a food source is abandoned the employed bee that occupied the particular food source transitions to a scout bee.

A scout bee aims to replace the abandoned food source with a new food source. In algorithm 5 this occurs on lines 18 – 22. The scout bee uses equation 5.8 to generate a new food source. The scout bee 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 like the rest of the employed bees do at the start of the next iteration.

5.4.3 ABC algorithm characteristics

In this subsection we will discuss the characteristics of the ABC algorithm that define the algorithm and make it unique. We will start of by first explaining how food sources are handled in the algorithm. We will then discuss how information is communicated to the colony. Finally, we will finish off this sub section with a more precise discussion on how foraging in modelled and used in the algorithm.

Food Sources

As discussed in the overview, 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. Therefore, initially a finite amount of food sources are randomly generated. Since each food source needs an employed bee to evaluate the nectar amount of the source, the parameter that defines the amount of food sources also defines the amount of employed bees.

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

Onlooker bees bear witness to a number of dances from a variety of employed bees. The onlooker bees therefore need to select a food source that is the most attractive while maintaining some diversity in the pool of solutions. Thus, an onlooker bee selects a food source based on a probability function which is formulated in equation (5.6) [55]:

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

The parameter p_i is the ith food source under consideration by the onlooker bee. As discussed above, the fit_i parameter represents the value of the cost function and is directly related to the nectar amount of food source i. The parameter SN is the maximum amount of food source and hence the maximum employed and/or onlooker bees [55].

In algorithm 4 the waggle dance can be seen being applied line 11 where the probability of all the employed bees solutions are calculated using equation 5.6. From line 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 outlined in the overview, when recruited onlooker bees reach the advertised food source that is stored in memory, they do not occupy the same food source. Instead, the bees explore the immediate neighbourhood of the food source that communicated to them. The onlooker bees seek to find a

food source that improves on the previous one. Equation (5.7) is used by the bees to generate new food sources in the neighbourhood of a food source x_i.

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

The subscripts $k \in \{1, 2, \ldots, SN\}$ and $j \in \{1, 2, \ldots, D\}$ are indexes which are randomly chosen. D represents 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 it *must* differ from the value i. The position of the new food source in the neighbourhood of x_{ij}, is controlled by the ϕ_{ij} parameter, which is a bounded random value between $[-1, 1]$.

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

After a new solution v_i is produced, the bee takes the new solution and the old solution 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. Otherwise, the bee abandons the new solution and keeps the old solution in memory. Thus, the bee seeks to always move towards a better solution and therefore, uses a greedy selection process.

One of the problems with the above approach is that little information about the food source is used in generating a neighbouring food source. In the research by Singh [107] a slight variation is proposed to generating food source neighbours by using more global information. The authors add 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 [107]. Scout and Onlooker bee generated solutions are not checked if it collides with other solutions in the system since the aim for them is exploration and not exploiting like with employed bees.

Scout bee

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

- The employed bee has reached the maximum allowed cycles to improve the nectar amount. The maximum cycles spends on a food source allows the algorithm to avoid local optima.

- The bee cannot improve the solution represented by the food source any futher.

When a food source in the algorithm is abandoned, it needs to be replaced with a new food source. An employed bee under goes a role transition when it abandons a food source. The bee transition from an employed bee to a scout bee.

It is the responsibility of the scout bee to replace the abandoned food source with a new randomly generated one. Scout be uses equation (5.8) to produce new food source that will replace food source x_i.

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

In research done by Gómez-Iglesias et. al [37] an extension is made to the scout bees. The Scout bee individuals are divided into two types of bees namely *Rovers* and *Cubs* bees [37].

- Rover bees are similar to traditional scout bees and hence use diversification strategies to explore the solution space.

- Cub bees explore the solution space relative to a good solution found by a Rover through randomly changing configuration parameters.

By using two different scout bees a good balance is achieved when searching the solution space in the beginning where diversity if preferred and late in the algorithm where intensification is preferred [37].

5.4.4 ABC algorithm on the FAP

The ABC algorithm and all its variants are relatively new. The algorithm has for now only been applied to a select few problems such as the Traveling Salesman Problem.

As of yet, no research has been done to apply the ABC algorithm to the FAP. After critically evaluating the basic ABC algorithm presented listed in algorithm 5 page 109 the following obstacles can be identified if one were to apply the algorithm on the FAP.

Food source representation Each food source can either represent a frequency plan or each food source can be a particular cell and a collective of food sources represent 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 search for neighbouring solutions. These neighbouring solutions are *also* frequency plan. Where as with each food source being a cell would require less memory. The problem with the food source as a cell approach is that the bees would then single out one cell as the optimum since they do not know 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 new 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 near lucrative enough for the rest of the bee colony to consider as it contains now knowledge gained by the algorithm.

Knowledge sharing As a food source gets more popular due to its high nectar amount indicating a good fitness, more onlooker bees will select it to search in its neighbourhood for slightly better solutions. Therefore, the bees are disregarding previous gained knowledge gained 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 their optimal frequencies, but due to the larger majority of the cells not being optimized, these *good*

cells are overshadowed. Thus due to the *bad* cells overshadowing the good cells, the food source gets a low nectar amount indicating a bad fitness and therefore the bees might abandon the food source and those optimal cells are lost.

The above obstacles present real relevant challenges that would require new techniques to be developed. As the algorithm has not been applied to a wide variety of problems and together with the above obstacles, it is difficult to gauge if the ABC is well suited to be applied. In future, when the algorithm has been applied to a more wide set of problems and has matured in the variety of approaches to problems one can revisit the algorithm and attempt on the FAP.

In this section a discussion was presented on the various obstacles one would encounter when applying the ABC class of algorithms on the FAP. This section concludes the discussion on the ABC algorithm. In the next section a discussion will be presented on the Particle Swarm Optimization algorithm.

5.5 Particle Swarm Optimization (PSO)

5.5.1 Overview

Particle Swarm Optimization (PSO) is a self-adaptive, population-based stochastic search technique that was developed by Kennedy and Ebenhart in 1995 [105, 105]. The basic model of the algorithm is based on simulations done to recreate the natural behaviour of a flock of birds [131].

In the early stages of the particle swarm development, simulations were developed to closely model the stigmergy (see page **??**) exhibited when a flock of birds cohesively move as one and is able to suddenly change direction in a unpredictable graceful manner only to regroup as one observed entity [52].

As the "leading" bird of the flock changes his movements the information is shared to the all birds that are in the immediate neighbourhood of the leading bird. As the information is shared locally among the birds, each bird modifies his own movement to that of the leading birds movement [52].

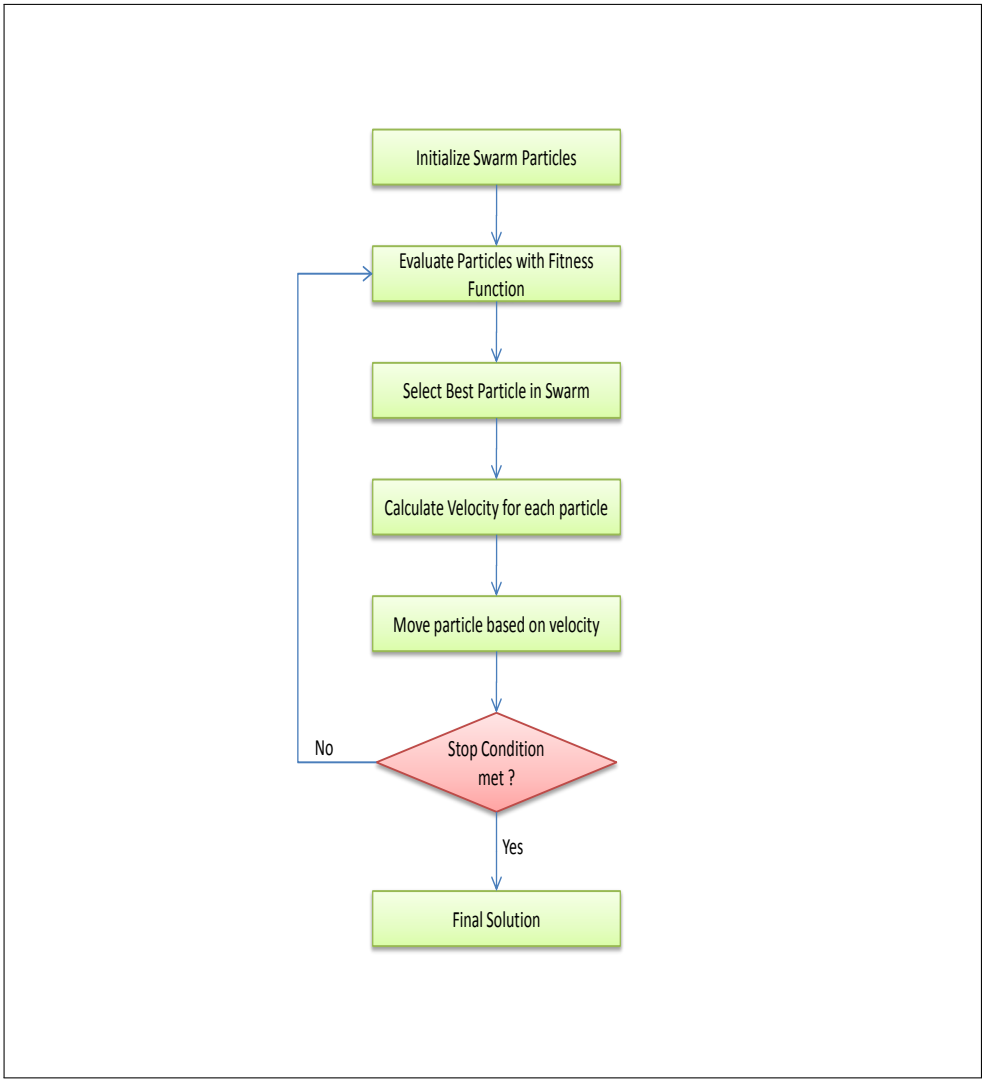

Figure 5.4: Flow Chart for PSO Algorithm

Algorithm 6 Basic Global Particle Swarm Optimization Algorithm

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

Thus, because birds obtain information through observation of their neighbouring birds, the stigmergy can be deemed of a physical nature. Therefore, the particular stigmergy used by birds is Sematectonic stigmergy (see page 94).

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 optimization algorithm [31].

In the algorithm a particle is an individual. A group of particles, referred to in the literature as swarm, fly through the solution space of the problem the algorithm is being applied to. Each particle changes his movement based on information shared to him by neighbouring particles in the swarm [30,31].

As information is shared among particles, the success of one particle ripples through the rest of the swarm and each particle is able to utilize shared information that lead to success of another particle. Thus, each particle own personal experience and knowledge of the search space has an effect on its neighbouring particles [30, 31].

There exist two primary PSO algorithms called Global PSO and Local PSO. The only difference between the two algorithms are how they go about sharing information to the rest of the swarm. The sharing models of these two algorithms along with other sharing models will be discussed in the PSO characteristics subsection [88].

In the swarm, each particle is a potential solution that is encoded in a D-dimensional vector [52, 95]. As a particle moves through the solution space, it continually evaluates its current position and adjusts it accordingly to move in the general direction of the best particle of the swarm.

Depending on the sharing model used, the best particle in the swarm is denoted as either *gbest* of the global PSO and *lbest* for the Local PSO [30, 31, 88].

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

A particle evaluates its current position by using a heuristic function or in more evolutionary algorithm terms, a fitness function. The fitness value indicates to the particle how far it is from an optimal position [31].

As a particle moves through the solution space it keeps a memory of the personal best position it has achieved since the start of the algorithm. In the literature and in the algorithm this personal best position is referred to as *pbest* [88].

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

A particle needs to systematically explore the solution space. Hence, when the particle needs to update its personal velocity, it doesn't 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 [30, 31, 88, 95].

The amount of global knowledge is referred to in the literature as the *social* component [30, 31, 88, 95]. The amount of personal information used by a particle is referred to as the *cognitive* component in the literature [30, 31, 88, 95].

The PSO algorithm is quick to converge on an optimum, which might not necessarily be the global optimum [95]. Hence, most of the research done on PSO has focused on the convergence of the algorithm as well as improving the diversity [30]. Most of these improvements and modifications will be discussed on the next sub section called PSO Characteristics.

In this section, an overview was given of the PSO algorithm. The discussion started on how and why the algorithm was developed. Furthermore a discussion was presented on what natural search procedure the algorithm incorporates and also how it is applied in the algorithm. After the search procedure was discussed, a general outline on how the particle moves and how its velocity is updated was given.

In the next section, a in depth discussion will be presented on the important PSO characteristics that was only touched upon in the overview, like Particle Velocity and Information sharing. This section on the PSO algorithm will conclude with a discussion on a relimanary analyses of applying the algorithm to the FAP.

5.5.2 Flow of the algorithm

The general concepts that is evident in the PSO algorithm have now been discussed. Using these concepts a general overview will now be given on the PSO algorithm flow using algorithm 6 as reference.

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

Before the swarm can start moving around in the problem space, it first needs to determine the global best (gbest) particle as well as each particles own personal best position. Therefore, as can be observed on line 3, each particle's current fitness $f(p_i)$ is determined using 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 personal best (pbest) since the algorithm started or not. This comparison can be seen to occur on line 5.

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

Regardless if the personal best of a particle has been updated or not, the algorithm performs another comparison also utilizing 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, which is to say, is it the *global* best (gbest). This comparison occurs on 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 on line 9.

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

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

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

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

The general flow of the PSO algorithm has now been outlined. In the next section a discussion will be presented on the some of the characteristics that makes the PSO algorithm unique.

5.5.3 PSO characteristics

In this sub section, a discussion will be presented on some of the characteristics of the PSO algorithm. The first discussion will be about the *swarm size*. After the discussion on swarm size a sub section will be presented on the *particle velocity* where the equation will be given that is used to update particle positions as well as their velocities. A discussion on *Inertia* follows the particle velocity section. This section will conclude with a discussion on the different *information sharing* models of PSO.

Swarm Size

The swarm size dictates the search breath the algorithm has to maintain each iteration [31, 137]. The initializing of particles in a swarm are the same as traditional population-based evolutionary algorithm use to initialize their respective populations [137]. At the start of the algorithm the swarm is initialized by randomly generating possible solutions that will represent the position of particles [31].

Even though the PSO algorithm in some aspects resembles evolutionary algorithms like the Genetic Algorithm, it does not continually generate new solutions to be re-inserted into the swarm to increase diversity [118]. 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 decrease rapidly as information is shared [30, 31].

A large swarm might increase diversity but at the expensive of computational time since the algorithm has to spend more time evaluating particles [30, 31]. Where as with a small swarm there might be low diversity but the algorithm requires little computational effort to evaluate and move the swarm around in the solution space [30, 31].

Diversity of the swarm gets lower due the as the swarm progresses, particles tend to converge toward the best particle shared to them. If no particle is able find a better position on its way towards the global position shared and the best particle is not able to improve its own position, then eventually all particles will converge on the best position.

With a small swarm the algorithm will converge faster to an optimum, which is not guaranteed to be the global minimum [30, 31]. Therefore, care must be taken with the selection of the swarm size, because if a large swarm

size is selected the algorithm will be slower to converge to an optimum but in turn, gains diversity which allows for a higher probability of finding the global optimum [81].

Particle Velocity

The velocity calculation of each particle is where the optimization procedure occurs in the PSO algorithm and is the only means by which the PSO algorithm searches the solution space, since its the only means by which particles are moved to new positions in the search space [31].

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

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

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

where $v_i(t+1)$ is the new velocity of a particle i for the next time step $t+1$. The cognitive component is represented by parameter c_1 and the social component is represented by the parameter c_2 (discussed on page 124) [30,31]. The current position of a particle in the solution space at time step t is represented by parameter $x_i(t)$ [30, 31]. After the new velocity is calculated the position of the particle is updated for time step $t+1$ using equation (5.10) [30, 31]. The velocity update can be visually depicted as shown in figure 5.5.

As discussed in the PSO algorithm overview pbest is the best position the particle has occupied since the start of the algorithm. In the literature there are two defined methods of determining gbest. The most common method used is where gbest is the best position obtained by a particle in the swarm since the start of the algorithm. Thus long term knowledge dictates the best position found which favours exploitation [30,31].

With 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 [30,31].

As can be observed in equation (5.9) the new velocity is added to the old velocity. Therefore, the velocity of particle can get very large, especially

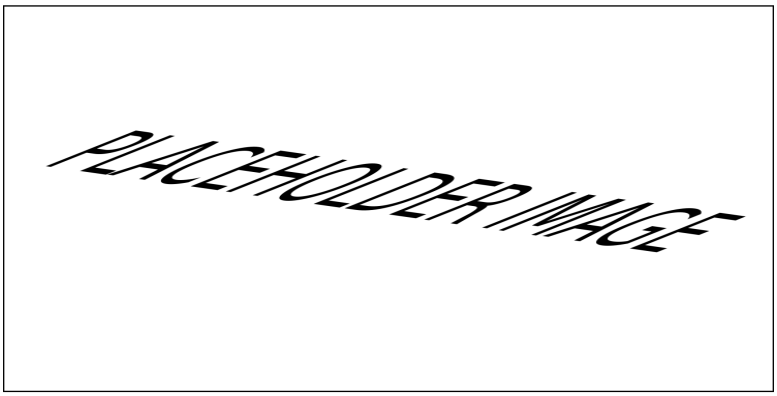

Figure 5.5: Visual particle velocity update [30, 31, 88, 95]

for those particles that are far from the pbest and gbest positions. Large velocities are necessary for early exploration, but if the velocity gets too large, the rate at which the particle moves in the solution space is to high and good solutions might be missed [30].

Thus, the velocity of a particle needs to be clamped to ensure it step size stays within acceptable bounds. Equation (5.11) is used to clamp the velocity of a particle *before* its position is updated [30].

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

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

Where V_{max} is the maximum allowed velocity and $\delta \in (0, 1]$. The values x_{max} and x_{min} are the respective minimums and maximums of the domain the algorithm is being applied to [30]. The value of δ is very problem dependent and must be carefully chosen as to maximise the exploration - exploitation trade off [30].

Most of the literature has concentrated on the velocity of the particle as a focal point because it is the main function performing the optimization. In research done by Ratnaweera et. al. [95] a particle positions in the solutions space are continually monitored. If the particle appears to be stagnant in the solution space, the velocity is first updated, and then the particle is reinitialized with a random position. The new position of the particle is then updated with the new velocity. Thus knowledge of the previous discarded particle is retained by using the velocity it had [95].

In research done by Kalivarapu et al. [53] a PSO algorithm is developed that seeks to incorporate the pheromone notion of ACO algorithms into the velocity updating of particles. The premise of the method is to allow greater sharing of information about promising areas between particles. The algorithm developed by the authors achieved promising results with the algorithm in certain cases finding solutions faster and better solutions than other PSO algorithms [53].

Other research done by Monson and Seppi [78] is more concerned with how particle presented. In the general PSO algorithm, particles have no physical form or volume, thus particles in the swarm move through each other. The authors changed this in their algorithm by letting each particle have a radius around itself. Therefore, as particles move through the solution space and another particle at a certain time step occupies the same space, the particles are said to collide. As one would expect, when a collision occurs both particles are deflected into random directions [78]. At a greater expense of computational time due to constant collision detection, the PSO gains greater exploration in the solution space.

Finally, in the research by Lenin and Monan a PSO algorithm is developed that is called the Attract and Repulse PSO (ARPSO). The algorithm continually monitors the solutions in the swarm. If the algorithm picks up that a certain percentage of the swarm is stagnating, the algorithm activates the repulse state. In the repulse state particle are repelled from other particles in the swarm, this 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 [61].

Inertia Weight

As an object moves with a certain velocity it carries momentum, if the object were to suddenly change direction, momentum would for a certain period still move the article in the previous direction. Inertia weight seeks to add type of behaviour to the particles of the PSO algorithm. It was initially developed to negate the use of clamping a particle velocity. The velocity update equation with added inertia is formulated in equation (5.13) [30].

$$v_i(t+1) = wv_i(t) + c_1\phi_1(t)[pbest - x_i(t)] + c_2\phi_2(t)[gbest - x_i(t)] \quad (5.13)$$

The addition of inertia (w in equation (5.13) to the general velocity update equation is simple and elegant which is why it has been adapted to a wide variety of PSO algorithms. The inertia value allows one to control the amount of control the social and cognitive components have with regard to velocity updates [30].

For values of w that are greater than 1, a large amount of momentum is preserved as the particles velocity are updated and hence, the particle explores more [30]. When $w < 1$, each time the particle updates it velocity is loses a certain amount of momentum, hence, the particle seems to slow down allowing it to exploit the current solution space in finer detail [30].

Even though inertia was developed to remove the use of velocity clamping, it did not entirely achieve its goal. For values of w that are greater than 1, the particle keeps a lot of momentum and accelerates even more. Therefore, as the particle accelerates its step size through the solution space gets larger and larger, increasing the probability that it will miss a good solution. This disadvantage is only applicable for algorithms that keep the inertia value static [30, 31].

To allow for a greater trade-off between exploration and exploitation, the inertia value was made dynamic. Exploration is favoured early on in an optimization algorithm and in return exploitation is favoured later on the algorithm when it is near an optimum. Thus, various methods linear decreasing and non-linear decreasing and have been developed that modify the inertia component as the algorithm moves around in the solution space [30, 31].

Finally, a similar inertia type component was developed from the analyse of particle dynamics [30]. This new component is called the *constriction coefficient* and like the inertia above, also modifies the velocity update equation slightly [30, 31, 78].

This modification can be observed in equation (5.14), which is the standard velocity equation with the constriction coefficient. The constriction coefficient is formulated in equation (5.15) [30, 31, 78].

$$v_i(t+1) = \chi[v_i(t) + c_1\phi_1(t)[pbest - x_i(t)] + c_2\phi_2(t)[gbest - x_i(t)]] \quad (5.14)$$

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \quad (5.15)$$

The constriction coefficient is represented by the value *phi* and allows

one to omit the usage of velocity clamping. The constriction coefficient evaluates to an ever decreasing value between $[0, 1]$. By using the constriction coefficient the PSO algorithm is also guaranteed to converge for values of $phi \geq 4$ and $\kappa \in [0, 1]$. As with the inertia discussed above, high values of κ allows for greater exploration and slow convergence. Where as low values of κ forces the algorithm to exploit the solution space and converge quickly [30, 31, 78].

5.5.4 PSO on the FAP

The PSO algorithm is also a relatively new algorithm and has been applied to only a handful of NP-Complete problems that includes the FAP. In this dissertation the PSO algorithm will be utilized on the FS-FAP to try and produce optimal solutions.

In the literature only two groups have presented research where the PSO has been applied on the FAP to produce a near optimal solution. The research presented by these groups concentrate on the MS-FAP variant of the FAP. Thus, the aim of their algorithm is to reduce the span of frequencies used, where as the problem this dissertation is concerned with is the FS-FAP where the amount of interference generated needs to be minimized.

To date, no PSO algorithm has been presented which has been designed to operate on the FS-FAP variant of the FAP. Therefore, the interest in the research presented 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 optimization procedure.

A general overview of the work produce by these groups will now be presented.

In the research presented by Elkamchouchi et. al [29] a PSO algorithm is applied to produce optimal solutions for the MS-FAP. The way the authors went about to assign frequencies in their algorithm is known as the Frequency Exhaustive Assignment (FEA). This method works by first generated a list of calls, called a *call list* denoting calls that occur in the system [29].

The method then iterates over the calls in the list and assigns the lowest possible frequencies to the calls without violating interference constraints

[29]. 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 [29].

As can be observed from the above discussion, the PSO has been successfully applied to the FAP. The particular variant of the FAP was the MS-FAP and also the authors PSO is tasked with minimizing constraint violations.

With 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. In addition to the successful application of the PSO to the FAP other factors also influenced the final decision to utilize the PSO instead of another algorithm.

The PSO algorithm is short and elegant. As discussed in the overview of the PSO, the algorithm utilizes the "flying" approach to search the problem space. With this approach one point progressively move towards another. Using this approach, the algorithm able to adequately explore the problem space much more thoroughly.

The algorithm also makes extensive use of knowledge gained by the various particles as the search the problem space. Not only does each particle keep personal history (with pbest) but the swarm as a whole keeps a history of the best particle with (with gbest). Thus, with regard to FAP, it is possible that even though a particle might be in a overall bad position, it might have some small bit of good knowledge being overshadowed by bad knowledge. Through the extensive use of historic knowledge good information is more likely to be shared or kept slightly longer in the algorithms collective knowledge.

For this dissertation the PSO will applied to the FS-FAP and thus the approach as used in by the authors in the above literature cannot be used. FS-FAP is concerned with interference generated and there are some constraints which cannot be broken, where the MS-FAP the performance measure is explicitly the amount of constraints violated.

The PSO algorithm might be short and elegant, but applying it the FS-FAP requires various new techniques in response to the following questions

- How to best represent a particle as a frequency plan ?
- How to "fly" one frequency plan to another ?
- How to avoid particles using forbidden frequencies when they fly towards a particular plan ?

The above questions are only the preliminary questions and are in fact problem that must be addressed for a successful application of the PSO to the FAP. In chapter 7 a discussion will be presented on how these problems were solved as well as how other problems were solved that were not anticipated.

In this section a discussion was presented which discussed literature that utilized the PSO on the FAP. Furthermore, it was formally stated that the PSO would be the primary algorithm in this dissertation to be applied to the FAP.

This section concludes the discussion on the PSO as well as the Swarm Intelligence chapter. In the following section a summary is presented on this chapter.

5.6 Summary

In this chapter a discussion was given on three swarm intelligence algorithms. The first section presented a discussion on the Ant Colony Optimization Algorithm.

The general flow of the algorithm was presented with the help of a diagram (see figure 5.1) as well as an explanation with regard to how the algorithm came about and the basic flow of the algorithm. The defining characteristics of the algorithm were also discussed. The discussion on the ACO algorithm concluded with a literature review of the ACO being applied to the FAP.

The second section that was presented provided an overview of the Artificial Bee Colony Optimization Algorithm. A discussion was presented on how the algorithm was developed and how the algorithm performs its search in a problem space. A diagram was also presented that outlines the general flow of the ABC algorithm.

Also within the second section, a series of defining characteristics are presented and explained. Each characteristic is a defining attribute of the algorithm that makes it unique with regard to other algorithms. No literature study was presented on the algorithm being applied on the FAP since to date; no research has been presented of such an ABC algorithm.

This chapter concluded with a discussion on the most important algorithm. The algorithm this dissertation will be using to apply it to the FAP,

namely the Particle Swarm Optimization algorithm. In the last section how the algorithm was developed was discussed as well as the general flow of the algorithm when search the problem space.

A diagram depicting the flow of the PSO algorithm was also presented (see figure 5.4). Furthermore, characteristics that make the algorithm unique were explained. The final section of this chapter concluded with a literature review of the PSO algorithm being applied to the FAP.

Part II

Implementation

Chapter 6

PSO on benchmark functions

6.1 Introduction

In this section a discussion will be presented on a series of optimisation benchmark functions. First a formulation of all the benchmark functions will be given. In the second section all the benchmark funcions will be plotted on a 3d graph. Finally this chapter will conclude with a presentation of the results obtained by the PSO algorithm.

The functions vary from being relatively easy to optimize, to functions that contain lots of local minima and a slightly concealed global minmima. In total fourteen benchmark functions will be formulated and benchmarked againts.

If one observed the following research [30, 31, 126]. Various optimisation algorithms are benchmarked such as Genetic Algorithm, Artificial Bee Colony algorithm, Tabu Search and Simulated Annealing. In the presented research the algorithms are benchmarked using numerical optimization functions.

Numerical optimization functions are good candidates to test optimisation algorithms as with a few slight changes the function operates in more dimensions or can have more or less optima [30,31,126]. Being able to alter these functions is a desirable traite as it enables one to accurately benchmark an algorithm, not only with regard to how the algorithm coupes with increased dimensionality but also in the algorithms accuracy in locating optima [30, 31, 126].

The reason why these functions have variate amount of local and global optima, it to test various factors on how good the algorithm is that is being applied on the function. The factors that are tested are [30, 31]:

- Rate of convergeance
- Exploration
- Exploitation
- Diversity
- Breaking out of local minima
- Information sharing

As discussed previously the numerical functions have predetermined optima, which means researchers are able to produce statistical information on how the algorithm performs. For instance, researchers will not be able to measure accurately the performance of the algorithm on a NP-Complete problem [30, 31]. Yes, they can compare results with what other algorithms have produced, but cannot with absolute certain say or measure the algorithm convergence,diversity etc on NP-Complete problems as their problem spaces are huge [130].

With these benchmark functions, the optima has been mathematically calculated and their position is known within the problem space [130]. Thus, researchers can now with surity measure the convergence rate, diversity and compare it with other algorithms, since the domain the algorithms operate it is not as specific as a NP-Complete problem [130]. Rather, the domain is mathematical and deterministic and therefore allows easy comparison [130].

In this dissertation, two PSO algorithm were developed specifically to measure the performance of the PSO algorithm and also to better understand the various underlying dynamics of the algorithm.

The first PSO that was developed was just the standard PSO algorithm with no constriction coefficient or inertia weight. The second PSO algorithm differed to the standard algorithm in the sense that it utilizes the notion of inertia to move particles.

Both of these algorithms have been applied to all fourteen benchmarks that are presented in the chapter and will be compared (where applicable)

to other optimisation algorithms that have also been applied to the same benchmark problems. The comparison between these algorithms will be presented in section 6.3.

In the next section, all the benchmarks that will be used for testing the PSO algorithms will be formulated. For the interested reader, 3D graphs along with the python code that generated the graphs is presented in the appendix. This chapter will conclude with a section on the results of the PSO algorithms that were developed being applied to the presented benchmarked as well as compared with other results that have been obtained by other algorithms and presented in the literature.

6.2 Test Functions

In this section all the test functions on which the two developed PSO algorithms will be benchmarked against will be presented. For each test function that is presented a mathematical formulation will be given and the global optimum will be explicitly stated. In addition to the formulation and global optimum, each function will also be classified whether it is a unimodal or modal function as well as whether it is seperable or non-seperable. Each of these classifications will now be explained.

Unimodal — A particular problem is classified as being unimodal when there is only clear solution. With only one clear solution, it means there is only one global optimum point in the solution space [30, 31, 57, 130].

Multimodal — A problem is multimodal when it has more than one defined solution. Thus, the particular problem space contains multiple global optima [30, 31, 57, 130].

Seperable — Functions that are classified as being separable have the characteristic when the function can be written as a series of summations of just one variable [57]. This quality makes the function easier to solve as the algorithm has only one variable to be concerend about [57, 130]. Seperable functions also have the inherent quality of being scalable, meaning they can be easily be adapted to higher dimensions [57, 130].

Non-seperable — Functions classifed as non-separable cannot be rewritten into a series of summation functions as the variables used in the functions have the characterisitc of being interrelated [57, 130]. The interrelation of the variables makes non-seperable functions more difficult than separable functions to solve since the algorithm has more interdependant variables to be concerned with [57, 130].

The De Jong test functions (F1, Shekel's Foxhole) are not considered to be the gold standard of testing optimisation algorithms [130]. The only reason for their extensive use in the literature is due to the functions being the first to be developed and applied to test an optimisation algorithm i.e. Genetic Algorithm [126, 130].

Since the inception of the De Jong test functions, additional functions have been developed which make it more difficult for an optimisation algorithm to locate the optimum [130]. These functions are more difficult in the sense, as they have multiple local optima, which does in actual fact leads the algorithm astray which is to say the problem space is deceptive [30, 31, 130].

Problems that have deceptive search spaces tests how good the algorithm is resistant to Hill-climbing[1] and hence how efficient the algorithm is in exploring the entire search space [130].

6.2.1 DeJong F1 Function

$$f(x) = \sum_{i=1}^{n} x_i^2, -5.12 \leq x_i \geq 5.12, i \in \mathbb{N} \tag{6.1}$$

The DeJong F1 Function has the following global minium when $f(x) = 0, x(i) = 0, i : n$ where n is the amount of dimensions [54, 55, 57, 58, 77, 98]. In the literature the function is classified as being unimodal and seperable [55, 77].

6.2.2 Shekel's Foxhole

$$f(x_1, x_2) = \{0.002 + \sum_{j=1}^{25} [j + (x_1 - a_{1j})^6 + (x_2 - a_{2j})^6]^{-1}\}^{-1} \tag{6.2}$$

[1] continously selecting what seems to be better moves, but in reality its moving towards a local optima peak

where

$$a = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & ... & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & ... & 32 & 32 & 32 \end{pmatrix}$$

the variables x_1 and x_2 are usually restricted to the square represented by $-65.356 \leq x_1 \leq 65.357, -65.357 \leq x_2 \leq 65.356$ [17, 55, 58, 77]. The global optimum is when $f(x_1, x_2) = 0, \{x_1, x_2\} = \{-32, -32\}$ [17, 55, 58, 77].

The matrix controls the holes that appear in the search space. The interested reader is directed to the 3D graph rendering of this function presented in the appendix on page 184 for a visual representation. In the literature the function is classified being multimodal and seperable [55, 77, 78].

6.2.3 Rastrigin

$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)],\ i \in \mathbb{N} \tag{6.3}$$

The values of the variable x_i is bounded by the hypercube $-5.12 \leq x_i \leq 5.12$ [50, 54, 55, 57, 77, 78, 98]. The global optimum for the function is when $f(x_i) = 0,\ x_i = 0,\ i = 1, \ldots, n$ [50, 55, 57, 77, 78].

Rastrigins function is based on DeJong's first function equation 6.1 adding a cosine term, which in turns alters the problem space by introducing many local minima [50, 54, 57, 77]. In the literature the function is classified being multimodal and separable [3, 50, 54, 55, 57, 77, 78, 98].

6.2.4 Schwefel

$$f(x) = 418.9829n - \sum_{i=1}^{n} [x_i \sin\sqrt{|x_i|}], \qquad i \in \mathbb{N} \tag{6.4}$$

The variable x_i is restricted to be in the hybercube $-500 \leq x_i \leq 500, i = 1, \ldots, n$ [35, 50, 55, 57, 77]. The global optimum for the function is $f(x) = 0$ when $x_i = 420.9687$ [35, 50, 55, 57, 77].

If one observes the 3D rendering of the schwefel function problem space on page 185, one can see that the search space contains a great number of peaks which might be local optima. The function also has the characteristic of having a second best optima far from the global optima which many algorithms get trapped in [35, 50, 55, 57, 77]. In the literature the function is classifed as being multimodal and separable [50, 55, 57, 77].

6.2.5 Griewank

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \qquad i \in \mathbb{N} \tag{6.5}$$

The variable x_i is bounded to within the hypercube $-600 \leq x_i \leq 600$ [54, 55, 57, 58, 77, 98]. The global optimum of the function is when $f(x) = 0$ which occurs when $x_i = 0, i = 1, \ldots, n$ [54, 55, 57, 58, 77, 98].

As with the schwefel function, the Griewank function also has a great number of peaks and valleys which many algorithm get trapped in. A particular quality of the griewank function is at low dimensions, the function is quite difficult to solve, whereas it has been shown that at higher dimensions the function becomes much easier due to there being less peaks and vallyes to nagivate in the search space [54,55,57,77,130]. In the literature the function is classified as being multimodal and non-separable [3, 54, 55, 57, 77, 78].

6.2.6 Salomon

$$f(x) = -\cos\left(2\pi \sum_{i=1}^{n} \sqrt{x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2} + 1, \quad i \in \mathbb{N} \tag{6.6}$$

Unlike the previous functions discussed in this section, the Salomon function imposes no constraint on the x_i variable. The global optimum is when $f(x) = 0$ and $x_i = 0$ where $i = 1, \ldots, n$. This particular function seems to have been applied as benchmarking function yet, as no literature can be found. Nontheless, the function is indeed a numerical optimisation function that is classified as being multimodal and non-separable [**?**].

6.2.7 Ackley

$$f(x) = -20e^{-0.2\sqrt{\frac{1}{2}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{2}\sum_{i=1}^{n} \cos 2\pi x_i} + 20 + e^1, \qquad i \in \mathbb{N} \tag{6.7}$$

The variable x_i is restricted to the hypercube represented by $-32.768 \leq x_i \leq 32.768$ [55, 57, 77, 98]. The global minimum is when $f(x) = 0$ and is obtainable for $x_i = 0, i = 1, \ldots, n$ [55, 57, 77, 98].

As can be observed from the mathematical formulation of the function, the function utilizes a exponential term. By using the exponential term the problem space contains numerous local optima which requires an algorithm to search much wider to avoid getting trapped. The literature classifies this function as being multimodal and non-separable [55, 57, 77, 78].

6.2.8 Six-Hump Camel Back

$$f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^{\frac{4}{3}})x_1^2 + (x_1x_2) + (-4 + 4x_2^2)x_2^2 \tag{6.8}$$

The variables x_1 and x_2 are subject to the following boundary constraints $-3 \leq x_1 \leq 3$ and $-2 \leq x_2 \leq 2$ [35, 77]. The global minimum is when $f(x_1, x_2) = -1.0316$ and is obtained when $x_1 = -0.0898$ and $x_2 = 0.7126$ or when $x_1 = 0.0898$ and $x_2 = 0.7126$ [35, 77].

True to its name the function has six peaks, four of which are local minima and two that are global minima as has already been defined. The literature classifies this function as being multimodal and non-separable [55, 77].

6.2.9 Shubert

$$f(x_1, x_2) = -\sum_{i=1}^{5}(i\cos(i+1)x_1 + 1)\sum_{i=1}^{5}(i\cos(i+1)x_2 + 1) \tag{6.9}$$

The search domain is constrained to $-10 \leq x_i \leq 10, i = 1, 2, \ldots, n$ [17, 55, 58,77]. The global optimum which is when $f(x_i) = -186.7309$ [17,55,58,77].

As can be observed from the 3D graph presented in the appendix on page 188 the landscape of the Shubert function contains various peaks and slopes. Thus, the function is classified as being multimodal and non-separable [55, 77].

6.2.10 Himmelblau

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{6.10}$$

The variables x_1, x_2 are constraint to be within the hypercube represented by $-6 \leq x_1 \leq 6, -6 \leq x_2 \leq 6$ [55, 77]. The Himmelblau function contains no local optima, but on the contrary, it has 4 global optima when $f(x_i) = 0$ which can be obtained at the following points [55, 77].

- $(x_1, x_2) = (-3.779310, -3.283185)$
- $(x_1, x_2) = (-2.805118, 3.131312)$
- $(x_1, x_2) = (3, 2)$
- $(x_1, x_2) = (3.584428, -1.848126)$

The literature classifies the function as being multimodal and non-separable [55, 77].

6.2.11 Rosenbrock Valley

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \tag{6.11}$$

The variable x_i is bounded to with the following constraint $-2.048 \leq x_i \leq 2.048$ [50, 54, 55, 57, 98, 116]. The global optimum is when $f(x) = 0$ and is obtained when $x_i = 1, i = 1, \ldots, n$ [50, 55, 57, 98, 116].

The rosenbrock search space has a curving valley which leads forces algorithms to coupe with the changing direction of the landscape [3, 50, 54, 55,57]. If the algorithm does not adapt to the changing direction it will fail in locating the global optimum. The function is classified as being multimodal and non-separable [3, 50, 54, 55, 57].

6.2.12 Dropwave

$$f(x) = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{\frac{1}{2}(x_1^2 + x_2^2) + 2} \tag{6.12}$$

The variables x_1 and x_2 are restricted to be within the following bounds $-5.12 \leq x_i \leq 5.12$ [77]. The landscape of this function resembles that of a droplet falling into a pool of water, as can be observed from the 3D graph presnted in the appendix on page 189. Due to its "wave" nature, the function has various local minima and only a single optima which is when $f(x_1, x_2) = 0$. The function is classified being multimodal and non-separable [77].

6.2.13 Easom

$$f(x_1, x_2) = -\cos(x_1)\cos(x_2)e^{(-(x_1-\pi)^2-(x_2-\pi)^2)} \tag{6.13}$$

The variables x_1 and x_2 are restricted to be within the hypercube represented by $-100 \leq x_1 \leq 100, -100 \leq x_2 \leq 100$ [58, 77, 116]. The global minimum is when $f(x_1, x_2) = -1$ and is obtainable if $(x_1, x_2) = (\pi, \pi)$ [58, 77, 116].

The easom function has a deceptive global minimum as it is very close to other local minima [17, 55]. As can be observed from the 3D graph of the function on page 190, the search space is a flat with the global minima

clearly visible. Flat search spaces are difficult to navigate by algorithms as the surrounding area gives no indication whether the algorithm is on the right track of not. The function is classied as being multimodal and non-separable in the literature [17, 55, 116].

6.2.14 Branins

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos x_1 + e \tag{6.14}$$

where

$$\begin{aligned} a &= 1 \\ b &= \frac{5.1}{4\pi^2} \\ c &= \frac{5}{\pi} \\ d &= 6 \\ e &= 10 \\ f &= \frac{1}{8\pi} \end{aligned}$$

The variables x_1 and x_2 are subject to the following boundary constraints $-5 \leq x_1 \leq 10, 0 \leq x_2 \leq 10$ [17, 55, 58, 77]. The global optimum is when $f(x_1, x_2) = 0.397887$ and is obtainable when x_1 and x_2 have the following values [17, 55, 58, 77]:

1. $x_1 = -\pi,\ x_2 = 12.275$
2. $x_1 = \pi,\ x_2 = 2.275$
3. $x_1 = 9.42478,\ x_2 = 2.475$

The literature classifies this function as being multimodal and separable [17, 55, 58, 77].

6.2.15 Michalewicz

$$f(x) = -\sum_{i=1}^{n} \sin(x_i)[\sin(\frac{(1 - x_i^2)}{\pi})]^{2m}, \qquad i, m \in \mathbb{N} \tag{6.15}$$

The variable x_i is usually constricted to the following defined boundary $0 \leq x_i \leq \pi, i = 1, \ldots, n$ [55, 77]. The parameter m defines the steepness of the valleys in the function. The function has two approximated global minima's which are difficult to locate since their size in comparison to the rest of the search space is relatively small [55, 77].

1. $f(x) = -4.687,\ n = 5$
2. $f(x) = -9.66,\ n = 10$

In the literature the function is classified as being multimodal and separable [55].

6.2.16 Goldstein

$$\begin{aligned} f(x_1, x_2) &= (1 + (x_1 + x_2 + 1)^2) \\ &= *(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \\ &= *(30 + (2x_1 - 3x_2)^2 \\ &= *(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2) \end{aligned}$$

The variables x_1 and x_2 are subject to the following boundary constraints $-2 \leq x_1 \leq 2, -2 \leq x_2 \leq 2$ [17, 55, 58, 77, 116]. The function has only one global minimum four local optima [17, 55]. The global optimum is when $f(x_1, x_2) = 3$ and is obtainable when $x_1 = 0$ and $x_2 = -1$ [17,55,58,77,116].

The literature classifies this function as being multimodal and non-seperable [55].

6.3 Results

In the previous section sixteen benchmark functions were mathematically defined, where applicable comments were given on the search space. For each function the following was explicitly stated:

- The global optimum and where is located.
- Whether the function is unimodal or multimodal.
- Whether the function is non-separable or separable.

In this section the results will be presented of how the two PSO algorithms that were developed performed on each of the presented benchmark functions.

The algorithms will be compared with the following criteria on each benchmark.

- The number of iterations the algorithm took to find the global optimum
- The accuracy of the algorithm, which means if the algorithm has indeed located the optimum or if not, how far off is the algorithm.
- The PSO is a population based algorithm, thus diversity is important. Diversity will be measured and compared on each benchmark function.

For each of the above criteria a subsection will be preseted. Where possible the results obtained will be compared to other algorithms that have been applied to similar test functions.

6.3.1 Iterations

Function	PSO	PSO*	GA	TS	ABC
Name	VALUES	VALUES	VALUES []	–	–
DeJong F1	–	–	–	–	–
Shekel's Foxhole	–	–	–	–	–
Rastrigin	–	–	–	–	–
Schwefel	–	–	–	–	–
Griewank	–	–	–	–	–
Salomon	–	–	–	–	–
Ackley	–	–	–	–	–
Six-Hump CamelBack	–	–	–	–	–
Shubert	–	–	–	–	–
Himmelblau	–	–	–	–	–
RosenbrockValley	–	–	–	–	–
Dropwave	–	–	–	–	–
Easom	–	–	–	–	–
Branins	–	–	–	–	–
Michalewicz	–	–	–	–	–
Goldstein	–	–	–	–	–

6.3.2 Diversity

Function	PSO	PSO*	GA	TS	ABC
Name	VALUES	VALUES	VALUES []	–	–
DeJong F1	–	–	–	–	–
Shekel's Foxhole	–	–	–	–	–
Rastrigin	–	–	–	–	–
Schwefel	–	–	–	–	–
Griewank	–	–	–	–	–
Salomon	–	–	–	–	–
Ackley	–	–	–	–	–
Six-Hump CamelBack	–	–	–	–	–
Shubert	–	–	–	–	–
Himmelblau	–	–	–	–	–
RosenbrockValley	–	–	–	–	–
Dropwave	–	–	–	–	–
Easom	–	–	–	–	–
Branins	–	–	–	–	–
Michalewicz	–	–	–	–	–
Goldstein	–	–	–	–	–

6.3.3 Accuracy

Function	PSO	PSO*	GA	TS	ABC
Name	VALUES	VALUES	VALUES []	–	–
DeJong F1	–	–	–	–	–
Shekel's Foxhole	–	–	–	–	–
Rastrigin	–	–	–	–	–
Schwefel	–	–	–	–	–
Griewank	–	–	–	–	–
Salomon	–	–	–	–	–
Ackley	–	–	–	–	–
Six-Hump CamelBack	–	–	–	–	–
Shubert	–	–	–	–	–
Himmelblau	–	–	–	–	–
RosenbrockValley	–	–	–	–	–
Dropwave	–	–	–	–	–
Easom	–	–	–	–	–
Branins	–	–	–	–	–
Michalewicz	–	–	–	–	–
Goldstein	–	–	–	–	–

Chapter 7

Applying the PSO to the FAP

7.1 Introduction

Particle Swarm Optimization (PSO) as discussed in the overview (see page 120) is an algorithm that is largely based on the flying behaviour exhibited by a flock of flying birds. Which is why the core of the algorithm is based upon vector math, with new positions and velocities being calculated after each iteration of the algorithm. Thus, each particle position, is represented by a D-dimensional vector and is then simulated flying through the D-dimensional space using the velocity equation (see **??**).

The performance of the Global PSO is benchmarked and outline in the previous chapter. Most of the problems to which PSO has been applied to, to date have been problems where the position of particles have a constant D-dimensional space, which is to formally state *the dimensionality of a particle position in its entirety, is constant.*

This constant dimensionality introduces a intriguing problem if you want to apply the PSO to an inherent multi-dimension problem like the Frequency Assignment Problem (FAP). In this chapter we will provide a discussion on how, we the authors, went about in applying the PSO to the FAP.

We will start of by first explaining what we deemed as a position for a particle in the Frequency Planning domain. This definition of the particle position is important because it plays a central part in how we "fly" our

particles through the Frequency planning domain. After the position definition, we will define how each position will be evaluated and formally define the fitness function our swarm will use.

Arguably the most important part of the swarm, is how we calculate the velocity of a particle and moving it to a new position in the search space. A discussion on the velocity function we have developed will be discussed in the section after the fitness function.

After the velocity function section we will discuss a new mechanism for selecting the global best which allowed us to get better fitness values and therefore direct the swarm more. Finally we will end this section of with a section on the how the swarm utilizes history to produce better results.

7.2 A Position in the Frequency Planning domain

In this section we will describe what a position is in the Frequency Planning domain. We will start of by first describing what a Frequency Plan is as well as provide the general structure to represent such a plan. We will also describe some of the hard and soft constraints and how it molds the plan to be suitable for a network.

A Frequency plan, is almost exactly as the name implies. A plan that outlines frequency usage for a wireless network. The benchmark problems we will be using all pertain to cellular phone networks. For Cellular networks, the frequency plan outlines what frequency must be allocated to what transceiver. With this basic definition, the problem seems relatively trivial to solve if one assumes that one either has a infinite number of frequencies or the amount of frequencies available to our disposal is more than the amount of transceivers in the network.

The reality is, that there are only a finite amount of frequencies available for cellphone transmissions. Hence a regulatory body needs to assign frequencies to cellphone network operators for use in their networks. A regulatory body is needed because, if a network operator just uses any frequency it wants, it is bound to interfere with someone else also utilising the frequency.

The assigned frequencies are also not a huge portion of the entire spectrum. If we look at one of our benchmark problems, Siemens1, the alloted spectrum is from frequency 16 to frequency 90. Which gives the network

operator 74 frequencies to use in its network without considering other constraints.

Besides the Electro-magnetic constraints that are also applicable here, there are regulatory constraints, like 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. There are also locally blocked frequencies, which apply only in certain regions of the geographical landscape of the network.

As discussed in chapter 2 (page 3) and 3 (see page 27). A Cellphone network is divided into a number of cells, and each cell requires a certain number of transceivers to service its corresponding area. This number of transceivers is based on the expected volume of traffic that a particular cell will experience at peak network usage. Due to this variability in the expected amount of traffic a cell is supposed to handle, the nature of the frequency assignment problem is of a multi dimensional nature. As can be seen

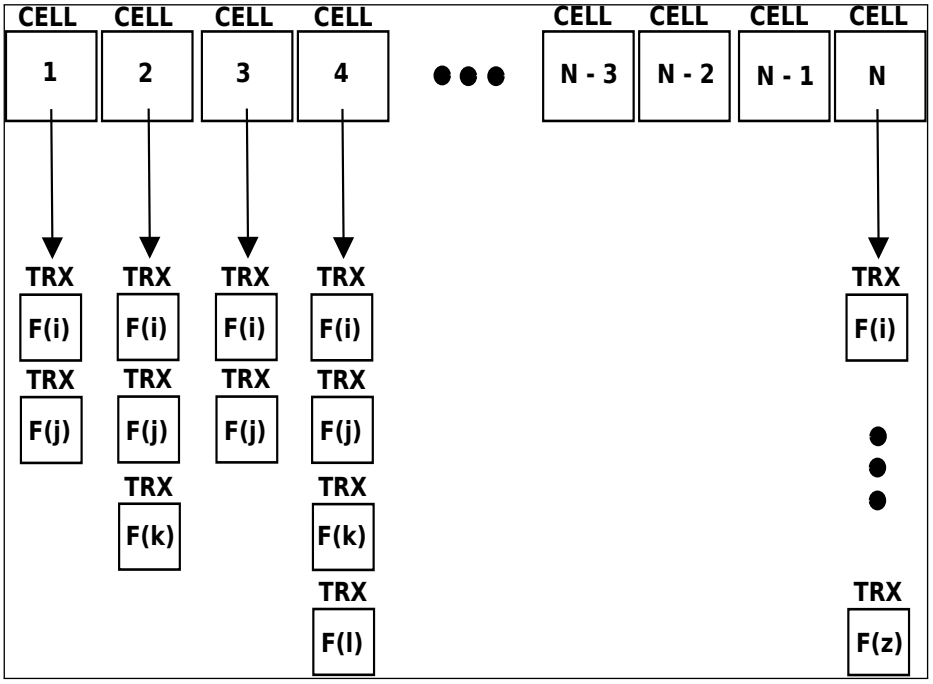

Figure 7.1: The Structure of a Frequency Plan

in figure 7.1 a cellular network can have any amount (N in the figure) of cells to attain the desired coverage over the geographical landscape. In our benchmarks problems the cellular networks have a number of cells ranging

from 500 to 1000+. The most important part of the plan, is the actual TRX's within each cell. In the figure we can clearly see, how the amount of TRX's vary from one cell to the next. $F(i)$ is a frequency at position i from the available usable spectrum.

Note 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 their is indeed interference, how much is inferred as a result. All this information isn't part of the plan. Instead this information, for purpose of this dissertation is supplied by the benchmark.

This information is referred to as the interference matrix. Within this interference matrix each entry references two cells entries Cell A and Cell B. The entry then lists the amount of interference that occurs when Cell B interferes with Cell A[1].

A Frequency Plan is a possible solution to the Frequency Assignment Problem. Therefore in the PSO, we the authors have developed, each Particle's position in the solution space is represented by a frequency plan. As mentioned earlier, moving particles through the frequency plan solution space introduces an interesting problem due to the multi-dimensionality of a plan. We will describe how particles are moved from one position to another through the solution space in section 7.4

In this section we have given a description of what a our PSO will use a position in the frequency planning domain. We outlined the general structure of what a frequency plan is as well as defined how the interference values are retrieved when two cells interfere. In the next section we will define the fitness function that our PSO's uses.

7.3 The Fitness Function

In this section we will provide a discussion on the fitness function we utilise in all the variants of our swarms to rate each individual particle position of the swarm.

As discussed in the previous section, the set of benchmark problems we apply our particle swarms to define the amount of interference incurred when two cells assigned frequencies interfere. This interference information is

[1]Interference occurs based on the electromagnetic constraints as defined in chapter 3

referred to as the interference matrix. Each pair of cells has two interference values defined, The first value referred to as Co-channel interference is when the frequency of one TRX is the equal to a TRX in the other cell. The second value, called adjacent channel interference is when the frequency of a TRX in one cell differs by 1 with another TRX from the other cell.

Evaluation of a frequency plan to determine its fitness value is a simple process. 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 TRX in the first cell is checked with all the TRX's of the other cell. Depending on whether how the frequencies differ from each other, the fitness procedure adds either the co-channel or the adjacent channel interference to a summing variable. This procedure is mathematically defined in chapter 3 see page 41 for the formal equation.

With regard to our benchmarks, not all interference values are added to the summing variable, since each of the benchmarks define a Minimum Tolerable interference variable. Which means that if a given interference value is either equal or less than this defined value it is acceptable and won't have a impact on the overall plan.

In this section we outlined the basic fitness function our PSO will use to rate the feasibility of particles after each itertation. In the next section we will define how the particle move from one iteration to the next 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 point in the solution space.

The velocity function doesn't 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 on page 127.

In this section we will provide a discussion on the process we the authors went through to develop a velocity function that is suitable for particles to

move from one Frequency plan to another. We will start of explaining our first and worst method. With each method we define we will give an outline of the problems associated with it. We will end of this section with our primary method which has obtained the best results.

7.4.1 Movement in the Frequency Planning domain

The standard velocity equation works on the basis of vector math. Each particle has a velocity and position which is represented by a standard mathematical vector. The standard equation basically just alters the direction the particle is moving to move to a more promising position in the solution space.

Vector math has standard basic operations defined for adding,subtracting and multiplication. Hence, applying the PSO to problems that are either mathematical functions or problems that map well to the vector domain is a straight forward trivial process. With regard to the Frequency planning domain an important question needs to be answered. How to move one multi dimension frequency plan to another ?

We the authors answered the question by thinking of the frequency plan and its dimensionality in a different manner. The most important realization is that, we don't have to develop a procedure that moves a whole plan to another taking into the account the dimensionality etc. Rather, we opted to disregard the plan as a whole and just apply the velocity function at a much smaller scale. Thus, the velocity function is applied at the lowest level of a frequency plan.

The basic idea about our velocity function is for the movement of the swarm to be at a much finer granularity. 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. Hence, the procedure goes into each cell defined in the frequency plan represented by the standard particle as well as the global best particle.

For each cell in the frequency plan, the standard PSO equation with inertia is applied to the TRX value. Thus each TRX value is moved in the general direction (on the frequency domain) towards the TRX value defined in the same cell of the global best frequency plan.

The velocity function we the authors have developed is therefore simplified from being a procedure that needs to move around in a multidimensional space, to move values in a 1 dimensional space. Therefore, our velocity function retains the simplicity of the original PSO algorithm as well as the general concept it is based upon.

7.4.2 Keeping frequencies bounded

We the authors, have defined the basic concept that we will use in our swarm to allow the individual particles to move around in the frequency domain. But this alone is not enough, since the swarm currently has no concept of the constraints that exist in the domain. These constraints include what frequencies are allowed to be used and which must be avoided. Therefore, the velocity function needs to be altered to make the swarm in some sense aware, and hence keep the particle positions bounded within the allowable search space.

The boundary check we implemented was fairly trivial. The check only applied when one of the following conditions were met after the calculated velocity was applied to the current position:

- If a TRX frequency is above the maximum allowable frequency (higher bound) given to the network.
- If a TRX frequency is below the minimum allowable frequency (lower bound) given to the network.

A mod operation is applied to the value to bring it within the allowable range. If for instance, the maximum allowable frequency is 50, and the TRX value (after velocity) is 56. The 56 value gets modded with 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 not so trivial case is when the frequency value is lower that 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 TRX value after movement is 15. Modding the TRX value of 15 with 20 has no effect. Therefore we have opted for the following methods to solve this problem:

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 whether it oversteps the bounds of the maximum allowable frequency and bounded accordingly.

2. Add the lower value to the minimum allowable frequency. The resultant value is checked whether it oversteps the bounds of the maximum allowable frequency and 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 in the realm of possibility that a TRX value after movement might contain a negative value. Our boundary check solves this problem by first taking the absolute value of negative TRX value. The boundary check then treats the now positive TRX value, as a normal value that needs to be bounded.

7.4.3 Using indices instead of frequencies

In the first iteration of our velocity equation the swarm worked with raw frequency values. But upon closer inspection and the frequency range the swarm was using to move around we noticed that the swarm wasn't prohibited from using Globally Blocked Channels or Locally Blocked Channels. Hence, the swarm increasingly moved towards allocating these prohibited values to TRX's since the fitness function doesn't penalize the use of these frequency values. This is due in part because these values are under no circumstances allowed to be used and thus the fitness function isn't designed to check for these values.

To solve this problem, we the authors proposed two solutions:

1. Modify the fitness function to penalize a frequency plan if it uses any Globally Blocked Channels or Locally Blocked Channels.

2. Instead of letting the swarm work with raw frequency values, rather let the swarm work with array indices. These array indices indicate positions in a 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 array.

With the first solution, the fitness function will have to be modified to levy a penalty if a prohibited frequency value is used. The first proposed solution was disregarded because it introduces complexity which can be completely avoided with the second proposed solution.

Where as with the second solution the fitness function will not have to be modified and the boundary check is simplified since there is no need to check for a lower bound anymore. The boundary check only now has to check for negative index values and if the higher bound is violated which is now, the size of the array.

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 in the velocity function section and in the Particle Velocity section on page 127 each particle position is then modified to move in the general direction of the global best and personal best position. Therefore the global best acts as a beacon for the rest of the swarm in the solution space to indicate where good solutions seem to be for the rest of the swarm.

Initially the method we the authors used for selecting the global best in our PSO for the FAP didn't not differ at all from the traditional global PSO algorithm. The Swarm would loop through all the particle and apply the fitness function to determine the fitness of the particles position. The algorithm would then iterate over the swarm to determine which particle has the lowest fitness or in Frequency Planning terms or in Frequency planning terms, which plan has the lowest interference overall. The particle with the lowest fitness would then be the global best for that iteration.

Selecting the global best by evaluating the position as a whole seems to be a natural fit. But if we were to analyse a frequency plan in more detail; specifically how a single value in a resident TRX of a particular cell can affect the interference generated by the whole cell. One can come to the conclusion that it is entirely in the realm of possibility that, one bad value

in a TRX can overshadow a potential good value of a neighboring TRX similarly the total interference of a cell can overshadow a cell that had very low interference.

Therefore, in the traditional method of selecting the global best, a particle is actually selected because it contains less overshadowing TRX's. Hence potential good TRX values get lost.

We the authors went about to rectify this problem by exploiting the information the fitness exposes to us much more thoroughly. The information exposed by the fitness function allows us to see what effects certain values have on the interference of the cell. Therefore, to make better use of this information we developed two methods, each one being more fine grained than the other.

1. Besides the particle knowing its fitness, we changed the structure of a cell to allow a cell to know the interference it resident TRX's generate.

2. Besides the particle also storing the total fitness, each TRX of a particular cell also stores the interference it has generated.

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. The scheme we the authors implemented to take advantage of the information does not look at a particle position fitness as a whole. Instead, the procedure builds a global best position with this new information.

The global best building scheme works slightly different for each method. For method 1, the scheme loops through the entire swarm and selects the cell with the lowest interference. It then copy this cell and places it in the global best position at the same position it was found at. For method 2 the scheme follows the same procedure with the only difference being that the scheme now copies a TRX value and places it at the same position in the global best position.

When the swarm executed using this new scheme initially it didn't produce good results. This is largely due to the interference information for a cell in method 1 and for a TRX in method 2 gets reset to 0 after each iteration. Which seems to be correct, but in essence what is occurring is that after each iteration the swarm is effectively discarding all information gathered in that iteration.

To enable to this information to direct the swarm a bit more, we changed the algorithm to not reset the interference values per TRX and per Cell to 0. Instead, the interference values for an iteration is now added to the previous iteration interference values stored by the cell and TRX. This also has the net effect, that bad decisions made by the swarm for a particular particle get progressively worse as the swarm progresses.

We the authors, experimented with reseting the gathered information after a certain number of iterations. But it had no significant impact on the overall solution being produced. In some cases, the swarm moved the much worse solution and wasn't able to improve best particle to the previous best particle levels before the information reset.

With this small change to the structure of cells and TRX's. The swarm produced much better frequency plans that had lower total interference that all previous generated plans. In the next chapter we will present these results.

In this section we discussed a new method for selecting the global best particle from a swarm of particles. We also outlined the structural changes we had to make to enable to algorithm to use these new structural changes to generate lower frequency plans. Finally we discussed some of the problems encountered and how we the authors went about solving it. In the next section we will discuss how we incorporated the notion of a particle or rather a cell, keeping history of previous TRX's values.

7.6 Keeping History

In the traditional PSO history is retained by using the particle personal best position to direct the next movement of the particle. Other methods such as inertia also allows history to direct the movement of the particle. With regard to our PSO on the FAP, the algorithm we have develop 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 been previously used in a cell.

In the algorithm we developed, we borrowed the concept of Tabu Lists from the Tabu Search Algorithm. Using Tabu lists a particle will be able to better exploit the solution space it currently finds itself in. In our algorithm,

we incorporated Tabu Lists by adding to each cell a list which keeps track of resident TRX values in the cell for 20 iterations.

Care must be taken to select a max size of the Tabu List since one wants to keep enough history so that the search space can be adequately exploited. The Max Tabu List size must be less than the amount of available frequencies. Finally the max Tabu List cannot be too large, since the amount of checks the algorithm has to do to see if a value is Tabu is a very expensive operation. The operation is expensive, since for each potential value the Tabu List needs to be iterated through to see if the value is Tabu.

Only after the newly calculated velocity has been applied to the current position of a particular particle is the position checked for *collisions*. We say a collision occurs once a value is found to be in the Tabu List. To resolve a collision, we simply randomly select a new valid value which is just a index to a valid frequency value. This randomly selected value is then also checked if it collides with another value in the Tabu List.

Generation of randomly values continues until a counter variable reaches a value of 50 or any other predefined value. The collision resolving procedure then just simply accepts the last generated value as valid. The reason for the counter variable is because since no track is kept on previously generated values, the collision resolving procedure might indefinitely keep generating values all of them colliding with values presently in the Tabu List.

By incorporating Tabu Lists and the collision resolving procedure, the efficiency of the algorithm reduced dramatically. To increase efficiency of the operations in the algorithm, we paralyzed the collision resolving procedure, the velocity function and the sanitation procedures [2]. Therefore, multiple cells are moved simultaneously to towards other corresponding cells.

The paralization of these procedures increased efficiency of the algorithm. We the authors also noticed a slight side effect because of these operations. The randomness of our random number generator decreased. This effect was noticed because we outputed the counter variable in the collision resolver procedure. When the value was being outputted our algorithm produced much better results.

The reason for this is because outputting the variable inherently introduces a delay and therefore, the random numbers generators in other threads

[2]The sanitation procedure consist of all the constraint checks

have different seed values. Hence, with a delay in each parallel thread the numbers generated by the random number generator is more distinct. Without the delay, the because the of the parallelization some threads might start of with similar seed values because the current time is used a seed value for our random number generator [3].

Keeping the delay in mind and the effect is has on the final result, we introduced a delay in our collision resolving procedure. The reason the particular procedure was selected was because it was where the effect of delay was first noticed. After performing tests with delays of 5 milliseconds (ms), 10 ms, 15, 20 we settled on 20 ms since it was gave just enough time for a reasonably distinction between seed values used by other parallel threads.

In this section we discussed how we used Tabu Lists from Tabu Search to keep history. We discussed the alterations we had to make to the algorithm to incorporate the new feature as well as outlined the necessary procedures needed to effectively exploit the Tabu Lists. Finally we discussed the effects the Tabu Lists had on the performance of the algorithm and how we the authors went about solving it.

7.7 Summary

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

Chapter 8

Results

8.1 Introduction

Chapter 9

Conclusion

9.1 Introduction

Appendices

Appendix A

Plotting functions in 3D

A.1 Introduction

In this section all the functions presented in chapter 6 page 137 will be plotted in 3D.

The following graphs were generated using Matplotlib [1] which is a python library that provides similar functionality to Matlab.

For each of the benchmark functions, the 3D graph along with the python code that was used to generate the graph will be presented.

A.2 Code

A.2.1 DeJongF1 Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150

def DeJongF1(x,y):
        return x**2 + y**2


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)
```

[1] address

```
print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = DeJongF1(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('DeJongF1')
print '----Complete----'
```

A.2.2 Shekel's Foxhole Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150

def insertValuesIntoMatrix(matrix,value,row,index,timesToInsert):
        if matrix.size / 2 >= index + timesToInsert:
                for i in range(timesToInsert):
                        matrix[row,index+i] = value


def createDeJongF5Matrix():
        a = np.array([])
        a.resize(2,25)
        for i in range(2):
                value = -32
                for j in range(25):
                        a[0,j] = value
                        value = value + 16
                        if j > 0 and (j+1) % 5 == 0:
                                value = -32
                                valueIndex = ((j + 1) / 5) - 1
                                startIndex = valueIndex * 5
                                insertValuesIntoMatrix(a,a[0,valueIndex],1,startIndex,5)
        return a

def DeJongF5(x,y,matrix):
        sumj = 0;
        for j in range(25):
                sumi = 0
                sumi = (x - matrix[0,j])**6 + (y - matrix[1,j])**6
                sumj = sumj + (j + sumi)**-1
        return (0.002 + sumj)**-1


fig = plt.figure()
ax = Axes3D(fig)
```

```
X = np.linspace(-65.356, 65.356, AMOUNT_OF_POINTS)
Y = np.linspace(-65.356, 65.356, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

dejongList = []
print 'Initializing Function'
deJongMatrix = createDeJongF5Matrix()
for i in range(AMOUNT_OF_POINTS):
        val = DeJongF5(X[i],Y[i],deJongMatrix)
        dejongList.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(dejongList)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

ax.set_zlim3d(-40,500)
#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Shekel Foxhole')
print '----Complete----'
```

A.2.3 Rastrigin Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150

def Rastrigin(x,y):
        rastriginSum = 0
        rastriginSum += x**2 + 10*np.cos(2*np.pi*x) + 10
        rastriginSum += y**2 + 10*np.cos(2*np.pi*y) + 10
        return rastriginSum


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Rastrigin(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'
```

```
#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Rastrigin')
print '----Complete----'
```

A.2.4 Schwefel Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
SCHWEFEL_CONSTANT = 418.9829 * 2
def Schwefel(x,y):
        schwefelSum = 0
        schwefelSum += (-1 * x)*np.sin(np.sqrt(abs(x)))
        schwefelSum += (-1 * y)*np.sin(np.sqrt(abs(y)))
        return SCHWEFEL_CONSTANT + schwefelSum


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-500, 500, AMOUNT_OF_POINTS)
Y = np.linspace(-500, 500, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Schwefel(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Schwefel')
print '----Complete----'
```

A.2.5 Griewank Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np
```

```
AMOUNT_OF_POINTS = 150

#http://www.geatbx.com/docu/fcnindex-01.html
def Griewank(x,y):
        griewankSum1 = (x**2)/4000 + (y**2)/4000

        griewankSum2 = np.cos(x / np.sqrt(1)) * np.cos(y / np.sqrt(2))
        return griewankSum1 - griewankSum2 + 1


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-600, 600, AMOUNT_OF_POINTS)
Y = np.linspace(-600, 600, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Griewank(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Griewank')
print '----Complete----'
```

A.2.6 Salomon Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 350

def Salomon(x,y):
        SalomonSum1 = 0
        SalomonSum1 += x ** 2
        SalomonSum1 += y ** 2
        SalomonSum1 = -np.cos(2*np.pi*np.sqrt(SalomonSum1))
        SalomonSum2 = 0
        SalomonSum2 += (x ** 2)# + 1
        SalomonSum2 += (y ** 2)# + 1
        SalomonSum2 = 0.1 * np.sqrt(SalomonSum2)+1
        return SalomonSum1 + SalomonSum2


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5, 5, AMOUNT_OF_POINTS)
Y = np.linspace(-5, 5, AMOUNT_OF_POINTS)
```

```
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Salomon(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('SalomonTest')
print '----Complete----'
```

A.2.7 Ackley Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
#http://www.geatbx.com/docu/fcnindex-01.html
def Ackley(x,y):
        a = 20
        b = 0.2
        c = 2* np.pi
        ndiv = 1.0 / 2.0
        AckleySum = -a * np.exp(-b * np.sqrt(ndiv*(x** 2 + y ** 2)))

        AckleySum -= np.exp(ndiv * (np.cos(c*x) + np.cos(c*y))) + a + np.exp(1)
        return AckleySum


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-32.768, 32.768, AMOUNT_OF_POINTS)
Y = np.linspace(-32.768, 32.768, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Ackley(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'
```

```
#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Ackley')
print '----Complete----'
```

A.2.8 Six-Hump Camel Back Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
#http://www.geatbx.com/docu/fcnindex-01.html
def Camel(x,y):
        CamelSum = (4 - 2.1 * (x ** 2) + (x ** 4.0)/3.0) * x ** 2 + (x * y) + (-4 + 4 * y **2)* y ** 2
        return CamelSum


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-3, 3, AMOUNT_OF_POINTS)
Y = np.linspace(-2, 2, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Camel(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Camel')
print '----Complete----'
```

A.2.9 Shubert Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
```

```
def Shubert(x,y):
        answ1 = 0.0
        for i in range(4):
                answ1 += i * np.cos((i+1) * x + i)
        answ2 = 0.0
        for j in range(4):
                answ2 += i*np.cos((i+1) * y + i)
        return answ1 * answ2


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Shubert(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Shubert')
print '----Complete----'
```

A.2.10 Himmelblau Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Himmelblau(x,y):
        answ1 = (x ** 2 + y - 11)**2 + (x + y ** 2 - 7) ** 2
        return answ1


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Himmelblau(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
```

```
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Himmelblau')
print '----Complete----'
```

A.2.11 Rosenbrock Valley Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Rosenbrock(x,y):
        answ1 = 100 * ((x - y**2) ** 2) + (1-x)**2
        return answ1


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-2.048, 2.048, AMOUNT_OF_POINTS)
Y = np.linspace(-2.048, 2.048, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Rosenbrock(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('RosenbrockTest')
print '----Complete----'
```

A.2.12 Dropwave Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
```

```
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Dropwave(x,y):
        powsum = x ** 2 + y ** 2
        answ = 1 + np.cos(12 * np.sqrt(powsum))
        answ /= 0.5 * powsum + 2
        return answ


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
Y = np.linspace(-5.12, 5.12, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Dropwave(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Dropwave')
print '----Complete----'
```

A.2.13 Easom Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Easom(x,y):
        answ = -np.cos(x)*np.cos(y)*np.exp(-1 * ((x - np.pi) ** 2) - ((y - np.pi) ** 2))
        return answ


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-100, 100, AMOUNT_OF_POINTS)
Y = np.linspace(-100, 100, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Easom(X[i],Y[i])
```

```
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Easom_-100_+100')
print '----Complete----'
```

A.2.14 Branins Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Branin(x,y):
        a = 1
        b = 5.1 / (4 * np.pi) ** 2
        c = 5 / np.pi
        d = 6
        e = 10
        f = 1 / 8 * np.pi
        answ = a * (y - b * (x**2) + c*x - d)**2 + e * (1 - f)*np.cos(x) + e
        return answ


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-5, 10, AMOUNT_OF_POINTS)
Y = np.linspace(0, 15, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Branin(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
```

```
plt.savefig('Branin')
print '----Complete----'
```

A.2.15 Michalewicz Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Michalewicz(x,y):
        m = 20
        sumAnswx = np.sin(x) * (np.sin((1 - x ** 2) / np.pi))**(2 * m)
        sumAnswy = np.sin(y) * (np.sin((1 - y ** 2) / np.pi))**(2 * m)
        return -1 * (sumAnswx + sumAnswy)


fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(0, np.pi, AMOUNT_OF_POINTS)
Y = np.linspace(0, np.pi, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Michalewicz(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Michalewicz')
print '----Complete----'
```

A.2.16 Goldstein Code

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FixedLocator, FormatStrFormatter
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np

AMOUNT_OF_POINTS = 150
def Goldstein(x,y):
        answ = (1 + (( x + y + 1) ** 2) * (19 - 14 * x + ((3*x)**2) - 14 * y + 6 * x * y + ((3 * y) ** 2)))
        answ *= (30 + ((2 * x - 3 * y)**2) * (18 -32 * x +  (12 * x)**2 + 48 * y - 36 * x * y + (27 * y)**2))
        return answ  / 1000000
```

```
fig = plt.figure()
ax = Axes3D(fig)
X = np.linspace(-2, 2, AMOUNT_OF_POINTS)
Y = np.linspace(-2, 2, AMOUNT_OF_POINTS)
X, Y = np.meshgrid(X, Y)

print 'Initializing Function'
zValues = []
for i in range(AMOUNT_OF_POINTS):
        val = Goldstein(X[i],Y[i])
        zValues.append(val)
print '----Complete----'
print 'Create Numpy array from Function results'
Z = np.array(zValues)
print '----Complete----'
print 'Plotting function'
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
print '----Complete----'
print 'Setting various graph properties'

#ax.w_zaxis.set_major_locator(LinearLocator(10))
ax.w_zaxis.set_major_formatter(FormatStrFormatter('%.01f'))
fig.colorbar(surf, shrink=0.5, aspect=5)
print '----Complete----'
print 'Saving the finished graph to disk'
plt.savefig('Goldstein')
print '----Complete----'
```

A.3 Graphs

In this section 3D graphs of all the formulated functions will be presented. The 3D graphs of these functions enable one to more clearly see the problem space the algorithm is searching in.

In each of the graphs presented, on the right hand side a coloured scale is presented and the graph follows this coloured scale. The scale ranges from the maximum value to the minimum value encountered in the particular functions problem space. The maximum value in the problem space is indicated by the colour red and the minimum value in the problem space is indicated by the colour blue.

Finally the graphs are in 3 dimensions, the x and y dimensions represent any numerical number. The z dimension represents the value produce by using the function to evaluate the particular x and y coordinates.

A.3.1 DeJong's First Function

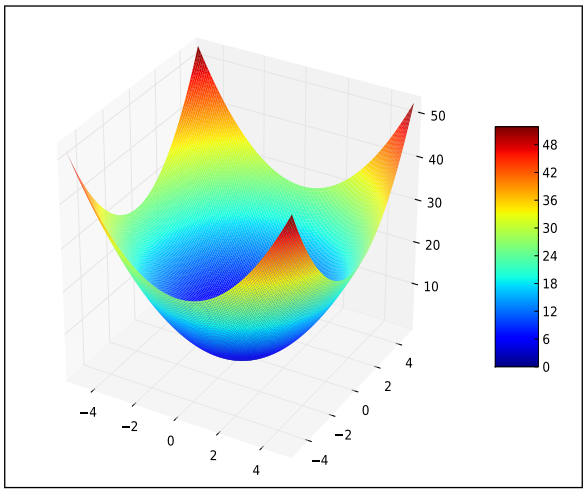

Figure A.1: DeJong's First Function

A.3.2 Shekel's Foxhole Function

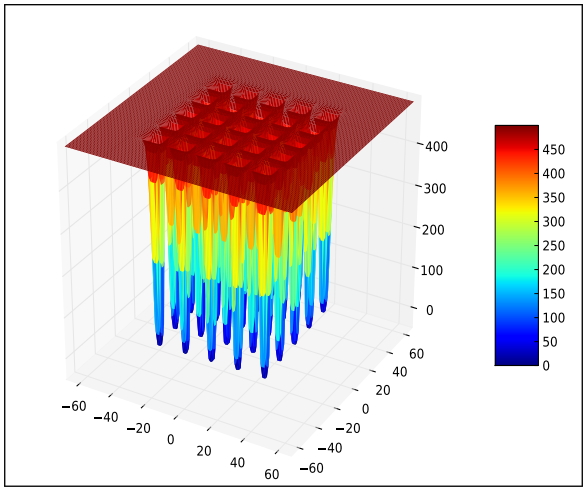

Figure A.2: Shekel's Foxhole Function

A.3.3 Rastrigin Function

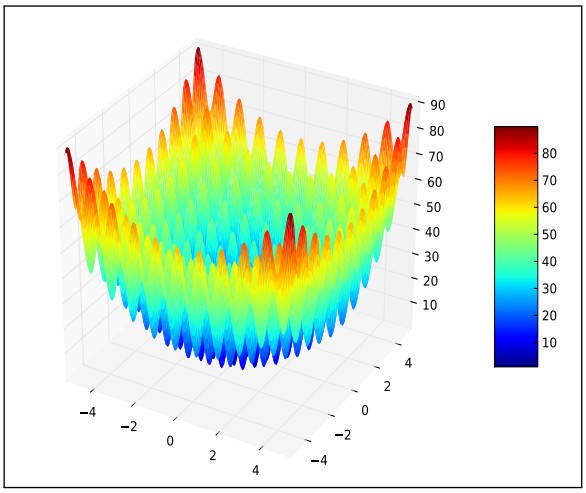

Figure A.3: The Function

A.3.4 Schwefel Function

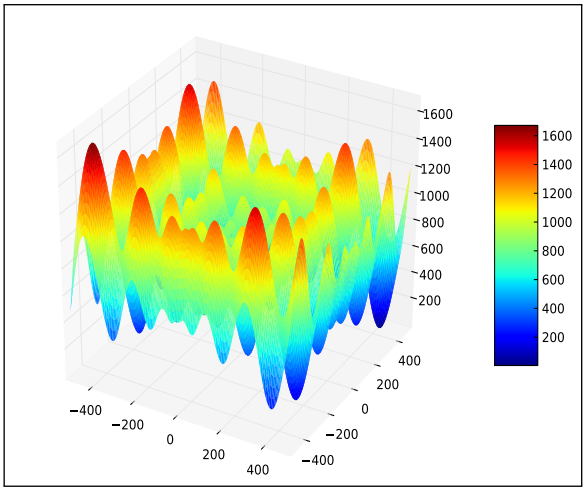

Figure A.4: Schwefel Function

A.3.5 Griewank Function

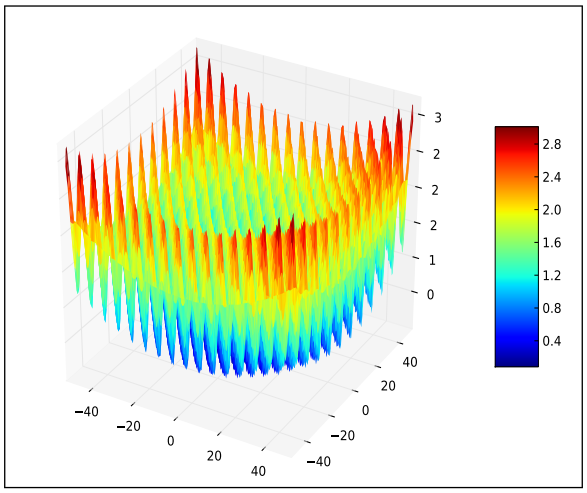

Figure A.5: Griewank Function

A.3.6 Salomon Function

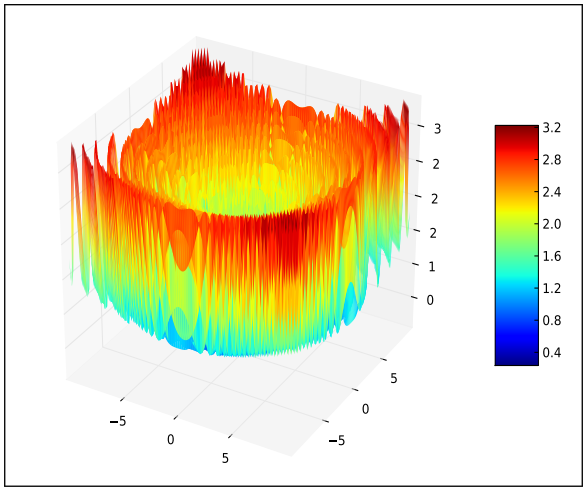

Figure A.6: Salomon Function

A.3.7 Ackley

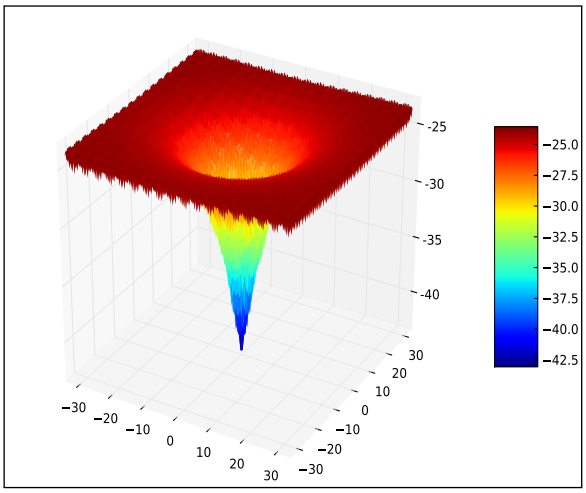

Figure A.7: Ackley Function

A.3.8 Six-Hump Camel Back Function

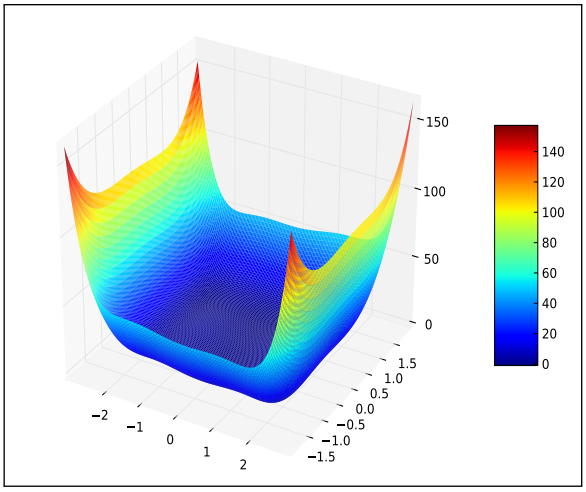

Figure A.8: Six-Hump Camel Back Function

A.3.9 Shubert Function

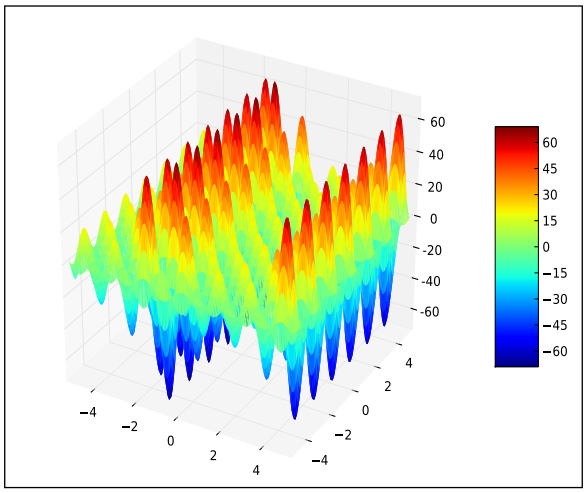

Figure A.9: Shubert Function

A.3.10 Himmelblau Function

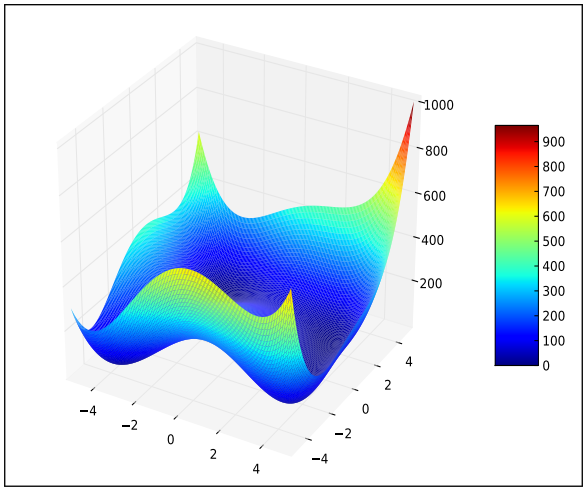

Figure A.10: Himmelblau Function

A.3.11 Rosenbrock Valley Function

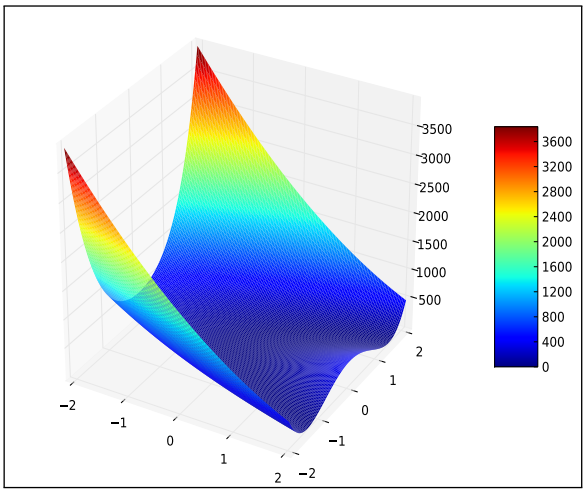

Figure A.11: Rosenbrock Valley Function

A.3.12 Dropwave Function

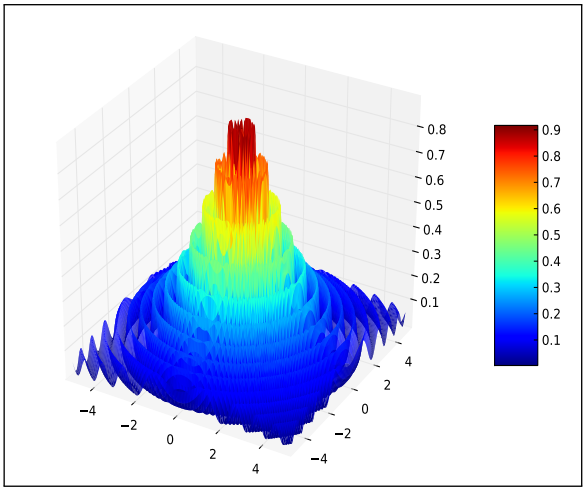

Figure A.12: Dropwave Function

A.3.13 Easom Function

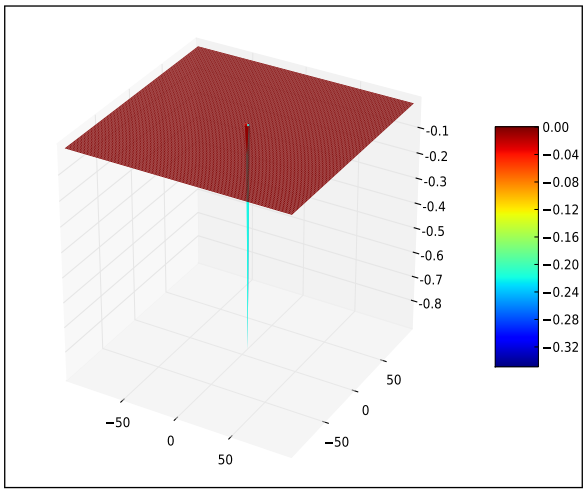

Figure A.13: Easom Function

A.3.14 Branin Function

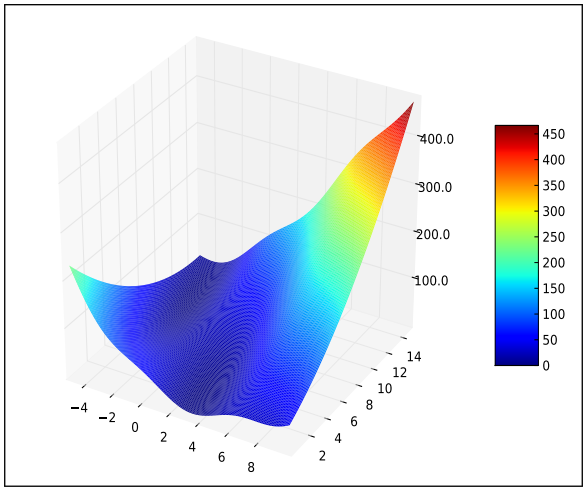

Figure A.14: Branin Function

A.3.15 Michalewicz Function

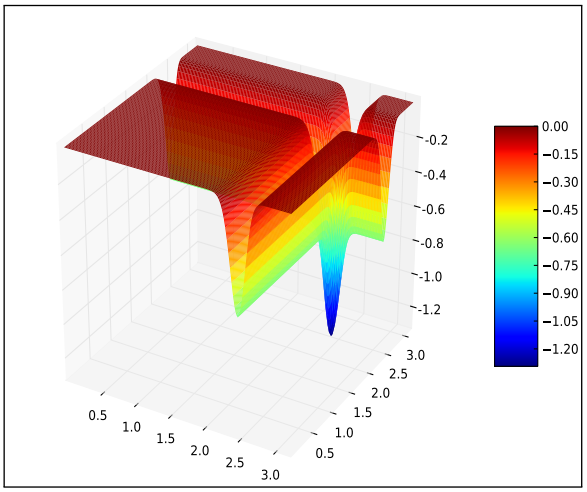

Figure A.15: Michalewicz Function

A.3.16 Goldstein Function

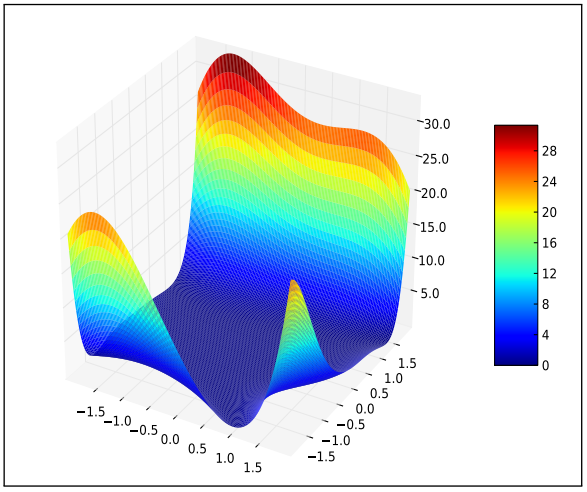

Figure A.16: THe Goldstein Function

Bibliography

[1] 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] Bilal Alatas. Chaotic bee colony algorithms for global numerical optimization. *Expert Syst. Appl.*, 37:5682–5687, August 2010.

[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] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353 – 373, 2005.

[11] 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.

[12] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin. Frequency assignment in cellular phone networks. *Annals of Operations Research*, (76):73–93, 1998.

[13] Ralf Borndörfer, Andreas Eisenblätter, Martin Grötschel, and Alexander Martin. The orientation model for frequency assignment problems. Technical report, Konrad-Zuse-Zentrum Berlin, 1998.

[14] 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.

[15] 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.

[16] 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.

[17] Rachid Chelouah and Patrick Siarry. Tabu search applied to global optimization. *European Journal of Operational Research*, 123:256–270, 2000.

[18] 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.

[19] 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.

[20] 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.

[21] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243 – 278, 2005.

[22] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243 – 278, 2005.

[23] Marco Dorigo, Eric Bonabeau, and Guy Theraulaz. Ant algorithms and stigmergy. *Future Gener. Comput. Syst.*, 16(9):851–871, 2000.

[24] 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.

[25] 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.

[26] A. Eisenblätter, M. Grötschel, and A. Martin. Frequency planning and rammifications of colouring. Technical report, Konrad-Zuse-Zentrum fu r Informationstechnik Berlin, 2000.

[27] Andreas Eisenblätter. Assigning frequencies in gsm networks. Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), 2001.

[28] Andreas Eisenblätter. *Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2001.

[29] 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.

[30] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006.

[31] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, 2007.

[32] Dr. Kamilo Feher. *Wireless Digital Communications: Modulation & Spread Sprectrum Applications*. Prentice Hall, 1995.

[33] 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.

[34] 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.

[35] Gautam Garai and B. B. Chaudhuri. A distributed hierarchical genetic algorithm for efficient optimization and pattern matching. *Pattern Recogn.*, 40(1):212–228, 2007.

[36] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] Julia Handl and Bernd Meyer. Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113, December 2007.

[43] 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.

[44] 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.

[45] http://fap.zib.de/. Fap web, 2010.

[46] Shun-Fa Hwang and Rong-Song He. Improving real-parameter genetic algorithm with simulated annealing for engineering problems. *Adv. Eng. Softw.*, 37(6):406–418, 2006.

[47] 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.

[48] C. Bettstetter J. Eberspächer, H.-J. Vögel and C. Hartmann. *GSM - Architecture, Protocols and Services*. Wiley Publishing, third edition, 2009.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Appl. Soft Comput.*, 8:687–697, January 2008.

[55] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108 – 132, 2009.

[56] 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.

[57] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *J. of Global Optimization*, 39:459–471, November 2007.

[58] Akbar Karimi, Hadi Nobahari, and Patrick Siarry. Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions. *Comput. Optim. Appl.*, 45:639–661, April 2010.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] Hongbo Liu, Ajith Abraham, and Maurice Clerc. Chaotic dynamic characteristics in swarm intelligence. *Applied Soft Computing*, 7(3):1019 – 1026, 2007.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] Carlo Mannino and Gianpaolo Oriolo. Solving stability problems on a superclass of interval graphs. Technical report, Centro Vito Volterra, 2002.

[70] 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.

[71] Asha Mehrotra. *GSM System Engineering*. Arteh House, Inc, 1997.

[72] 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.

[73] 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.

[74] 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.

[75] Marie-Bernadette Pautet Michel Mouly. *The GSM System for Mobile Communications*. Cell & Sys, 1992.

[76] 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.

[77] M. Molga and C. Smutnicki. Test functions for optimization needs, 2005.

[78] 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.

[79] Roberto Montemanni. *Upper and Lower bounds for the fixed spectrum frequency assignment problem*. PhD thesis, School of Tecnology, University of Glamorgan, 2001.

[80] Roberto Montemanni and Derek H. Smith. Heuristic manipulation, tabu search and frequency assignment. *Comput. Oper. Res.*, 37(3):543–551, 2008.

[81] Angel E. Mu noz Zavala, Arturo Hernández Aguirre, and Enrique R. Villa Diharce. Constrained optimization via particle evolutionary swarm optimization algorithm (peso). In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 209–216, New York, NY, USA, 2005. ACM.

[82] Garry Mullet. *Wireless telecommunications systems and networks*. Thomsan Delmar Learning, 2006.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] 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.

[88] Michael O’Neill and Anthony Brabazon. Self-organising swarm (soswarm). *Soft Comput.*, 12(11):1073–1080, 2008.

[89] W. Paszkowicz. Properties of a genetic algorithm equipped with a dynamic penalty function. *Computational Materials Science*, 45(1):77 – 83, 2009. Selected papers from the E-MRS 2007 Fall Meeting Symposium G: Genetic Algorithms in Materials Science and Engineering - GAMS-2007.

[90] Thomas La Porta Patrick Traynor, Patrick McDaniel. *Security for Telecommunications Networks*, volume 40 of *Advances in Information Security*. Springer US, 2008.

[91] 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.

[92] Jean-Yves Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63:337–370, 1996.

[93] 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.

[94] 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.

[95] 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.

[96] Andrea Reese. Random number generators in genetic algorithms for unconstrained and constrained optimization. *Nonlinear Analysis: Theory, Methods & Applications*, 71(12):e679 – e692, 2009.

[97] D. J. Reid. Genetic algorithms in constrained optimization. *Mathematical and Computer Modelling*, 23(5):87 – 111, 1996.

[98] J. Riget and J.S. Vesterstrm. A diversity-guided particle swarm optimizer - the arpso. Technical report, Aarhus Universitet, 2002.

[99] 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.

[100] Stuart Russel and Peter Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall, second edition, 2003.

[101] 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.

[102] 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.

[103] H.G. Sandalidis, P.P. Stavroulakis, and J. Rodriguez-Tellez. An efficient evolutionary algorithm for channel resource management in cel-

lular mobile systems. *Evolutionary Computation, IEEE Transactions on*, 2(4):125 –137, nov 1998.

[104] Mischa Schwartz. *Mobile Wireless Communications*. Cambridge University Press, 2005.

[105] 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.

[106] Patrick Siarry, Alain Pétrowski, and Mourad Bessaou. A multipopulation genetic algorithm aimed at multimodal optimization. *Adv. Eng. Softw.*, 33(4):207–213, 2002.

[107] Alok Singh. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing*, 9(2):625 – 631, 2009.

[108] 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.

[109] 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.

[110] 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.

[111] Gordon L. Stuüber. *Principles of Mobile Communication*. Kluwer Academic Publishers, 1996.

[112] 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.

[113] 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.

[114] 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.

[115] 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.

[116] Y. S. Teh and G. P. Rangaiah. Tabu search for global optimization of continuous functions with application to phase equilibrium calculations. *Computers & Chemical Engineering*, 27(11):1665 – 1679, 2003.

[117] 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.

[118] 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.

[119] 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.

[120] 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.

[121] Nguyen Thanh Trung and Duong Tuan Anh. Comparing three improved variants of simulated annealing for optimizing dorm room assignments. In *Computing and Communication Technologies, 2009. RIVF '09. International Conference on*, pages 1 –5, 13-17 2009.

[122] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[123] 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.

[124] Roger L. Wainwright. A family of genetic algorithm packages on a workstation for solving combinatorial optimization problems. *SIGICE Bull.*, 19(3):30–36, 1994.

[125] 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.

[126] Z. G. Wang, Y. S. Wong, and M. Rahman. Development of a parallel optimization method based on genetic simulated annealing algorithm. *Parallel Comput.*, 31:839–857, August 2005.

[127] 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.

[128] 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.

[129] 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.

[130] Darrell Whitley, Soraya Rana, John Dzubera, and Keith E. Mathias. Evaluating evolutionary algorithms. *Artif. Intell.*, 85:245–276, August 1996.

[131] 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.

[132] Wayne L. Winston and Munirpallam Venkataramanan. *Introduction to Mathematical Programming.* Thomson Learning, 2003.

[133] Lihua Wu and Yuyun Wang. An introduction to simulated annealing algorithms for the computation of economic equilibrium. *Computational Economics*, 12:151–169, 1998.

[134] 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.

[135] 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.

[136] Xin-She Yang. *Nature-Inspired Metaheuristic Algorithms.* Luniver Press, 2008.

[137] 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.

[138] Quan Yuan, Feng Qian, and Wenli Du. A hybrid genetic algorithm with the baldwin effect. *Information Sciences*, 180(5):640 – 652, 2010.

[139] 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.