Decision Support for Generator Maintenance Scheduling in the Energy Sector

Evert Barend Schlünz

Thesis presented in partial fulfilment of the requirements for the degree of
Master of Science (Operations Research)
in the Department of Logistics at Stellenbosch University

Supervisor: Prof JH van Vuuren

December 2011

Declaration

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

December 2011

Abstract

As the world-wide consumption of electricity continually increases, more and more pressure is put on the capabilities of power generating systems to maintain their levels of power provision. The electricity utility companies operating these power systems are faced with numerous challenges with respect to ensuring reliable electricity supply at cost-effective rates. One of these challenges concerns the planned preventative maintenance of a utility's power generating units.

The *generator maintenance scheduling* (GMS) problem refers to the problem of finding a schedule for the planned maintenance outages of generating units in a power system (*i.e.* determining a list of dates corresponding to the times when every unit is to be shut down so as to undergo maintenance). This is typically a large combinatorial optimisation problem, subjected to a number of power system constraints, and is usually difficult to solve.

A mixed-integer programming model is presented for the GMS problem, incorporating constraints on maintenance windows, the meeting of load demand together with a safety margin, the availability of maintenance crew and general exclusion constraints. The GMS problem is modelled by adopting a reliability optimality criterion, the goal of which is to level the reserve capacity. Three objective functions are presented which may achieve this reliability goal; these objective functions are respectively quadratic, nonlinear and linear in nature.

Three GMS benchmark test systems (of which one is newly created) are modelled accordingly, but prove to be too time consuming to solve exactly by means of an off-the-shelf software package. Therefore, a metaheuristic solution approach (a *simulated annealing* (SA) algorithm) is used to solve the GMS problem approximately. A new ejection chain neighbourhood move operator in the context of GMS is introduced into the SA algorithm, along with a local search heuristic addition to the algorithm, which results in hybridisations of the SA algorithm.

Extensive experiments are performed on different cooling schedules within the SA algorithm, on the classical and ejection chain neighbourhood move operators, and on the modifications to the SA algorithm by the introduction of the local search heuristic. Conclusions are drawn with respect to the effectiveness of each variation on the SA algorithm. The best solutions obtained during the experiments for each benchmark test case are reported. It is found that the SA algorithm, with ejection chain neighbourhood move operator and a local search heuristic hybridisation, achieves very good solutions to all instances of the GMS problem.

The hybridised simulated annealing algorithm is implemented in a computerised *decision support system* (DSS), which is capable of solving any GMS problem instance conforming to the general formulation described above. The DSS is found to determine good maintenance schedules when utilised to solve a realistic case study within the context of the South African power system. A best schedule attaining an objective function value within 6% of a theoretical lowerbound, is thus produced.

Uittreksel

Met die wêreldwye elektrisiteitsverbruik wat voortdurend aan die toeneem is, word daar al hoe meer druk geplaas op die vermoë van kragstelsels om aan kragvoorsieningsaanvraag te voldoen. Nutsmaatskappye wat elektrisiteit opwek, word deur talle uitdagings met betrekking tot betroubare elektrisiteitsverskaffing teen koste-effektiewe tariewe in die gesig gestaar. Een van hierdie uitdagings het te make met die beplande, voorkomende instandhouding van 'n nutsmaatskappy se kragopwekkingseenhede.

Die *generator-instandhoudingskeduleringsprobleem* (GISP) verwys na die probleem waarin 'n skedule vir die beplande instandhouding van kragopwekkingseenhede binne 'n kragstelsel gevind moet word ('n lys van datums moet tipies gevind word wat ooreenstem met die tye wanneer elke kragopwekkingseenheid afgeskakel moet word om instandhoudingswerk te ondergaan). Hierdie probleem is tipies 'n groot kombinatoriese optimeringsprobleem, onderworpe aan 'n aantal beperkings van die kragstelsel, en is gewoonlik moeilik om op te los.

'n Gemengde, heeltallige programmeringsmodel vir die GISP word geformuleer. Die beperkings waaruit die formulering bestaan, sluit in: venstertydperke vir instandhouding, bevrediging van die vraag na elektrisiteit tesame met 'n veiligheidsgrens, die beskikbaarheid van instandhoudingspersoneel en algemene uitsluitingsbeperkings. Die GISP-model neem as optimaliteitskriterium betroubaarheid en het ten doel om die reserwekrag wat gedurende elke tydperk beskikbaar is, gelyk te maak. Drie doelfunksies word gebruik om laasgenoemde doel te bereik (naamlik doelfunksies wat onderskeidelik kwadraties, nie-lineêr en lineêr van aard is).

Drie GISP-maatstaftoetsstelsels (waarvan een nuut geskep is) is dienooreenkomstig gemodelleer, maar dit blyk uit die oplossingstye dat daar onprakties lank gewag sal moet word om eksakte oplossings deur middel van kommersiële programmatuur vir hierdie stelsels te kry. Gevolglik word 'n metaheuristiese oplossingsbenadering ('n *gesimuleerde temperingsalgoritme* (GTA)) gevolg om die GISP benaderd op te los. 'n Nuwe uitwerpingsketting-skuifoperator word in die konteks van GISP in die GTA gebruik. Verder word 'n lokale soekheuristiek met die GTA vermeng om 'n basteralgoritme te vorm.

Uitgebreide eksperimente word uitgevoer op verskeie afkoelskedules binne die GTA, op die klassieke en uitwerpingsketting-skuifoperators en op die verbasterings van die GTA meegebring deur die lokale soekheuristiek. Gevolgtrekkings word oor elke variasie van die GTA se effektiwiteit gemaak. Die beste oplossings vir elke toetsstelsel wat gedurende die eksperimente verkry is, word gerapporteer. Daar word bevind dat die GTA met uitwerpingsketting-skuifoperator en lokale soekheuristiek-verbastering baie goeie oplossings vir die GISP lewer.

Die verbasterde GTA word in 'n gerekenariseerde *besluitsteunstelsel* (BSS) geïmplementeer wat 'n gebruiker in staat stel om enige GISP van die vorm soos in die wiskundige programmeringsmodel hierbo beskryf, op te los. Daar word bevind dat die BSS goeie skedules lewer wanneer dit gebruik word om 'n realistiese gevallestudie binne die konteks van die Suid-Afrikaanse kragstelsel, op te los. 'n Beste skedule met 'n doelfunksiewaarde wat binne 6% vanaf 'n teoretiese ondergrens is, word ondermeer bepaal.

Acknowledgements

The author hereby wishes to express his deepest gratitude towards those who played a significant role during the progress of work towards this thesis:

- My supervisor, Prof Jan van Vuuren, for his guidance and support throughout the duration of this project. I appreciate his time, dedication and hard work in ensuring that work of a high standard is delivered.
- The Department of Logistics for the use of their excellent computing facilities and office space.
- The South African Nuclear Energy Corporation (NECSA) for their financial support over the past two years.
- All of my GOReLAB office colleagues over the past two years for their support and technical assistance, for the new friendships that have formed and for a number of wonderful experiences that I could share with them.
- Finally, my friends and family for their moral support and encouragement during the past two years, and their understanding during times (especially the last two months) of great pressure and unavailability on my part.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AC	Alternating current
ACO	Ant colony optimisation
ACS	Ant colony system
AIM	Average increase method
B&B	Branch-and-bound
BSS	Besluitsteunstelsel
CSP	Constraint satisfaction problem
DC	Direct current
DP	Dynamic programming
DPSA	Dynamic programming with successive approximations
DSS	Decision support system
ED	Economic dispatch
ES	Expert system
GA	Genetic algorithm
GISP	Generator-instandhoudingskeduleringsprobleem
GMS	Generator maintenance scheduling
GTA	Gesimuleerde temperingsalgoritme
GUI	Graphical user interface
IEEE	Institute of Electrical and Electronics Engineers
IP	Integer program
LP	Linear program
MILP	Mixed-integer linear program
MINP	Mixed-integer nonlinear program
MIQP	Mixed-integer quadratic program
MW	Megawatt
NLP	Nonlinear program
RTS	Reliability Test System
SA	Simulated annealing
SDM	Standard deviation method
SLP	Successive linear programming
TS	Tabu search
TSP	Travelling salesman problem
UC	Unit commitment

List of Reserved Symbols

The symbols listed below are reserved for a specific use, unless specified otherwise in a localised section where its meaning is apparent. Other symbols may be used throughout the thesis in an unreserved fashion.

Symbols in this thesis conform to the following font conventions:		
$\mathcal{A}$	Symbol denoting a **set**	(Calligraphic capitals)
$\boldsymbol{a}, \boldsymbol{A}$	Symbol denoting a **vector**	(Boldface lower case letters or capitals)

Symbol	Meaning
Indices	
i	The index of generating units.
j	The index of time periods.
k	The index of generating unit subsets.
Sets	
$\mathcal{I}$	The set of indices of generating units.
$\mathcal{J}$	The set of indices of time periods.
$\mathcal{K}$	The set of indices of generating unit subsets.
$\mathcal{I}_k$	The subset of indices of generating unit subset k.
Variables	
$x_{i,j}$	A binary variable taking the value 1 if maintenance of generating unit i commences at time period j, or zero otherwise.
$y_{i,j}$	A binary variable taking the value 1 if generating unit i is in maintenance at time period j, or zero otherwise.
r_j	The unused power during time period j, excluding the safety margin capacity.
$\overline{r}$	The mean reserve load.
o_j	A slack variable defined as the overachievement of the actual reserve from the mean reserve level during time period j.
u_j	A slack variable defined as the underachievement of the actual reserve from the mean reserve level during time period j.
P_w^i	The maintenance window constraint violation penalty term for unit i.
P_w	The overall maintenance window penalty.

P_ℓ^j	The load and reliability constraint violation penalty term during time period j.
P_ℓ	The overall load and reliability penalty term.
P_c^j	The maintenance crew constraint violation penalty term during time period j.
P_c	The overall maintenance crew penalty term.
$P_e^{k,j}$	The exclusion constraint violation penalty term for generating unit subset k during time period j.
P_e	The overall exclusion penalty term.
Parameters	
n	The number of generating units.
m	The number of time periods in the planning horizon.
e_i	The earliest time period during which maintenance of generating unit i may start.
ℓ_i	The latest time period during which maintenance of generating unit i may start.
W_{ext}	The number of time periods by which the earliest and latest starting times for each unit are extended in a soft constraint approach.
d_i	The maintenance duration for generating unit i.
$g_{i,j}$	The power generating capacity of unit i during time period j.
$g'_{p,i,j}$	The power generating capacity lost during time period j if maintenance of unit i commenced at time period p.
D_j	The demand at time period j.
S	The safety margin as a proportion of the demand for the power system.
$m_{i,j}$	The manpower required by unit i when undergoing maintenance during time period j.
m_i^k	The manpower required by unit i during its k-th period of maintenance.
$m'_{p,i,j}$	The manpower required by unit i when undergoing maintenance during time period j if maintenance commenced at time period p.
M_j	The maximum available manpower during time period j.
K	The number of generating unit subsets.
K_k	The maximum number of units within generating unit subset k that are allowed to be in simultaneous maintenance during any time period.
w_w^i	The maintenance window violation penalty weight for unit i.
w_ℓ	The load and reliability penalty weight.
w_c	The maintenance crew penalty weight.
w_e	The exclusion penalty weight.

CHAPTER 1

Introduction

Contents

The first written account of an electrical effect — the shocks from electric fish — is found in ancient Egyptian texts, dating from 2750 BC. In these texts, the fish are referred to as the "Thunderers of the Nile" [90]. However, knowledge of electricity only developed in later millenia. It was known by ancient cultures around the Mediterranean that certain objects, such as rods of amber, could attract light objects like feathers after they had been rubbed in cat's fur [90]. Only in 1600 AD did William Gilbert coin the New Latin word *electricus*, meaning "of amber" (from the word *ήλεκτρον* [*elektron*], Greek for "amber"), to refer to amber's attractive properties [92]. The introduction of the word *electric* into the English language was used to describe materials like amber that attracted other objects. This led to the first use of the English word *electricity* in 1646, which at that stage, referred to the property of behaving like an electric [92]. The term "electricity" has changed in definition since then, due to non-scientific usage by electric utility companies and the general public. According to the Oxford Dictionaries Online, electricity today refers to a form of energy resulting from the existence of charged particles (such as electrons or protons), either statically as an accumulation of charge or dynamically as a current [69].

1.1 Background

The production of electricity in early years was an expensive and inefficient process, since electricity could only be produced by means of the chemical reactions in electrochemical cells[1] or electrostatic generators[2]. However, in 1831, Michael Faraday created a machine capable of

[1] A Galvanic cell, or Voltaic cell, is an electrochemical cell which converts chemical energy into electrical energy through spontaneous chemical reactions taking place at the electrodes of the cell [91]. The first electrical battery was the voltaic pile — a set of individual Galvanic cells placed in series, invented by Alessandro Volta in 1800 [100].

[2] An electrostatic generator operates by using moving electrically charged belts, plates and disks to carry charge to a high potential electrode [88]. Such a generator typically generates very high voltages and low currents. The Van de Graaff generator is an example of such a machine.

generating electricity by means of a rotary motion [96]. The machine (a generator) converts mechanical energy into electrical energy by using electromagnetic principles. A number of years later, the technology became commercially viable.

The first power station in the world became operational in 1878 and was built in the Bavarian town of Ettal. The station consisted of 24 dynamo electrical generators, driven by a steam engine and its purpose was to provide electricity for illuminating the "Venus Grotto" in changing colours, located in the gardens of Linderhof Palace[3] [96]. The Venus Grotto is shown in Figure 1.1 with its different colours.

Figure 1.1: *The Venus Grotto within the gardens of Linderhof Palace [95].*

In September 1881, the world's first public electricity supply was established in the town Godalming in England. This electrical system was powered by a water wheel on the river Wey, driving a Siemens alternator and it provided electricity to light up a number of lamps within the town [89]. However, it was Thomas Edison (see Figure 1.2(a)) who opened the world's first public power station in London, January 1882 [96]. A 27 ton generator, called Jumbo, was driven by a steam-powered engine in the power station and the electricity supply was *direct current*[4] (DC). Later in the same year, Edison opened a power station in New York to provide the lower Manhattan Island area with electrical lighting, and again, the electricity supply was DC.

Although DC power supply had a number of advantages in the early years of electricity distribution, it also had flaws — the greatest being its distribution capability. Using a higher voltage reduces the current, resulting in less power loss caused by resistance in the transmission cables. Edison did not have any means of voltage conversion for his DC power supply and the result was that the electricity generation had to occur close to the consumer [89].

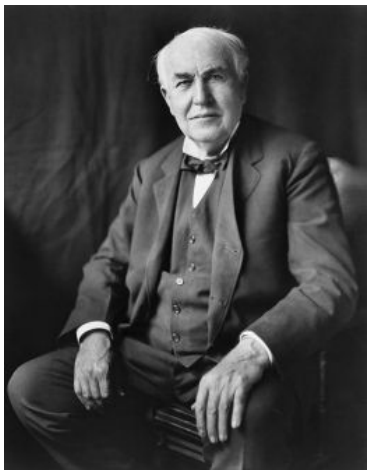
The other form of electricity supply, is *alternating current*[5] (AC). A former employee of Edison, named Nikola Tesla (see Figure 1.2(b)), devised an electrical system using AC, which remains the primary means of electricity distribution throughout the world today [89]. The AC system allows for the transformation between voltage levels in different parts of the system, thereby allowing efficient distribution of electricity over long distances by means of high-voltage AC current.

When Edison's DC system was introduced, there was no practical AC electrical motor available and his DC distribution became the standard for the United State of America [101]. By 1887, there were 121 power stations in the United States of America using Edison's DC system.

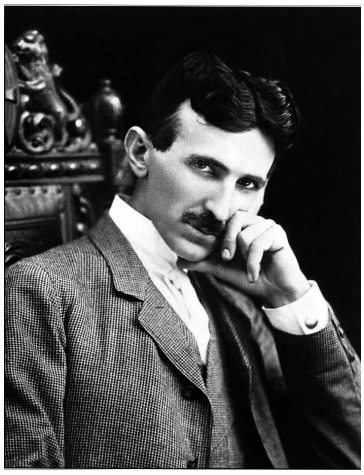
[3] Linderhof Palace is the smallest of three palaces built by King Ludwig II of Bavaria [95].
[4] Direct current is the undirected flow of electric charge.
[5] In alternating current, the flow of electric charge periodically reverses in direction.

However, firm believers in AC technology started emerging at this time. George Westinghouse invested in the technology [101] and he partnered with Tesla in 1888, commercialising Tesla's AC system, which included a practical AC motor. This led to the so-called *War of Currents* over electrical power distribution, with Edison's DC on the one side and Tesla's AC on the other.

The *War of Currents* involved companies in Europe and the United States of America which had invested large amounts of resources into AC or DC power supply. The most notable business rivalries developed between Westinghouse Electric, Siemens and Oerlikon (favouring AC), and the mighty Edison General Electric (favouring DC). However, the war is often personified by the personal rivalry which developed between Tesla and Edison [101]. This rivalry originated from events that occurred while Tesla was an employee of Edison. During the *War of Currents*, Edison carried out a publicity campaign, primarily focused on the notion that AC systems were more dangerous than his DC systems, so as to discourage the use of AC [101]. Included in this campaign, was the public AC-driven killings of animals. Edison even became involved in the development and promotion of the electric chair (AC-driven) for capital punishment in order to promote the idea that AC had greater lethal potential than DC [101].



(a) Thomas Edison (c. 1932)



(b) Nikola Tesla (c. 1896)

Figure 1.2: *Thomas Edison [98] and Nikola Tesla [101].*

Ultimately, the *War of Currents* resolved in favour of AC and the end of the war was marked by the International Electro-Technical Exhibition in 1891, held in Frankfurt. The first long distance transmission of high-power, three-phase alternating electric current was featured at the exhibition, being generated by a power station in Lauffen am Neckar, 175 kilometers away [101]. Corporate technical representatives (including from Thomson-Houston Electric Company) were thoroughly impressed by this demonstration. The following year, General Electric was formed by the merger of Edison General Electric and Thomson-Houston Electric Company, and it immediately invested in AC. Thomas Edison could no longer influence the company direction, as the General Electric president, Charles Coffin, and the board of directors muted his opinions [101]. In 1893, Almarian Decker designed a new three-phase generator and system (based on Tesla's experimental work) which was used for the Mill Creek No. 1 Hydro-electric Plant in California [85]. It was the first commercial power plant in the United States of America using three-phase alternating current. The design of Decker's three-phase system established the standards for the complete system of generation, transmission and motors used today [86].

Power stations became increasingly larger during the early 20th century and relied on interconnections of a number of stations to improve the reliability and cost of electricity generation. The steam turbine arrived on the power generation scene around 1906 and it allowed for significant expansion of generating capacity in power stations [96]. Over the years, new and improved power generation technologies appeared, increasing the efficiency of electricity generation and the number of generation methods and fuel sources. Having originally only used water power and coal, power stations today rely on a variety of different fossil fuels, nuclear fission, biomass, geothermal power, water power, wind power or solar power.

In order to meet the rising demand for electricity, the generating capacities of power stations have increased considerably over the years, where technology and fuel sources have allowed such expansion. Seven of the ten largest power stations in the world today are hydro-electric stations. At present, the largest power station in the world is the hydro-electric Three Gorges Dam power station in China (see Figure 1.3) with a capacity of 18 460 MW [7]. The Kashiwazaki-Kariwa Nuclear Power Plant in Japan, at number five on the list of largest power stations, is the largest non-renewable power station in the world with a capacity of 8 206 MW. The future may herald in an even larger power station than the Three Gorges Dam for the world. The proposed Grand Inga Dam on the Congo River in the Democratic Republic of Congo has been earmarked for hydro-electric power generation [93]. This dam has an expected generating capacity of 39 000 MW, more than double the capacity of the currently largest power station. To put this figure into perspective — the total electricity consumption of the African continent in 2007 was estimated at 58 090 MW [103]. Therefore, the Grand Inga Dam, with its 39 000 MW capacity, would have provided 67% of the African continent's power demand in 2007.

(a) **(b)**

Figure 1.3: *The Three Gorges Dam in China [12, 80].*

1.2 Informal problem description

In South Africa, the state-owned electricity utility *Eskom* generates approximately 95% of the country's electricity. Eskom generates, transmits and distributes electricity to customers in all sectors of society. The company was established by the government in 1923 as the *Electric Supply Commission* (ESCOM) and was responsible for establishing and maintaining electricity supply undertakings on a regional basis [44]. Over the years, the company has grown into one of the largest electricity utilities in the world in terms of generating capacity and sales today.

However, in recent years, Eskom has faced challenges with respect to sufficient power generation for South Africa. In 2006 and 2007, power outages arose due to higher than expected electricity

demand, unplanned generating unit outages, and a diminished reserve capacity [45]. The reserve margin for generating capacity had decreased from the desired 15% down to less than 8%. Then President Thabo Mbeki publicly apologised in 2007 for the government not heeding Eskom's timeous recommendation to build new power stations to match the country's growth rate. Eskom warned that power interruptions were highly likely over the following five years as new base-load stations were expected to come online only from 2012 onwards.

Emergency load shedding was implemented between October 2007 and February 2008 in order to avoid a potential nationwide blackout. A national electricity emergency was declared on 24 January 2008 [45]. According to Eskom's annual report of 2008 [27], load shedding may be attributed mainly to the low reserve margin of the power system. This low margin meant that the system could not adequately deal with the external events affecting its efficiency at that time. These events comprised increased unplanned outages[6], coal stock piles reaching unacceptably low levels and unusually high rainfall, causing moisture levels in the coal so as to severely hamper efficient power generation.

Since the events of 2008, South Africa has not experienced any load shedding, mainly due to the recovery plan implemented by Eskom. It comprised three phases — the first two phases being short-term solutions which ended in 2008. The current phase is a medium-term solution scheduled to last to 2012 when the first of the new base-load stations is expected to come online. Ultimately, the challenge remains to achieve and maintain a reserve margin of 15% [27].

In view of the South African electricity challenge described above, a key area of concern is the planned maintenance outages of generation plants. Since planned maintenance is a power system requirement, it is an unavoidable duty for an electricity utility to perform. The relatively old age and higher load factor of the South African power stations, significantly increase the need for plant maintenance, thereby reducing the opportunity (leverage) for planned maintenance. Combined with the diminished safety margin of the capacity, these two factors render the task of scheduling planned maintenance outages of power generating units a daunting endeavour at best. Furthermore, scheduling the planned maintenance outages in such a way that the system supply still satisfies the demand, is the simplest form of the problem — additional factors and constraints may also influence the scheduling process, such as limited maintenance resources.

The problem of finding a schedule for the planned maintenance outages of generating units in a power system is known as the *generator maintenance scheduling* (GMS) problem. The challenges currently faced by Eskom in South Africa may easily occur in other power systems across the world. As power systems become larger and demand for electricity increases continually, so does the difficulty in finding maintenance schedules increase in complexity, especially in systems with small reserve margins and/or high levels of constriction.

1.3 Scope and objectives

The following objectives are persued in this thesis in order to lend decision support to a power system operations planner tasked with generator maintenance scheduling:

- **Objective I:**
  To *provide* a list of essential general modelling considerations for the GMS problem in order to be able to formulate a suitable model for the problem.

[6]South Africa's system of power stations is relatively old and requires above-average levels of maintenance. In view of South Africa's system conditions, the stations also run continuously at very high load factors (how hard a plant is being run on a percentage basis). These two aspects contribute significantly to unplanned outages.

- **Objective II:**
  To *perform* a literature survey of previous formulations of GMS models and their extensions.
- **Objective III:**
  To *perform* a literature survey of typical solution techniques previously applied with a view to solving the GMS problem.
- **Objective IV:**
  To *develop* a suitable model for the GMS problem and *present* and *motivate* its mathematical problem formulation.
- **Objective V:**
  To *propose* an approximate solution approach towards solving the GMS problem and to *compare* its effectiveness in solving benchmark GMS test problems of the form described in Objective IV to that of exact solution approaches currently commercially available.
- **Objective VI:**
  To *propose* a new neighbourhood move operator in the context of the GMS problem and to *investigate* its effectiveness, as well as the effectiveness of classical variations and *proposed* modifications to the solution algorithm.
- **Objective VII:**
  To *design* a generic decision support system capable of solving a GMS problem instance of the form in Objective IV approximately, based on the results of persuing Objectives V and VI.
- **Objective VIII:**
  To *implement* the decision support system of Objective VII on a personal computer and to use it to *solve* approximately a GMS case study in the South African power system.

The scope of this thesis shall be restricted to the GMS problem, contained within the broader area of power system operations scheduling, and shall exclude the remaining scheduling problems of transmission line maintenance, unit commitment and economic dispatch.

Deterministic power system values shall be presented in any results, but the reliability key performance indicators of Eskom shall not be used.

The decision support system shall be designed for any electricity utility with a power generating system of the form described in Objective IV.

Finally, the decision support system shall be implemented to assist a power system operations planner tasked with generator maintenance scheduling, by suggesting maintenance schedules; the idea is not to replace him/her.

1.4 Thesis organisation

This thesis comprises seven further chapters and a number of appendices, following this introductory chapter. In Chapter 2, the reader is introduced to the different modelling considerations that have to be addressed in order to derive a model formulation for the GMS problem. A comprehensive literature review on GMS model formulations is presented, containing the mathematical programming formulations of individual constraint sets and objective functions.

Formulations other than that of a mathematical program are briefly presented, along with model extensions to the problem — moving beyond the consideration of only generator maintenance in the problem. The final section of the chapter contains a concise, but thorough literature review on the solution techniques that are typically applied to the GMS problem. Both exact and approximate solution approaches of varying complexity — from simple heuristics to more complicated mathematical programming techniques, and from artificial intelligence techniques to fuzzy logic and expert systems — are presented.

Chapter 3 contains the derivation of the GMS model adopted for further use in this thesis. In the first section, the GMS problem is placed in its proper context within the broader area of power system operations scheduling. The next section contains a motivation of the necessary assumptions in order to reduce the GMS problem into a managable power system operations scheduling subproblem for use in this thesis. Two GMS models — a simple model, and a more advanced model — are presented in the concluding sections of the chapter, each with a choice of three objective functions. The difference between the two models involves the constraint sets that are included. Furthermore, a significant change in the mathematical programming formulation occurs from the one model to the next, namely the introduction of a second set of dependent variables, thereby increasing the problem dimensions considerably. A total of six formal mathematical programming formulations are therefore presented for the GMS problem adopted in this thesis.

The exact and the approximate solution approaches to be considered in this thesis are presented in Chapter 4. The first section contains descriptions of the algorithms typically employed by an off-the-shelf software package to solve the GMS problem exactly. Following the section, the full approximate solution approach adopted in this thesis towards solving the GMS problem, is explained. This approach comprises a random search heuristic implementation, mainly used for comparative purposes, and a simulated annealing algorithmic implementation — the primary solution technique employed in this thesis. Additionally, the details of a new neighbourhood move operator in the context of GMS are presented, along with other functions containing modifications to the standard simulated annealing approach, including different cooling schedules. Every noteworthy function and algorithm described in the chapter, is presented additionally as a pseudo-code listing for ease of grasp.

Three GMS benchmark test systems are introduced in Chapter 5, two of which have been previously studied in the literature. The third test system is newly established by the author. Within the approximate solution approach considered in this thesis, a soft constraint approach is adopted, necessitating the use of a penalty weight associated with each constraint violation. The second section of this chapter contains a description of the methodology for determining these weight values and the subsequent calculation thereof for each test system. Typically, the application of the two approximate solution techniques introduced in the previous chapter contain parameters whose values are problem instance-specific. A detailed parameter optimisation procedure for each test system is presented in the final section of the chapter, along with the optimised parameter values for each of the three benchmark test systems and each solution technique variation.

In the sixth chapter, the performance of each solution technique variation is analised. These variations comprise the different cooling schedules within the simulated annealing algorithm, a new ejection chain neighbourhood move operator and a number of proposed modifications to the simulated annealing algorithm by means of the introduction of a local search heuristic and good initial solutions. In the section following the performance analyses of the various solution technique variations, the results of the exact solution approach to the benchmark test systems

are presented, as well as the best solutions obtained for each benchmark test system during the course of the work towards this thesis — in each case obtained via the approximate solution approach.

In Chapter 7, a computerised *decision support system* (DSS) for solving GMS problem instances, in any power system having the form described in Chapter 3, is presented. The general framework of the DSS is described in the first section, as well as how the difficulties in creating a generic solution scheme for GMS problems were overcome, in order to create the DSS. The implementation, appearance and working of the DSS are described in the second section via the use of screenshots and bulleted lists containing procedural steps to be followed by the user. Finally, the chapter is concluded with an application of the DSS to a realistic GMS scenario case study within the context of the South African national power generating system.

The thesis closes in Chapter 8 with a summary of the work contained therein, the contributions that were made in the thesis, as well as suggestions for future work in the field of generator maintenance scheduling.

CHAPTER 2

Literature Review

Contents

This chapter introduces the reader to the different modelling considerations that are required to formulate the *generator maintenance scheduling* (GMS) problem. A literature review is presented to illustrate how these different considerations are typically modelled and formulated in the literature. This is followed by a description of the popular solution techniques that may be employed to solve the GMS problem.

2.1 General model considerations

The GMS problem is formulated in [46] as the time sequence of preventive maintenance outages for a given set of generating units in a power system over a planning period, so that all constraints are satisfied and the objective function obtains an extreme value. There are four general

considerations which have to be addressed before formulating a model for the GMS problem, namely *the time sequence*, *the planning period*, *various constraints* and *the objective function*.

2.1.1 The planning period

The simplest attribute is the *planning period*. It depends mainly on the electric power utility how far it wants to plan into the future. Generally, the planning period or planning horizon is taken as one year [11, 21, 47] since power generating units are usually serviced annually [49]. Since load demand is typically calculated in year-format so as to include all the seasons, this could also bias the planning horizon towards one year. However, there is nothing that prohibits one to choose any length of time as the planning period. Mathematically, it has no effect other than increasing or decreasing the dimensionality of the problem. In the literature, the planning horizon varies from eight weeks for a small test problem [23] to five years for a captive power plant case study [62]. It is also possible to wrap the maintenance around to the following planning horizon [9] which endows the formulation an air of continual maintenance or periodicity.

The length of a time unit within the planning horizon, likewise, has no mathematical effect other than influencing the dimensionality of the model formulation. What is more relevant, however, is the availability of data corresponding to the time unit (hourly, daily, weekly, monthly, *etc.*) and practical implications such as the minimum maintenance time (one day, five days, *etc.*). Weekly time units are most commonly used in practice and in the literature [46]. Examples of other time units are single-day [32], five-day [36] and monthly time units [2]. Typically, shorter time units allow for greater flexibility in schedules but at the cost of increased dimensionality.

2.1.2 The time sequence

The *time sequence* indicates when a generating unit is in service (*i.e.* producing electricity) or out of service for maintenance. Most GMS models have decision variables corresponding to the starting time of maintenance for each unit [46]. Two representations can be used, namely a binary variable $x_{i,j}$ taking the value 1 if maintenance of generating unit i commences at time period j (and zero otherwise) or an integer variable x_i taking the value j if maintenance of generating unit i commences at time period j. Auxiliary variables may be introduced in the form of a binary variable $y_{i,j}$ taking the value 1 if generating unit i is in maintenance at time period j (and zero otherwise). This formulation with auxiliary variables increases the problem dimensionality in the sense that it necessitates the inclusion of more constraints which explicitly dictate that maintenance is not interrupted for the required duration [46].

2.1.3 Model constraints

The *constraints* of a GMS problem can vary in number depending on the complexity of the model, assumptions and the requirements of the utility. Probably the most basic form of the problem requires *maintenance window constraints* which ensure that each unit is scheduled for maintenance between an earliest and latest time period, along with so-called *load constraints* in order to ensure that the load demand is met for each time period. Depending on the choice of variables, it may be necessary to add constraints so as to ensure that maintenance is performed during consecutive time periods and only once during a maintenance window for each unit. The planning horizon may consist of multiple maintenance windows for each unit, but these windows are not allowed to overlap.

Reliability constraints may be added by including a reserve/safety margin in the load constraints. General *resource constraints* typically specify limits on the resources required for maintenance during each period. In a broader sense, these constraints limit the number of units that may be in maintenance simultaneously due to some "resource" being in scarce supply. Maintenance *crew constraints* consider the availability of manpower to perform the maintenance tasks at a given time and may also be considered as a resource constraint (the resource is manpower). If certain units are not allowed to be in a state of simultaneous maintenance (*e.g.* in the same power station or geographical region) then *exclusion constraints* may be added. Similarly, *precedence constraints* ensure that certain units are scheduled for maintenance before others.

The above-mentioned constraints are listed in [17, 46] and almost all GMS models contain a subset of these. A fairly recent addition to GMS constraint set are *transmission/network constraints* which receive attention in [2, 49, 51, 59, 60]. Lastly, specific constraints not mentioned above, arise whenever the given GMS environment is adapted to be more problem-specific. As an example, consider [64] where an equilibrium constraint is added for pumped-storage units.

2.1.4 The objective function

The *objective function* attribute represents an optimality criterion. Unfortunately, GMS naturally presents conflicting requirements which means that a trade-off solution must typically be found. Ultimately this makes the GMS problem multiobjective in nature. The dominant objective is usually chosen as the optimality criterion, with the lesser objectives being incorporated into the model as constraints [46]. However, alternative multiobjective approaches will be discussed later in this section.

The optimality criteria most often used may be grouped into three categories, namely *convenience criteria*, *economic criteria* and *reliability criteria* [17, 46] and may be employed in single or multiobjective settings.

Within the class of *convenience criteria*, the objective could be to minimise the degree of constraint violations or to minimise possible disruptions to the existing schedule. The author could not find any reference in the literature to single objective formulations using this criterion. However, the multiobjective formulations in [47, 48, 51] include it as an optimality criterion.

Under the heading of the *economic criteria* in GMS, the objective is commonly chosen as the minimisation of the operating cost which consists of the production cost and maintenance cost [17, 46]. The deregulation of the electric power market in many countries has perhaps shifted the focus away from operating cost and reliability more towards profitability. Competitive market environments typically cause an objective change to the maximisation of profit [34, 42, 106]. GMS formulations which consider economic criteria (operating cost of some composition) as a single objective formulation are wide-spread [9, 11, 19, 21, 25, 33, 49, 59, 65].

Reliability criteria may either be deterministic or stochastic [63]. Deterministic reliability objectives are commonly chosen as the levelling of the reserve load over all the time periods, which is generally achieved by minimising the sum of squares of the reserve. This approach is successfully used in the single objective formulations found in [14, 15, 17, 18, 32, 62, 63]. An alternative objective is to maximise the minimum reserve during any time period [68]. As stochastic reliability objective, the *effective load carrying capacity* (ELCC) for each unit and an *equivalent load* (EL) is used to level the risk of exceeding the available capacity [17], and it is applied as a single objective model formulation in [63]. Alternatively, the *loss of load probability/expectation* (LOLP/LOLE) is a commonly used reliability index. Adopting as objective the minimisation of the total LOLP for the planning horizon [17, 82] is effective for the single objective formulation.

In a multiobjective context, objectives from any of the above three criterion categories may be combined to form a multiobjective formulation for the GMS problem. Examples of combinations are: economic and reliability objectives in [36, 38, 64]; economic, reliability and convenience objectives in [47, 48], and economic and convenience objectives in [31, 51].

2.2 Model formulations in the literature

Due to the suitable nature of the GMS problem, it can easily be formulated as a mathematical program in which one or more objective functions are optimised, subject to certain constraints, as described in the previous section. Many scheduling problems share similarities with classical optimisation problems, such as the *assignment problem*[1], the *travelling salesman problem*[2] and the *vehicle routing problem*[3] (and all their variations). As a result, many scheduling problems have traditionally been formulated in terms of one of these optimisation problems. The GMS problem, however, does not follow these traditional scheduling problem formulations. The author could find no reference in the literature containing such a formulation for a GMS problem and, furthermore, no explanation why such formulations are not (or may not be) used.

A possible reason why the GMS problem is not formulated as one of the classical optimisation problems may be because the GMS problem differs from typical maintenance problems. The authors in [46] describe it as "*The peculiarity of maintenance scheduling of ... units ...*" and attribute it to the following properties of power systems: generated electricity is impossible to store, the transmission network is limited and the required amount of electricity must be generated at every instant, an adequate amount of reserve capacity has to be available, and the parallel nature of electricity supply within a power system (due to multiple generating units). Furthermore, the GMS problem can be highly variable in its composition — different objective functions may be employed, the formulation may be linear or nonlinear, different combinations of various constraints may be included, and adopting a deterministic or stochastic approach all lead to the conclusion that the GMS problem should be modelled in its own fashion. A last possible reason may be that a classical problem formulation adds unnecessary complexity to the GMS problem, whereas a direct mathematical programming formulation of the GMS constraints and objective function tends to be of a simpler form.

Since many different formulations for the GMS problem appear in the literature, depending on the scope of the model, the individual constraint and objective function formulations found in the literature are presented seperately in this section, in a similar fashion as in §2.1. These formulations are all presented in the context of a mathematical program.

[1]The classical assignment problem consists of assigning a number of *agents* to perform the same number of *tasks*. Any agent can perform any task at some cost, depending on the agent-task assignment. In order to solve the problem, an assignment of exactly one agent to each task has to be obtained such that the total cost of the assignment is minimised [87].

[2]Given a number of *cities* and the *distances* between these cities, the travelling salesman problem asks for a shortest (closed) tour in which each city is visited exactly once [99].

[3]The vehicle routing problem contains a set of *customers* to be serviced by a fleet of *vehicles*, which are initially located in one or more *depots*. The vehicles may move along a *road network* associated with travelling costs and the task is to find all the vehicles' servicing routes, starting and ending at their respective depots, such that the total travelling cost is minimised [78].

2.2.1 Constraint formulations in the literature

In order to present mathematical representations of the various GMS problem constraints, define $x_{i,j}$ as the binary decision variable taking the value 1 if maintenance of generating unit i commences at time period j, and zero otherwise. Alternatively, define x_i as the integer decision variable denoting the starting time period of the maintenance of generating unit i. Let $y_{i,j}$ be a binary variable taking the value 1 if generating unit i is in maintenance at time period j, and zero otherwise. If there are n generating units and m time periods in the planning horizon, let $\mathcal{I} = \{1, \ldots, n\}$ be the set of indices of generating units and let $\mathcal{J} = \{1, \ldots, m\}$ be the set of indices of time periods. Then $i \in \mathcal{I}$ and $j \in \mathcal{J}$ in the variables defined above.

The maintenance window constraint set ensures that the maintenance of a generating unit occurs in a pre-specified time-window. This is achieved by specifying that maintenance of unit i must start between an earliest time period (denoted by e_i) and latest period (denoted by ℓ_i), both inclusive. The binary variable constraint set

$$\sum_{j \in \mathcal{J}_i} x_{i,j} = 1, \qquad i \in \mathcal{I} \tag{2.1}$$

achieves this requirement, where $\mathcal{J}_i$ is the set of time periods during which the maintenance of unit i may start [15]. Therefore, $\mathcal{J}_i = \{j \in \mathcal{J} \mid e_i \leq j \leq \ell_i\}$. One may also explicitly specify that the variables should be zero when maintenance is not allowed [2, 3], that is requiring that

$$x_{i,j} = 0, \qquad i \in \mathcal{I},\ j \notin \mathcal{J}_i, \tag{2.2}$$

$$y_{i,j} = 0, \qquad i \in \mathcal{I},\ j < e_i \text{ or } j > \ell_i + d_i - 1. \tag{2.3}$$

If the integer variables are used, the constraint set simply bounds the variables [2, 48, 83] by requiring that

$$e_i \leq x_i \leq \ell_i, \qquad i \in \mathcal{I}. \tag{2.4}$$

If the maintenance of unit i starts, the maintenance must occur for a given duration d_i. Furthermore, this maintenance must occur contiguously for the given duration. When the binary variables are used [11], the duration constraint set is

$$\sum_{j \in \mathcal{J}} y_{i,j} = d_i, \qquad i \in \mathcal{I}, \tag{2.5}$$

while the non-stop maintenance constraint set is given by

$$y_{i,j} - y_{i,j-1} \leq x_{i,j}, \qquad i \in \mathcal{I},\ j \in \mathcal{J} \backslash \{1\}, \tag{2.6}$$

$$y_{i,1} \leq x_{i,1}, \qquad i \in \mathcal{I}. \tag{2.7}$$

A simpler formulation may be given if the integer variables are used. The constraint set is then formulated [19, 25] as

$$y_{i,j} = \begin{cases} 1 & \text{for } x_i \leq j \leq x_i + d_i - 1 \\ 0 & \text{for all other } j. \end{cases} \tag{2.8}$$

This constraint set may be written more quantitively than in (2.8). The formulation presented in [48] presents two constraint sets. The first set is identical to the duration constraint set (2.5). The second set formulates the non-stop restriction as

$$\prod_{j=x_i}^{x_i+d_i-1} y_{i,j} = 1, \qquad i \in \mathcal{I}. \tag{2.9}$$

However, this approach is undesirable since the constraints are nonlinear. If possible, one should avoid nonlinear constraints as it becomes much more difficult to solve a problem bound by such restrictions. There is a linear constraint formulation that can include the duration and continuous maintenance requirements into one constraint [46, 65]. This elegant constraint set is

$$\sum_{j=x_i}^{x_i+d_i-1} y_{i,j} = d_i, \qquad i \in \mathcal{I}. \tag{2.10}$$

The load constraints ensure that the forecasted load demand for each time period is met. Two approaches are presented in the literature. Firstly, let $g_{i,j}$ denote the generating capacity of unit i during time period j. To ensure the demand is at least met, the total generating capacity less the capacities in maintenance should generate enough electricity [32], that is

$$\sum_{i\in\mathcal{I}} g_{i,j} - \sum_{i\in\mathcal{I}} g_{i,j}y_{i,j} \geq D_j, \qquad j \in \mathcal{J}, \tag{2.11}$$

where the demand at time period j is denoted by D_j. If R_j denotes the reserve margin/safety level required during time period j, the reliability constraints may be combined with the load constraints [23, 32] in (2.11) as

$$\sum_{i\in\mathcal{I}} g_{i,j} - \sum_{i\in\mathcal{I}} g_{i,j}y_{i,j} \geq D_j + R_j, \qquad j \in \mathcal{J}. \tag{2.12}$$

A formulation, similar in form to (2.12), but using the binary decision variables $x_{i,j}$ is presented in [15]. Let $\mathcal{S}'_{i,j}$ be the set of start time periods such that if the maintenance of unit i starts at time period k then unit i will be in maintenance during time period j, therefore $\mathcal{S}'_{i,j} = \{k \in \mathcal{J}_i \mid j - d_i + 1 \leq k \leq j\}$. Also, let $\mathcal{I}'_j$ be the set of indices of generating units which are allowed to be in maintenance during time period j, so $\mathcal{I}'_j = \{i \mid j \in \mathcal{J}_i\}$. The constraint set (2.12) may then be replaced by

$$\sum_{i\in\mathcal{I}} g_{i,j} - \sum_{i\in\mathcal{I}'_j} \sum_{k\in\mathcal{S}'_{i,j}} g_{i,k}x_{i,k} \geq D_j + R_j, \qquad j \in \mathcal{J}. \tag{2.13}$$

The second approach to formulate the load constraints is to assume that the output level of a generating unit is not fixed at its capacity [9, 19, 66]. This leads to the introduction of a variable $p_{i,j}$ which denotes the output level of generating unit i during time period j. Since the specific output levels are computed, the generated output must equal the demand. This may be achieved by requiring that

$$\sum_{i\in\mathcal{I}} p_{i,j} = D_j, \qquad j \in \mathcal{J}, \tag{2.14}$$

while the additional constraint set specifying the limits of each unit's output level is given by

$$0 \leq p_{i,j} \leq g_{i,j}(1 - y_{i,j}), \qquad i \in \mathcal{I},\ j \in \mathcal{J}. \tag{2.15}$$

The output level must be less than the generating capacity and zero whenever the unit is in maintenance. In this case, the reliability constraint, incorporating a reserve margin, must be specified separately from the load constraints [9, 19, 66]. However, the constraint set is identical to (2.12).

General resource constraints ensure that the maximum resources available for maintenance during any time period are not exceeded. Let $q_{i,j}$ denote the quantity of the resources required

by generating unit i during time period j when in maintenance. The resource constraint set [3, 32] may then be formulated as

$$\sum_{i \in \mathcal{I}} q_{i,j} y_{i,j} \leq Q_j, \qquad j \in \mathcal{J}, \tag{2.16}$$

where Q_j denotes the available resources during time period j. Since manpower may be considered as a resource, crew constraints [25] are typically formulated in this fashion. The constraint set may also be formulated in terms of the decision variables $x_{i,j}$, according to the crew constraints in [15], as

$$\sum_{i \in \mathcal{I}'_j} \sum_{k \in \mathcal{S}'_{i,j}} q_{i,k} x_{i,k} \leq Q_j, \qquad j \in \mathcal{J}. \tag{2.17}$$

There may be a number of reasons why certain units are not allowed to be in maintenance simultaneously. Typically, there are certain sets of units (for example, units within the same power plant, units within the same class or units within the same geographical region). As a result, define $\mathcal{I}_k$ as the subset of generating units that belong to some specific grouping k of units. If there are K different groupings, let $\mathcal{K}$ denote the set of indices of these groupings with $\mathcal{K} = \{1, \ldots, K\}$. Then $k \in \mathcal{K}$. The exclusion constraint set then takes the form

$$\sum_{i \in \mathcal{I}_k} y_{i,j} \leq K_k, \qquad j \in \mathcal{J},\ k \in \mathcal{K}, \tag{2.18}$$

where K_k denotes the maximum number of units within grouping k that is allowed to be in maintenance simultaneously [11, 48].

It may be required that some generating units should be in maintenance before others (*i.e.* implementing different priority levels). If maintenance of unit i_1 has to start before that of unit i_2, the following pair of constraint sets ensures this precedence when using the binary decision variables

$$\sum_{p=1}^{j} x_{i_1,p} - x_{i_2,j} \geq 0, \qquad j \in \mathcal{J}, \tag{2.19}$$

$$x_{i_1,j} + x_{i_2,j} \leq 1, \qquad j \in \mathcal{J}. \tag{2.20}$$

Constraint set (2.19) ensures that the maintenance of unit i_2 does not start before the maintenance of unit i_1, while constraint set (2.20) prevents the maintenance of two units from starting simultaneously [11]. The precedence constraints are much simpler to formulate using the integer decision variables. The following precedence constraint formulation specifies that maintenance of unit i_2 may only start after the maintenance of unit i_1 has been completed [48, 66], *i.e.*

$$x_{i_1} + d_{i_1} \leq x_{i_2}. \tag{2.21}$$

Should one require that the maintenance of unit i_1 only starts before the maintenance of unit i_2, the constraint formulation becomes

$$x_{i_1} < x_{i_2}. \tag{2.22}$$

Lastly, as mentioned in §2.1.3, transmission/network constraints have recently been receiving attention in GMS problem formulations. These constraints ensure that the flow on all the transmission lines are kept within their limits (*i.e.* capacity constraints). The output levels

of the generating units are necessary for implementation of such constraints — therefore the transmission contraints may only be applied in conjunction with constraint sets (2.14) and (2.15). From these output levels, a load flow problem is solved to obtain the line flows. Let $\mathcal{L} = \{1, \ldots, L\}$ be the set of indices of transmission lines. These computed line flows must adhere to the transmission constraint set

$$|f_\ell| \leq F_\ell, \qquad \ell \in \mathcal{L}, \tag{2.23}$$

where f_ℓ denotes the flow on transmission line ℓ and F_ℓ denotes the maximum flow limit of transmission line ℓ in the power network [24, 51].

2.2.2 Objective function formulations in the literature

A number of economic objectives appear in the literature within a single objective environment. Let $c^p_{i,j}$ denote the production cost associated with generating unit i during time period j and let $c^m_{i,j}$ denote the maintenance cost incurred if unit i is in maintenance during time period j. An objective function presented in [49] is formulated with the requirement to

$$\text{minimise} \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} \left(c^p_{i,j}(1 - y_{i,j}) + c^m_{i,j} y_{i,j} \right). \tag{2.24}$$

A similar objective function is presented in [21], but the authors only considered the production cost. On the other hand, in [65] the only cost considered in the objective function is the maintenance cost. However, the maintenance cost is multiplied by the generating capacity of the unit (in their case $g_{i,j} = g_i$), giving rise to an objective that should

$$\text{minimise} \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} c^m_{i,j} g_i y_{i,j}. \tag{2.25}$$

According to the model, the cost coefficients are at a minimum when a unit is in maintenance during its preferred period of maintenance. Outside this preference window, the costs increase linearly. The generating capacity factor in the objective function results in a heavier penalty attributed to larger generating units. Therefore, a good schedule will give preference to the larger units' preferred maintenance periods.

Since fuel costs tend to dominate the value of production costs, some models simply use the fuel costs directly. This approach is adopted in the objective function in [19]. Let c^f_j denote the fuel cost per unit megawatt output during time period j. The objective function employs the integer decision variables and is defined to

$$\text{minimise} \sum_{j\in\mathcal{J}} \left(c^f_j \sum_{i\in\mathcal{I}} p_{i,j} \right) + \sum_{i\in\mathcal{I}} c^m_{i,(x_i)}, \tag{2.26}$$

targeting the minimisation of the overall fuel cost (production cost) and overall maintenance cost — therefore having the same form as (2.24). The maintenance cost $c^m_{i,(j)}$ in this objective function, however, is defined as the maintenance cost of unit i if its maintenance commenced during time period j, which is slightly different than in (2.24).

If a GMS formulation includes start-up and shut-down considerations, an important financial quantity is the start-up cost of a generating unit, as it may be a considerable amount. The objective function presented in [11] includes such costs. In that model development, the author determined that the maintenance costs were insignificant in comparison with the start-up costs

and production costs and hence, the maintenance costs were disregarded. The objective function has the same general form as (2.26) with production costs replacing the fuel costs and start-up costs replacing the maintenance costs.

The deterministic reliability objective approaches in the literature are all concerned with the reserve load. In almost all single objective cases, the objective is to level the reserve load over the planning period. The most common objective function is the sum of squares of the reserve loads, which is to be minimised. Since the reserve load during a given time period j is calculated by substracting the demand from the available generating capacity (*i.e.* total capacity minus capacity in maintenance minus demand), the objective function is to

$$\text{minimise} \sum_{j\in\mathcal{J}} \left(\sum_{i\in\mathcal{I}} g_{i,j} - \sum_{i\in\mathcal{I}} g_{i,j} y_{i,j} - D_j \right)^2, \tag{2.27}$$

as presented in [32]. The formulation in [15] involves this same sum of squares objective function, only in terms of the starting binary variables $x_{i,j}$. One may also include the reserve margin within the sum of squares objective function, whereby a pseudo-reserve load is levelled [63]. The reserve margin R_j is simply subtracted as well.

Another objective function that can also level the reserve is the sum of the differences between the average reserve load and the reserve loads for every time period [23]. The average reserve load $\bar{r}$ is calculated by

$$\bar{r} = \frac{1}{m} \sum_{j\in\mathcal{J}} r_j, \tag{2.28}$$

where the reserve load during time period j is given by

$$r_j = \sum_{i\in\mathcal{I}} g_{i,j} - \sum_{i\in\mathcal{I}} g_{i,j} y_{i,j} - D_j, \qquad j \in \mathcal{J}. \tag{2.29}$$

Note that $\bar{r}$ and r_j are merely used here for notational purposes to correspond closely to the formulation presented in [23]. The objective is then to

$$\text{minimise} \sum_{j\in\mathcal{J}} (\bar{r} - r_j). \tag{2.30}$$

This objective function is not as strong as the sum of squares objective function for two reasons. Firstly, each term in (2.27) is positive, whereas terms in (2.30) may be negative. In the summation, large negative or positive terms may be cancelled by other terms and result in a low objective function value when minimised. However, the reserve loads might be very different from one another. This cannot happen in the objective function in (2.30). Secondly, outlier reserve loads are penalised much more in (2.27) than in (2.30) because of the square.

A different reliability objective than levelling the reserve loads is to maximise the smallest reserve load during any time period [68]. The objective is therefore to

$$\text{maximise} \quad \min_{j\in\mathcal{J}} \left\{ \sum_{i\in\mathcal{I}} g_{i,j} - \sum_{i\in\mathcal{I}} g_{i,j} y_{i,j} - D_j \right\} \tag{2.31}$$

and is an attractive objective if a utility wishes to maintain the largest possible reserve during any given time period.

The various single objective GMS models in the literature are formulated from the constraint and objective function formulations presented in this section. Since all the presented economic

objective functions are linear, GMS problems may be formulated as integer or mixed-integer linear programs, depending on the decision variables that are chosen and assuming that constraint set (2.9) is not included. However, all the reliability objective functions are nonlinear. Depending on the reliability objective and choice of decision variables, GMS problems may also be formulated as integer/mixed-integer quadratic/nonlinear programs.

Numerous multiobjective formulations also appear in the literature. The approach followed in [47] defines an objective function vector $\boldsymbol{F}(X) = [F_1(X) \dots F_t(X) \dots F_T(X)]$ containing the values of T different objectives under a maintenance schedule X. Each objective is calculated by

$$F_t(X) = \sum_{j \in \mathcal{J}} f_{t,j}(y_{1,j}, \dots, y_{n,j}) \tag{2.32}$$

where $f_{t,j}(y_{1,j}, \dots, y_{n,j})$ denotes the value of objective t during time period j. The formulation specified three objectives, namely fuel costs, expected unserved energy and constraint violations in terms of a penalty function. More objectives may, however, be added to the vector. A solution is found by minimising the Euclidian distance between the ideal point $\boldsymbol{F}^*(X)$ and the objective function vector.

A goal programming approach is presented in [64] involving two objectives, namely minimising an economic cost and levelling the reserve margin. The economic cost includes the fuel, start-up and storage costs, as well as penalties for constraint violations. In the model, the reserve margin for each period is defined as the total available generating capacity divided by the period's peak load demand. The first stage of the problem is solved adopting the cost as objective function. This function is similar in nature to (2.24) but with many more different components. The second stage of the problem is solved adopting the reserve margin as objective function for a fixed minimum cost objective function value. Using the notation in [64], let mr_p denote the reserve margin at time period p. The relation between consecutive periods' reserve margins is given by the constraint

$$mr_p - mr_{p+1} + h_p - a_p = 0, \qquad \text{for all periods } p. \tag{2.33}$$

The objective is to minimise the sum of the slack variables in this constraint over all time periods, *i.e.* to

$$\text{minimise} \sum_p (h_p + a_p)\,. \tag{2.34}$$

Therefore the sum of the differences between the reserve margins of consecutive periods is minimised. By minimising this objective function, the reserve margins may be levelled.

2.2.3 Other problem formulations

A different problem formulation than the ones mentioned in the previous section is a *constraint satisfaction problem* (CSP) formulation [33]. The GMS problem may be encoded into the constraint satisfaction framework, consisting only of *hard* constraints — ones that must be satisfied for a solution to be feasible. The GMS problem is encoded as a CSP with $3 \times n \times m$ variables. Three sets of variables may be distinguished and the notation in [33] is used to define them. The variables $X_{i,j}$ may take on values in the set {ON, OFF, MAINT} corresponding to the state of unit i during time period j. Variables $Y_{i,j} \in$ {FIRST, SUBSEQUENT, NOT} signify the state of unit i during time period j. If $Y_{i,j} =$ FIRST, then maintenance of unit i starts during time period j; if $Y_{i,j} =$ SUBSEQUENT then unit i is scheduled for maintenance during time period j and for at least one prior time period, while if $Y_{i,j} =$ NOT, then the unit is not in

maintenance. Lastly, the variables $Z_{i,j}$ are boolean variables that may take on values in the set $\{\text{NONE}, \text{FULL}\}$, indicating whether unit i is producing power during time period j or not. With these variable definitions and domain values, the GMS problem is encoded in a CSP framework.

Another class of problem formulations in the literature is found by applying fuzzy logic to the GMS problem [14, 25, 36]. If one uses a fuzzy approach, the objective(s) and/or certain constraints may be *fuzzified* to incorporate uncertainty into the problem. A membership function must be defined for the chosen fuzzy numbers (for example, the load demand, manpower requirements or operating costs). An explanation of the basic concepts of fuzzy sets is presented later in this chapter. Typically, the constraint set and objective function formulations are the same as in §2.2.1 and §2.2.2; the only difference is that the fuzzy numbers are used instead of the deterministic values. The variable definitions remain exactly the same.

2.3 Model extensions

The GMS problem has received considerable attention in the literature over the years [1, 46] and continues to play a central role in decision making at electrical power utilities world-wide. As technologies evolve, electricity demand invariably increases and competitive markets emerge, and so the effectiveness of GMS models also has to stay abreast of developments by expanding the models so as to include other processes.

As mentioned in §2.1.3, transmission considerations have only recently been introduced into GMS modelling and then only in the form of constraints. Since electricity provision depends on both the generating units and the transmission lines, a logical step would be to formulate the maintenance scheduling of both these components as a single problem. This notion has been proposed in [24, 59]. In order to model these two scheduling problems in a single optimisation problem, the objective function will have to be altered and certain constraints will have to be added. The details of this formulation may be found in the two mentioned sources. However, a brief description of these additions is given here. Since transmission lines do not induce production costs, the economic objective function is defined as the sum of the production costs, maintenance costs of the generating units and maintenance costs of the transmission lines. The objective is to

$$\text{minimise} \sum_{j\in\mathcal{J}} \left(\sum_{i\in\mathcal{I}} c^p_{i,j} p_{i,j}(1-y_{i,j}) + \sum_{i\in\mathcal{I}} c^m_{i,j} y_{i,y} + \sum_{\ell\in\mathcal{L}} c^t_{\ell,j} T_{\ell,j} \right), \tag{2.35}$$

where $T_{\ell,j}$ is a binary variable taking the value 1 if transmission line ℓ is in maintenance during time period j and $c^t_{\ell,j}$ denotes the maintenance cost if transmission line ℓ is in maintenance during time period j. The additional constraints include constraint sets identical to (2.4), (2.8) and (2.16), but now also in terms of coefficients and variables of the transmission system. Furthermore, for the specific formulation in [24], the power output constraints (2.11) and (2.15) are replaced with

$$\sum_{i\in\mathcal{I}} p_{i,j}(1-y_{i,j}) = D_j, \qquad j\in\mathcal{J}, \tag{2.36}$$

$$0 \le p_{i,j} \le g_{i,j}, \qquad i\in\mathcal{I},\ j\in\mathcal{J}, \tag{2.37}$$

in order to correspond with the new objective function.

Coordination between separate generation and transmission companies with respect to their maintenance schedules is proposed in [34]. Maintenance scheduling problems for a generation

and a transmission company are formulated separately as mathematical programs with constraints (notably with transmission constraints included) similar to those presented earlier in this chapter. These two solutions are compared with so-called preferred schedules from a co-ordinator. After possible modifications to the schedules, the generation company's schedule is inserted as a constraint into the transmission company's problem, or *vice versa*. The combined problem is resolved and should any violations occur, the inserted schedule is modified until a final solution is found.

Another extension that has received attention in the literature is generation planning along with maintenance scheduling. The formulation proposed in [70] combines the constraints for each problem and considers a quadratic profit objective function.

2.4 Typical solution techniques

Large modern-day power systems typically exhibit a number of generating units — generally exceeding 100 [64] — with typical planning horizons of 52 weeks. The number of independent variables in a GMS problem are determined by these values, giving rise to very large combinatorial optimisation problem. Suitable solution techniques should be able to obtain good or optimal solutions for such large problems while keeping the computational time within reason. A comprehensive survey of GMS literature is given in [1], stating many of the solution techniques employed in the context of GMS, and is used as base reference for this section.

2.4.1 Heuristics

The application of heuristic techniques[4] are based on trail and error. Sequential maintenance is usually scheduled in a unit-by-unit manner [46] and possible corrections are made according to a previously defined scheduling order. In its simplest form, the scheduling order/priority list is found by arranging the units in decreasing order of capacity. These methods are typically simple to understand and very straight-forward to apply, requiring very little computational time. An example of a heuristic with the objective of levelling the reserve is presented here, as described in [13].

The units in the power system are classified into to six categories:

1. Units on fixed maintenance,
2. Units available only part of the year and requiring one maintenance occurence,
3. Units available only part of the year and requiring two maintenance occurences,
4. Units fully available for the year and requiring one maintenance occurence,
5. Units fully available for the year and requiring two maintenance occurences,
6. Units not requiring maintenance during the year.

The assumption is that if a unit requires two maintenance occurences, one occurence must be scheduled during the first half of the year, while the other occurence must be scheduled during

[4]Heuristic is derived from the Greek word *heuriskein* which means "to find." A heuristic technique finds (or attempts to find) a solution to a problem by means of loosely defined rules-of-thumb.

the second half. It is assumed that no unit requires more than two maintenance occurences. Furthermore, the problem constraints are ranked according to their severity. The heuristic proceeds as follows:

STEP 1: Determine a scheduling priority list for the units.

> Arrange the units in ascending order of category. For units within the same category, arrange them in descending order of maintenance duration. For units within the same category and with equal maintenance durations, arrange them in descending order of capacity.

STEP 2: Develop the maintenance schedule according to the priority list.

> Schedule a unit for maintenance over successive intervals (equal to the maintenance duration) such that the sum of reserves is a maximum without violating any of the constraints. If this causes an infeasibility, schedule the maintenance over the intervals which violates the least severe constraints.

Many of these heuristic methods were designed years ago when the complexity of the GMS problem seemed overwhelming. Today, despite the existence of sophisticated optimisation algorithms, many heuristic models are nevertheless still used [1].

2.4.2 Mathematical programming techniques

As demonstrated in §2.2, the GMS problem can easily be formulated as a mathematical program, thereby paving the way for the use of mathematical programming solution techniques.

The single-objective mixed-integer programs in [2, 21, 65] are all solved by algorithms that employ the branch-and-bound method. Although not formulated as a mathematical program, [47] employs a multiobjective branch-and-bound algorithm to solve the GMS problem. Unfortunately this algorithm is used in conjunction with successive approximations, and hence the solution is not necessarily globally optimal. Modern software packages that are available for solving integer optimisation problems generally use branch-and-bound methods, as they are still the most viable methods known for integer problems. Other popular and effective algorithms are used within these software packages to solve general mathematical programs.

The single-objective programs in [11, 34, 49, 60] employ a decomposition method called Benders' decomposition to solve the GMS problem. As GMS typically gives rise to a large-scale problem, decomposition seems a very attractive approach in order to reduce computational solution time.

The branch-and-bound method

The *branch-and-bound* (B&B) method is a general algorithm that may be applied to combinatorial optimisation problems and it guarantees an optimal solution. The method was first proposed by Land and Doig [50] in 1960. A general description of the method is presented here. Assume, without loss of generality, that an objective function $f(\boldsymbol{x})$ has to be minimised, where the decision variable $\boldsymbol{x}$ lies within some set $\mathcal{S}$ of candidate solutions (known as the search space or feasible region). The algorithm applies two procedures to the problem. Firstly, the *branching* procedure (given a subset $\mathcal{S}' \subseteq \mathcal{S}$) returns two or more smaller sets $\mathcal{S}'_1, \mathcal{S}'_2, \dots$ whose union covers $\mathcal{S}'$. The minimum value of $f(\boldsymbol{x})$ over $\mathcal{S}'$ is then attained at $\min\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}$, where

$f(\boldsymbol{x}_i)$ is the minimum value of $f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{S}_i'$. The recursive application of branching leads to a datastructure resembling a rooted tree, whose nodes are the subsets of $\mathcal{S}$, hence the name *branching*. The second procedure is known as *bounding* and it provides a method for calculating upper and lower bounds on the minimum value of $f(\boldsymbol{x})$ within a given subset $\mathcal{S}' \subseteq \mathcal{S}$.

The main idea of the B&B method is that if the lower bound for some node (subset of candidate solutions) A is greater than the upper bound for some other node B, then A may be discarded from the search (*i.e.* node A and its entire subtree may be removed). This step is known as *pruning*. The recursion (branching) stops when the current set of candidate solutions $\mathcal{S}$ is reduced to a single element or when the upper bound on $f(\boldsymbol{x})$ over $\mathcal{S}$ matches its lower bound. In both cases, a minimum value of $f(\boldsymbol{x})$ is found over $\mathcal{S}$.

Since the GMS problem requires maintenance of units to occur over discrete integer time intervals, its mathematical programming formulations employ integer variables for the starting times of maintenance or binary variables indicating when a unit is in maintenance, as stated in §2.1.2. Therefore, the GMS problem is often moulded in the shape of an *integer program* (IP). IPs are often solved by using the branch-and-bound method [102]. A brief description of the method in terms of an integer programming paradigm is presented below.

Consider a general integer program and assume that the objective is to minimise a linear function of the decision variables. The B&B method starts by solving the *linear programming* (LP) relaxation[5] of the IP. If an optimal solution to the LP relaxation comprises integer values for all the integer variables, this relaxed optimal solution is also an optimal solution to the IP. Should this not be the case, branching occurs on any integer variable with a non-integer value in the relaxed optimal solution. For a pure integer problem, branching may occur on any of the variables, however, in a mixed-integer problem the branching should only occur on integer variables.

The *branching* procedure partitions the feasible solution space into smaller, mutually exclusive subsets [40]. Each partition is represented by a *node* in a datastructure resembling a rooted tree and corresponds to a new subproblem. The root node represents the LP relaxation problem of the IP. If an integer variable x has a non-integer value a, then branching may occur by steering the value away from the interval $(\lfloor a \rfloor < x < \lceil a \rceil)$, leading to a branch for $x \leq \lfloor a \rfloor$ and a branch for $x \geq \lceil a \rceil$. Each inequality is included as a constraint in the respective new subproblems.

The *bounding* procedure computes a lower bound on the optimal objective function value of the IP at any node (*i.e.* for that subset of the solution space). This is achieved by solving the LP relaxation of the subproblem. Generally, branching occurs from the unterminated node with the smallest lower bound [40], a traversal protocol called jumptracking. However, there are also other traversal protocols (*e.g.* depth-first). If the solution to the LP relaxation of a node has integer values for all the integer variables, that solution is feasible for the IP and further branching on the node is terminated. Should any node in the search tree achieve a lower bound no better than the value of the best feasible solution uncovered up to that point, further branching on such a node may also be terminated (in which case the node is said to be *fathomed*), because subsequent branching will not lead to a solution with a better objective function value than that of the best encountered feasible solution. Branching continues until all nodes are fathomed or the value of the best lower bound is no better than that of the best feasible solution [40].

[5] The linear program obtained by omitting all integer or binary constraints on variables is called the LP relaxation of the integer program [102].

An advantage of the B&B method is that it guarantees a globally optimal solution. The method is also computationally acceptable (it should terminate within a acceptable amount of time) even for problems with a large number of variables and constraints [46]. The reason lies in the fact that potentially large subsets of solutions are discarded by means of the lower bounds (bounding procedure), thereby reducing computational time. This is, however, not necessarily always the case as it is possible to enumerate almost all of the solutions of the subproblems for pathologically difficult IPs (if most bounds obtained during the search are not useful). Different traversal protocols may also be beneficial in reducing computational time, since it allows a tailored approach to the problem, depending on the branch structure of the decision tree.

The simplex algorithm

The simplex algorithm was developed by George Dantzig in 1947 and it is generally an efficient method for solving linear programming problems. For most problems in practice, the simplex algorithm converges in expected polynomial time; however, its worst-case time complexity is exponential, as demontrated with specific examples by Klee and Minty [43] in 1972. The algorithm searches for an optimal solution along the extreme points of the feasible region of an LP[6]. The relevant terminologies are defined below, followed by a description of the basic algorithm when applied to a maximisation problem, following in [102].

An LP is said to be in *standard form* if all its constraints are equalities and all variables are nonnegative. The conversion of an LP into standard form is performed as follows:

- If constraint i is a "$\leq$" constraint, a slack variable s_i is added to the i-th constraint together with the sign restriction $s_i \geq 0$.
- If constraint i is a "$\geq$" constraint, an excess variable e_i is subtracted from the i-th constraint and the sign restriction $e_i \geq 0$ is imposed.
- If a variable x_i is unrestricted in sign, it is replaced by $x_i' - x_i''$ in the objective function, as well as all constraints, together with the sign restrictions are $x_i' \geq 0$ and $x_i'' \geq 0$.

Consider a system $A\boldsymbol{x} = \boldsymbol{b}$ of m linear equations in n variables (assume $n \geq m$). A *basic solution* to the system is obtained by setting $n - m$ variables equal to 0 and solving for the values of the remaining m variables. This assumes that setting the $n - m$ variables equal to 0 yields unique values for the remaining m variables or, equivalently, the columns for the remaining m variables are linearly independent [102]. The $n - m$ variables set to equal 0 are referred to as the *nonbasic variables*, while the m variables solved for values are referred to as the *basic variables*. Any basic solution in which all the variables are nonnegative is called a *basic feasible solution*[7].

The simplex algorithm proceeds as follows:

1. The LP is converted into standard form. Furthermore, a *canonical form*[8] of the LP is determined from the standard form. The objective function is added as a equation in row

[6]The feasible region for any LP is a convex set. For any convex set $\mathcal{S}$, a point $P \in \mathcal{S}$ is an *extreme point* if each line segment that lies completely in $\mathcal{S}$ and contains the point P has P as an endpoint of the line segment [102]. Furthermore, if an LP has an optimal solution, then it has an optimal solution corresponding to an extreme point [102].

[7]A point in the feasible region of an LP is an extreme point if and only if it is a basic feasible solution to the LP [102].

[8]A system of linear equations in which each equation has a variable with a coefficient of 1 in that equation, and a zero coefficient in all other equations, is said to be in canonical form.

zero above the canonical form, expressed as $z - \boldsymbol{c}^T\boldsymbol{x} = 0$, where z is the objective function value, $\boldsymbol{c}$ is the problem coefficient matrix and $\boldsymbol{x}$ the solution vector.

2. An initial basic feasible solution is obtained, if possible.

3. The current basic feasible solution is tested for optimality. This is achieved by determining whether a unit increase in any nonbasic variable (while keeping the other nonbasic variables at zero) will increase the objective function value. A nonbasic variable with a negative coefficient in row 0 will achieve this. The solution is optimal if the objective function value cannot be increased.

4. A nonbasic variable is sought to enter the basis (*entering variable*) and a basic variable is sought to exit the basis, if the current solution is not optimal. The entering variable is chosen as the nonbasic variable which will increase the objective function value the most (*i.e.* the nonbasic variable with the most negative coefficient in row 0). A ratio test is used to determine how large the entering variable can be made. This is achieved by computing, for any row in which the entering variable has a positive coefficient, the ratio

$$\frac{\text{Right-hand side of row}}{\text{Coefficient of entering variable in row}}. \tag{2.38}$$

   The row (constraint) with the smallest ratio is the winner of the ratio test.

5. A new basic feasible solution is determined. The entering variable is made a basic variable in the winning row of the ratio test. This is done by a procedure called *pivoting* which consists of elementary row operations to render a coefficient of 1 for the entering variable in the winning row. A new canonical form is created after the procedure is performed. The new basic feasible solution is tested for optimality in step 3 and the process repeats.

The algorithm continues until an optimal solution is found, provided the LP is not unbounded or that cycling does not occur. Furthermore, note that the right-hand side of any row (except the objective function row) may not be negative.

The dual simplex algorithm

Any LP (called the *primal* problem) is associated with another LP, called its *dual* problem. Consider the (primal) maximisation problem:

$$\begin{aligned} \text{maximise} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b}, \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{aligned}$$

Then its dual is given by:

$$\begin{aligned} \text{minimise} \quad & \boldsymbol{y}^T\boldsymbol{b} \\ \text{subject to} \quad & A^T\boldsymbol{y} \geq \boldsymbol{c}, \\ & \boldsymbol{y} \geq \boldsymbol{0}. \end{aligned}$$

If $\boldsymbol{x}^*$ is a feasible solution to the primal problem, and $\boldsymbol{y}^*$ is a feasible solution to the dual problem and $\boldsymbol{c}^T\boldsymbol{x}^* = \boldsymbol{y}^{*T}\boldsymbol{b}$, then $\boldsymbol{x}^*$ is optimal for the primal problem and $\boldsymbol{y}^*$ is optimal for the dual problem [102].

The simplex algorithm described above (applied to a maximisation problem, referred to here as the primal problem) starts with a primal feasible solution, since all the constraints have nonnegative right-hand sides in the initial iteration. Furthermore, at least one nonbasic variable may increase the objective function value (negative coefficient in row 0); therefore, the initial primal solution is not dual feasible. Since the right-hand sides never become negative during the simplex algorithm, primal feasibility is maintained, and the algorithm terminates at an optimal solution when a nonnegative row 0 is attained (no further improvement is possible), *i.e.* when dual feasibility has been achieved.

The dual simplex method may be used when each variable in row 0 has a nonnegative coefficient (dual feasibility) and at least one constraint (row) has a negative right-hand side (primal infeasible). The dual simplex algorithm maintains the nonnegative row 0 (dual feasibility) and at some point reaches an iteration where all the right-hand sides are nonnegative (primal feasibility). At this point, an optimal solution has been obtained [102].

The generalised reduced gradient algorithm

The generalised reduced gradient algorithm is a solution method that extends the reduced gradient algorithm for *nonlinear programming* (NLP) problems having the standard form that aims to

$$\begin{aligned} &\text{minimise} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}, \\ & && \boldsymbol{a} \leq \boldsymbol{x} \leq \boldsymbol{b}, \end{aligned}$$

where $\boldsymbol{h}(\boldsymbol{x})$ has dimension m. Note that a general NLP problem can always be expressed in the standard form above by including slack variables and allowing some components of $\boldsymbol{a}$ and $\boldsymbol{b}$ to take the values $+\infty$ or $-\infty$, if necessary [58]. The description of the generalised reduced gradient algorithm presented here, is paraphrased from [58].

The algorithm requires a *nondegeneracy* assumption stating that, at each point $\boldsymbol{x}$, a partition of $\boldsymbol{x}$ into $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$ exists with the properties:

- $\boldsymbol{y}$ has dimension m and $\boldsymbol{z}$ has dimension $n - m$,
- if $\boldsymbol{a} = (\boldsymbol{a_y}, \boldsymbol{a_z})$ and $\boldsymbol{b} = (\boldsymbol{b_y}, \boldsymbol{b_z})$ are the corresponding partitions of $\boldsymbol{a}$ and $\boldsymbol{b}$, then $\boldsymbol{a_y} < \boldsymbol{y} < \boldsymbol{b_y}$,
- the $m \times m$ matrix $\boldsymbol{\nabla_y h}(\boldsymbol{y}, \boldsymbol{z})$ is nonsingular at $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$.

The vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ consist of the dependent and independent variables, respectively. The idea behind the generalised reduced gradient method is to consider the problem only in terms of the independent variables $\boldsymbol{z}$ at each stage. If $\boldsymbol{z}$ is specified, $\boldsymbol{h}(\boldsymbol{y}, \boldsymbol{z}) = \boldsymbol{0}$ can be solved uniquely for the dependent variables $\boldsymbol{y}$ and as such, the objective function may be considered a function of $\boldsymbol{z}$ only. By accounting for the constraints, a simple modification of the method of steepest descent may be executed. The reduced gradient, with respect to the independent variables $\boldsymbol{z}$, is

$$\boldsymbol{r}^T = \boldsymbol{\nabla_z} f(\boldsymbol{y}, \boldsymbol{z}) + \boldsymbol{\lambda}^T \boldsymbol{\nabla_z h}(\boldsymbol{y}, \boldsymbol{z}), \tag{2.39}$$

where $\boldsymbol{\lambda}$ satisfies

$$\boldsymbol{\nabla_y} f(\boldsymbol{y}, \boldsymbol{z}) + \boldsymbol{\lambda}^T \boldsymbol{\nabla_y h}(\boldsymbol{y}, \boldsymbol{z}) = \boldsymbol{0}. \tag{2.40}$$

Equivalently, the reduced gradient is

$$\boldsymbol{r}^T = \boldsymbol{\nabla}_{\boldsymbol{z}} f(\boldsymbol{y},\boldsymbol{z}) \overbrace{-\boldsymbol{\nabla}_{\boldsymbol{y}} f(\boldsymbol{y},\boldsymbol{z})[\boldsymbol{\nabla}_{\boldsymbol{y}} \boldsymbol{h}(\boldsymbol{y},\boldsymbol{z})]^{-1}}^{\boldsymbol{\lambda}^T} \boldsymbol{\nabla}_{\boldsymbol{z}} \boldsymbol{h}(\boldsymbol{y},\boldsymbol{z}). \tag{2.41}$$

Therefore, moves are taken by changing $\boldsymbol{z}$ in the direction of the negative reduced gradient and components of $\boldsymbol{z}$ on their boundary are held fixed if the movement would violate the bound. The independent variables vector $\boldsymbol{z}$ moves along a straight line; however, the dependent variables vector $\boldsymbol{y}$ must move nonlinearly in order to satisfy the equality constraints. This is achieved by first moving linearly along the tangent to the constraint surface by means of the moves $\boldsymbol{z} \to \boldsymbol{z} + \Delta\boldsymbol{z}$, $\boldsymbol{y} \to \boldsymbol{y} + \Delta\boldsymbol{y}$ where $\Delta\boldsymbol{y} = -[\boldsymbol{\nabla}_{\boldsymbol{y}}\boldsymbol{h}]^{-1}\boldsymbol{\nabla}_{\boldsymbol{z}}\boldsymbol{h}\Delta\boldsymbol{z}$. Next, a correction procedure is applied to return to the constraint surface and the magnitude bounds on the dependent variables are tested for feasibility. Furthermore, a feasibility tolerance has to be introduced to address the enforced return to the constraint surface.

An iterative scheme is used to return to the surface. If $\boldsymbol{x}_k$ is the point at the previous step, then from any point $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{w})$ near $\boldsymbol{x}_k$ one returns to the constraint surface by solving the nonlinear equation

$$\boldsymbol{h}(\boldsymbol{y}, \boldsymbol{w}) = \boldsymbol{0}, \tag{2.42}$$

for $\boldsymbol{y}$ with $\boldsymbol{w}$ being fixed. This is done through the iterative process

$$\boldsymbol{y}_{j+1} = \boldsymbol{y}_j - [\boldsymbol{\nabla}_{\boldsymbol{y}}\boldsymbol{h}(\boldsymbol{x}_k)]^{-1}\boldsymbol{h}(\boldsymbol{y}_j, \boldsymbol{w}), \tag{2.43}$$

which produces a sequence $\{\boldsymbol{y}_j\}$ with $\boldsymbol{y}_j \to \boldsymbol{y}$, solving (2.42), if it is started close enough to $\boldsymbol{x}_k$.

The method of successive linear programming

The method of *successive linear programming* (SLP), also known as sequential linear programming, is an extension of linear programming in order to solve NLPs. In the method of SLP, the nonlinear functions in an NLP are linearised so that the resulting problem is linear and may be solved using traditional linear programming techniques.

A general nonlinear function $f(\boldsymbol{x})$ may be linearly approximated about the point $\boldsymbol{x}_c$ by the first-order Taylor polynomial

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_c) + [\nabla f(\boldsymbol{x}_c)]^T(\boldsymbol{x} - \boldsymbol{x}_c). \tag{2.44}$$

The method of SLP starts by linearising the NLP about an initial estimate of the optimal solution. The resulting LP is solved and the solution (an improved estimate of the optimal) is used to create a new linearisation of the NLP. Continuing in this manner, a sequence of NLP linear approximations are solved whereby each new solution is closer to the optimal solution of the NLP. However, the linearisations need not be bounded; therefore, trust regions[9] or similar techniques may be required to ensure convergence to a globally optimal solution [97].

Barrier methods

Barrier methods, also known as interior point methods, approximate constrained problems (linear or nonlinear) by unconstrained problems. A term is added to the objective function that

[9] If an objective function is replaced by a suitable model of it (*i.e.* an approximation) the model may only be "trusted" within a certain trust region around the current point, specified by a trust region radius.

favours points lying in the interior of the feasible region over those that lie near the boundary, thereby creating a "barrier" on the boundary that prevents a search procedure from leaving the feasible region.

Barrier methods may be applied to problems that

$$\left.\begin{array}{ll} \text{minimise} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{S}, \end{array}\right\} \tag{2.45}$$

where the constraint set $\mathcal{S}$ has a nonempty interior such that any boundary point may be reached by approaching it from the interior. A *barrier function* B is defined on the interior of $\mathcal{S}$, with the properties that

- B is continuous,
- $B(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathcal{S}$, and
- $B(\boldsymbol{x}) \to \infty$ as $\boldsymbol{x}$ approaches the boundary of $\mathcal{S}$.

The corresponding barrier method approximation for the problem in (2.45) may require one to

$$\begin{array}{ll} \text{minimise} & f(\boldsymbol{x}) + \dfrac{1}{c}B(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \text{interior of } \mathcal{S}, \end{array}$$

where c is a large positive constant. In the formulation, the problem is still constrained, but the advantage is that it may be solved using unconstrained search techniques [58].

The barrier method for a problem of the form presented in (2.45) proceeds as follows. Let $\{c_k\}$ be a sequence of real numbers tending to infinity such that $c_k \geq 0$ and $c_{k+1} > c_k$ for each $k \in \{1, 2, 3, \dots\}$. Furthermore, define the function

$$r(c, \boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{c}B(\boldsymbol{x}), \tag{2.46}$$

and for each k, solve the problem

$$\begin{array}{ll} \text{minimise} & r(c_k, \boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \text{interior of } \mathcal{S}, \end{array}$$

to obtain a sequence $\{\boldsymbol{x}_k\}$. The result [58] is that any limit point of a sequence $\{\boldsymbol{x}_k\}$ generated by the barrier method is a solution to the problem in (2.45).

A decomposition method

Benders' decomposition is a method that allows one to solve hard optimisation problems which have a certain substructure. As with any decomposition method, Benders' decomposition partitions the problem into multiple smaller problems which are easier (and faster) to solve. Therefore it is especially suitable for large-scale problems [76]. Benders' decomposition was initially developed for solving linear programs and, as such, is explained here within the context of a *mixed-integer linear program* (MILP).

In order to apply Benders' decomposition to a linear program, the problem has to have the following structure:

$$\left.\begin{array}{ll} \text{minimise} & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{y} \\ \text{subject to} & A\boldsymbol{x} + B\boldsymbol{y} = \boldsymbol{b}, \\ & \boldsymbol{x} \geq \boldsymbol{0}, \\ & \boldsymbol{y} \in Y \subseteq \mathbb{R}^q, \end{array}\right\} \tag{2.47}$$

where q is the number of y-variables and $\mathbb{R}^q$ is the subspace of vectors having q real-valued elements. This means the problem may be partitioned in terms of the variables, where the y-variables are *complicating variables* in the sense that the problem becomes easier to solve if the y-variables are fixed [76]. In the MILP context, $\boldsymbol{y} \in \mathbb{Z}^q$ with $\mathbb{Z}^q$ being the subspace of vectors having q integer-valued elements. Benders' decomposition partitions (decomposes) the problem in (2.47) into a master problem and a subproblem. The master problem is an MILP containing the y-variables. It contains only a few constraints and may therefore be considered as a relaxation of the original problem [49]. By solving the master problem, a preliminary solution is generated which attains a lower bound on the optimal objective function value for the original problem. This trial solution (*i.e.* values for $\boldsymbol{y}$) is used to solve the subproblem, an LP containing only the x-variables.

If the subproblem has an infeasible solution, a constraint referred to as a *Benders feasibility cut* is added to the relaxed master problem. If the subproblem has a feasible solution, but the objective function value of the original problem is not sufficiently close to the lower bound, a constraint referred to as a *Benders optimality cut* is added to the relaxed master problem. In this manner, the master problem is iteratively solved with additional constraints until a solution is found, satisfying all the constraints, and which is close enough to the lower bound. Since there are a finite number of Benders cuts and since cuts are generated in each iteration, the method converges to an optimal solution in a finite number of iterations [76].

A flow diagram of the Benders' decomposition method is presented in Figure 2.1. According to [76], the master problem may also be nonlinear or a constraint programming problem. Likewise, the subproblem may be extended to a convex program. These characteristics make Benders' decomposition method an effective method to use in respect of a wide range of problems.

2.4.3 A dynamic programming variant

It has been suggested in [1, 46, 104, 105] that *dynamic programming* (DP) suits maintenance problems best, based on the following properties: "*(1) DP is especially suitable for problems where a sequence of decisions is involved, (2) the objective function need not be a continuous function of decision and state variables, and (3) analytic forms for the objective functions are not required (provided one can obtain the function values at a given state)* [104]."

In order to apply DP to the GMS problem it has to be formulated as a dynamic programming model. A control vector $\boldsymbol{U}(j) = [u_1(j)\ u_2(j)\ \ldots\ u_n(j)]$ is defined, with $u_i(j)$ being control binary variables, for all $i \in \mathcal{I}, j \in \mathcal{J}$, attaining a value of 1 if unit i is in maintenance during time period j, and 0 otherwise. The state vector is defined as $\boldsymbol{X}(j) = [x_1(j)\ x_2(j)\ \ldots\ x_n(j)]$ where $x_i(j)$ is the state variable indicating the degree of maintenance completion for unit i at the beginning of time period j. The maintenance scheduling process may then be expressed by the recursive relation

$$\boldsymbol{X}(j+1) = \boldsymbol{X}(j) + \boldsymbol{U}(j), \qquad j \in \mathcal{J}, \tag{2.48}$$

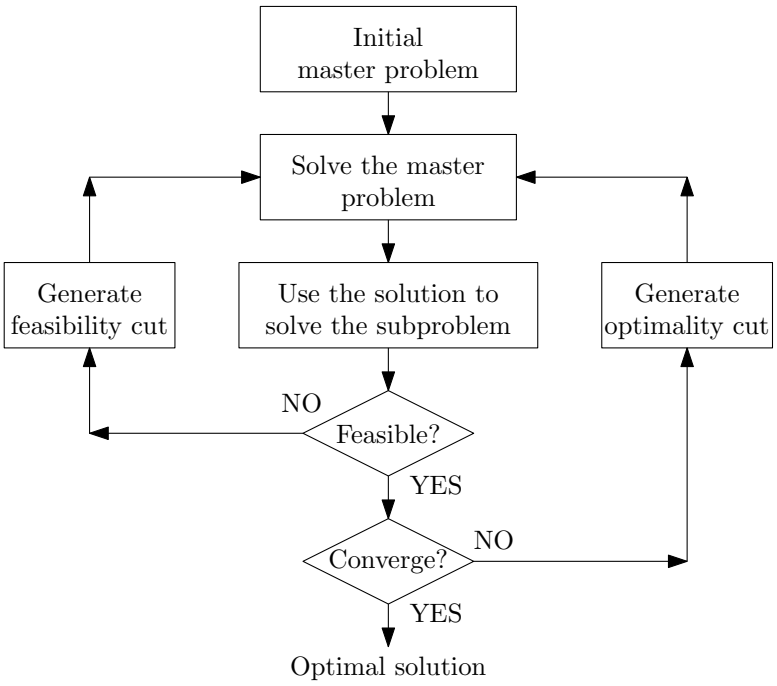


Figure 2.1: *Flow diagram of Benders' decomposition method.*

together with the boundary conditions

$$\boldsymbol{X}(1) = \mathbf{0} \quad \text{and} \quad \boldsymbol{X}(m+1) = [d_1 \ d_2 \ \ldots \ d_n], \tag{2.49}$$

where d_i denotes the maintenance duration of unit $i \in \mathcal{I}$.

The DP model for the GMS problem contains a large number of control and state variables and, as such, the DP approach suffers greatly under the "curse of dimensionality". As a result, a direct application of dynamic programming is not manageable on account of computational requirements. This has led to the application of *dynamic programming with successive approximations* (DPSA) to reduce the problem dimensions. This technique converges, but does not guarantee a global optimum [46].

The DPSA algorithm achieves a trade-off between scheduling units sequentially (as heuristic methods do according to a priority list) and applying simultaneous scheduling of units. The algorithm solves a sequence of DP problems, where the elements of the sequence are disjoint subsets of the units. If the units are grouped into small subsets, a reduction of the state space is achieved. The algorithm starts with a user-given initial solution. In each iteration, most of the control and state variables (*i.e.* all those not in the current subset) have fixed values and values are found for the remaining variables by means of DP. Each iteration continues onto the next subset of units. The algorithm stops if two successive iterations produce the same schedule or if no significant change in the objective function value is achieved. A flow chart describing the working of the DPSA algorithm for the GMS problem is presented in Figure 2.2. Ultimately, the DPSA method for solving the GMS problem should only be considered as an advanced heuristic, due to the use of successive approximations. The method was successfully used in the earlier years of GMS, as in [104, 107].

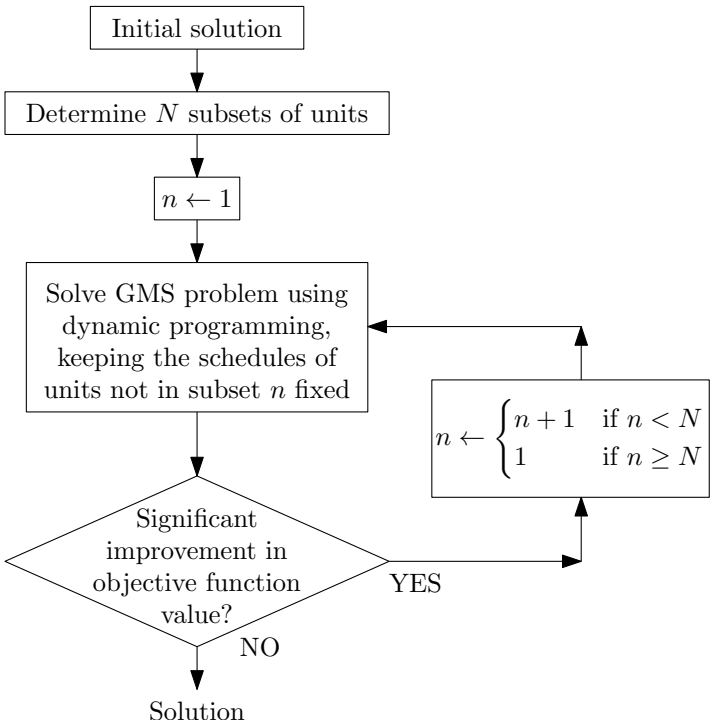


Figure 2.2: *Flow chart of the DPSA algorithm for the GMS problem.*

2.4.4 Metaheuristics

The "curse of dimensionality" of the GMS problem prohibits exact solution methodologies to obtain results within acceptable computation time frames. A relatively new approach to counter problems (typically combinatorial in nature) with this attribute is to apply metaheuristic[10] techniques. These techniques typically obtain very good solutions (although not necessarily optimal) within more acceptable computation time frames. Most metaheuristics are inspired by analogies from nature [20], *e.g.* from physics, biology or ethology. Unfortunately, there is no exact definition of the notion of a metaheuristics. However, the United States National Institute of Standards and Technology [8] defines a metaheuristic as

1. *"A high-level algorithmic framework or approach that can be specialized to solve optimization problems."*
2. *"A high-level strategy that guides other heuristics in a search for feasible solutions."*

In recent years, metaheuristics have been shown to be capable of solving GMS problems to close optimality within very limited computation time budgets. A review of some of these techniques may be found in [17].

Genetic algorithms

A *genetic algorithm* (GA) is an example of a metaheuristic also oftenly called an evolutionary algorithm and is based on the biological analogy of evolution and natural selection proposed

[10]The word metaheuristic is derived from the Greek prefix *meta* which means "position beyond or something of a higher or second-order kind" and the Greek word *heuriskein* which means "to find."

by Charles Darwin [20]. His theory states that evolution in a species occurs as a result of the competition which selects the best adapted individuals to survive, ensuring the continuation of the species by transmitting the useful characteristics of these individuals to their offspring. A form of cooperative sexual reproduction ensures that this inheritance carries on strongly [20].

In terms of the analogy, the solutions of a given problem are the *individuals* of a species and constitute candidate solutions in the search space. A *fitness* level is associated with each individual. This fitness specifies its desirability of being chosen for reproduction or replacement and is determined by a *fitness function* which depends on the objective function. The subset of individuals that are considered simultaneously by the algorithm is referred to as a *population*. The population iteratively evolves until some termination criterion is met and each iteration is called a *generation*.

During each generation, specific operators act on the individuals of a population to create the new population for the next generation. A new population is generated by individuals reproducing, surviving or disappearing from the current population. If an operator acts on one or more individuals, they are called *parents* while individuals that are created by these operators are called *offspring*.

The *selection* operator determines how many times an individual will be chosen for reproduction, while the *replacement* operator determines which individuals will have to be discarded from the new population so as to maintain a fixed population size. These two operators are known as the *selection operators*. Since the aim of the genetic algorithm is to obtain better solutions at each generation until a best solution is found, the individuals in a population must undergo some transformation. This is achieved by applying *variation operators* or *search operators* to the population. Two categories of variation operators exist, namely *mutation* operators and *crossover* operators. The mutation operators modify one individual to form another while the crossover operators generate one or more offspring from combinations of two or more parents (*i.e.* analogous to sexual reproduction).

Lastly, GAs also utilise a genotype-phenotype transcription of individuals, inspired by genetics. According to Wikipedia, "*Genotype is an organism's full hereditary information, even if not expressed. Phenotype is an organism's actual observed properties*" [84]. In terms of the algorithm, a *genotype* is typically a binary string which is used by all the operators. The string is then decoded to form an actual solution of a problem represented in its natural formalism. This represents the *phenotype* of an individual. This phenotype is evaluated by the fitness function to obtain an individual's fitness.

A flow chart illustrating the working of a generic GA is given in Figure 2.3. The squares indicate the application of the selection operators, while the hexagonal forms indicate the application of the variation operators. The GA is also presented as pseudo-code form in Algorithm 2.1. For more detail on GAs and their variations, the reader is referred to [20]. GAs are the most commonly applied metaheuristic to the GMS problem [9, 82, 83]. The binary nature of the problem (a unit either being in maintenance or not) facilitates an easy genotype-phenotype representation, although developing a suitable crossover operator is much more difficult.

Simulated annealing

Simulated annealing (SA) is a metaheuristic technique for solving combinatorial optimisation problems based on the physical phenomenon of annealing. Annealing is a process whereby a physical system is led to a low energy state by controlling its temperature [20]. It is explained

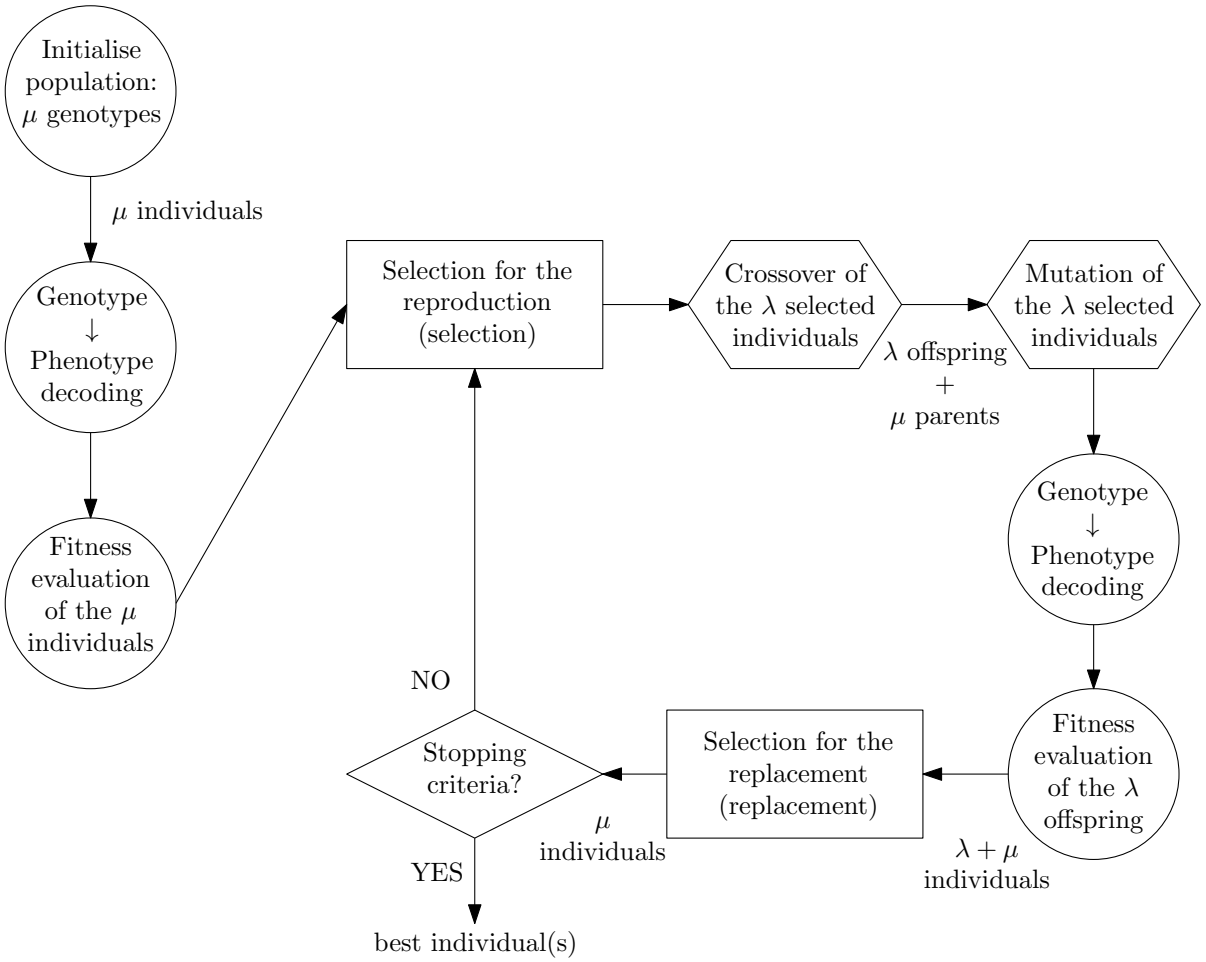


Figure 2.3: *Flow chart of a generic genetic algorithm.*

here in the context of strengthening materials (*e.g.* metals). The technique of annealing starts by heating a material to bestow a high energy to it. The material is then cooled slowly in stages by keeping the temperature constant during each stage for a sufficient duration. This controlled decrease in temperature leads to the material obtaining a stable solid state, corresponding to an absolute minimum energy configuration. In this state, the material becomes uniform in density and typically contains very few defects.

Simulated annealing may be used to solve combinatorial optimisation problems in a manner that is analogous to the process of annealing as described above. A control parameter is introduced to mimic the temperature of the system. This *temperature* controls the number of accessible energy states and should lead towards an optimal state when lowered gradually. The objective function in a minimisation problem corresponds to the free *energy* in the system, while a feasible solution to the problem corresponds to a certain state of the material. The final (possibly globally optimal) solution corresponds to the system being *frozen* in its ground state.

The SA algorithm is based on two results from statistical physics. Firstly, when *thermodynamic balance* is reached, the probability of the system having a given energy E, is proportional to the Boltzmann factor $\exp(-E/k_BT)$, where k_B denotes the Boltzmann constant corresponding to the material in question and T is the temperature in Kelvin. The energy states follow a Boltzmann distribution at the given temperature [20]. Secondly, the so-called *Metropolis algorithm* is utilised to simulate the evolution of a physical system towards its thermodynamic

Algorithm 2.1: Generic genetic algorithm outline

```
1  Initialise population as genotypes;
2  Decode genotypes to phenotypes;
3  Evaluate fitness of individuals;
4  while stopping criteria not met do
5  |  Select individuals for reproduction;
6  |  Apply crossover operator;
7  |  Apply mutation operator;
8  |  Decode genotypes to phenotypes;
9  |  Evaluate fitness of offspring;
10 |  Select individuals for replacement;
11 end
12 Solution ← best individual;
```

balance at a given temperature [20]: from a given configuration (feasible solution), the system experiences a small modification (*e.g.* elements of the solution are exchanged or relocated) — if the modification causes a decrease in energy (objective function value), the configuration is accepted with probability 1 (*i.e.* with certainty); however, an increase ΔE in energy is only accepted with a probability of $\exp(-\Delta E/T)$. This occasional increase in objective function value results in the system avoiding becoming trapped in a local minimum. Repeated iterations of the Metropolis algorithm gives rise to a sequence of configurations which constitutes a Markov chain (each configuration depends only on the previous configuration). If the chain is of *infinite* length, the system can *reach* thermodynamic balance at a given temperature, thereby leading to a Boltzmann distribution of the energy states at the given temperature. Practically, the word "infinite" above is substituted with the word *sufficient* and the word "reach" is substituted with the word *approach*.

The effect of the temperature is now apparent — at a high temperature, the factor $\exp(-\Delta E/T)$ is close to 1, causing the Metropolis algorithm to accept the majority of modifications (similar to a rondom walk), whereas a low temperature results in $\exp(-\Delta E/T)$ being close to 0, hence leading to the rejection of the majority of increasing energy (worsening objective function value) modifications. In this manner, the SA algorithm starts at a high temperature and considers as many solutions as possible in a bid to explore the solution space, after which the temperature is gradually lowered in order to converge to a configuration which gives a minimum objective value (local or possibly global).

The working of the SA technique is illustrated by means of a flow chart in Figure 2.4. It may be seen that the technique consists of an outer temperature loop and an inner Metropolis loop. A pseudo-code listing for the SA technique is given in Algorithm 2.2 and provides an outline for the implementation. There are different approaches in SA with respect to choosing the initial temperature, cooling schedule and termination criteria. More detail is available in [20]. Simulated annealing has been implemented successfully in a GMS context in [9] and is often used in *hybrid metaheuristic* techniques such as GA/SA combinations [15, 51, 63]. These hybrid techniques may lead to improved results.

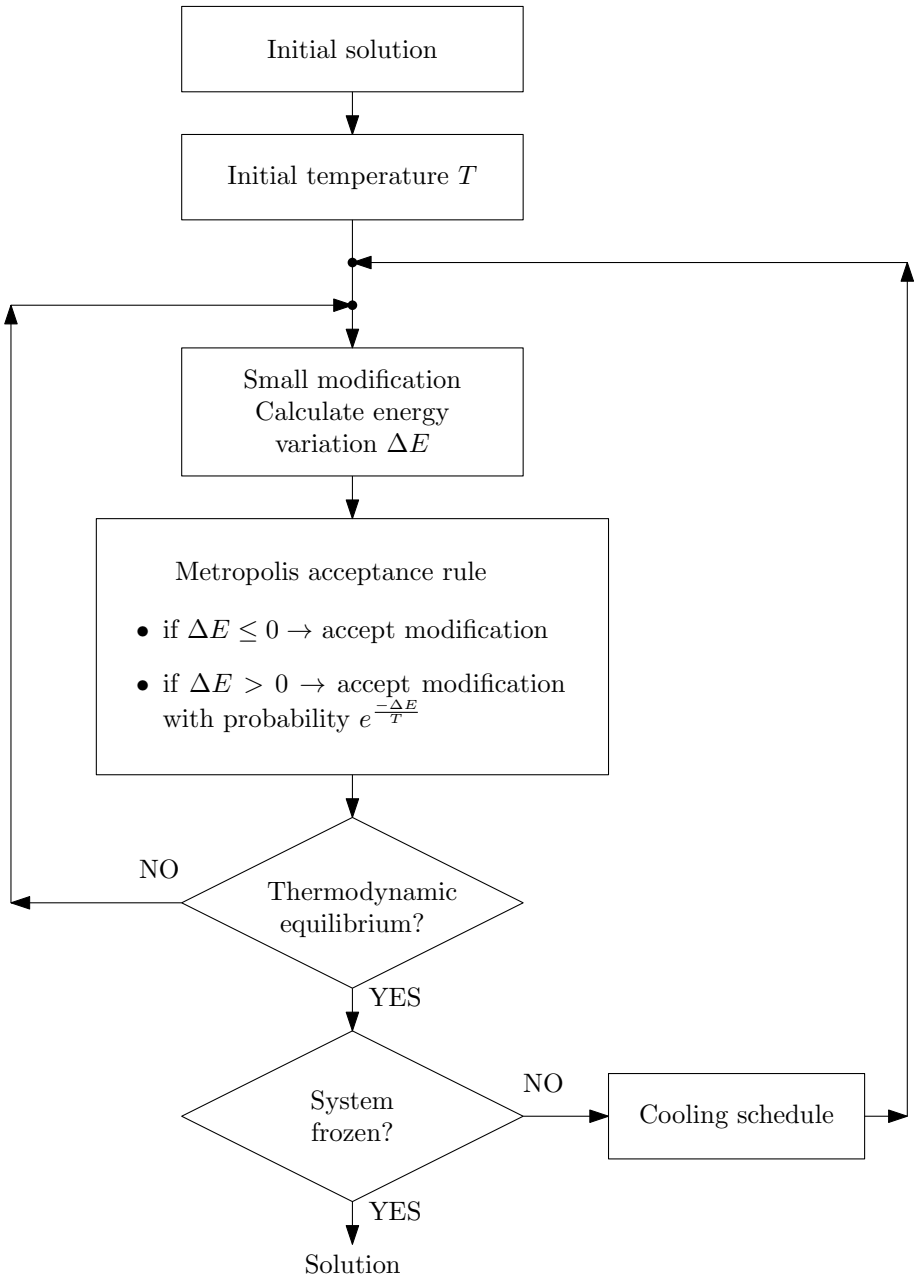


Figure 2.4: *Flow chart of the simulated annealing technique for a minimisation problem.*

Tabu search

Tabu search (TS) is a metaheuristic that falls within the category of local search methods. The main idea behind the algorithm is that it incorporates a memory structure in order to prohibit revisiting certain recent solutions or moves. Non-improving solutions are allowed during the search only if no improving solution is found in the neighbourhood of the current solution, thereby allowing the search to possibly escape from a local optima. The memory structure employed by the search ensures that it does not periodically return to the same local optima [20].

As mentioned, a TS explores the solutions in a neighbourhood of a current candidate solution, in search of a better solution. A *neighbourhood* N of a solution is obtained from a *move set* M.

Algorithm 2.2: Simulated annealing algorithm outline

```
Generate initial solution;
Determine initial temperature T;
while system not frozen do
    while thermodynamic balance not achieved do
        Apply small modification to solution;
        Calculate energy variation ΔE;
        if ΔE ≤ 0 then
            Accept modified solution;
        else
            Accept modified solution with probability e^(−ΔE/T);
        end
    end
    Decrease temperature T;
end
```

This set defines how elementary modifications should be made to a solution in order to generate solutions that are "close" to it, which constitutes the neighbourhood. Mathematically, if s is any feasible solution and S is the entire set of feasible solutions, then $N(s)$ is the neighbourhood of s and $N(s) \subseteq S$. An alternative approach, instead of using the solutions in $N(s)$, is to consider the move set of s. A *move* is a single modification of s. Expressing the neighbourhood in terms of the moves may reduce the size of the neighbourhood (*e.g.* candidate solutions in the form of permutations with a transposition neighbourhood[11]).

Generally, a TS does not necessarily have to evaluate the full neighbourhood. If $N(s)$ is large, a subset of $N(s)$ may be evaluated. One policy may be to randomly select a small number of solutions from $N(s)$. Another approach is to use a *candidate list* which keeps track of high quality moves. After the subset of neighbouring solutions has been calculated, the best solution therein is chosen as the new solution, provided it does not appear in the *tabu list*. The tabu list is the short term memory of the TS algorithm and has a specified size n, called the *tabu tenure*. The memory may be recency-based (the last n solutions or moves) or frequency-based (the n most frequently visited solutions or moves). The tabu list uses a *first-in-first-out* (FIFO) rule when resetting previous tabu solutions as non-tabu.

An *incumbent solution* keeps track of the best solution found thus far and the termination criterion is typically activated when the incumbent solution remains unchanged for a pre-specified number of iterations. Additional considerations for a TS are *aspiration criteria, intensification strategies* and *diversification strategies*. Aspiration criteria allow execution of a move that appears in the tabu list to be considered (*e.g.* if a move leads to a better solution than the incumbent). Intensification and diversification strategies may be implemented to equip the TS with long term memory. Both are frequency-based, with intensification strategies exploring promising areas in the solution space more in-depth, while diversification strategies steer the search towards areas seldomly visited, should the search start to stagnate [20].

The working of a simple TS algorithm is illustrated in the flow chart in Figure 2.5. Notice that the additional considerations, as mentioned above, are not included. The outline of the simple

[11] A transposition neighbourhood is obtained from a move set which interchanges two elements in a candidate solution (*e.g.* a transposition neighbour of the permutation 5812 may be 5218 because the elements in positions two and four interchanged).

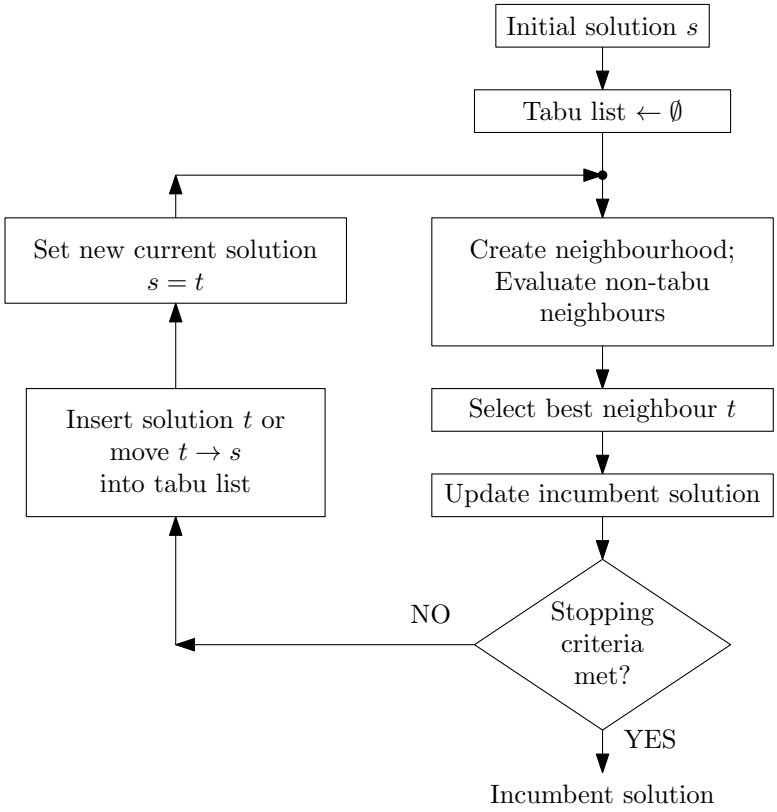


Figure 2.5: *Flow chart of a simple tabu search algorithm.*

tabu search is given in pseudo-code form in Algorithm 2.3. A more detailed description of the TS, including implementation issues and a worked example, is presented in [20]. The TS is very successful in its application to the *travelling salesman problem* (TSP) and has been applied traditionally to scheduling and routing problems. It has also been utilised in the GMS context in [23].

Algorithm 2.3: Simple tabu search algorithm outline

```
1  Generate initial solution s;
2  Tabu list ← ∅;
3  while stopping criteria not met do
4  |  Generate neighbourhood N(s);
5  |  Evaluate non-tabu elements of N(s);
6  |  Select best neighbour t;
7  |  Update incumbent solution;
8  |  Update tabu list;
9  |  s ← t
10 end
```

Ant colony optimisation

Ant colony optimisation (ACO) techniques are quite new to the study of difficult optimisation problems. It is inspired by the behaviour of ants, always seemingly to be able to find the

shortest path between a food source and their anthill/nest. Moreover, all the ants follow this path. The key to the ants' success lie in their indirect communication with one another by means of dynamic modifications of their environment, which is known as *stygmergy* [20].

The ants naturally optimise their routes by means of pheremone trails. Ants use volatile substances called *pheromones* to communicate and these pheromones may be deposited on the ground to form an odorous trail which other ants may follow. Pheromone levels, however, *evaporate* over time, unless other ants deposit more of it along the same path. This is exactly where the optimisation occurs. Ants typically follow paths along which pheromone levels are higher with a larger probability, and since the quantity of pheromone on a shorter path will be slighty more significant than on a longer path due to evaporation, ants will travel along the shorter paths more frequently. As a result, the pheromone deposits along these shorter paths will increase due to more ants travelling along the paths, while the pheromone levels will decrease along longer paths. Therefore, the system reinforces itself until all the ants follow a single path.

The original ant colony algorithm, the "Ant System," was specifically designed for the travelling salesman problem [20]. It is easy to see the analogy — the cities are different foodsources and the ants have to find the shortest path connecting all the cities, with each city being visited only once. Since then, many variations and improvements have been made on the algorithm, and it has been applied successfully to problems such as scheduling, routing, assignment and graph colouring problems. In theory, any problem that may be represented by a connected graph with an objective of minimising a path in the graph, may be solved by the method of ACO.

In terms of an algorithm, the *ant colony system* (ACS) algorithm is presented here, as it was originally introduced as an improvement on the Ant System. Consider a complete graph $G(N, A)$ where the cities are the N nodes and the paths between cities are the A edges. Each ant $k \in \{1, \ldots, m\}$ traverses the graph and builds a path at each iteration $t \in \{1, \ldots, t_{max}\}$ of the algorithm. For each ant, the path between city i and city j depends on the list of remaining cities J_i^k when ant k is at city i, the *visibility* of each city $\eta_{ij} = \frac{1}{d_{ij}}$, and the quantity of pheromone deposited on each edge, called the *trail intensity* $\tau_{ij}(t)$. An ant k at city i will travel to city j according to the rule

$$j = \begin{cases} \arg\max_{u \in J_i^k} \left[(\eta_{iJ})^\beta (\tau_{iu}(t)) \right] & \text{if } q \leq q_0 \\ J & \text{if } q > q_0 \end{cases} \tag{2.50}$$

where $q_0 \in [0, 1]$ is a parameter which defines the balance between *diversification* and *intensification*, q is a random variable uniformly distributed on the interval [0,1], and $J \in J_i^k$ is a city randomly selected according to the probability

$$p_{iJ}^k(t) = \frac{(\eta_{iJ})^\beta (\tau_{iJ}(t))}{\sum_{l \in J_i^k} (\eta_{il})^\beta (\tau_{il}(t))}. \tag{2.51}$$

If $q > q_0$, the system leans toward a diversification and if $q \leq q_0$, the system leans towards an intensification. In fact, for $q \leq q_0$, the algorithm exploits the information collected by the system (pheromone trails) more and it cannot choose a hitherto unexplored path [20].

The pheromone levels on the trails are updated on both a local and a global level. When an ant traverses an edge (i, j), the pheromone level on that edge is locally updated immediately according to the rule

$$\tau_{ij}(t + 1) = (1 - \rho)\tau_{ij}(t) + \rho\tau_0, \tag{2.52}$$

where τ_0 is the initial pheromone level of the trail and $\rho \in (0, 1)$ a parameter at which the pheromone level drops. Diversification takes place because only the visited edges decrease in

pheromone quantity. The global update occurs at the end of each iteration, according to the rule

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t), \tag{2.53}$$

where the edge (i, j) only belongs to the best tour found T^+, which has length L^+, and where $\Delta\tau_{ij}(t) = \frac{1}{L^+}$. Only the best trail is updated, thus promoting an intensification of the best solution.

As the best tour found is constantly updated, the algorithm produces that tour as the answer when the termination criteria are met. In this case, the termination criterion is a pre-specified number of iterations.

A flow chart of a simple ant colony system algorithm is presented in Figure 2.6. A pseudo-code listing for an ACS algorithm is presented in Algorithm 2.4. There are many variations of ant colony algorithms based on elitism, different pheromone update rules and a hybridisation with local search algorithms. These variations and more detailed information regarding ACO may be found in [20]. Since the GMS problem may be represented as a graph, ant colony optimisation algorithms have been implemented in [31, 32] to good effect.

Algorithm 2.4: Simple ant colony system algorithm outline

```
Calculate initial pheremone level;
for t ← 1 to t_max do
    for each ant k ← 1 to m do
        Choose a node at random;
        while node i not visited do
            Choose a node j ∈ J_i^k according to the rule of transition;
            Apply local pheromone update on edge (i, j);
        end
    end
    Update best tour found;
    Apply global pheromone update on the best trail;
end
```

2.4.5 Fuzzy systems

It has already been established that the GMS problem may have multiple conflicting objectives. Furthermore, there is an inherent uncertainty in GMS as a result of uncertain load demand, maintenance windows, manpower and other resources. These constraints are not necessarily as rigid as conventional deterministic techniques treat them because of assumptions and simplifications that are typically made to treat these parameters in a deterministic manner. By introducing fuzzy sets, both the objective function and constraint concerns are addressed since a fuzzy environment is able to deal with multiple objectives and the uncertainties in many of the constraints [17].

According to [57], *"Fuzzy-set theory is a mathematical theory designed to model the vagueness or imprecision of human cognitive processes."* Fuzzy-set theory (or fuzzy logic) is concerned with the *degree of truth* according to which an outcome belongs to a certain category. It should not be confused with the degree of likelihood that the outcome will be observed. The building block of fuzzy logic is the membership function of a fuzzy set or fuzzy number. Consider the

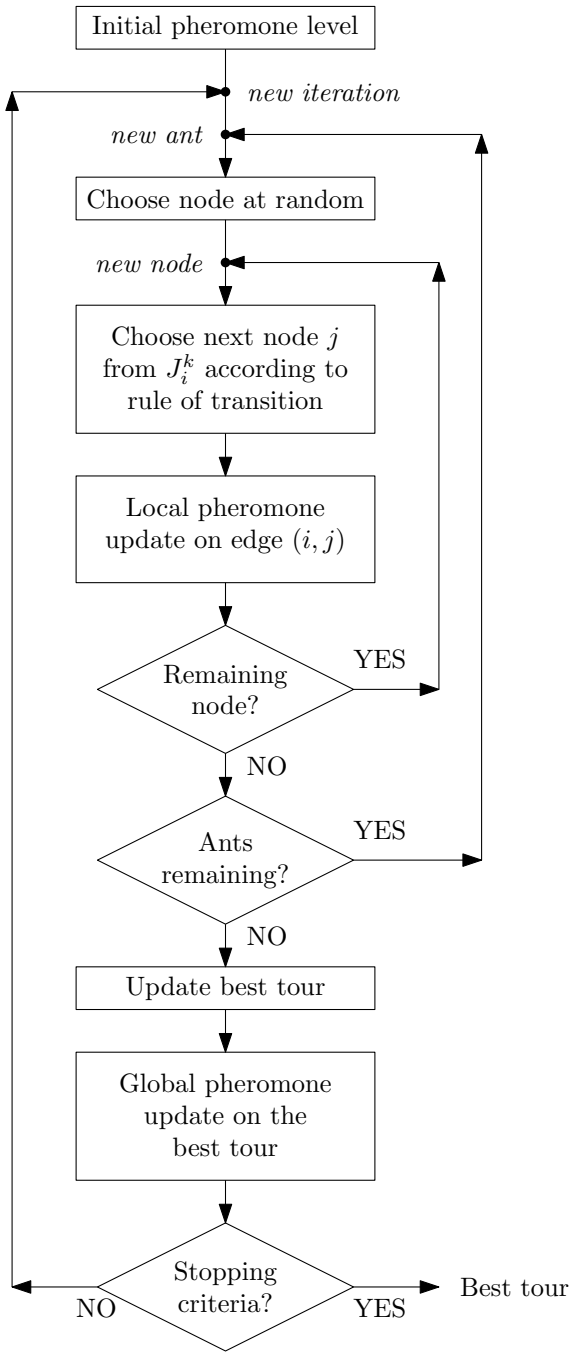


Figure 2.6: *Flow chart of a simple ant colony system algorithm.*

statement: *the element x belongs to the set A*. In classical set theory, the characteristic function χ_A of a set A in the universe of discourse U is defined by

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A. \end{cases} \tag{2.54}$$

Therefore the characteristic function has only a true or false answer to the statement, for each element in U. In this context the set A is referred to as a crisp set. However, this concept may be extended towards a fuzzy set A where the statement x *belongs to* A is not necessarily true

or false only. The fuzzy set A has a *membership function* μ_A defined as

$$0 \leq \mu_A(x) \leq 1 \qquad \text{for any } x \in U, \tag{2.55}$$

where the *truth value* $\mu_A(x)$ represents the degree of truth, subjectively assigned by a human referee, of the statement [57]. An interpretation of the truth value may be that it is the fraction of a sufficiently large number of referees agreeing with the statement x *belongs to* A.

In order to define a fuzzy number, the following definition is necessary: the α-level subset of a fuzzy set A is the crisp set of elements in U such that $\mu_A(x) \geq \alpha$. Now, a fuzzy number is a fuzzy set A in the one-dimensional universe of discourse U such that

(a) the α-level subsets are intervals monotonously shrinking as $\alpha \to 1$, and

(b) there is at least one $x \in U$ such that $\mu_A(x) = 1$.

A fuzzy number generally has a membership function associated with it which increases monotonically from 0 to 1 on the left-hand side, has a single top or plateau at 1 therafter, and finally decreases monotonically to 0 on the right-hand side. As an example, consider a triangular fuzzy number $\tilde{a}$ — that is, the number has a triangular membership function. The fuzzy number $\tilde{a}$ is characterised by three parameters, namely the lower value a_l, the modal value a_m and the upper value a_u. Figure 2.7 illustrates this triangular membership function.

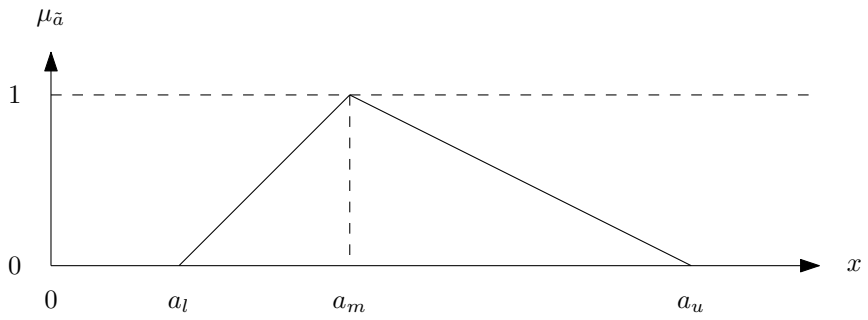


Figure 2.7: *Membership function of a triangular fuzzy number.*

A fuzzy approach towards GMS is taken in [36] by *fuzzifying* the multiple objectives and soft constraints. This means that a membership function is obtained for each of the above. The problem is then typically solved by means of a fuzzy dynamic programming technique as the problem is now too difficult for conventional mathematical algorithms to solve. A combination of metaheuristics and a fuzzy approach is adopted in [14, 38, 39]. These combinations typically fuzzify either the objective function(s) for evaluation purposes or some of the constraints that may have an inherent uncertainty associated with them, while the metaheuristic obtains different solutions. Fuzzy logic seems to be a promising approach to GMS [1] as many of the uncertainties in GMS can be addressed, as well as multiobjective goals.

2.4.6 Knowledge-based/expert systems

Power systems have been managed for many years and invaluable experience is locked inside the field experts charged with this task. Their knowledge may be included in an *expert system* (ES) in order to attempt solving the GMS problem. It is imperative that the system should

have powerful rules in its knowledge base [1] in order to ensure that correct and logical decisions are made.

As an example, an expert system for GMS has been presented in [52] and applied to the *Taiwan Power Company's* (TPC's) network. Its developement is solely based on TPC constraints and the solution does not consider a generalised application to other systems. After the ES was applied to the TPC case study, the resulting GMS solution closely resembled the nature of historical decisions and it required only 30 minutes for execution.

In their expert system, the objective function varies as the power system conditions change. The ES calculates an operation index, based on expert experience, so as to select a suitable objective function under different load conditions and constraints. There are four *conditions* that the ES has to consider, depending on the value of the operation index value. Within each condition there are *steps* and *rules* to follow in order to satisfy the system demands and possibly obtain a new operation index. A branch-and-bound or dynamic programming technique is used within the ES during the optimisation process, depending on the objective function. The structure of the ES for solving the GMS problem is illustrated in Figure 2.8.

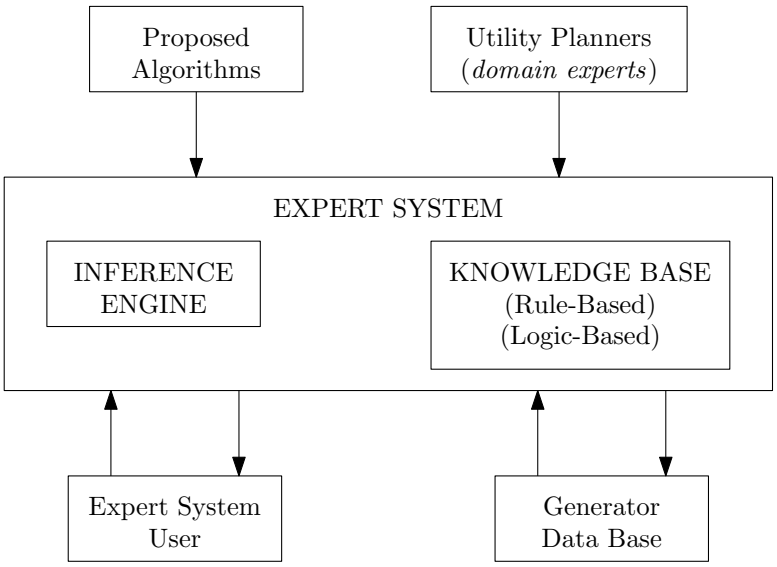


Figure 2.8: *Expert system structure [52].*

Expert systems have not been applied widely in the GMS context. However, some advances have been made (for example, by combining expert knowledge with a fuzzy environment [75] and probabilistic rules [6]).

2.5 Chapter summary

The aim in this chapter was to provide the reader with the necessary background in terms of the rich array of facets of the GMS problem in the literature. Having done so, one may proceed to formulate a model for a new case of the GMS problem with the necessary awareness in respect of previous approaches towards solving the general problem.

The general GMS model considerations were stated in §2.1. These considerations form the basis of providing the model formulations from the literature in §2.2. Since there has been a

considerable number of different approaches toward tackling the GMS problem, the formulations have been subdivided into individual constraint and objective function formulations.

In §2.3 some extensions to the GMS problem were briefly discussed. An assortment of different solution techniques applicable to the GMS problem was presented in §2.4. Each technique was explained in moderate detail and in some cases were accompanied by an example in the literature. Where algorithms were introduced, basic pseudo-code listings were provided.

CHAPTER 3

Mathematical Problem Formulation

Contents

A brief description of where the *generator maintenance scheduling* (GMS) problem fits into the operations scheduling of a power system is presented in this chapter. A number of necessary assumptions are presented in order to formulate the GMS problem mathematically. Several mathematical formulations are presented, based on different objective functions.

3.1 The GMS problem in context

Before a mathematical model may be developed for any problem, certain assumptions are necessary in order to simplify the problem and/or address uncertainties. Maintenance scheduling is only one component of the overall operations scheduling in a power system. A basic diagram of the components of operations scheduling in a power system, and their interdependencies, may be found in Figure 3.1. The outer blocks, with arrows pointing toward the operations scheduling

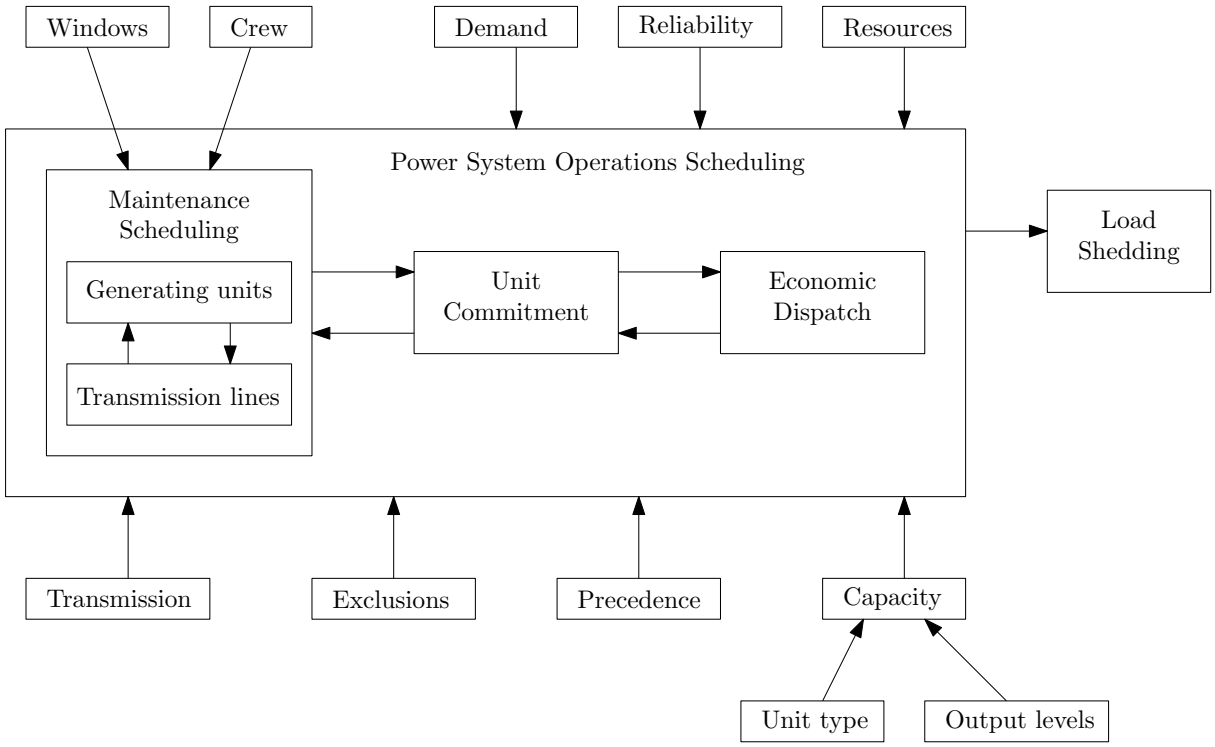


Figure 3.1: *Dependency diagram for operation scheduling in a power system.*

block, represent various constraints that may influence operations scheduling. Within power system operations scheduling, there are generally three components: maintenance scheduling, unit commitment and economic dispatch. The generating units and transmission lines of a power system are considered within the maintenance scheduling component. Should the process of operations scheduling fail, the power system typically has to resort to load shedding in order to compensate for any shortage in electricity supply.

3.2 Problem assumptions

Based on the context of the GMS problem within the broader operations scheduling problem in a power system, a number of assumptions are made in this section to reduce the complexity of the problem. The aim is to arrive at managable levels of dependencies such that the GMS problem may be modelled effectively, without significant short-comings in the broader operations scheduling problem in which it resides. The following general assumptions are made in order to develop a model.

3.2.1 Unit commitment

The problem of determining which generating units should be in service during each time interval of the planning period in order to meet the system demand, is referred to as *unit commitment* (UC) [74]. It is performed on a small time scale, typically on a day-to-day or week-to-week basis, and is based on advanced load demand forecasting. The UC problem is thus very similar to the GMS problem. However, it functions on a much smaller time scale (planning period) and may have different constraint sets. For example, in the UC problem, the start-up and shut-down

times of generating units are very important, whereas in the GMS environment, these start-up and shut-down times are significantly less than a single time unit of the problem and, as such, are unimportant.

In short, the UC problem determines which units should be in service during each time period over the short-term, while the GMS problem determines which units should be in maintenance during what time period over the mid- to long-term. Clearly, these two problems affect each other and may therefore be modelled as a single problem. Should this approach be taken, an integrated schedule is obtained. However, there are drawbacks to such an integrated approach. Due to the large planning period of GMS, less accurate load demand forecasts have to be used and the long-term UC schedule may be too inaccurate for a given time period. Furthermore, for UC a very small time unit has to be used in the problem which, in combination with a large planning period, causes the dimensionality of the problem to escalate beyond practicality.

If one adopts a separate segregated modelling approach, the UC problem has no influence on the GMS problem and the maintenance schedule obtained is then typically used as a constraint in the UC problem — an availability constraint. The GMS problem is not significantly affected by excluding the UC problem, because GMS lies much higher in the temporal hierarchy of operations scheduling. Therefore, it is assumed that UC has no influence on the GMS problem and hence may be excluded from the problem. The UC problem is therefore considered to be a separate problem for the purposes of this thesis.

3.2.2 Economic dispatch

The *economic dispatch* (ED) problem is the allocation of the load demand among the generating units that are in service during each time interval, as determined by the UC problem. ED focuses on the cost-efficiencies of the available generating units and is determined for short time periods. The UC and ED problems are interdependent and may be modelled as a single problem. Therefore, ED has an indirect effect on GMS. However, since it is assumed that UC and GMS are separate problems, the ED problem may be excluded from the model.

3.2.3 Transmission line maintenance

Like generating units, transmission lines and substations must also undergo regular maintenance. The scheduling of such transmission maintenance is contained in the "maintenance scheduling" component in Figure 3.1. If a generating unit is in maintenance, the transmission lines connected to it is not in use and it may provide an opportunity to perform maintenance on these lines, and *vice versa*. Therefore, generator and transmission line maintenance may be modelled as a single problem, because of this interdepency between the two problems. Transmission line maintenance does not, however, fall within the scope of this thesis. As such, it is considered to be a separate problem, not influencing generator maintenance, and is excluded from the GMS problem formulations that follow.

3.2.4 Transmission constraints

There are two types of transmission constraints. The first type concerns the transmission capabilities of the electrical network (*e.g.* bus loads, voltage, *etc.*). A transmission network is designed to accomodate the power flow of a system within its normal operating conditions. Since it has already been assumed that no transmission line maintenance occurs, one may also

assume that the transmission network will always function at its proper capability. As such, any transmission constraints which ensure that the power flow in the system is maintained within specified limits, are assumed to be satisfied at all times and may therefore be excluded from the GMS problem formulations.

The second type of transmission constraint involves the physical transmission network. Since power stations typically serve multiple cities and towns, the transmission network ensures that the electricity from a power station is sent to the respective geographic regions in its service area. However, should a power station be offline (in this case in maintenance), electricity is provided to the affected geographic regions from other power stations through the transmission lines. Should a line failure occur under these circumstances, the result would be that some areas are left without any electricity and cause the system demand to be not met. To avoid such infeasibilities, it is assumed that there will always be full connectivity in the transmission network at all times. This assumption is very reasonable because line failures are random events of short duration and should not influence long term maintenance scheduling of generating units.

3.2.5 Resources

The most important resource pertaining to electricity generation is the fuel. Different types of units require different kinds of fuels *e.g.* coal, natural gas, water or uranium. Since a unit cannot produce electricity without sufficient fuel, it is safe to assume that having sufficient fuel available at a power station is of the highest priority to a power utility. Therefore, it is assumed that a generating unit will always have a sufficient fuel stockpile in order to enable it to generate electricity at a given capacity. This may be an unrealistic assumption for renewable energy units such as hydro, solar or wind generating units because their fuel sources are less controllable. However, the assumption is made across the board for the sake of simplicity.

Resources required specifically for maintenance during any time period, are assumed to be limited to crew availability. Since the GMS problem may be viewed as *planned, preventative* maintenance, all preparations are made well in advance of a unit's maintenance window. As a result, the necessary spare parts for a unit, maintenance equipment and any other resources required to complete the maintenance process, are assumed to be available. It may therefore be assumed that all these resources, necessary for generator maintenance, are available and within their limits during any time period.

3.2.6 Load shedding

It is very possible for a country/region to have a higher power demand than supply at a given time. There may be different factors that cause such a shortage, but ultimately the operations scheduling of the power system fails. One of the strategies that a power electrical utility may use in such a scenario, usually as a last resort [71], is load shedding. This strategy consists of causing deliberate power outages to smaller regions on a rotating basis, thereby avoiding a total blackout of the entire power system. These power outages effectively reduce the load demand to a level less than the power supply. The load shedding problem is governed by its own set of constraints and objectives, and is therefore a problem on its own.

Load shedding may last for days, weeks or even months, depending on the power system and its demand. As a result, any unit that is in maintenance during times of load shedding will have a significant effect on the system. It may be more beneficial to postpone any generator maintenance until the load shedding is discontinued. Should this not be possible, an alternative

approach may be to schedule the maintenance such that it has the least negative effect on the power system.

The intricacies and unpredictability of load shedding make it very complex to incorporate such requirements along with the problem of GMS in a single model. For the sake of simplicity, it is assumed that load shedding will not occur and may therefore be excluded from the GMS problem formulations.

3.2.7 Generating capacity

The generating capacity of a generating unit may fluctuate over time and decrease as the unit ages, regardless of adequate levels of maintenance throughout its lifetime. The fluctuation may be due to the fuel quality [28] and auxiliary power consumption of the unit. As a result, all generating units provide less power to the grid than their nominal installed capacities. Furthermore, generating units typically do not have to operate at a 100% output level — the longevity of a unit may be increased if it is operated at lower levels. These lower output levels may especially occur during times of low demand and is typically part of the ED problem. Lastly, the unit type also has a significant effect on its generating capacity. Renewable energy generating units vary much more in output levels than conventional generating units, due to the variability in fuel supply.

Ideally, the generating capacity of a generating unit should be treated stochastically. However, within the scope of this thesis, a deterministic approach is followed. Therefore it is assumed that the generating capacity of each unit is fixed for a given time period. This assumption does not exclude the use of a probabilistic simulation of the power system to calculate expected values for the generating capacities that may be used as input for the GMS model. Variability may be introduced by fixing different capacities during different time periods (*e.g.* a unit may have a capacity of 200 MW during time period 1, but 150 MW during time period 2, *etc.*). How these capacities are calculated, be it via a probabilistic simulation or some other method, is up to the power utility, as the model only depends on deterministic parameters. By using deterministic parameters, discrete optimisation techniques may be utilised to solve the GMS problem. Monte Carlo simulations with respect to the generating capacities may be used to solve a stochastic version of the GMS problem, but are very time-consuming and as such have not been considered here.

3.2.8 Precedence constraints

In order to simplify the model, precedence constraints are not considered in this thesis. These constraints are not very common in practice and are therefore excluded. The assumption still allows one to include a certain degree of precedence requirements (should it be required). The maintenance windows of units may be specified such that certain windows are explicitly earlier than others, thereby enforcing precedence constraints. However, this approach is only successful if the maintenance windows do not overlap.

3.3 A simple GMS model

In any modelling environment, it is good practice to start with the simplest form of the problem. This observation may be motivated as follows. Firstly, a simpler form of the problem is much

easier to model, because less detail and fewer constraints have to be taken into account. Secondly, a successful simpler model provides a solid foundation on which to build more complexity. A simple model also provides an effective means for understanding the underlying structure of the problem and typically improves one's ability to draw conclusions with respect to the behaviour of the problem variables in responce to variation of its parameters. As such, a simple model for the GMS problem is provided as a basis in this section.

The GMS problem has an inherent structure which calls naturally for a mathematical programming modelling approach: a schedule must be obtained that optimises some goal, subject to restrictions or constraints on that schedule. To this end, the mathematical model of scheduling the generator maintenance in a power network is formulated as a mathematical program below.

3.3.1 Model constraints

As stated in §2.2, even the simplest GMS model must include a set of maintenance window constraints to specify when each unit may be scheduled for maintenance, as well as load constraints which ensure that the load demand is met. The decision variables are chosen to represent the starting time of maintenance of each unit. Define $x_{i,j}$ as a binary decision variable taking the value 1 if maintenance of unit i commences during time period j, and zero otherwise. If there are n generating units and m time periods in the planning horizon, and $\mathcal{I} = \{1, \ldots, n\}$ is the set of generating unit indices and $\mathcal{J} = \{1, \ldots, m\}$ is the set of time period indices, then $i \in \mathcal{I}$ and $j \in \mathcal{J}$.

A maintenance window is defined by two time periods. Let e_i and ℓ_i denote the earliest and latest time periods, repectively, during which maintenance of generating unit i may start. If d_i is the maintenance period duration of generating unit i, then $e_i, \ell_i \in \{1, \ldots, m - d_i\}$. Since maintenance is allowed only once during a window, the maintenance window constraint set may be formulated mathematically by requiring that

$$\sum_{j=e_i}^{\ell_i} x_{i,j} = 1, \qquad i \in \mathcal{I}. \tag{3.1}$$

It is known that a unit will not be in maintenance outside its maintenance window. To this effect, one may include the explicit constraints

$$x_{i,j} = 0, \qquad j < e_i \text{ or } j > \ell_i, \qquad i \in \mathcal{I}, \tag{3.2}$$

which will reduce the number of decision variables [21]. Such an inclusion decreases the solution space of the problem and may lead to shorter solution times. The constraints are not mandatory, since the objective function is expected to drive a good solution to the point where the decision variables take the value zero outside the maintenance window, but it is recommended.

The load constraints restrict the maintenance schedule so that the total demand for electricity is at least met during every time period. Therefore, all the generating units less those in maintenance should produce enough power to meet the forecasted demand. Let $g_{i,j}$ denote the power generating capacity of unit i during time period j, and let $g'_{p,i,j}$ denote the power generating capacity lost during time period j if maintenance of unit i commenced at time period p. These parameters are calculated by

$$g'_{p,i,j} = \begin{cases} g_{i,j} & \text{if } j - p < d_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

If the load demand at time period j is denoted by D_j, then the relevant constraint set may be formulated as

$$\sum_{i=1}^{n} g_{i,j} - \sum_{i=1}^{n} \sum_{p=1}^{j} g'_{p,i,j} x_{i,p} \geq D_j, \qquad j \in \mathcal{J}. \tag{3.4}$$

With constraint sets (3.1) and (3.4), the requirements for the GMS problem are stated in its most simplistic form. There is another constraint set which may be added without increasing the complexity, namely the reliability constraint set. Reliability is obtained by specifying a reserve or safety margin that needs to be adhered to. Let S denote the safety margin as a proportion of the demand for the power system. This constraint may be incorporated into the load constraints without affecting the model complexity, because the safety margin only induces a higher demand level. The resulting constraint set is

$$\sum_{i=1}^{n} g_{i,j} - \sum_{i=1}^{n} \sum_{p=1}^{j} g'_{p,i,j} x_{i,p} \geq D_j(1+S), \qquad j \in \mathcal{J} \tag{3.5}$$

and replaces (3.4) in the formulation.

An important quantity for an electrical utility is the amount of reserve power. Although a safety margin has been introduced to compensate for variations in the actual demand or unplanned unit outages, the actual reserve levels at any given time hold significant value. To this end, the reserve level variables r_j are defined as the unused power during time period j, excluding the safety margin. Of course, a separate variable is not necessary for the incorporation of reserve levels into the model, because the reserve levels may easily be calculated as the available power less the demand. However, for ease of use, reserve level variables are nevertheless used in the model. Constraint set (3.5) changes from inequalities to a set of equalities of the form

$$\sum_{i=1}^{n} g_{i,j} - \sum_{i=1}^{n} \sum_{p=1}^{j} g'_{p,i,j} x_{i,p} = D_j(1+S) + r_j, \qquad j \in \mathcal{J}, \tag{3.6}$$

where it is required that $r_j \geq 0$ for all $j \in \mathcal{J}$. To obtain the actual reserve levels at any given time period, calculating $D_j S + r_j$ is now much simpler than calculating the quantity

$$\sum_{i=1}^{n} g_{i,j} - \sum_{i=1}^{n} \sum_{p=1}^{j} g'_{p,i,j} x_{i,p} - D_j. \tag{3.7}$$

Lastly, constraint sets that specify the nature of the variables are given by

$$x_{i,j} \in \{0,1\}, \qquad i \in \mathcal{I},\ j \in \mathcal{J} \tag{3.8}$$

$$r_j \geq 0, \qquad j \in \mathcal{J}, \tag{3.9}$$

as mentioned above.

To summarise, the simple GMS model presented here contains the constraint sets (3.1), (3.2), (3.6), (3.8) and (3.9) in its mathematical program formulation. It is thus further assumed that no exclusions are present and that there are always sufficient maintenance crews available. The GMS problem has therefore been simplified from being a component in operations scheduling, as presented in Figure 3.1, to the singular problem presented in Figure 3.2. The remaining task is to define a suitable objective function.

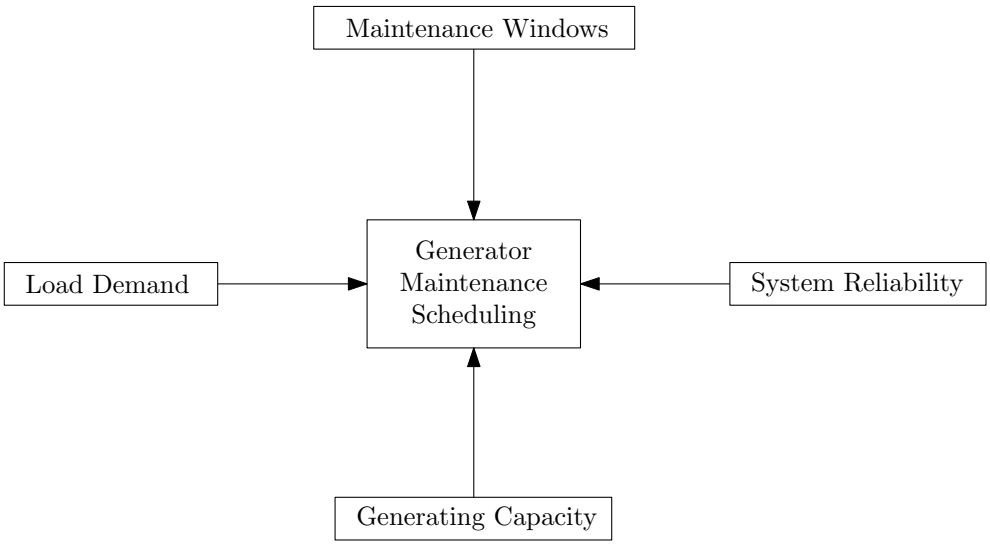


Figure 3.2: *Dependency diagram for a simple GMS problem.*

3.3.2 The objective function

The choice of an optimality criterion lies between convenience, economic and reliability considerations, or a combination of these, as stated in §2.1.4. In keeping with the notion of a simple model, the optimality criterion will be a single objective. In terms of priority in a single objective environment, an economic or reliability criterion significantly outweighs a convenience criterion. The risk of extreme behaviour within a schedule is too great if a schedule is computed solely on the basis of convenience (*e.g.* too expensive or reserve levels reaching a minimum). Therefore, convenience will not be considered as an objective for the simple GMS model.

The choice between economic and reliability considerations is more difficult. Typically, economic considerations include production costs (which may consist of fuel costs, salaries/wages, start-up and shut-down costs, *etc.*) and maintenance costs (which may include wages, replacement parts, *etc.*). All of these costs may vary over some or even over all the generating units and, as such, combined with the potentially high variability in some of these costs (*e.g.* fuel prices), the necessary problem data may be very difficult to obtain. In strong contrast to this observation, reliability considerations typically rely on the power reserve levels — a quantity that is very easily obtainable. Ultimately, the choice depends on the power utility — which consideration is more important in their strategy or business philosophy.

Since the GMS problem is investigated here from a South African viewpoint (although not restricted to such a point of view), the views of Eskom carry significant weight. According to [61], Eskom considers reliability considerations much more important than economic considerations for the GMS problem. This may be due to the fact that South Africa still has a regulated electric power market, with Eskom being a state-owned enterprise providing approximately 95% of the country's electricity. One of Eskom's strategic objectives is "Ensuring reliable supply of electricity to all South Africans" [28]. In light of this, and along with the possible data complications mentioned in the previous paragraph, the optimality criterion is chosen as reliability for the purposes of this thesis.

As stated in §2.1.4, the reliability objective is commonly chosen as levelling the reserve load over the planning horizon; a view shared by Eskom [61]. This may be achieved in a number of ways. The typical method is to minimise the sum of squares of the reserves. However, this method

results in a nonlinear (quadratic) objective function. An attempt is made in this section to derive a linear objective function yielding similar results, from this quadratic objective function. A method using the sum of over- and underachievements from the mean reserve level is presented; which is analogous to the sum of the absolute differences between the reserve levels and the mean reserve.

Sum of squares approach

The sum of squares of the reserves should not be confused with the statistical criterion of the sum of squares, perhaps more appropriately called the *sum of squared differences*. The objective function for the simple GMS model, may be expressed mathematically as

$$\sum_{j=1}^{m}(D_jS + r_j)^2, \tag{3.10}$$

whereas the *sum of squared differences* would be given by

$$\sum_{j=1}^{m}(D_jS + r_j - \overline{r})^2, \tag{3.11}$$

where $\overline{r}$ denotes the mean reserve level over the planning horizon. If one adopts the sum of squares approach, a solution is obtained for which the reserve levels are close to one another in value during all the time periods, thereby producing a level reserve. The main advantage of using a sum of squares approach is that outliers are penalised more severely and thus add a much larger value to the objective.

Therefore, the first GMS problem formulation to be considered in this thesis is given by the *mixed-integer quadratic program* (MIQP) in which the objective is to

$$\text{minimise} \qquad \sum_{j=1}^{m}(D_jS + r_j)^2$$

subject to the constraints

$$\left.\begin{aligned} \sum_{j=e_i}^{\ell_i} x_{i,j} &= 1, && i \in \mathcal{I} \\ x_{i,j} &= 0, && j < e_i \text{ or } j > \ell_i,\ i \in \mathcal{I} \\ \sum_{i=1}^{n} g_{i,j} - \sum_{i=1}^{n}\sum_{p=1}^{j} g'_{p,i,j}x_{i,p} &= D_j(1+S) + r_j, && j \in \mathcal{J} \\ x_{i,j} &\in \{0,1\}, && i \in \mathcal{I},\ j \in \mathcal{J} \\ r_j &\geq 0, && j \in \mathcal{J}. \end{aligned}\right\} \tag{3.12}$$

Sum of absolute differences approach

Unlike the sum of squares approach, using absolute values requires some baseline to level the reserve. The sum of squares of the reserves naturally levels the reserves towards the mean reserve without actually calculating that mean. However, the same is not necessarily true when

using absolute values instead of squares. If one only uses the absolute differences of consecutive reserve levels, one might find a solution which forms hills and valleys in the reserve. To rectify this situation, one should use the absolute differences between each reserve level and every other reserve level. This, however, leads to a very inefficient objective function. Instead, one may follow the "statistical" route, mentioned above. This approach measures the absolute difference between the reserve level during each time period and mean reserve. In this case the objective may be to

$$\text{minimise} \qquad \sum_{j=1}^{m} |D_j S + r_j - \overline{r}|. \tag{3.13}$$

By minimising the above sum of absolute differences, the solution is steered towards reserve levels which are close to the mean reserve (and one another), thus inducing a level reserve. Unfortunately in this case, outliers are penalised with the same severity as any other values. The additional constraint

$$\overline{r} = \frac{1}{m} \sum_{j=1}^{m} (D_j S + r_j) \tag{3.14}$$

should also be added to the formulation in order to define the mean reserve level.

Therefore, the second GMS problem formulation to be considered in this thesis is given by the *mixed-integer nonlinear program* (MINP) in which the objective is to

$$\text{minimise} \qquad \sum_{j=1}^{m} |D_j S + r_j - \overline{r}|$$

subject to the constraints

$$\left.\begin{aligned} \overline{r} &= \frac{1}{m} \sum_{j=1}^{m} D_j S + r_j, && \\ \sum_{j=e_i}^{\ell_i} x_{i,j} &= 1, && i \in \mathcal{I} \\ x_{i,j} &= 0, && j < e_i \text{ or } j > \ell_i,\ i \in \mathcal{I} \\ \sum_{i=1}^{n} g_{i,j} - \sum_{i=1}^{n} \sum_{p=1}^{j} g'_{p,i,j} x_{i,p} &= D_j(1+S) + r_j, && j \in \mathcal{J} \\ x_{i,j} &\in \{0,1\}, && i \in \mathcal{I},\ j \in \mathcal{J} \\ r_j &\geq 0, && j \in \mathcal{J}. \end{aligned}\right\} \tag{3.15}$$

Sum of over- and underachievements approach

Adopting the sum of absolute differences approach above may be viewed as an intermediate step towards formulating a linear objective function which attempts to level the reserve. To this effect, the same principle may be applied — the difference between the reserve level and mean reserve during each time period is calculated. Negative differences cause trouble and is the reason why absolute values are used in the formulation above. One can, however, reformulate the problem in order to achieve positive deviations only. This may be achieved by introducing slack variables in a similar fashion as in goal programming. Define o_j as the overachievement of the actual reserve from the mean reserve level during time period j and define u_j as the

underachievement of the actual reserve from the mean reserve level during time period j. Both these sets of variables are non-negative. The constraint set

$$\overline{r} = (D_j S + r_j) + u_j - o_j, \qquad j \in \mathcal{J} \tag{3.16}$$

is added to the formulation in order to define the slack variables' relationships to the reserve levels along with the constraint set

$$o_j, u_j \geq 0, \qquad j \in \mathcal{J} \tag{3.17}$$

to specify the nature of the slack variables. Since there can never be both an over- and under-achievement at any given time, at least one of the two variables o_j and u_j should be zero during time period j, for all $j \in \mathcal{J}$. By transforming the differences between the reserve levels and the mean reserve into non-negative deviations, the objective is to

$$\text{minimise} \qquad \sum_{j=1}^{m} (o_j + u_j), \tag{3.18}$$

which is a linear objective function. Solutions obtained using this objective function may be compared directly with solutions obtained using the objective function (3.13).

The third GMS problem formulation to be considered in this thesis is given by the *mixed-integer linear program* (MILP) in which the objective is to

$$\text{minimise} \qquad \sum_{j=1}^{m} (o_j + u_j)$$

subject to the constraints

$$\left.\begin{aligned}
\overline{r} &= \frac{1}{m} \sum_{j=1}^{m} D_j S + r_j, && \\
\sum_{j=e_i}^{\ell_i} x_{i,j} &= 1, && i \in \mathcal{I} \\
x_{i,j} &= 0, && j < e_i \text{ or } j > \ell_i,\ i \in \mathcal{I} \\
\sum_{i=1}^{n} g_{i,j} - \sum_{i=1}^{n} \sum_{p=1}^{j} g'_{p,i,j} x_{i,p} &= D_j(1+S) + r_j, && j \in \mathcal{J} \\
\overline{r} &= (D_j S + r_j) + u_j - o_j, && j \in \mathcal{J} \\
x_{i,j} &\in \{0,1\}, && i \in \mathcal{I},\ j \in \mathcal{J} \\
r_j, o_j, u_j &\geq 0, && j \in \mathcal{J}.
\end{aligned}\right\} \tag{3.19}$$

3.4 A more advanced GMS model

Now that three base models have been established in §3.3, the GMS problem may be expanded to include crew and exclusion constraints. In terms of the GMS problem's original context, as presented in Figure 3.1, the more advanced model simplifies it to the singular problem presented in Figure 3.3. In order to model these additional constraints, auxiliary variables are introduced. Define $y_{i,j}$ as a binary variable taking the value 1 if unit i is in maintenance during time period j,

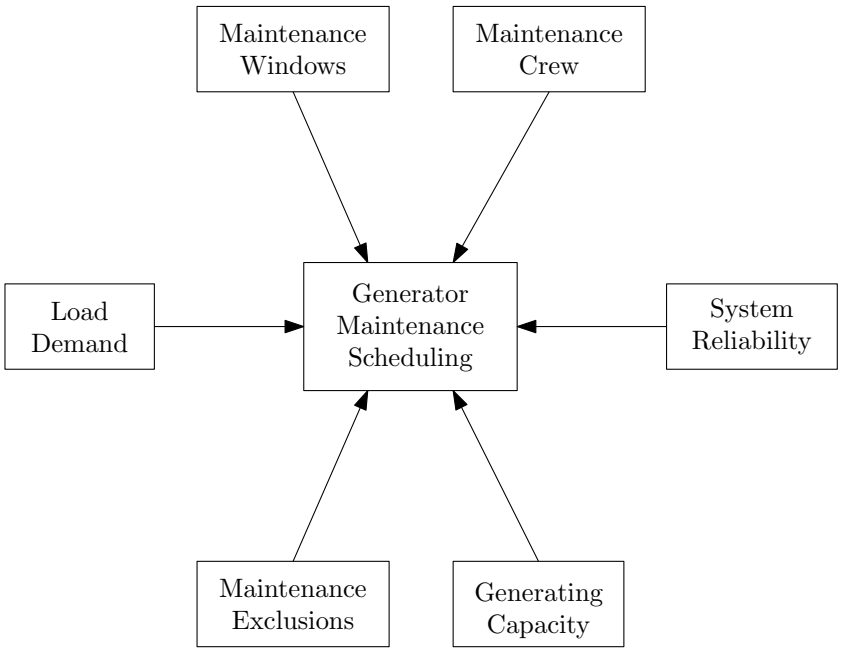


Figure 3.3: *Dependency diagram for a more advanced GMS problem.*

and zero otherwise. Before the new constraints are formulated, it is necessary to reformulate some of the existing constraints in terms of the new auxiliary variables as well as to present additional technical constraints. The formulation of the model is loosely based on the ones presented in [11, 48].

3.4.1 Model constraints

The maintenance window constraint sets (3.1) and (3.2) remain unchanged. An additional constraint set, similar to (3.2), may be added in order to ensure that the auxiliary variables are also set to zero outside the maintenance window of a unit. The constraint set

$$y_{i,j} = 0, \qquad j < e_i \text{ or } j > \ell_i + d_i - 1, \qquad i \in \mathcal{I} \tag{3.20}$$

fulfills the role of reducing the number of variables.

Technical constraints concerned with when a unit is in maintenance, have to be included in the problem. Since the maintenance of each unit must last for a given duration, the maintenance duration constraint set

$$\sum_{j=e_i}^{\ell_i+d_i-1} y_{i,j} = d_i, \qquad i \in \mathcal{I} \tag{3.21}$$

should be included. However, the maintenance of a generating unit must occur over consecutive time periods (continuously). A non-stop maintenance constraint set of the form

$$\begin{aligned} y_{i,j} - y_{i,j-1} &\leq x_{i,j}, \qquad i \in \mathcal{I},\ j \in \mathcal{J} \setminus \{1\}, \\ y_{i,1} &\leq x_{i,1}, \qquad i \in \mathcal{I} \end{aligned} \tag{3.22}$$

should therefore also be included.

The combination of the load and reliability constraints in constraint set (3.6) may be reformulated in terms of the auxiliary variables as a slightly simpler set of equalities. The structure

of the constraints remains exactly the same, but the parameters $g'_{p,i,j}$ are no longer necessary. Constraint set (3.6) is therefore replaced by

$$\sum_{i=1}^{n} g_{i,j}(1 - y_{i,j}) = D_j(1 + S) + r_j, \qquad j \in \mathcal{J}. \tag{3.23}$$

The first new constraint set deals with the availability of manpower for maintenance work during any given time period. There are two possible approaches, depending on the crew requirements of the power system. A simple approach may be used if a generating unit has crew requirements that only depend on the current time period in the planning horizon (*i.e.* if unit i requires a specified level of manpower when undergoing maintenance during time period j). If $m_{i,j}$ denotes the manpower required by unit i when undergoing maintenance during time period j, the crew constraint set to be included is

$$\sum_{i=1}^{n} m_{i,j} y_{i,j} \leq M_j, \qquad j \in \mathcal{J}, \tag{3.24}$$

where M_j denotes the maximum available manpower during time period j.

However, if the crew requirements of a generating unit depend on how much time it has already spent in maintenance (*i.e.* unit i requires a specified level of manpower in its k-th time period of maintenance) the constraints are slightly more complicated. Let $m'_{p,i,j}$ denote the required manpower of unit i when in maintenance during time period j if unit i commenced at time period p. If m_i^k denotes the required manpower of unit i in its k-th period of maintenance, the parameters $m'_{p,i,j}$ are calculated as

$$m'_{p,i,j} = \begin{cases} m_i^{j-p+1} & \text{if } j - p < d_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.25}$$

The crew constraint set may now be formulated as

$$\sum_{i=1}^{n} \sum_{p=1}^{j} m'_{p,i,j} x_{i,p} \leq M_j, \qquad j \in \mathcal{J}. \tag{3.26}$$

Exclusion constraints restrict certain units to be in a state of simultaneous maintenance. Consider the more general exclusion constraints where at most some specified number of units, within some subset of units, are allowed to be in a state of simultaneous maintenance. Let $\mathcal{K}$ denote the set of indices of generating unit subsets. These subsets typically contain homogeneous units (*e.g.* all units are nuclear, all units are coal-fired, *etc.*). If there are K subsets then $\mathcal{K} = \{1, \ldots, K\}$. Define $\mathcal{I}_k \in \mathcal{I}$ as the k-th subset of generating units with $k \in \mathcal{K}$. The exclusion constraint set may be formulated as

$$\sum_{i \in \mathcal{I}_k} y_{i,j} \leq K_k, \qquad j \in \mathcal{J},\ k \in \mathcal{K} \tag{3.27}$$

where K_k denotes the maximum number of units within subset k that are allowed to be in simultaneous maintenance during any time period.

Again, the final additional constraint set specifies the binary nature of the auxiliary variables as

$$y_{i,j} \in \{0, 1\}, \qquad i \in \mathcal{I},\ j \in \mathcal{J}. \tag{3.28}$$

3.4.2 The objective function

The three objective functions, defined for the simple GMS model in §3.3.2, remain unchanged with the introduction of auxiliary variables in the more advanced model. Therefore, the objective functions (3.10), (3.13) and (3.18) may be applied directly in the more advanced model. Furthermore, the constraint sets that define the quantities within these objective functions also remain unchanged, that is constraint (3.14) and constraint sets (3.16) and (3.17).

Problem formulations (3.12), (3.15) and (3.19) are now extended into more advanced models by problem formulations (A.1), (A.2) and (A.3), respectively. These formulations may be found in Appendix A.

3.5 Chapter summary

In this chapter, six formal mathematical model formulations were presented for the GMS problem. The formulations took the form of mathematical programs and vary in terms of their objective functions and complexity. Before the problem assumptions were presented in §3.2, the context of the GMS problem was briefly discussed with respect to power system operations scheduling in §3.1.

Following the GMS problem assumptions, three simple models were presented in §3.3 with different objective functions. These objective functions all aim to level the reserve. However, the formulations differ in classification — the first formulation is quadratic, the second is nonlinear while the third is linear.

In §3.4, the simple models were extended to more advanced models. This was achieved by the introduction of additional constraints to the model, because the simple models only considered the bare essentials in terms of the constraints for the GMS problem. These more advanced formulations provide one with an essential mathematical platform for generator maintenance scheduling.

CHAPTER 4

Solution Methodology

Contents

In this chapter, an exact and an approximate solution approach are proposed for solving the GMS problem. The exact approach consists of utilising an off-the-shelf software package to solve the problem; the algorithms employed by the software package are described in §4.1. A random search heuristic and a simulated annealing algorithm are proposed for solving the GMS problem approximately. The specific implementations of these proposed methods for the GMS problem are presented in §4.2. As most parameter values within the methods are typically problem-dependent, these values are not estimated here.

4.1 Exact solution approach

Since the GMS problem may be formulated as a mathematical program, as was done in Chapter 3, mathematical programming solution techniques may theoretically be applied to solve the GMS problem to optimality. However, due to the integer-valued, and large scale nature of the GMS problem, it may take an unpractical amount of time to obtain such an optimal (or even a good) solution, especially if the objective function is nonlinear.

Even so, the inclusion of an exact solution approach to the GMS problem in this thesis is for comparative purposes and to provide possible bounds on the objective function values of the problem for a given instance. An off-the-shelf software package, featuring different solvers, called *LINGO* [53, 55] was used to implement the mathematical programming formulations and to solve the GMS problem. However, restrictions were placed on the allowed computational time for practical reasons.

LINGO utilises various built-in solvers to solve a wide variety of problems [54, 56]. It interprets problem formulations and automatically passes the problem to the appropriate solver, based on the type of problem. Alternatively, the user may specify the solver to be used for a particular problem instance. Short descriptions of the relevant solvers, which may be used for the GMS problem, are presented below.

4.1.1 LINGO's simplex solvers

The Primal and Dual Simplex solvers feature advanced implementations of the primal and dual simplex algorithms (see §2.4.2). These solvers are the primary means for solving linear models.

4.1.2 LINGO's integer solver

Models containing general and/or binary integer variables are solved via an integer solver that works in conjunction with the linear and nonlinear solvers. LINGO uses a branch-and-bound solution procedure (see §2.4.2) when integer models are solved. If the model is linear, the integer solver employs preprocessing strategies, heuristics and advanced constraint "cut" generation routines that may reduce solution times significantly.

4.1.3 LINGO's general nonlinear solver

The primary underlying technique used by the nonlinear solver is based upon a generalised reduced gradient algorithm (see §2.4.2). LINGO also incorporates successive linear programming (see §2.4.2) to compute new search directions and to find a good feasible solution quickly. The nonlinear solver is classified as a local search solver and can only guarantee locally optimal solutions.

4.1.4 LINGO's global solver

Instead of discontinuing the search when a local optimum is found, as the nonlinear solver would do, the global solver continues searching until a global optimum is confirmed. The global solver uses a series of range bounding and range reduction techniques to convert the original non-convex, nonlinear problem into several convex, linear subproblems [56]. An exhaustive search is then done within a branch-and-bound framework over these subproblems to find a global optimum.

4.1.5 LINGO's quadratic solver

The quadratic solver uses a barrier or interior point method (see §2.4.2) to solve a model with a quadratic objective function and/or some constraints with quadratic terms. LINGO takes advantage of the quadratic structure to solve such problems much faster than the nonlinear solver would.

4.2 Approximate solution approach

A number of different solution techniques for the GMS problem were reviewed in §2.4, most of which were non-exact methods. Two local search methods for solving the GMS problem are presented in more detail here — a random search heuristic designed to serve as a simple starting point, followed by a simulated annealing algorithm which is more advanced (a meta-heuristic method). In the implementation of these methods, a solution to the GMS problem (*i.e.* maintenance schedule) is denoted by a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ of length n where the entry x_i is an integer value representing the starting time period of maintenance for unit $i \in \mathcal{I}$.

Two constraint approaches to the GMS problem may be adopted, namely a hard constraint approach and a soft constraint approach. Within the hard constraint approach, all the constraints of the GMS problem are satisfied. A candidate solution that violates any of the constraints is deemed infeasible and not considered by the algorithm. On the other hand, the soft constraint approach allows for limited constraint violation. If a candidate solution violates any of the constraints, a corresponding penalty value is added to the objective function value associated with the solution and the algorithm continues the search process. A more detailed explanation of the soft constraint approach is presented below as this is the approach adopted in this thesis.

4.2.1 The soft constraint approach

When a candidate solution violates a constraint in the GMS problem, a penalty is incurred for that violation. This penalty is imposed in such a manner that a larger violation receives a larger penalty value in order to discourage infeasible solutions. The total penalty value of a solution is calculated as a weighted sum of the various constraint penalty values. This total penalty value P is then added to the objective function value associated with the candidate solution.

For the maintenance window constraint set, the earliest and latest maintenance commencement times for each unit are extended by a parameter W_{ext}. As such, the allowable maintenance window $[e_i, \ell_i]$ is extended to $[e_i - W_{ext}, \ell_i + W_{ext}]$ for each unit $i \in \mathcal{I}$. The window violation penalty term P_w^i for unit i is the number of time periods before the earliest starting time e_i or after the latest starting time ℓ_i within this extended window at which maintenance of the unit commences, that is

$$P_w^i = \begin{cases} e_i - x_i & \text{if} \quad x_i < e_i, \\ 0 & \text{if} \quad e_i \leq x_i \leq \ell_i, \\ x_i - \ell_i & \text{if} \quad x_i > \ell_i, \end{cases}$$

for all $i \in \mathcal{I}$. A different weight w_w^i (linearly increasing as a function of P_w^i) is associated with each of these maintenance window violations. The overall maintenance window penalty P_w is calculated as the sum of the weighted window violation penalty terms P_w^i over all the units, that is

$$P_w = \sum_{i=1}^{n} w_w^i P_w^i. \tag{4.1}$$

The load and reliability constraint violation penalty term P_ℓ^j during time period j is simply the shortfall in load, that is

$$P_\ell^j = \max\{-r_j,\ 0\},$$

for all $j \in \mathcal{J}$, since r_j is negative during a shortfall in power supply. The overall load and

reliability penalty term P_ℓ is the sum of these violations over all time periods, that is

$$P_\ell = \sum_{j=1}^{m} P_\ell^j. \tag{4.2}$$

Similarly, the crew constraint violation penalty term P_c^j during time period j is the shortfall in manpower during that time period, that is

$$P_c^j = \begin{cases} \max\left\{\sum_{i=1}^{n} m_{i,j}y_{i,j} - M_j,\ 0\right\} & \text{if constraint set (3.24) is used,} \\ \max\left\{\sum_{i=1}^{n}\sum_{p=1}^{j} m'_{p,i,j}x_{i,p} - M_j,\ 0\right\} & \text{if constraint set (3.26) is used,} \end{cases}$$

for all $j \in \mathcal{J}$. The overall crew penalty term P_c is the sum of these crew constraint violation penalty terms over all time periods, that is

$$P_c = \sum_{j=1}^{m} P_c^j. \tag{4.3}$$

Lastly, the exclusion constraint violation penalty term $P_e^{k,j}$ for the subset of units k during time period j is the number of units in simultaneous maintenance beyond the maximum of the subset. Thus

$$P_e^{k,j} = \max\left\{\sum_{i\in\mathcal{I}_k} y_{i,j} - K_k,\ 0\right\},$$

for all $k \in \mathcal{K}$ and $j \in \mathcal{J}$. The overall exclusion penalty term P_e is calculated as the sum of these exclusion constriant violation penalty terms over all subsets and time periods, that is

$$P_e = \sum_{k=1}^{K}\sum_{j=1}^{m} P_e^{k,j}. \tag{4.4}$$

As mentioned earlier, the total penalty value associated with a candidate solution is the weighted sum of all the penalty terms above. If w_ℓ, w_c and w_e are the corresponding weights for the load and reliability, crew and exclusion penalties, respectively, then the total penalty term is

$$P = P_w + w_\ell P_\ell + w_c P_c + w_e P_e, \tag{4.5}$$

where the values of P_w, P_ℓ, P_c and P_e are taken as in (4.1)–(4.4). Note that the window penalty P_w is already weighted due to the varying values of window violation weights. The method adopted to determine values for all these weights is described later as it is typically problem instance-dependent. The function `checkFeasibilityAndCalculatePenalty`, for which a pseudo-code listing is given in Function 4.1, may be called after a candidate solution has been generated in order to determine its corresponding penalty value.

4.2.2 The neighbourhood move operators

Since both the random search heuristic and the simulated annealing algorithm are local search methods, some form of neighbourhood structure or move operator has to be defined. The

Function 4.1: `checkFeasibilityAndCalculatePenalty`$(\boldsymbol{x}, dataset)$

Input: The current solution vector, the problem's full dataset
Output: The total penalty term for the current solution

$P_w \leftarrow 0$
$P_\ell \leftarrow 0$
$P_c \leftarrow 0$
$P_e \leftarrow 0$
if $\boldsymbol{x}$ *violates maintenance window constraint* **then** Calculate overall maintenance window penalty term P_w
if $\boldsymbol{x}$ *violates load and reliability constraint* **then** Calculate overall load and reliability penalty term P_ℓ
if $\boldsymbol{x}$ *violates crew constraint* **then** Calculate overall crew penalty term P_c
if $\boldsymbol{x}$ *violates exclusion constraint* **then** Calculate overall exclusion penalty term P_e
totalPenalty $\leftarrow P_w + w_\ell P_\ell + w_c P_c + w_e P_e$
return totalPenalty

purpose of a move operator is to generate a new candidate solution by perturbing the current solution in some specified manner. The solutions obtained from all possible perturbations according to the operator are collectively known as the neighbourhood of the current solution.

The author could find only two neighbourhood move operators present in GMS literature [10, 15, 17, 23, 41, 72, 73], with the one being a simplification of the other. According to the first of these move operators, one unit is randomly selected according to a uniform distribution and its maintenance starting time is then randomly changed to a new value within the allowed range (maintenance window) according to a uniform distribution. This neighbourhood move operator will hereafter be referred to as the *classical* neighbourhood move operator. The function `createClassicalNeighbourhoodList`, for which a pseudo-code listing is given in Function 4.2, illustrates how to generate the list that contains all the moves to create the classical neighbourhood of a candidate solution. The other operator restricts the new value to the two adjacent time periods. These two moves are classified as *elementary* moves.

Function 4.2: `createClassicalNeighbourhoodList`$(n, \boldsymbol{e}, \boldsymbol{\ell}, W_{ext})$

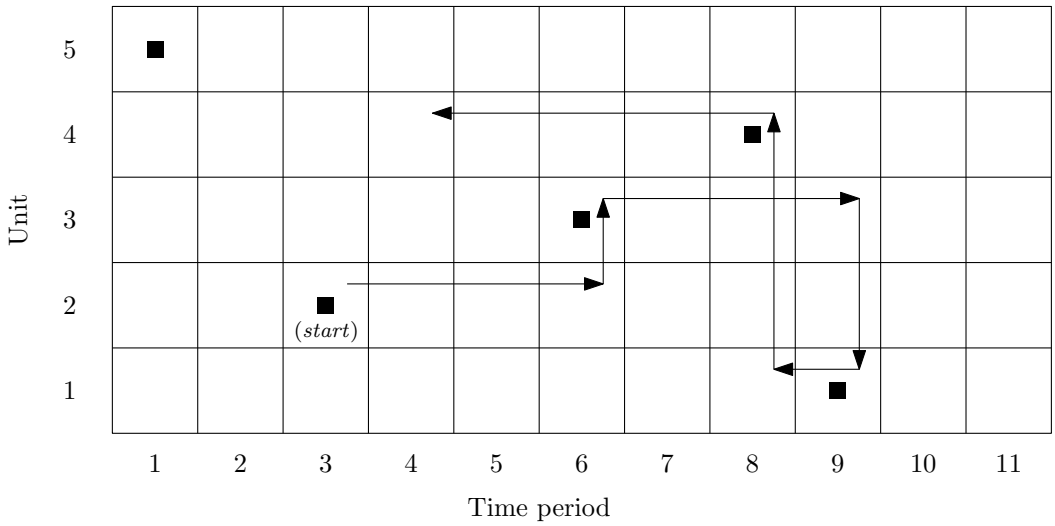
Input: The number of units, the vectors containing the earliest and latest maintenance starting times for all units, the maintenance commencement extension parameter
Output: The list of elementary moves that creates the full classical neighbourhood

counter $\leftarrow 1$
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow (e_i - W_{ext})$ **to** $(\ell_i + W_{ext})$ **do**
    moves(counter) $\leftarrow \{i, j\}$
    counter $\leftarrow$ counter $+ 1$
  end
end
return moves

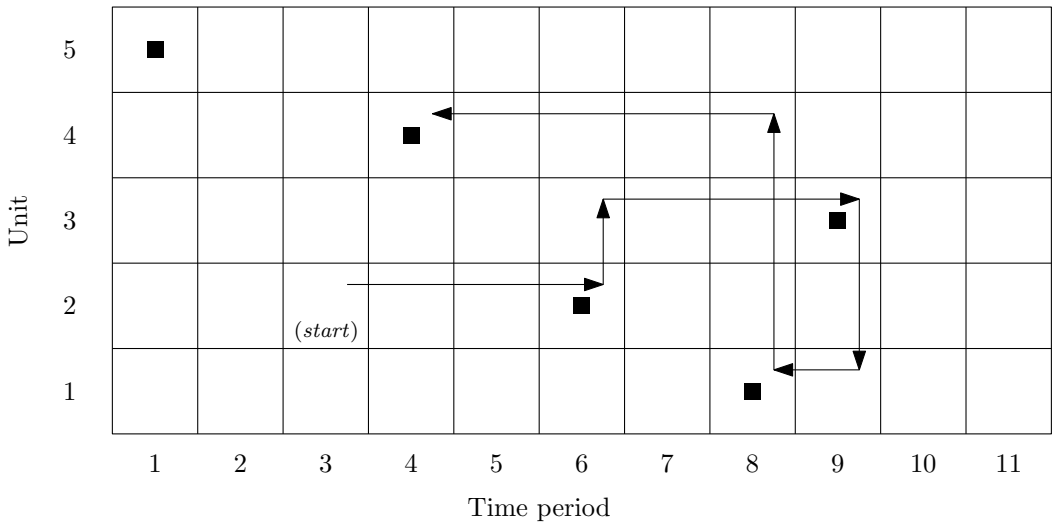
Although the elementary neighbourhood move operators described above have been used with success in the literature, a new neighbourhood move operator for solutions in a GMS context is

proposed in this thesis. This operator diminishes the locality of elementary moves by including more global information on the entire maintenance schedule in order to search the solution space more effectively.

The move operator presented below generates a so-called *ejection chain* and it may be classified as a *compound* move. Consider a solution vector $\boldsymbol{x}$ represented on a $n \times m$ grid where the rows represent the generating units and the columns represent the time periods of the maintenance schedule. An occupied cell in row i and column j indicates that unit i commences its maintenance during time period j, that is $x_i = j$. The ejection chain move operator generates a sequence of alternating horizontal and vertical steps within the grid, an example of which may be seen in Figure 4.1. In the description of the ejection chain that follows, any reference to a



(a) Schedule before the ejection chain move is applied



(b) Schedule after the ejection chain move is applied

Figure 4.1: *Illustration of the ejection chain move on a GMS schedule.*

random selection is assumed to be performed according to a uniform distribution. The starting point of the sequence (*start*) is determined by randomly selecting an occupied cell (*i.e.* choosing a unit at random and selecting an occupied cell in its row randomly). From this occupied cell,

an unoccupied time period (column) is randomly selected within the same row (unit) from the allowable range of starting times for the unit. This selection step is referred to as *taking a horizontal step*. From this unoccupied cell, a random occupied cell in the same column is selected, referred to as *taking a vertical step*. This process of taking horizontal and vertical steps in alternating fashion is repeated until either a horizontal step is made from an occupied cell to an unoccupied cell in the same column as *start*, thus closing the sequence, or until a vertical step cannot be performed due to a lack of occupied cells in the relevant column. The neighbouring solution is generated by applying the ejection chain move along the sequence — every cell value at the head of an arrow in the chain, is replaced by the cell value at the tail of that same arrow in Figure 4.1. Should the chain be open-ended, as in the example in Figure 4.1, the starting cell value is replaced by an empty cell value. The function `createEjectionChainList`, for which a pseudo-code listing is given in Function 4.3, illustrates how to create such an ejection chain.

Function 4.3: `createEjectionChainList`$(unit, n, \boldsymbol{e}, \boldsymbol{\ell}, W_{ext}, \boldsymbol{x})$

Input: The unit at the head of an ejection chain, the number of units, the vectors containing the earliest and latest maintenance starting times for all units, the maintenance commencement extension parameter, the current solution vector

Output: The list of moves in an ejection chain

```
chain ← ∅
possibleTimes ← {(e_unit − W_ext), ..., (ℓ_unit + W_ext)} \ x_unit
newTime ← rand(possibleTimes)
counter ← 1
chain(counter) ← {unit, newTime}
remainingUnits ← {1, ..., n} \ unit
notDone ← true
while notDone = true do
    potentialUnits ← ∅
    for i ∈ remainingUnits do
        if x_i = newTime then
            potentialUnits ← potentialUnits ∪ i
        end
    end
    if potentialUnits ≠ ∅ then
        counter ← counter + 1
        newUnit ← rand(potentialUnits)
        possibleTimes ← {(e_newUnit − W_ext), ..., (ℓ_newUnit + W_ext)} \ x_newUnit
        newTime ← rand(possibleTimes)
        chain(counter) ← {newUnit, newTime}
        remainingUnits ← {1, ..., n} \ newUnit
        if newTime = chain(1, 2)                    // first entry, second element
        then
            notDone ← false
        end
    else
        notDone ← false
    end
end
return chain
```

To summarise, the ejection chain move operator generates a list of units whose maintenance starting times are randomly altered (horizontal steps). However, adjacent units in the list are connected in such a way that the preceding unit's new maintenance starting time is the same as the succeeding unit's old maintenance starting time (vertical steps). Furthermore, there are no restrictions with respect to horizontal and vertical steps crossing one another or, as a result, on the direction of any of the steps. The example in Figure 4.1 illustrates this fact.

A natural question that arises from the ejection chain move operator described above is how one guarantees that a significant number of non-trivial ejection chains are generated during the course of the SA algorithm? If the length of an ejection chain is defined as the number of altered maintenance starting times, *i.e.* the number of horizontal moves, then a trivial ejection chain has a length of one. This is the minimum length of any ejection chain for a solution of the GMS problem, provided that all the units have a maintenance window larger than one time period. Due to the typical large-scale nature of the GMS problem, it is a natural occurence that many columns in a solution grid are completely unoccupied. This occurence presents a problem for any ejection chain. However, the problem may be resolved if more columns are available for the vertical steps. As such, the proposed SA algorithm is implemented by additionally including a certain number of adjacent columns (depending on the problem instance) when performing the vertical steps in an ejection chain.

4.2.3 Generating a random initial solution

Consider firstly the hard constraint approach. For unit i, a random maintenance starting time x_i is chosen between its earliest and latest starting times, according to a uniform distribution. The resulting candidate solution vector $\boldsymbol{x}$ is then tested for feasibility. Note that the maintenance window constraint set will always be satisfied. If the solution is feasible, it is adopted as the initial solution. Otherwise, all units involved in unsatisfied constraints are isolated. From these units, one is chosen randomly according to a uniform distribution. A new random maintenance starting time for this unit is chosen in the same manner as before, with the current time excluded. The new candidate solution vector is again tested for feasibility. This process is repeated until a feasible solution is obtained. Unfortunately, this process may be very time consuming (even taking impractically long), especially if the problem instance is highly constrained.

Consider now the soft constraint approach. Again, for each unit i, a random maintenance starting time x_i is chosen according to a uniform distribution. However, the starting time is chosen between the extended earliest and latest starting times. The constraint violations are calculated from this solution vector $\boldsymbol{x}$ in order to determine the solution's feasibility. If the solution is feasible, its penalty value is set to zero, otherwise, the value of the penalty is calculated accordingly as in (4.5). Furthermore, the solution is immediately set as the initial solution. Clearly, this process is much faster than that of the hard constraint approach. The function `generateRandomSolution`, for which a pseudo-code listing is given in Function 4.4, may be called to create a random solution and its objective function value which may then be set as the initial solution.

4.2.4 Random search heuristic implementation

A random search method is one of the simplest solution methods one may adopt in order to solve a combinatorial optimisation problem approximately. Such an approach is often used as a baseline for further solution methods which employ more complicated techniques, since a random

Function 4.4: `generateRandomSolution(`*dataset*`)`

Input: The problem's full dataset
Output: A random solution vector and it's objective function value

for $i \leftarrow 1$ **to** n **do**
  possibleTimes $\leftarrow \{(e_i - W_{ext}), \ldots, (\ell_i + W_{ext})\}$
  $x_i \leftarrow$ `rand`(possibleTimes)
end
$P \leftarrow$ `checkFeasibilityAndCalculatePenalty`$(\boldsymbol{x}, dataset)$
Calculate objFunctionValue
objFunctionValue $\leftarrow$ objFunctionValue $+ P$
return [$\boldsymbol{x}$, objFunctionValue]

search is simply a method that randomly searches the solution space of a problem by generating new solutions according to some neighbourhood structure without any other enhancements. The algorithmic implementation of the random search heuristic adopted in this thesis for the GMS problem is presented in this section.

Initialisation

The random search heuristic is initialised by generating an initial solution to an instance of the GMS problem. A random initial solution is generated, as described in §4.2.3, by the function `generateRandomSolution`.

Determining the next candidate solution

In order to find the new candidate solution for the next iteration in the heuristic, a *best improvement* approach is adopted. A truncated neighbourhood of user-defined size is generated from the current solution. Both neighbourhood move operators described in §4.2.2 are implemented in the random search heuristic to create the neighbourhood (for comparative purposes). The candidate solution within this truncated neighbourhood having the best objective function value (*i.e.* the best neighbour) is selected as the new current solution for the next iteration. Note that the best neighbour does not have to be an improving solution.

Termination criteria

The random search heuristic terminates as soon as one of the following two conditions is satisfied:

- the number of iterations exceeds a pre-specified number,
- the number of consecutive iterations without uncovering an improving solution reaches a pre-specified number,

During each iteration, the best neighbour is compared with the best solution found so far (the *incumbent solution*), and when necessary, the incumbent solution is updated. When the heuristic terminates, the incumbent solution is the final solution returned. The pseudo-code listing for the generator maintenance scheduling random search heuristic with an ejection chain neighbourhood is provided in Algorithm 4.5. Similarly, the pseudo-code listing for the generator maintenance scheduling random search heuristic with a classical neighbourhood is provided in Algorithm B.1 in Appendix B.

Algorithm 4.5: The GMS random search heuristic with ejection chain neighbourhood

Input: A power system scenario for which to solve the generator maintenance scheduling problem
Output: The best maintenance schedule found

```
dataset ← declareSystemData()
[current, currentObj] ← generateRandomSolution(dataset)
[incumbent, incumbentObj] ← [current, currentObj]
iterationCounter ← 0
nonImproveCounter ← 0
while (iterationCounter < maxIteration) and (nonImproveCounter < maxNonImprove) do
    bestNeighbour ← ∅
    bestNeighbourObj ← some very large number
    for neighbourCounter ← 1 to neighbourhoodSize do
        neighbour ← current
        unit ← rand([1, n])
        chain ← createEjectionChainList(unit, n, e, ℓ, W_ext, neighbour)
        Apply chain on neighbour to create new neighbour
        P ← checkFeasibilityAndCalculatePenalty(neighbour, dataset)
        Calculate neighbourObj
        neighbourObj ← neighbourObj + P
        if neighbourObj < bestNeighbourObj then
            [bestNeighbour, bestNeighbourObj] ← [neighbour, neighbourObj]
        end
    end
    if bestNeighbourObj < incumbentObj then
        [incumbent, incumbentObj] ← [bestNeighbour, bestNeighbourObj]
        nonImproveCounter ← 0
    else
        nonImproveCounter ← nonImproveCounter + 1
    end
    [current, currentObj] ← [bestNeighbour, bestNeighbourObj]
    iterationCounter ← iterationCounter + 1
end
```

4.2.5 Simulated annealing algorithmic implementation

The general framework of working for the method of SA was presented in §2.4.4. A more detailed description of the SA algorithmic implementation adopted for scheduling purposes in this thesis is presented here.

Initialisation

The initialisation step of the SA algorithm consists of generating an initial solution to an instance of the GMS problem, as well as computing an initial temperature for the system. A random initial solution is generated, as described in §4.2.3, by the function `generateRandomSolution`. After this has been done, an initial temperature is computed by using this initial solution.

Given an initial solution, an initial temperature T_0 may be determined by using the method described in [79]. This method, hereafter referred to as the *average increase method*, relies on an initial acceptance ratio χ_0, which is defined as the number of accepted worsening solutions[1] divided by the number of attempted worsening solutions, and on the average increase in energy (worsening of objective function value), denoted by $\overline{\Delta E}^{(+)}$. The value of $\overline{\Delta E}^{(+)}$ is estimated by executing a random walk over the solution space, using the initial solution is as the starting point. Once the value of $\overline{\Delta E}^{(+)}$ is estimated, the initial temperature may be calculated as

$$\chi_0 = \exp\left(-\frac{\overline{\Delta E}^{(+)}}{T_0}\right),$$

which yields

$$T_0 = -\frac{\overline{\Delta E}^{(+)}}{\ln(\chi_0)}. \tag{4.6}$$

A second method for the initial temperature, namely White's formula, is also considered [79], hereafter referred to as the *standard deviation method*. According to this formula, the initial temperature is set as the standard deviation σ_0 in the objective function value during an initial random walk over the solution space, that is

$$T_0 = \sigma_0. \tag{4.7}$$

The function `initialTemperature`, for which a pseudo-code listing is given in Function 4.6, illustrates these two methods.

The cooling schedule

The outer loop of the SA algorithm, starting at line 7 in Algorithm 4.7, consists of the gradual lowering of the temperature of the system. A decreasing temperature function, also known as the law of decrease of the temperature, is thus defined. The well-known and widely adopted geometric law of decrease [20] was initially chosen for use in the proposed SA algorithm due to its simplicity and effectiveness. The updating rule for this law is

$$T_{s+1} = \alpha T_s, \tag{4.8}$$

where T_s is the temperature at stage s and $\alpha \in (0, 1)$ is a constant called the cooling parameter. Typically, the value of α is taken between 0.8 and 0.99 when used in practice [22]. This cooling law was also adopted in [9, 15, 73] within a GMS context.

Three additional temperature functions reported in [79] are investigated for use in the context of the GMS problem. These methods are called *adaptive* as they use feedback from the algorithm to calculate the next temperature. Such an adaptive schedule attempts to meet two contradicting goals, namely to

- keep the annealing as close to equilibrium as possible, and
- execute the annealing process in as short a time as possible.

[1] In a minimisation problem, a worsening solution increases the objective function value.

Function 4.6: `initialTemperature`$(\boldsymbol{x}, xObj, dataset)$

Input: The initial solution vector, the initial objective function value, the problem's full dataset

Output: Two initial temperatures calculated using the average increase method and using the standard deviation method

```
[current, currentObj] ← [x, xObj]
j ← 0
for i ← 1 to length of random walk do
    previousObj ← currentObj
    unit ← rand([1, n])
    chain ← createEjectionChainList(unit, n, e, ℓ, W_ext, current)
    Apply chain on current to create new solution
    P ← checkFeasibilityAndCalculatePenalty(current, dataset)
    Calculate currentObj
    currentObj ← currentObj + P
    ΔE ← currentObj − previousObj
    if ΔE > 0 then
        j ← j + 1
        increases_j ← ΔE
    end
    values_j ← currentObj
end
avgIncTemperature ← −average(increases) / ln(χ_0)
stdDevTemperature ← stdDeviation(values)
return [avgIncTemperature, stdDevTemperature]
```

Firstly, the updating rule proposed by Huang *et al.* [37] is given by

$$T_{s+1} = T_s \exp\left(-\frac{\lambda T_s}{\sigma_s}\right), \tag{4.9}$$

where $\lambda \in (0, 1]$ is a constant with a typical value of 0.7 and σ_s is the standard deviation observed in the changing values of the objective function when reaching stage s. The rule is based on the average objective function values achieved during consecutive temperature stages. It is expected that by setting the difference between the average objective function value at the current temperature stage and at the next temperature stage to be less than the standard deviation of the objective function value at the current temperature stage, quasi-equilibrium[2] will be maintained.

Secondly, Van Laarhoven *et al.* [81] proposed the updating rule

$$T_{s+1} = T_s \frac{1}{1 + \frac{\ln(1+\delta)}{3\sigma_s} T_s}, \tag{4.10}$$

[2]Based on the analogy of simulated annealing to real annealing, as described in §2.4.4, the quasi-equilibrium is analogous to the thermodynamic balance at a temperature stage, *i.e.* the probability distribution of objective function values is close to a Boltzmann distribution.

where δ is a "small" real number. If $q_{\boldsymbol{x}}(T_s)$ represents the stationary distribution[3] at temperature stage s, then the decrement rule guarantees that

$$\frac{1}{1+\delta} < \frac{q_{\boldsymbol{x}}(T_s)}{q_{\boldsymbol{x}}(T_{s+1})} < 1+\delta, \qquad \text{for all } \boldsymbol{x}.$$

It is expected that by maintaining the homogeneous Markov chains close to each other, the number of attempted solutions should be few in order to reach equilibrium between each temperature decrement.

Finally, the updating rule proposed by Triki *et al.* [79] is given by

$$T_{s+1} = T_s \left(1 - T_s \frac{\Delta}{\sigma_s^2}\right), \tag{4.11}$$

where Δ is the expected decrease in the average objective function value when reaching the next temperature stage of the search process. The theoretical progression of the expected objective function value is set once for all the temperature stages by the choice of Δ. If the observed average expected objective function value at temperature T_s corresponds well with the theoretical value during the search process, then equilibrium is still achieved. However, if the gap between the observed and theoretical values becomes significant, equilibrium has not been achieved and the temperature decrement must be adapted. Four options may be considered: increasing the number of attempted solutions at the current temperature stage, increasing the current temperature, choosing a new smaller expected decrease Δ, or terminating the algorithm and initialising a greedy algorithm. The authors in [79] provide an algorithm in which the rule (4.11) may be implemented efficiently. A slightly modified version of their algorithm was applied in this thesis, a pseudo-code of which may be found in Algorithm B.2 in Appendix B. Typically, the initial value of the expected decrease in the objective function value is taken as σ_0/μ_2, with $\mu_2 \in [1, 20]$. Equilibrium has not been reached if the ratio between the average objective function value and the expected average objective function value is larger than ζ, with $\zeta \in [1, 1.1]$. Lastly, if the expected decrease Δ is modified during some temperature stage, it is decreased by a factor of μ_1 with $\mu_1 \in [2, 20]$.

The length of the inner loop of the SA algorithm is determined by the number of repeated iterations of the Metropolis algorithm (length of the Markov chain). In other words, this length is the criterion for the change of temperature stage [20]. According to Eglese [22], the length of the Markov chain is determined by a sufficient number of solutions being accepted subject to a constant upper bound in order to bring the system close to equilibrium at that temperature. Following the suggested scheme presented in [20], the inner Metropolis loop in the proposed SA algorithm terminates during each temperature stage when one of the following two conditions is satisfied

- a maximum of $12N$ solutions are accepted,
- a maximum of $100N$ solutions are attempted,

where N denotes the number of degrees of freedom of the problem. In this case $N = n$.

[3]The existence of, and convergence to, a stationary distribution for the Markov chains formed at each temperature stage have been formally established in the literature [81] by various authors [20].

The rule of acceptance

Since candidate solutions that cause a decrease in energy are accepted with probability 1, the rule of acceptance only applies to solutions causing an increase in energy. Although different rules of acceptance (or acceptance probabilities) may be adopted in the method of SA, the proposed algorithm implements the classical Metropolis rule described in §2.4.4. Practically, this rule is implemented as follows. Whenever a candidate solution causes an increase in energy ($\Delta E > 0$), a random number $r \in (0, 1)$ is generated according to a uniform distribution. If $r < \exp(-\Delta E/T_s)$, then the candidate solution is accepted as the new current solution; otherwise it is rejected and the current solution remains the same.

The neighbourhood move operator

Both neighbourhood move operators described in §4.2.2 are implemented in the SA algorithm for comparative purposes. The aim with the new ejection chain move operator is to improve upon the results obtained by using the elementary move operator. Note that candidate solutions within the classical neighbourhood of a solution are sampled without re-insertion. This means that a neighbourhood move applied to the current solution may only be repeated once all possible moves in the neighbourhood have been attempted. This is not the case with candidate solutions in the ejection chain neighbourhood, since the nature of the ejection chain moves does not make this an easy task — if there are k units within an ejection chain, the neighbourhood size is

$$(m-1)^k \prod_{i=0}^{k-1} (n-i),$$

while the size of the classical neighbourhood is only nm.

Termination criteria

The SA algorithm terminates when the outer temperature loop terminates. This may occur if the system has reached its ground state, corresponding to its lowest energy (smallest objective function value). As stated in §2.4.4, the system is said to be *frozen* in this case because no further change in the solution is likely.

Two termination criteria are implemented in the proposed SA algorithm. The temperature loop terminates as soon as one of the following two conditions is satisfied:

- the temperature at the current stage reaches the pre-specified minimum temperature T_{min},
- a pre-specified number, Ω_{frozen}, of successive temperature stages occur without the occurence of any acceptance.

On completion of a standard SA algorithm, the current solution is reported as an approximate solution to the GMS problem instance. However, the proposed SA algorithm is implemented with a slight modification of the final solution, as described in below.

Modification to the standard SA algorithm

A modification in the proposed SA algorithm is to store the best solution found so far, called the *incumbent solution*. This simple modification [22] is not computationally expensive and may pro-

duce improved solutions at a similar computational time than that of a standard SA algorithm. The reason why this may occur, is the fact that the algorithm can accept worsening solutions and it is therefore possible that the final solution is worse than the best solution obtained during the entire algorithmic execution. A pseudo-code listing for the generator maintenance scheduling simulated annealing algorithm with an ejection chain neighbourhood is provided in Algorithm 4.7. Since the difference in pseudo-code between using the ejection chain neighbourhood versus using the classical neighbourhood is so small, as seen between Algorithm 4.5 and Algorithm B.1, no listing is provided for the simulated annealing algorithm with a classical neighbourhood. The neighbourhood implementation would be similar to that in Algorithm B.1.

4.2.6 Proposed modifications for investigation

Additional modifications to the simulated annealing algorithm are investigated in this thesis. These modifications attempt to improve the solution method without incurring excessive computational costs.

Introduction of a local search heuristic

A local search heuristic may be introduced into the SA solution method described above in order to obtain a hybrid solution method. The proposed hybridisation may be applied in two different ways. In the first, the local search is applied on the incumbent solution each time a new incumbent solution is encountered during the algorithm's execution, that is after line 26 in Algorithm 4.7. Only the incumbent solution is updated by means of the local search, the current solution remains unaffected in order to prevent premature convergence. In the second, the local search is applied at the end of the algorithm, on the final incumbent solution, that is after line 42 in Algorithm 4.7. It may be possible to improve on the solution, since its full neighbourhood has typically not been explored in the SA algorithm.

The implementation of the local search heuristic adopts the classical neighbourhood move operator presented in §4.2.2. That is, the elementary move in which one generating unit is randomly selected according to a uniform distribution and its maintenance starting time is then changed to a random new value within the allowed range (maintenance window). The local search heuristic receives the incumbent solution as initial solution. The full neighbourhood of the solution (with a maximum size of nm) is then searched in order to find the best neighbour. If the best neighbour improves the current solution, it is set as the new current solution and the process repeats. The search terminates if no further improvement can be made. A pseudo-code listing of the local search heuristic is provided in Algorithm 4.8.

Improved initial solution

Instead of simply using a random solution as initial solution, the local search heuristic introduced above may be applied to the random solution. This is expected to improve the objective function value of the solution, resulting in a "good" random initial solution. The possible impact of applying this enhancement to the solution methods may include faster solution times (fewer iterations) and improved incumbent solutions because the algorithm already started with a good solution. In the implementation, a user-defined number of random solutions are generated according to the process described in §4.2.3, each undergoing the local search heuristic presented above. The best solution from this set is then chosen as the initial solution for the algorithm to

Algorithm 4.7: The GMS simulated annealing algorithm

Input: A power system scenario for which to solve the generator maintenance scheduling problem

Output: The best maintenance schedule found

```
1  dataset ← declareSystemData()
2  [current, currentObj ] ← generateRandomSolution(dataset)
3  [avgT_0, stdT_0] ← initialTemperature(current, currentObj, dataset)
4  T ← avgT_0                                                    // or stdT_0
5  [incumbent, incumbentObj] ← [current, currentObj]
6  notAcceptCounter ← 0
7  while (T > T_min) and (notAcceptCounter < Ω_frozen) do
8      numberAccept ← 0
9      numberAttempt ← 0
10     accepted ← false
11     while (numberAccept < 12n) and (numberAttempt < 100n) do
12         numberAttempt ← numberAttempt + 1
13         neighbour ← current
14         unit ← rand([1, n])
15         chain ← createEjectionChainList(unit, n, e, ℓ, W_ext, neighbour)
16         Apply chain on neighbour to create new neighbour
17         P ← checkFeasibilityAndCalculatePenalty(neighbour, dataset)
18         Calculate neighbourObj
19         neighbourObj ← neighbourObj + P
20         ΔE ← neighbourObj − currentObj
21         if ΔE ≤ 0 then
22             [current, currentObj] ← [neighbour, neighbourObj]
23             numberAccept ← numberAccept + 1
24             accepted ← true
25             if currentObj < incumbentObj then
26                 [incumbent, incumbentObj] ← [current, currentObj]
27             end
28         else
29             if rand((0, 1)) < exp(ΔE/T) then
30                 [current, currentObj] ← [neighbour, neighbourObj]
31                 numberAccept ← numberAccept + 1
32                 accepted ← true
33             end
34         end
35     end
36     if accepted = true then
37         notAcceptCounter ← 0
38     else
39         notAcceptCounter ← notAcceptCounter + 1
40     end
41     Update temperature T
42 end
```

Algorithm 4.8: The GMS local search heuristic

Input: The incumbent solution vector, the incumbent objective function value, the problem's full dataset
Output: The possibly improved incumbent solution vector and corresponding objective function value

```
1  [current, currentObj] ← [incumbent, incumbentObj]
2  improved ← true
3  moves ← createClassicalNeighbourhoodList(n, e, ℓ, W_ext)
4  while improved = true do
5  |  bestNeighbour ← ∅
6  |  bestNeighbourObj ← some very large number
7  |  for i ← 1 to number of elements in moves do
8  |  |  neighbour ← current
9  |  |  Apply moves(i) on neighbour to create new neighbour          // i-th move
10 |  |  P ← checkFeasibilityAndCalculatePenalty(neighbour, dataset)
11 |  |  Calculate neighbourObj
12 |  |  neighbourObj ← neighbourObj + P
13 |  |  if neighbourObj < bestNeighbourObj then
14 |  |  |  [bestNeighbour, bestNeighbourObj] ← [neighbour, neighbourObj]
15 |  |  end
16 |  end
17 |  if bestNeighbourObj < incumbentObj then
18 |  |  [incumbent, incumbentObj] ← [bestNeighbour, bestNeighbourObj]
19 |  else
20 |  |  improved ← false
21 |  end
22 end
23 return [incumbent, incumbentObj]
```

follow. The function `generateGoodRandomSolution`, for which a pseudo-code listing is given in Function 4.9, illustrates how the initial solution is improved and it replaces the call to function `generateRandomSolution` in line 2 in Algorithm 4.7.

Function 4.9: `generateGoodRandomSolution`(*number*, *dataset*)

Input: The number of solutions to compare, the problem's full dataset
Output: The good random solution vector, the objective function value

```
1  bestObj ← some very large number
2  for i ← 1 to number do
3  |  [solution, solutionObj] ← generateRandomSolution(dataset)
4  |  Apply the local search heuristic in Algorithm 4.8 on [solution, solutionObj]
5  |  if solutionObj < bestObj then
6  |  |  [best, bestObj] ← [solution, solutionObj]
7  |  end
8  end
9  return [best, bestObj]
```

4.3 Chapter summary

The details of the solution methodology employed in this thesis were presented in this chapter. An exact off-the-shelf solution approach to the GMS problem was described in §4.1. The various solvers utilised by the software package LINGO, in which the mathematical programming formulations were implemented, were briefly described.

In §4.2, the algorithmic implementations of a random search heuristic and a simulated annealing algorithm for solving the GMS problem were presented. Functions utilised within these two solution methods were detailed, as well as a new neighbourhood move operator for solutions in a GMS context. Two modifications to the algorithms were also proposed for investigation, the aim being to enhance the solution quality obtained by the algorithms. Finally, pseudo-code listings were provided for all the algorithms and functions in the chapter.

CHAPTER 5

Parameter evaluation

Contents

Three GMS benchmark test systems are presented in this chapter. As the application of the approximate solution approach requires problem instance-dependent parameter settings in the solution techniques, an extensive evaluation of the parameter values is presented in the chapter. This computational evaluation was performed on a personal computer with a 3.0 GHz Intel® Core™ 2 Duo E8400 processor and 3.25 GB RAM, running on Microsoft Windows XP Professional (Version 2002, Service Pack 3). The approximate solution approach was implemented in the software package *MATLAB* [77].

5.1 Benchmark test systems

The solution techniques described in Chapter 4 are applied to two GMS benchmark test systems that have previously been studied in the literature [14, 15, 16, 18, 23, 29, 30]. In addition, a new GMS test system is established here, based on the IEEE-RTS data set [4, 5], and the solution techniques are also applied to this new benchmark instance.

5.1.1 The 21-unit system

Dahal and McDonald [17] created a 21-unit GMS test system loosely derived from the system presented in [104] with some simplifications and additional constraints. The GMS problem assumes reliability as the optimality criterion and the objective is to minimise the sum of squares of the reserve levels over the planning period. This objective attempts to level the reserves over the planning period. Constraints of the test problem are restricted to the adherence to maintenance windows of each unit, the system meeting the load demand and the availability of maintenance crew.

Unit	Capacity (MW)	Earliest starting time (week)	Latest starting time (week)	Duration (weeks)	Manpower required during each week of maintenance
1	555	1	20	7	10, 10, 5, 5, 5, 5, 3
2	555	27	48	5	10, 10, 10, 5, 5
3	180	1	25	2	15, 15
4	180	1	26	1	20
5	640	27	48	5	10, 10, 10, 10, 10
6	640	1	24	3	15, 15, 15
7	640	1	24	3	15, 15, 15
8	555	27	47	6	10, 10, 10, 5, 5, 5
9	276	1	17	10	3, 2, 2, 2, 2, 2, 2, 2, 2, 3
10	140	1	23	4	10, 10, 5, 5
11	90	1	26	1	20
12	76	27	50	3	10, 15, 15
13	76	1	25	2	15, 15
14	94	1	23	4	10, 10, 10, 10
15	39	1	25	2	15, 15
16	188	1	25	2	15, 15
17	58	27	52	1	20
18	48	27	51	2	15, 15
19	137	27	52	1	15
20	469	27	49	4	10, 10, 10, 10
21	52	1	24	3	10, 10, 10

Table 5.1: *Data for the 21-unit test system.*

The test system consists of 21 power generating units with specifications and maintenance requirements as presented in Table 5.1. The planning period covers 52 weeks and the system's peak load demand remains constant at 4739 MW during the entire planning period. The allowed windows, during which maintenance may occur, are either during the first half of the year (weeks 1–26) or during the second half of the year (weeks 27–52). Lastly, a maximum of 20 maintenance personnel are available for maintenance work during each week.

Due to the complexity of the test system, an optimal solution for this problem is still unknown. A theoretical lower bound may be calculated from the average weekly reserve level of 477.6 MW. This average reserve level provides a uniform reserve margin over the planning period, disregarding the discrete unit capacities, maintenance windows and crew constraints [15]. A lower bound for the objective function value (sum of squares of the reserves) is therefore 11 861 100 MW^2.

5.1.2 The 22-unit system

A second GMS test system, containing 22 units, was reportedly [29] first solved by Escudero *et al.* [26] in 1980 using an implicit enumeration algorithm. Two optimality criteria are specified, namely one of reliability and one of economic considerations. However, only the reliability criterion is considered in this thesis. Again, the goal is to level the reserves, but the objective in this GMS test problem is to minimise the sum of absolute differences between the reserve levels during the various time periods and the mean reserve. The planning period covers 52 weeks and the constraints present in the problem include the specification of maintenance windows for each unit, the system meeting the load demand together with a safety margin, the availability of maintenance crew and precedence relationships.

Unit	Capacity (MW)	Earliest starting time (week)	Latest starting time (week)	Duration (weeks)
1	100	1	47	6
2	100	1	50	3
3	100	1	50	3
4	100	1	50	3
5	90	1	47	6
6	90	1	49	4
7	95	1	50	3
8	100	1	49	4
9	650	27	48	5
10	610	6	11	12
11	91	1	49	4
12	100	1	45	8
13	100	1	50	3
14	100	1	47	6
15	220	1	48	5
16	220	1	47	6
17	100	1	48	5
18	100	1	48	5
19	220	1	50	3
20	220	1	50	3
21	240	1	50	3
22	240	1	48	5

Table 5.2: *Data for the 22-unit test system.*

The specifications of this 22 power generating unit system are presented in Table 5.2. Only two crew constraints are applicable for this test system and they are, in fact, exclusion constraints. Units 15 and 16 are not allowed to be in a state of simultaneous maintenance, and neither are units 21 and 22. The precedence constraints for this test problem state that the maintenance of unit 2 has to precede that of unit 3, and that the maintenance of unit 5 has to precede that of unit 6. Table 5.3 contains the weekly peak load demand of the power system. Finally, a minimum safety margin of 20% of the peak load demand has to be maintained throughout the planning period.

Unfortunately, due to the objective function no longer being the sum of squares of the reserve levels, no meaningful lower bound can be obtained for this test system. If one uses the average reserve level of 1791.4 MW, the lower bound is simply zero.

Week	Demand (MW)	Week	Demand (MW)	Week	Demand (MW)	Week	Demand (MW)
1	1 694	14	1 396	27	1 737	40	1 982
2	1 714	15	1 443	28	1 927	41	1 672
3	1 844	16	1 273	29	2 137	42	1 782
4	1 694	17	1 263	30	1 927	43	1 772
5	1 684	18	1 655	31	1 907	44	1 556
6	1 763	19	1 695	32	1 888	45	1 706
7	1 663	20	1 675	33	1 818	46	1 806
8	1 583	21	1 805	34	1 848	47	1 826
9	1 543	22	1 705	35	2 118	48	1 906
10	1 586	23	1 766	36	1 879	49	1 999
11	1 690	24	1 946	37	2 089	50	2 109
12	1 496	25	2 116	38	1 989	51	2 209
13	1 456	26	1 916	39	1 999	52	1 779

Table 5.3: *The weekly peak load demands for the 22-unit system.*

5.1.3 The IEEE-RTS inspired system

In order to fully demonstrate the efficiency and effectiveness of the approximate solution techniques presented in this thesis, a test system containing all the constraint sets in the advanced model formulations, as presented in Appendix A, was additionally created.

In 1979, the *IEEE Reliability Test System* (RTS-79) was developed and published [4] by the Application of Probability Methods Subcommittee of the IEEE Power System Engineering Committee. The report was developed to satisfy the need for standardised data in power system reliability evaluation. It describes a load model, generation system and transmission network. Two revisions of the original test system were published in later years, namely the RTS-86 [5] and the RTS-96 [35] systems.

The author created a test system derived from the load model and generation system in the RTS-79 with additional constraints and parameter values. As in the 21-unit system, the objective in the new system is to level the reserves by minimising the sum of squares of the reserve levels over the planning period (*i.e.* assuming a reliability optimality criterion). The constraints of the system consist of the specification of maintenance windows, the meeting of the load demand together with a safety margin, adhering to the availability of maintenance crew and respecting exclusion constraints.

A total of 32 generating units have to be in maintenance over a planning period of 52 weeks. The specifications of the generating system are presented in Tables 5.4 and 5.5. In Table 5.4, the generating unit capacities and maintenance durations are consistent with the RTS-79 system. A maintenance schedule was included in the RTS-86 revision system to analyse the effect of scheduled maintenance and was derived using a levelled risk criterion [5], complying with the maintenance rate of the original RTS system. Given this schedule, the author roughly extrapolated maintenance windows around each unit's scheduled maintenance for use in the new test system.

The manpower parameters were selected such that the test system is relatively highly constrained by manpower considerations. A maximum of 25 maintenance personnel are available for maintenance work during each week. The exclusion sets presented in Table 5.5 correspond to the units within each power station in the RTS-79. The author selected the maximum number

Unit	Capacity (MW)	Earliest starting time (week)	Latest starting time (week)	Duration (weeks)	Manpower required during each week of maintenance
1	20	1	25	2	7, 7
2	20	1	25	2	7, 7
3	76	1	24	3	12, 10, 10
4	76	27	50	3	12, 10, 10
5	20	1	25	2	7, 7
6	20	27	51	2	7, 7
7	76	1	24	3	12, 10, 10
8	76	27	50	3	12, 10, 10
9	100	1	50	3	10, 10, 15
10	100	1	50	3	10, 10, 15
11	100	1	50	3	15, 10, 10
12	197	1	23	4	8, 10, 10, 8
13	197	1	23	4	8, 10, 10, 8
14	197	27	49	4	8, 10, 10, 8
15	12	1	51	2	4, 4
16	12	1	51	2	4, 4
17	12	1	51	2	4, 4
18	12	1	51	2	4, 4
19	12	1	51	2	4, 4
20	155	1	23	4	5, 15, 10, 10
21	155	27	49	4	5, 15, 10, 10
22	400	1	21	6	15, 10, 10, 10, 10, 5
23	400	27	47	6	15, 10, 10, 10, 10, 5
24	50	1	51	2	6, 6
25	50	1	51	2	6, 6
26	50	1	51	2	6, 6
27	50	1	51	2	6, 6
28	50	1	51	2	6, 6
29	50	1	51	2	6, 6
30	155	1	23	4	12, 12, 8, 8
31	155	1	49	4	12, 12, 8, 8
32	350	1	48	5	5, 10, 15, 15, 5

Table 5.4: *Data for the IEEE inspired test system.*

of units allowed in simultaneous maintenance within each set as being no more than half of the units in each set. The load model is taken directly from the RTS-79 system and has a peak load

Exclusion set	Units	Maximum
1	1, 2, 3, 4	2
2	5, 6, 7, 8	2
3	9, 10, 11	1
4	12, 13, 14	1
5	15, 16, 17, 18, 19, 20	3
6	24, 25, 26, 27, 28, 29	3
7	30, 31, 32	1

Table 5.5: *Exclusion data for the IEEE inspired system.*

demand of 2850 MW during week 51. Table 5.6 contains the weekly peak load demand of the power system and it represents a typical pattern with two seasonal peaks [4]. Finally, a safety

margin of 15% of the peak load demand has to be maintained throughout the planning period, as selected by the author.

Week	Demand (MW)	Week	Demand (MW)	Week	Demand (MW)	Week	Demand (MW)
1	2 457	14	2 138	27	2 152	40	2 063
2	2 565	15	2 055	28	2 326	41	2 118
3	2 502	16	2 280	29	2 283	42	2 120
4	2 377	17	2 149	30	2 508	43	2 280
5	2 508	18	2 385	31	2 058	44	2 511
6	2 397	19	2 480	32	2 212	45	2 522
7	2 371	20	2 508	33	2 280	46	2 591
8	2 297	21	2 440	34	2 078	47	2 679
9	2 109	22	2 311	35	2 069	48	2 537
10	2 100	23	2 565	36	2 009	49	2 685
11	2 038	24	2 528	37	2 223	50	2 765
12	2 072	25	2 554	38	1 981	51	2 850
13	2 006	26	2 454	39	2 063	52	2 713

Table 5.6: *The weekly peak load demands for the IEEE inspired system.*

A theoretical lower bound for the objective function value (sum of squares of the reserves) has been determined from the average weekly reserve level of 801 MW. This lower bound is 33 363 252 MW^2.

5.2 The penalty weights

As described in §4.2.1, a soft constraint approach is adopted in the approximate solution approach for the GMS problem. Accordingly, a penalty value defined in (4.5) is added to the objective function value associated with a candidate solution that violates any of the constraints. The penalty weights within this expression are typically problem instance-dependent and must therefore be calculated for each test system accordingly. The method that was used to determine these weights is described below.

Consider a hard constraint approach (*i.e.* all the constraints of the problem have to be satisfied). A feasible initial solution is determined via a selected solution technique, either by setting the penalty weights in the soft approach to be extremely large (in order to strongly discourage infeasible solutions) or by only using hard constraints. The latter option may take an unpractical amount of time to obtain such a solution if the problem is heavily constrained.

Each constraint set is allowed to be violated (*i.e.* viewed as a soft constraint set) while keeping all the other constraint sets hard (*i.e.* assuming extremely large penalty weights for these constraint sets). The problem instance is then solved over a range of penalty weights for that constraint set. This range is the problem instance-specific element in the process as the objective function values for different problem instances will vary in order of magnitude and the problem instances may be more (or less) constrained, thereby eliminating the use of a general rule of thumb for the GMS problem. As the weights in the range become larger, more incumbent solutions will be feasible (*i.e.* have a zero penalty value) and the aim is to select a weight such that most incumbent solutions returned by a solution technique are feasible. It is not desirable that the weight should be too large as this discourages a broad exploration of the solution space of

a problem, but also not too small as this increases the likelihood of an infeasible incumbent solution. The process is repeated for each constraint set, resulting in penalty weights for each set to use in the original soft constraint approach.

The results of each test system's penalty weight analysis is presented below. For each weight in each range, a total of 20 problem instances were solved in order to obtain average objective function and penalty values as well as the frequency of infeasible solution occurences.

5.2.1 The 21-unit system

The constraint sets involved in the 21-unit system, as presented in §5.1.1, include adherence to specified maintenance windows, load demand requirements and maintenance crew limitations. The window penalty weights considered ranged from 250 000 to 2 000 000 in increments of 250 000. In Figure 5.1, the average and minimum objective function values of both feasible and infeasible incumbent solutions obtained over this range of weights in the 20 instances are illustrated. The frequency of infeasible solutions was not significant enough to represent visually — at the weights of 250 000 and 500 000 there were three and one infeasible solutions, respectively, and for the remaining weights all solutions were feasible. Based on these results, the window penalty weight was chosen as 500 000 to ensure a high probability of a feasible incumbent solution while maintaining an opportunity for the solution technique to explore the infeasible solution space.

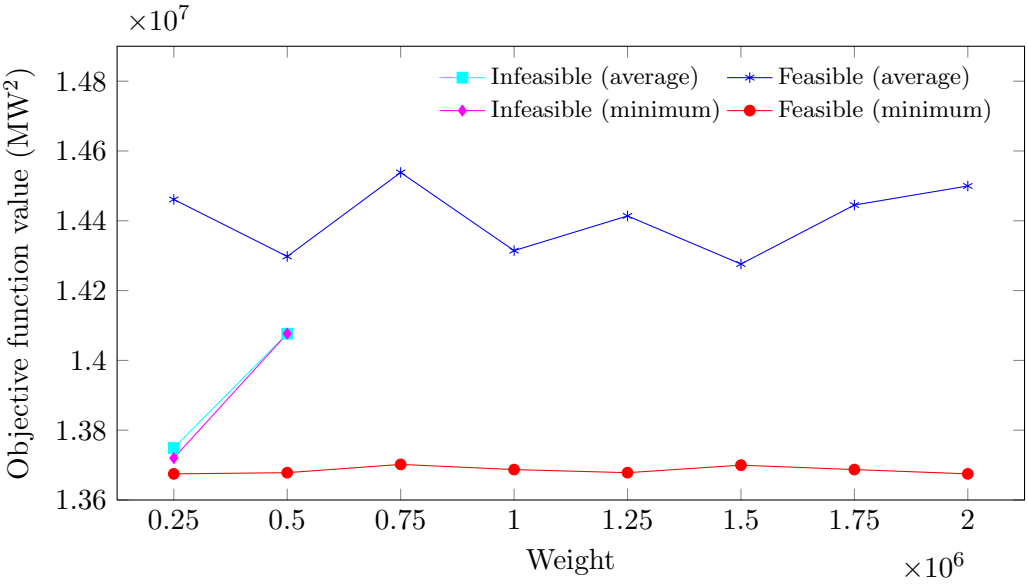
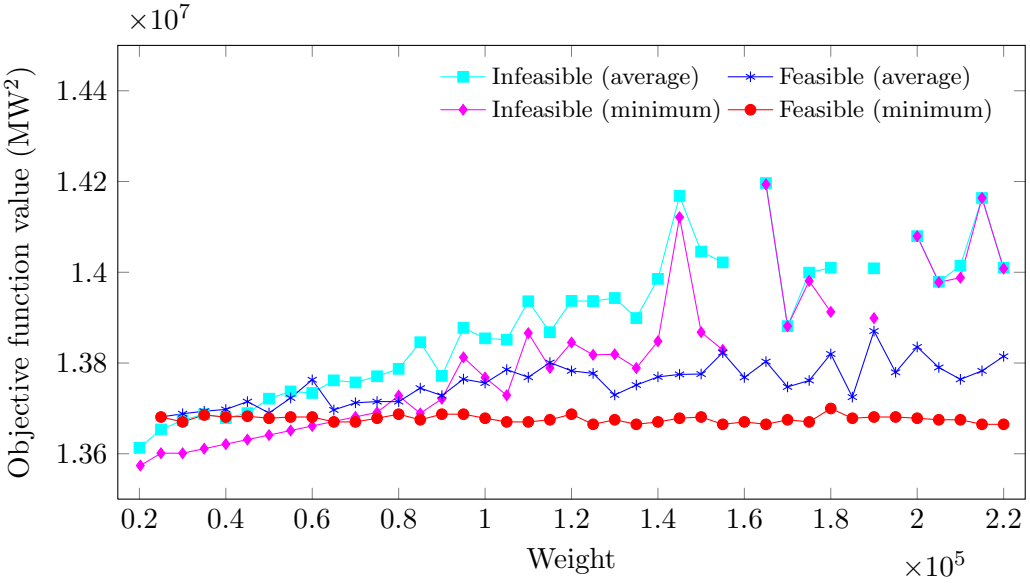


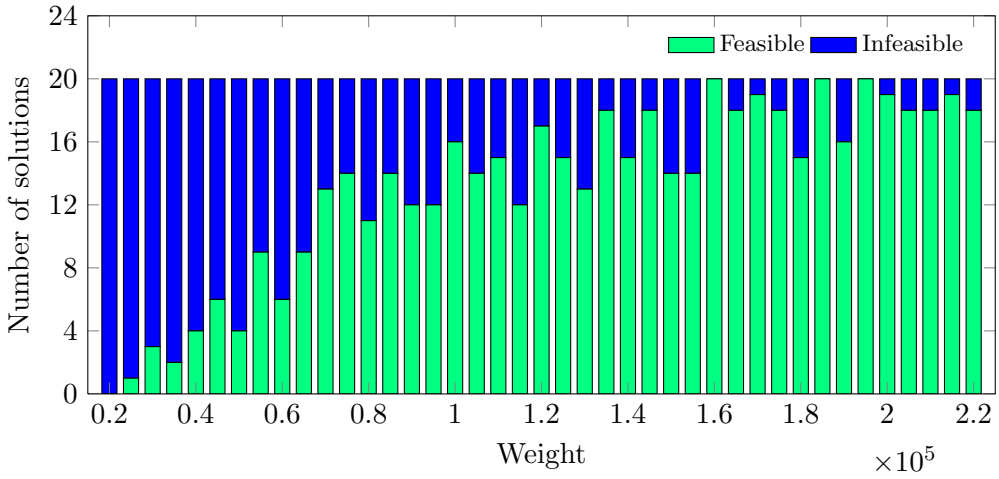
Figure 5.1: *Penalty weight analysis involving the maintenance window constraint set for the 21-unit system.*

The penalty weight for the load demand constraint set was chosen as 1, because the analysis uncovered a single infeasible solution at a weight of 1 with no infeasible solutions present in the rest of the range (intervals of 100 up to 1 000). An advantage of having a load demand penalty term present is that the constraint violation has the same scaling as the reserve used in the objective function. Therefore, it does not come as a surprise that the penalty weight is chosen as 1, since the penalty requires no scaling.

For the penalty weights of the maintenance crew constraint set, values ranging from 20 000 to 220 000 in increments of 5 000 were considered. The graph in Figure 5.2(a) shows the average and minimum objective function values of both feasible and infeasible incumbent solutions obtained over the range of weights. The minimum feasible objective function values only cross the minimum infeasible objective function values at a weight of 65 000 and as such, weights below that point do not have to be considered. From there onwards, the minimum infeasible values slowly deteriorate, while the minimum feasible values remain approximately level. Purely from the frequency perspective, one would at least want 75% of the incumbent values to be feasible. As indicated in Figure 5.2(b), this only occurs consistently at a weight of 160 000 or more. The penalty weight for the maintenance crew constraint set was therefore chosen as 200 000 to ensure a high probability of a feasible incumbent solution while maintaining an opportunity for the solution technique to explore the infeasible solution space.



(a) Objective function values of the average and minimum feasible and infeasible solutions.



(b) The number of feasible and infeasible solutions.

Figure 5.2: *Penalty weight analysis involving the maintenance crew constraint set for the 21-unit system.*

5.2.2 The 22-unit system

The constraint sets involved in the 22-unit system, as presented in §5.1.2, include adherence to specified maintenance windows, load demand with safety margin requirements, respecting exclusion constraints and adherence to precedence relationships. However, the precedence relationships were implemented as hard constraints in order to restrict the solution techniques to consider constraints specified in the advanced GMS model formulation presented in §3.4 only.

The penalty weights for the maintenance window constraint set and exclusion constraint set ranged from 10 to 100 in increments of 10. In both cases, the analysis uncovered intermittent single occurences of infeasible solutions spread over the entire range of weights (5 and 3 occurences, respectively). As a result, the penalty weight for the maintenance window constraint set was chosen as 10 and the penalty weight for the exclusion constraint set was also chosen as 10.

For the load demand with safety margin constraint set, the penalty weight ranged from 1 to 10 and the analysis uncovered no infeasible solutions. Therefore, the penalty weight for the load demand with safety margin constraint set was chosen as 1.

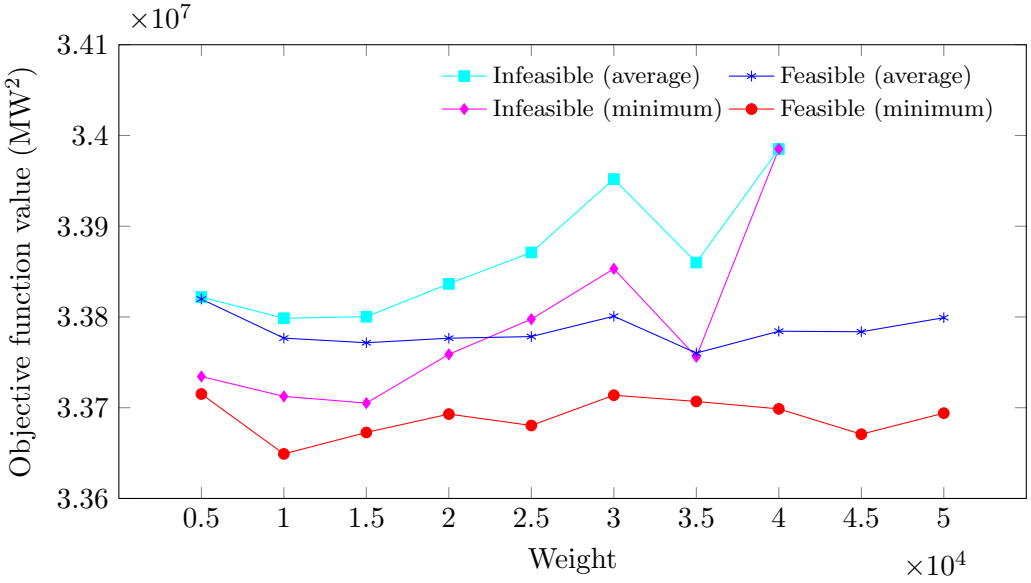
5.2.3 The IEEE-RTS inspired system

Lastly, the constraint sets involved in the IEEE-RTS inspired system, as presented in §5.1.3, include adherence to specified maintenance windows, load demand with safety margin requirements, restrictions on maintenance crew availability and respecting exclusion constraints. The weights for the maintenance window constraint set ranged from 5 000 to 50 000 in increments of 5 000. As illustrated in Figure 5.3(b), the frequency of feasible solutions only becomes acceptable from a weight of 20 000 and above. The difference between the minimum feasible and infeasible (as well as average feasible and infeasible) objective function values also increases more from this weight on, as illustrated in Figure 5.3(a). In view of the above information, the penalty weight of the maintenance window constraint set was chosen as 40 000 to ensure that a high probability of the minimum objective function value is obtained from a feasible solution.

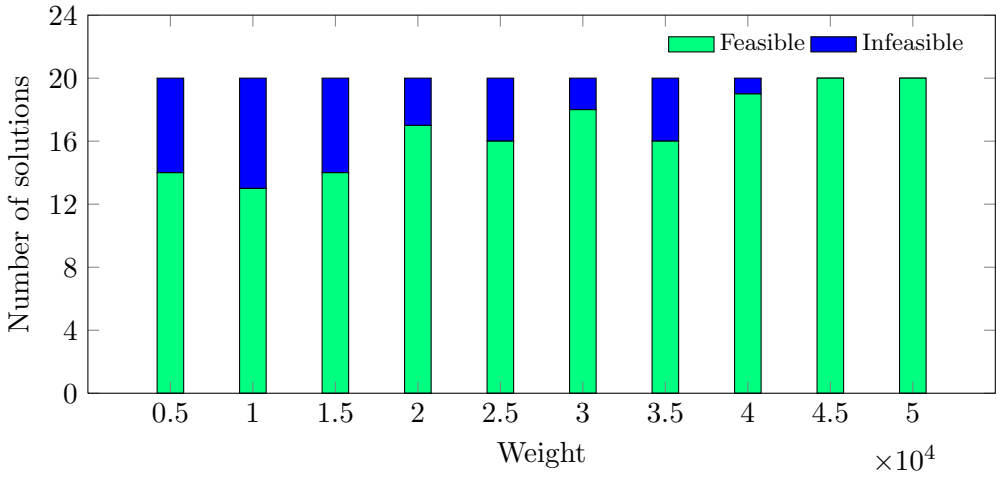
Again, the penalty weight for the load demand with safety margin was chosen as 1, because the analysis uncovered no infeasible solutions over an interval of values ranging from 1 to 500 in increments of 100.

In the analysis involving the maintenance crew constraint set, penalty weight values ranging from 2 500 to 20 000 in increments of 2 500 were considered. The graph in Figure 5.4(a) shows that the minimum and average infeasible objective function values follow a steady deterioration, except at the weight of 12 500, with the difference between the minimum feasible and infeasible objective function values being very small. Even though the minimum feasible objective function values over the range are mostly smaller than this infeasible value, one should be careful not to choose a weight around this point as it indicates that outlier behaviour still occurs. From the frequency results in Figure 5.4(b), a progression from an entire set of infeasible solutions to an entire set of feasible solutions over a relatively short range is visible. Since the differences in objective function values are not very large (as mentioned above), the choice of penalty weight for the maintenance crew constraint set was largely based on the frequency result and was therefore chosen as 20 000 to ensure a feasible incumbent solution with very highy likelihood.

The penalty weights for the exclusion constraint set ranged from 2 500 to 30 000 in increments of 2 500 (similar to the range of the maintenance crew penalty weights discussed above). The



(a) Objective function values of the average and minimum feasible and infeasible solutions.



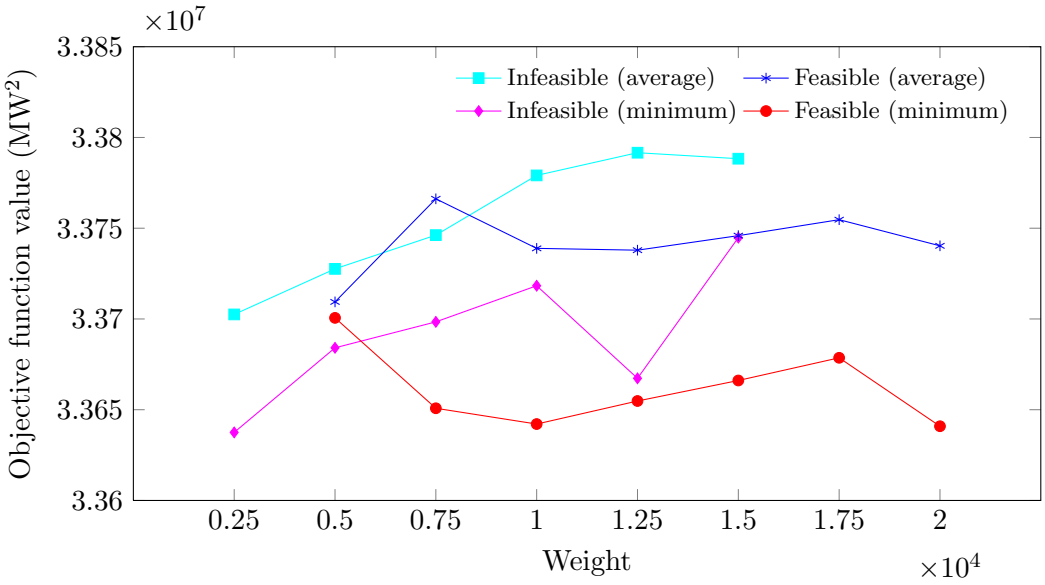
(b) The number of feasible and infeasible solutions.

Figure 5.3: *Penalty weight analysis involving the maintenance window constraint set for the IEEE-RTS inspired system.*

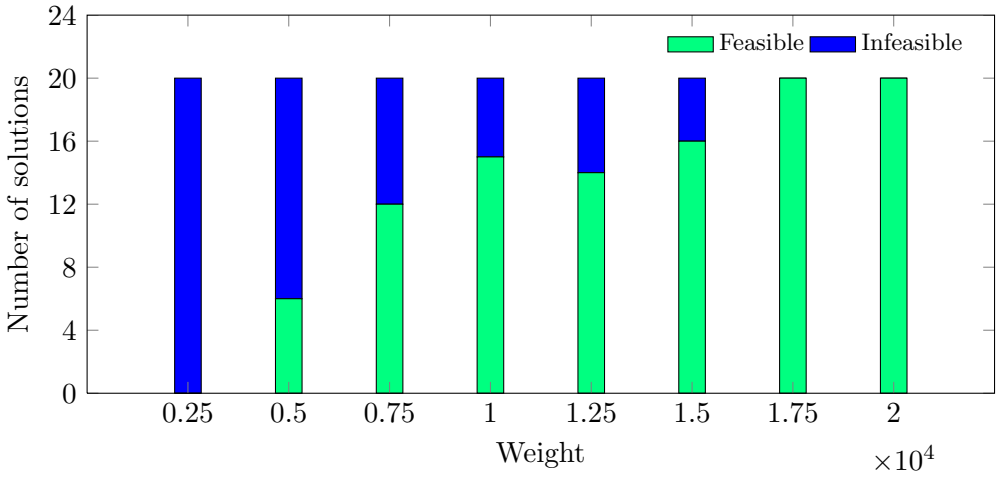
minimum and average objective function values of the feasible and infeasible solutions are presented in Figure 5.5(a). As before, a clear deterioration pattern may be seen with a sharp increase in infeasible objective function values from a weight of 12 500 and above. It is also from this weight onwards that the frequency of feasible incumbent solution become acceptable. The penalty weight of the exclusion constraint set was chosen as 20 000 because of the large difference in minimum feasible and infeasible objective function values, while maintaining the opportunity for the solution technique to explore the infeasible solution space.

5.3 Parameter optimisation

The methodology followed and results obtained by choosing the best parameter values for each solution technique are described in this section. These values are typically problem instance-



(a) Objective function values of the average and minimum feasible and infeasible solutions.

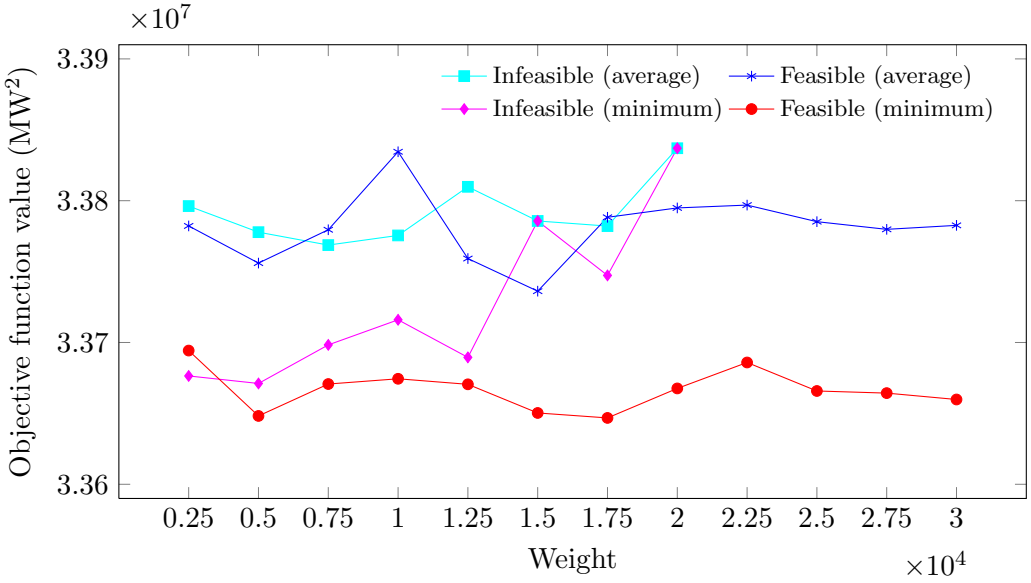


(b) The number of feasible and infeasible solutions.

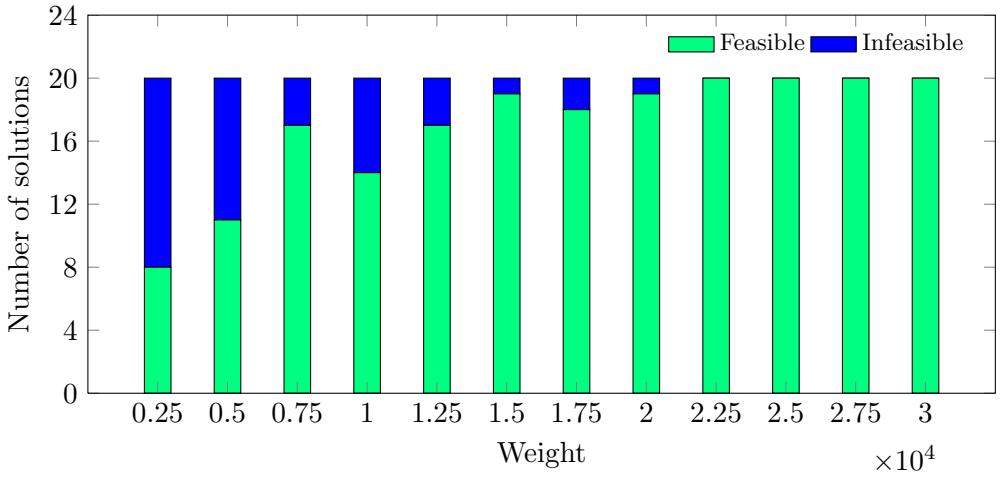
Figure 5.4: *Penalty weight analysis involving the maintenance crew constraint set for the IEEE-RTS system.*

dependent and therefore have to be analysed for each benchmark test system separately. Furthermore, since two different neighbourhood structures are compared in this thesis, the parameter values may be different for each structure in the solution techniques. Within this section, only the results of the ejection chain neighbourhood structure are presented in detail. The parameter values for the classical neighbourhood are simply presented in table form at the conclusion of this section. However, the classical neighbourhood underwent the same analysis as the ejection chain neighbourhood.

The following notation regarding each test system and solution technique is used in the remainder of this chapter. An instance of the 21-unit test system, solved by the random search heuristic using the ejection chain neighbourhood, is referred to as *21-RS-E*, while when using the classical neighbourhood, is referred to as *21-RS-C*. For the simulated annealing algorithm, the notations are *21-SA-E* and *21-SA-C*. Likewise for the 22-unit test system, the notations are



(a) Objective function values of the average and minimum feasible and infeasible solutions.



(b) The number of feasible and infeasible solutions.

Figure 5.5: *Penalty weight analysis involving the exclusion constraint set for the IEEE-RTS system.*

22-RS-E, *22-RS-C*, *22-SA-E* and *22-SA-C*. Finally, the notations for the IEEE-RTS inspired test system are *IEEE-RS-E*, *IEEE-RS-C*, *IEEE-SA-E* and *IEEE-SA-C*.

5.3.1 Random search heuristic

The random search heuristic has three parameters that may be optimised, namely the neighbourhood size, the number of iterations, I, in a solution instance and the number of iterations without an improving solution (the latter two being the only termination criteria). A total of 50 problem instances were solved for each parameter value combination in order to obtain average results.

The ejection chain neighbourhood size was varied over the range $\{0.5n, n, 1.5n, 2n\}$, the number of iterations I from 1 000 to 10 000 in increments of 1 000 and the number of iterations without improvement over the range $\{0.5I, 0.6I, 0.7I, 0.8I, 0.9I, I\}$, thus giving rise to 240 parameter

combinations. The reason for choosing the neighbourhood size in factors of n is to relate the number of generated neighbours to the number of units in the problem, since a neighbour (ejection chain) starts at a randomly chosen unit. For example, if the neighbourhood size is n, one may roughly think of every unit as being a starting point for an ejection chain (although this is not truly the case due to the use of random choice).

For the classical neighbourhood, its size was varied over the range $\{m, 1.5m, 2m, 2.5m, 3m\}$, while the number of iterations, I, and the number of iterations without improvement was varied in the same fashion as described above. This gave rise to 300 parameter combinations. The neighbourhood size was chosen as factors of m to relate the number of created neighbours to the number of units in the problem and their maintenance windows, since the maintenance of a unit during any time period in the year has m possible starting points. For example, if the neighbourhood size is $2m$, one may roughly think of the situation as two units whose starting times are varied over their entire allowable range (although this is not truly the case due to the use of random choice and the fact that many maintenance windows are smaller than m).

The 21-unit system

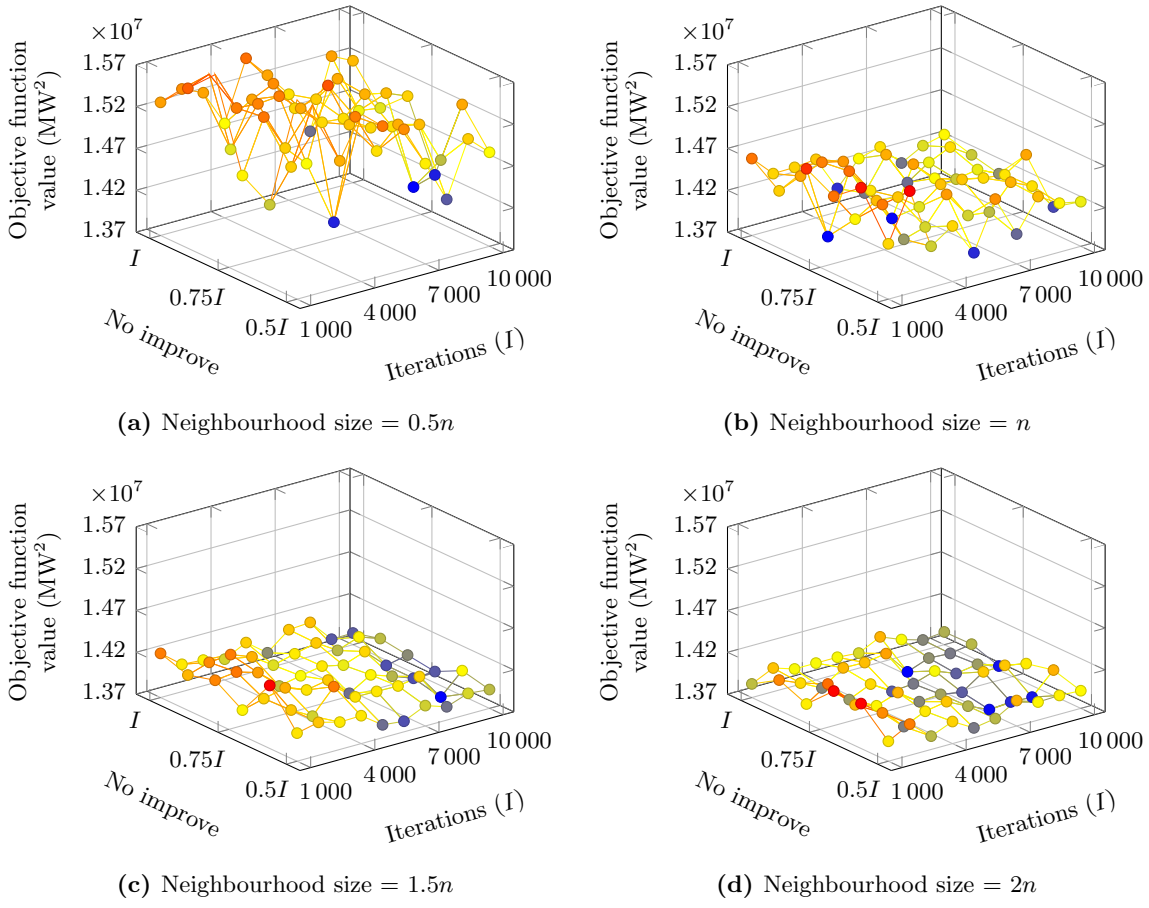
The progression of how the neighbourhood size affects the minimum incumbent objective function value of the problem is illustrated in Figure 5.6. A high level of degeneracy in solution quality is observed at the smaller two neighbourhood sizes. As the neighbourhood size increases, the objective function values level off at much smaller values than at the larger neighbourhood sizes. The axes are all scaled the same over the neighbourhood sizes. Therefore, the poor objective function values at the small neighbourhood size are easily visible. Solely based on these four graphs, the neighbourhood size should be chosen as $2n$.

In Figure 5.7, additional confirmation with respect to the choice of $2n$ as neighbourhood size is provided. The graphs illustrate the progression of the average penalty values corresponding to the average incumbent solutions. Most likely, none of the incumbent solutions were feasible (nonzero penalty) at a neighbourhood size of $0.5n$, as indicated in Figure 5.7(a). In contrast, at a neighbourhood size of $2n$, as shown in Figure 5.6(d), most of the incumbent solutions after 7 000 iterations were feasible — a result not easily detected from Figure 5.6. The effect of the number of iterations without improvement is yet to be seen in the results.

The average solution times required at a neighbourhood size of $2n$, restricted by the number of iterations ranging between 7 000 and 10 000, are considered in Figure 5.8. The average solution times level off after about $0.7I$ iterations without improvement. This is an indication that the incumbent solution does not necessarily improve if more iterations are performed. From this observation, in combination with the feasibility observation from Figure 5.6(d), the number of iterations was chosen as 7 000 and the number of iterations without an improving solution was chosen as $0.8I$. There is not enough of a benefit in the solution quality beyond this choice to warrant more of a trade-off with the average solution time.

The 22-unit system

The effect of the neighbourhood size on the minimum incumbent objective function value of the problem is illustrated in Figure 5.9. Unlike the results in 21-RS-E, no significant degeneracy is present in these objective function values over all the neighbourhood size instances. However, the observation is still made of the dramatic improvement in collective objective function values as the neighbourhood size increases to $2n$. Again, this progression of graphs is more than sufficient to warrent a choice of the neighbourhood size as $2n$.



(a) Neighbourhood size = $0.5n$

(b) Neighbourhood size = n

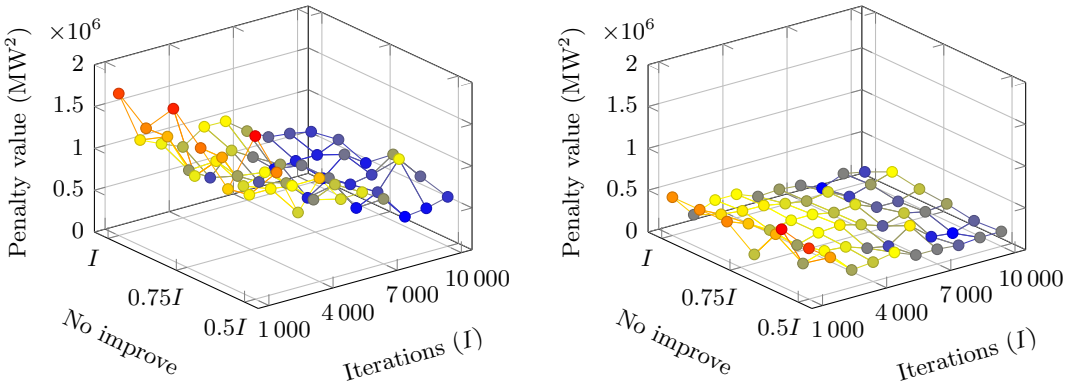
(c) Neighbourhood size = $1.5n$

(d) Neighbourhood size = $2n$

Figure 5.6: *The minimum incumbent objective function value for each parameter combination in 21-RS-E.*

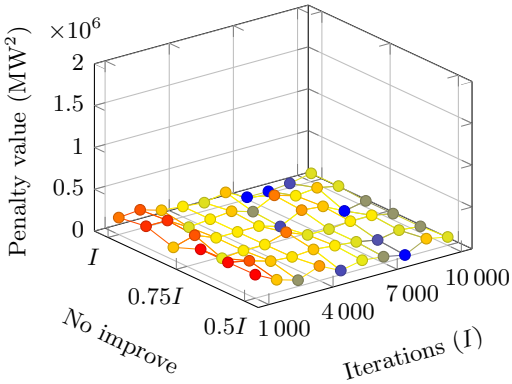
In the 22-RS-E instance case, the graphs of the average penalty values corresponding to the average incumbent solutions provide inconclusive results and therefore, no conclusions may be drawn from them. However, in Figure 5.10, the progression of the average incumbent objective function values as the neighbourhood size increases is illustrated. The collective improvement in objective function values is much more noticable in these four graphs, but more importantly, the superior solution quality obtained at the top end of the number of iterations is clear to see, even at the small neighbourhood size of $0.5n$ in Figure 5.10(a). An acceptable range of choice would therefore be between 7 000 and 10 000 iterations.

The graph in Figure 5.11 illustrates how the average solution times vary at a neighbourhood size of $2n$ and a total number of iterations, I, of between 7 000 and 10 000 (the acceptable range). Note how the solution times level off from $0.8I$ iterations without an improvement above this value, indicating that there is little benefit in allowing more iterations. Since no noticable improvement in solution quality may be seen between the 9 000 and 10 000 iteration counts, a value of 9 000 iterations was chosen as the number of iterations, since the associated solution time is less. Furthermore, as illustrated in Figure 5.11, the number of iterations without an improvement was chosen as $0.8I$.

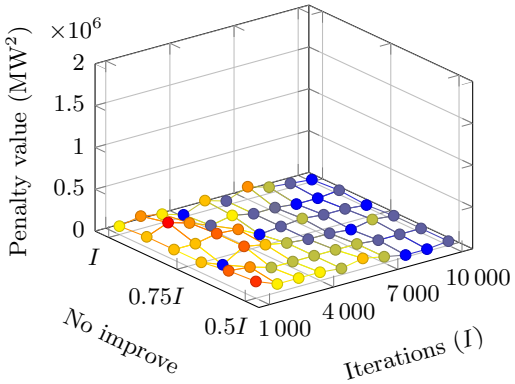


(a) Neighbourhood size = $0.5n$

(b) Neighbourhood size = n



(c) Neighbourhood size = $1.5n$



(d) Neighbourhood size = $2n$

Figure 5.7: *The average incumbent penalty value for each parameter combination in 21-RS-E.*

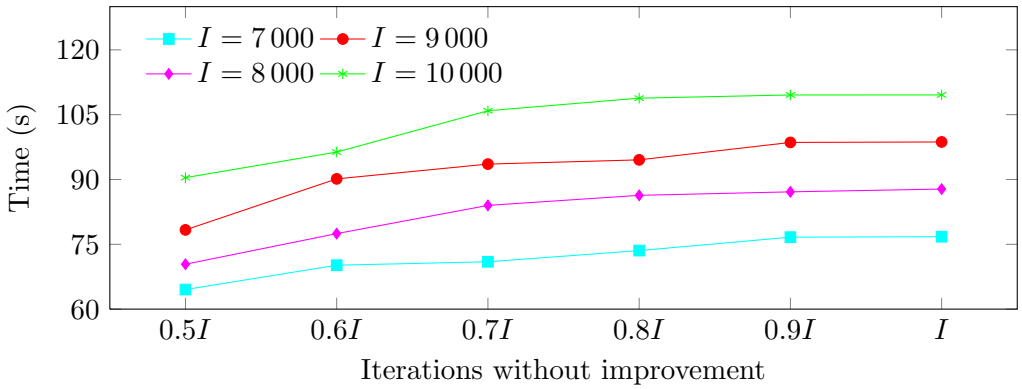
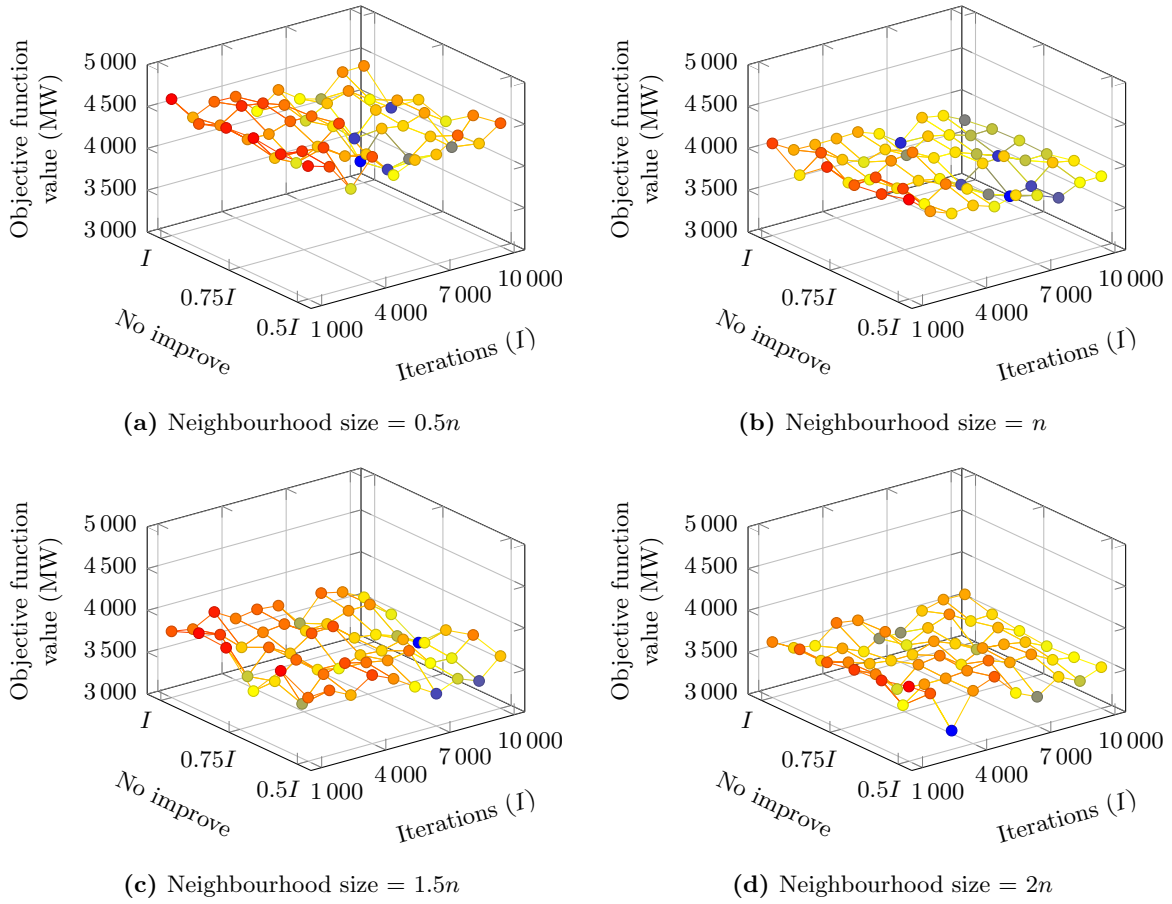


Figure 5.8: *The average solution times for a truncated set of parameter combinations in 21-RS-E.*

The IEEE-RTS inspired system

The average incumbent objective function values are presented in Figure 5.12, with the subgraphs progressing over the neighbourhood size. Unlike the corresponding graphs in the 21- and 22-unit systems, the effect of the neighbourhood size is not apparent at all, as the objective function values all lie within the same range. However, what is noticable is the continuous improvement in solution quality up to the limit of the iteration range of 10 000.



(a) Neighbourhood size = $0.5n$

(b) Neighbourhood size = n

(c) Neighbourhood size = $1.5n$

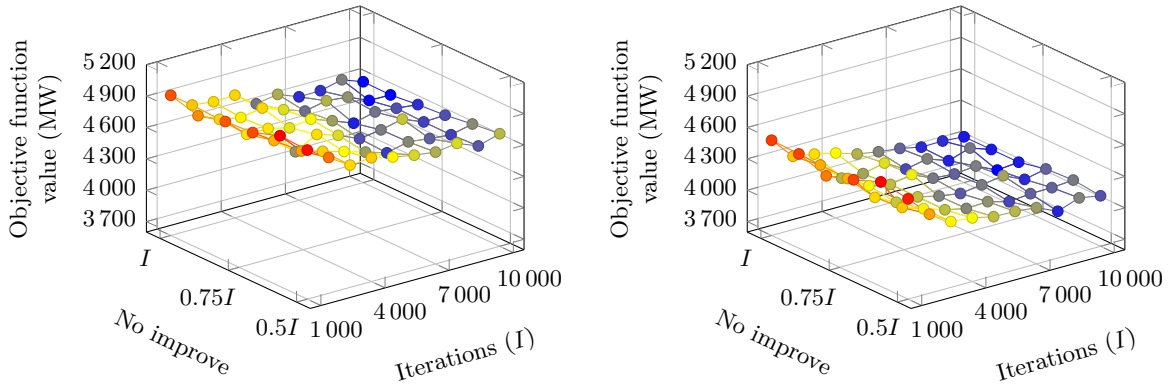
(d) Neighbourhood size = $2n$

Figure 5.9: *The minimum incumbent objective function values for each parameter combination in 22-RS-E.*

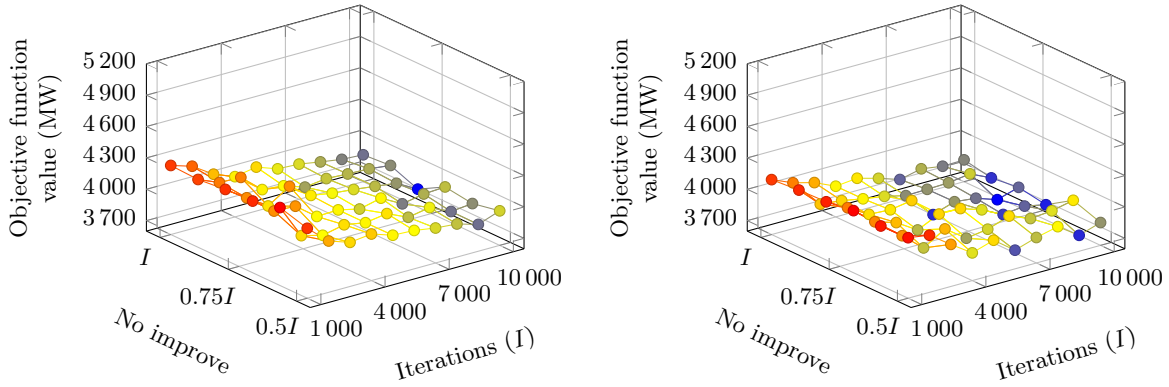
In order to see this effect more clearly, assume that the maximum number of iterations without an improving solution does not play a role and consider determining its average objective function value at each iteration count. The resulting graph is displayed in Figure 5.13. Now, the continuous improvement in average incumbent objective function value over the number of iterations is apparent. As such, the number of iterations may be chosen as 10 000 in order to obtain the best solution quality.

In Figure 5.14, the average solution times for each neighbourhood size, at an iteration count of 10 000 is considered. The graph shows that the average solution times remain level after about $0.7I$ iterations without improvement. As before, this indicates that the solution technique does not necessarily improve upon the incumbent solution if the search continues for longer. From this observation, the number of iterations without an improving solution was chosen as $0.8I$.

As mentioned above, the neighbourhood size seems not to have an apparent effect on the solution quality. Due to this observation, the choice in neighbourhood size may be restricted to $0.5n$ and n, based on the average solution times in Figure 5.14. If the attention is turned to the minimum incumbent objective function value obtained by these two neighbourhood sizes in Figure 5.15, it is observed that the neighbourhood size of n obtained the superior results between the two sizes. Therefore, the neighbourhood size was chosen as n.



(a) Neighbourhood size = $0.5n$ **(b)** Neighbourhood size = n



(c) Neighbourhood size = $1.5n$ **(d)** Neighbourhood size = $2n$

Figure 5.10: *The average incumbent objective function values for each parameter combination in 22-RS-E.*

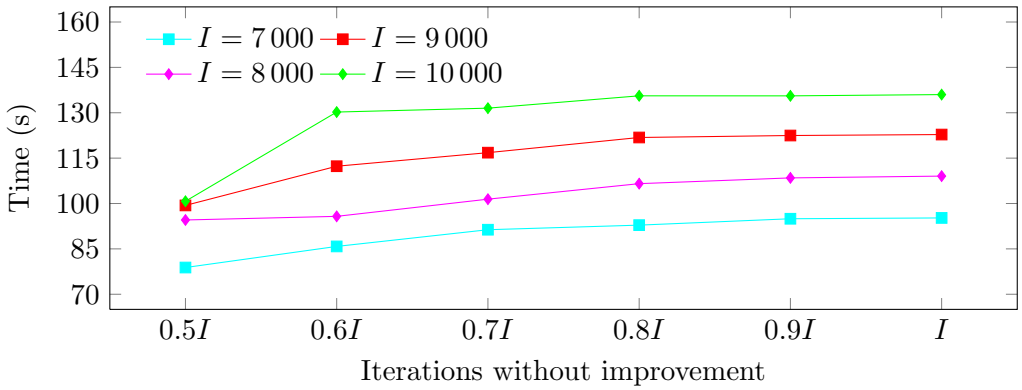
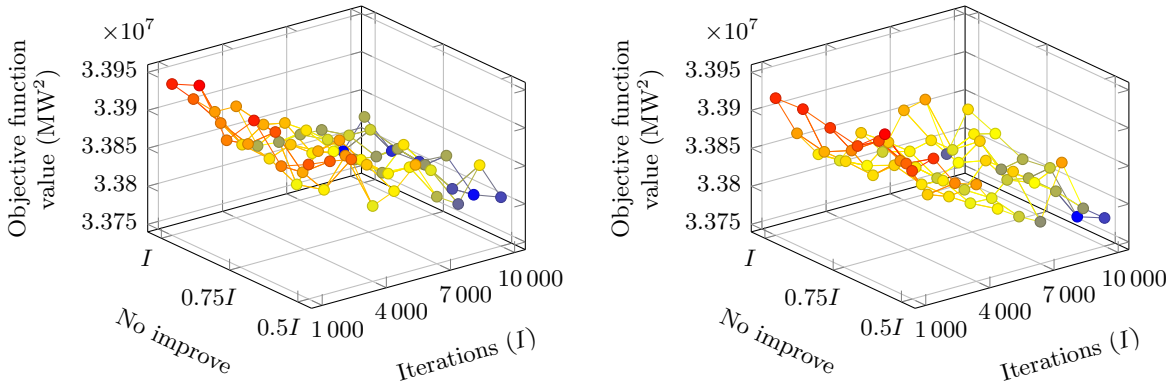


Figure 5.11: *The average solution times for a truncated set of parameter combinations in 22-RS-E.*

5.3.2 Simulated annealing algorithm

The simulated annealing algorithm employs two sets of parameters. The first set contains the parameters involved in the specific cooling schedule, while the second set contains the termination criteria parameters of the algorithm. A total of 50 problem instances were solved for each cooling schedule parameter value combination, followed by 20 problem instances for each termination criteria parameter combination in order to obtain average results.



(a) Neighbourhood size = $0.5n$

(b) Neighbourhood size = n

(c) Neighbourhood size = $1.5n$

(d) Neighbourhood size = $2n$

Figure 5.12: *The average incumbent objective function values for each parameter combination in IEEE-RS-E.*

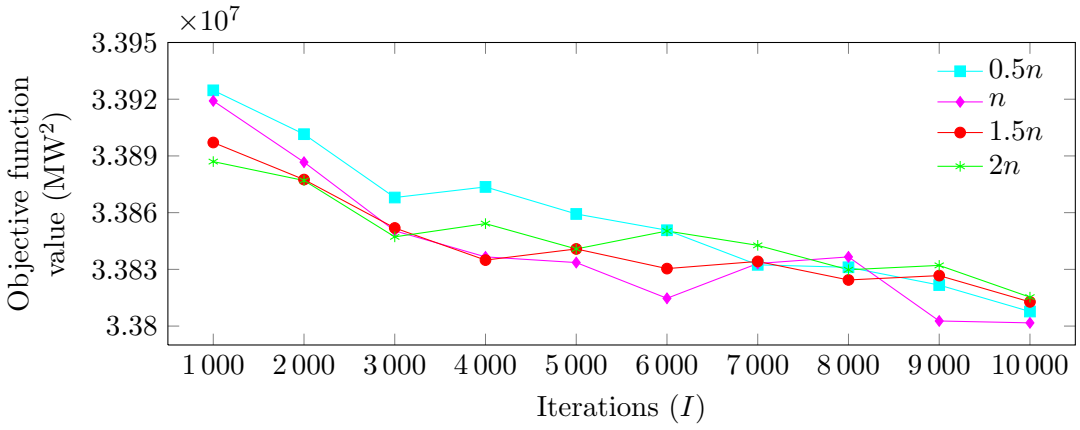


Figure 5.13: *The effect of the number of iterations on the average incumbent objective function values in IEEE-RS-E when the number of iterations without an improving solution is averaged out.*

As stated in §4.2.5, four different cooling schedules are investigated in this thesis, each employing its own parameters. An initial temperature parameter T_0 is present in all four schedules and may be determined by the two methods described in §4.2.5, herafter referred to as the *average increase method* (AIM) and the *standard deviation method* (SDM). In the schedule proposed by

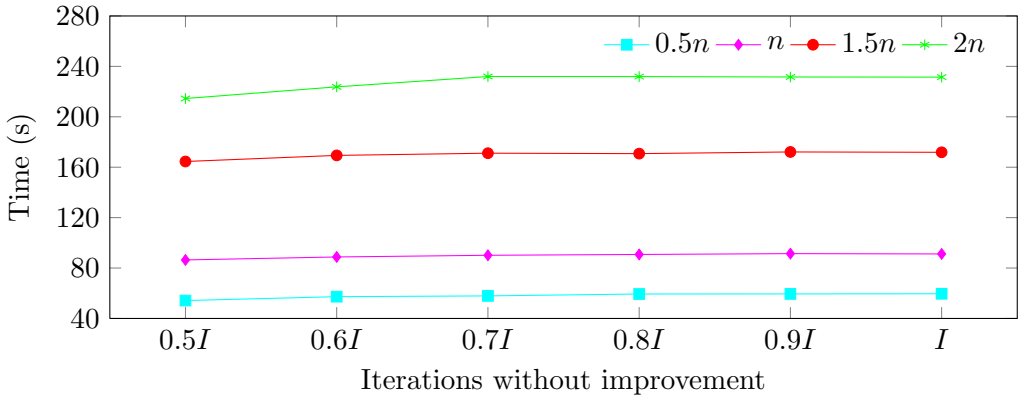
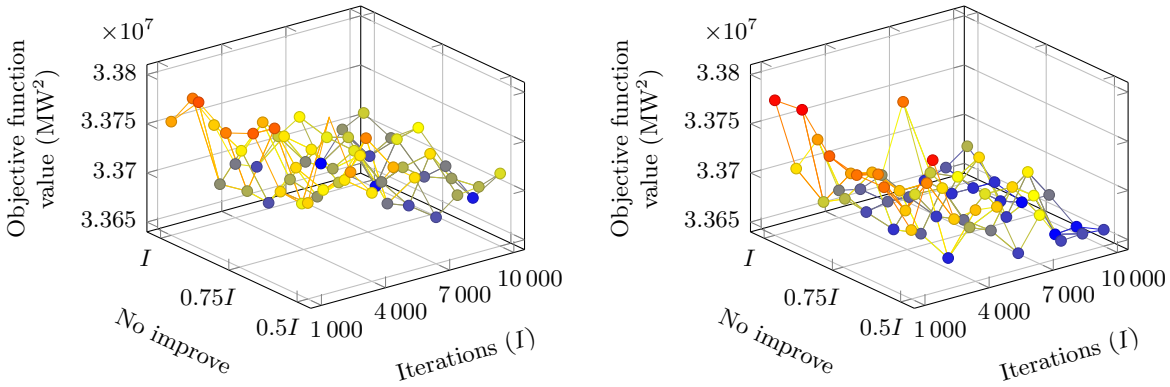


Figure 5.14: *The average solution times for a truncated set of parameter combinations in IEEE-RS-E.*



(a) Neighbourhood size = $0.5n$ **(b)** Neighbourhood size = n

Figure 5.15: *The minimum incumbent objective function values obtained from two neighbourhood sizes in IEEE-RS-E.*

Triki *et al.* [79] the initial temperature has to be determined by the SDM. Therefore, the initial temperature was only varied over the other three schedules between the AIM and the SDM. The ranges for each parameter in the different cooling schedules, were varied as follows:

- For the geometric schedule, the parameter α was varied over the range $\{0.8, 0.81, \ldots, 0.99\}$ for 21-SA-E and 22-SA-E, and over the range $\{0.8, 0.82, \ldots, 0.96\}$ for IEEE-SA-E.
- The parameter λ within the schedule proposed by Huang *et al.* [37] was varied over the range $\{0.5, 0.52, \ldots, 0.9\}$ for 21-SA-E and 22-SA-E, and over the range $\{0.5, 0.54, \ldots, 0.9\}$ for IEEE-SA-E.
- In the schedule proposed van Van Laarhoven *et al.* [81] the parameter δ was varied over the range $\{0.1, 0.12, \ldots, 0.7\}$ for 21-SA-E and 22-SA-E, and over the range $\{0.1, 0.15, \ldots, 0.7\}$ for IEEE-SA-E.
- Lastly, in the schedule proposed by Triki *et al.* [79] the parameter ζ was varied over the range $\{1.02, 1.04, 1.06, 1.08\}$ for 21-SA-E and 22-SA-E, and over the range $\{1.02, 1.04, 1.06\}$ for IEEE-SA-E. For all three problems, the parameters μ_1 and μ_2 were varied over the range $\{5, 10, 15\}$.

Identical parameter ranges were used in the analysis of the instances with the classical neighbourhood move operator.

Three termination criteria parameters are present in the SA algorithm. These are the final temperature T_{min}, the maximum number of attempted solutions during each temperature stage (hereafter referred to as *max_attempt*) and the maximum number of accepted solutions during each temperature stage. However, to reduce the number of adjustable parameters, the maximum number of accepted solutions was fixed at 12% of the maximum number of attempted solutions (a generalisation of the proposed number suggested in [20] and proposed in §4.2.5). Therefore, only the final temperature and the maximum number of attempted solutions during each temperature stage were varied — T_{min} over the range $\{0.5T_0, 0.4T_0, 0.3T_0, 0.2T_0, 0.1T_0, 1\}$ and *max_attempt* over the range $\{10n, 20n, \ldots, 100n\}$.

Before the analysis results for each test system is reported, a concept regarding the initial temperature analysis is introduced. Since a problem instance is solved for each possible parameter combination, a minimum and average incumbent objective function value is obtained for both initial temperature methods at the same cooling schedule parameter. In the analysis, the objective function value obtained by the SDM is subtracted from the objective function value obtained from the AIM. Since the GMS problem being considered is a minimisation problem, a *positive difference* necessarily indicates that the SDM obtained the better solution, while a *negative difference* indicates the AIM obtained the better solution. These differences over the entire parameter range are then used to determine which method was superior. Furthermore, the average solution times do not factor into the decision process as both methods take approximately the same amount of time to execute.

Finally, a number of graphs below have two vertical axes — in all such cases, time is measured on the right-hand axis and all other data on the left-hand axis.

The 21-unit system

Consider firstly the geometric cooling schedule. In Figure 5.16, the difference in incumbent objective function values (average and minimum) between using the AIM and the SDM to calculate the initial temperature is illustrated. Since the negative bars are more prevalent in this case, the AIM was chosen for determining the initial temperature.

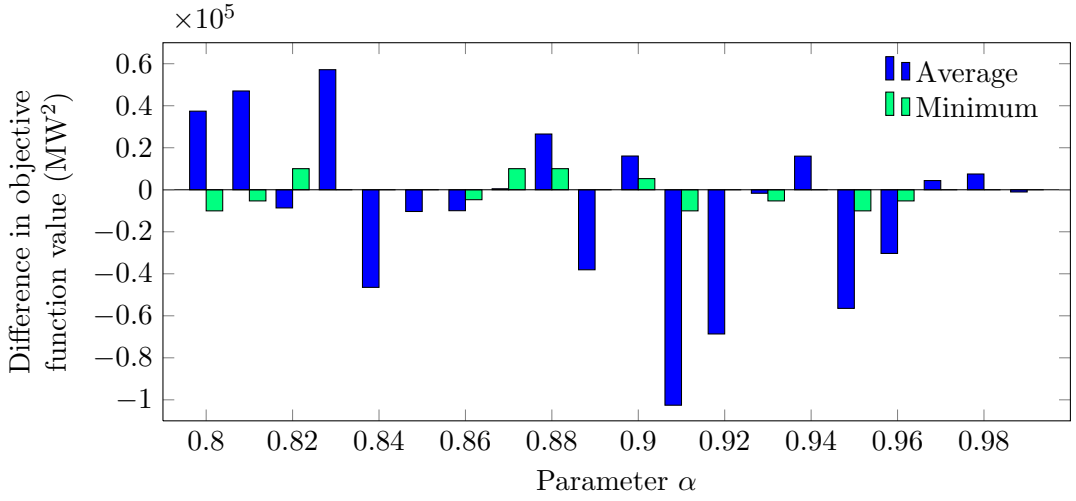


Figure 5.16: *Initial temperature analysis for the geometric cooling schedule in 21-SA-E.*

The graph in Figure 5.17 illustrates how the incumbent solution quality and average solution time varies over the different parameter values of α. The minimum incumbent objective function value remains very consistent over the whole range and even unchanged at a minimum level from an α-value of 0.91 onwards. The average incumbent objective function value improves significantly as α increases and the average feasibility of the incumbent solutions also improve. This is indicated by the average penalty values which correspond to the average incumbent solutions. In the graph, the penalty values are added to the average incumbent objective function values for scaling purposes in order to render it visible on the same set of axes. This feasibility improvement culminates in all 50 problem instances obtaining a feasible incumbent solution at $\alpha = 0.98$ and $\alpha = 0.99$. However, the average solution time increases exponentially as α increases, making 0.98 or 0.99 undesirable choices as values of α due to their long solution times. Any value in the range between [0.9, 0.95] would be a good choice for α, considering the trade-off between solution quality and execution time. The value of α was chosen as 0.92.

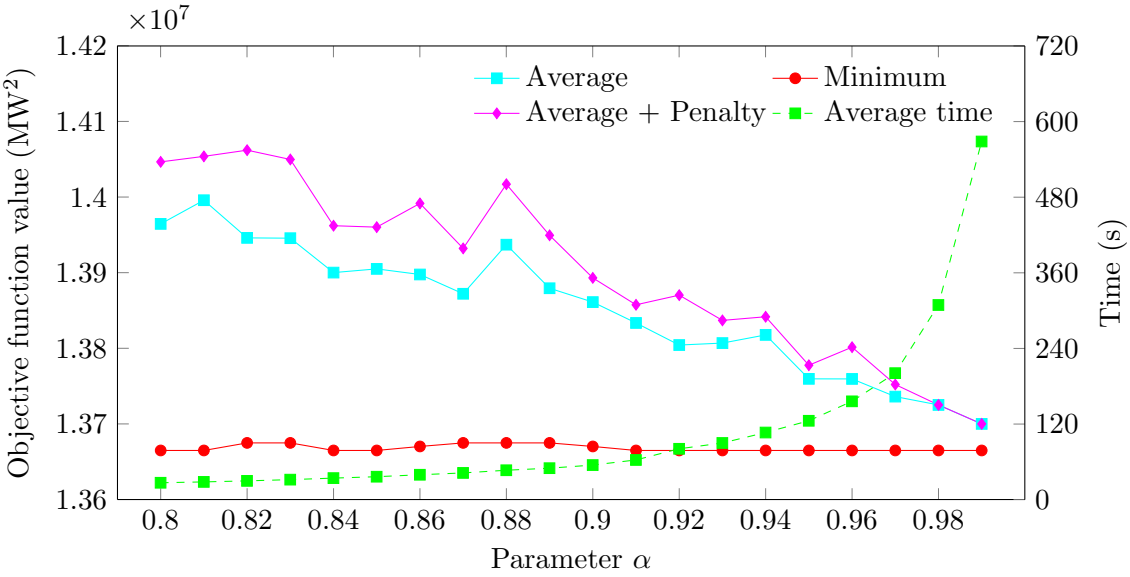
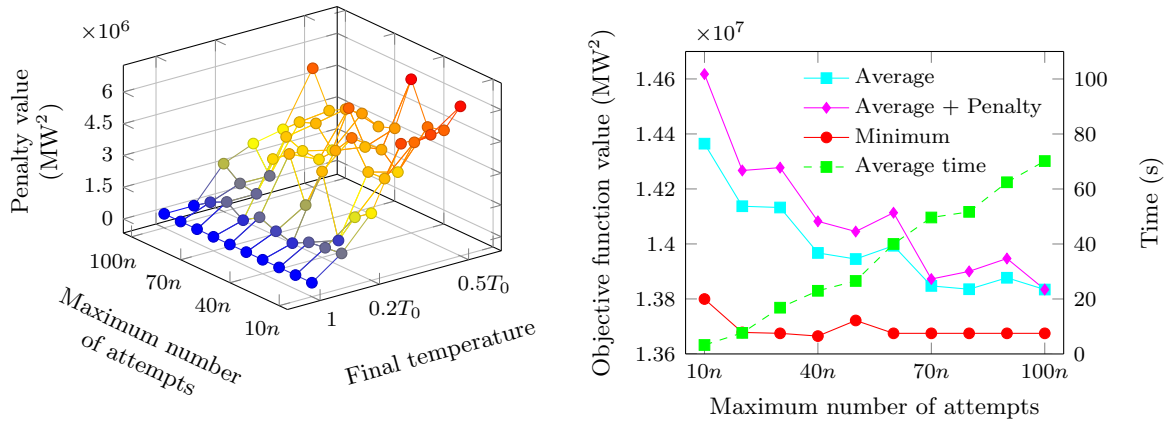


Figure 5.17: *Parameter optimisation for the geometric cooling schedule in 21-SA-E.*

The final step in the parameter optimisation process for the geometric cooling schedule is to choose the termination criteria parameter values. The penalty values corresponding to the minimum incumbent solutions are illustrated in Figure 5.18(a). One may observe that instances with a final temperature of 1 represent the only parameter combination for which the algorithm consistently finds feasible incumbent solutions. The final temperature was therefore chosen as $T_{min} = 1$. In order to choose *max_attempt*, one may analyse the graph in Figure 5.18(b) which illustrates the incumbent solution quality and average solution time over the different *max_attempt*-values, fixed at $T_{min} = 1$. The average solution time increases linearly as *max_attempt* increases, which is to be expected as each attempt typically requires the same amount of execution time. The minimum incumbent objective function value remains level from a parameter value of $60n$ and more, while the average incumbent objective function value approximately levels out only from $70n$ onwards. Taking these observations into account, the maximum number of attempted solutions during each temperature stage was chosen as $max\_attempt = 80n$.

The next cooling schedule to consider is the one proposed by Huang *et al.* [37]. The difference in incumbent objective function values (average and minimum) between those obtained using



(a) Minimum incumbent solution's penalty values.

(b) Parameter analysis at $T_{min} = 1$.

Figure 5.18: *Termination criteria parameter analysis for the geometric cooling schedule in 21-SA-E.*

the AIM and those using the SDM is shown in Figure 5.19. In this case, there is not much to choose between the two methods, as neither the positive nor negative bars seem to dominate. Therefore, the method to determine the initial temperature was arbitrarily chosen as the AIM.

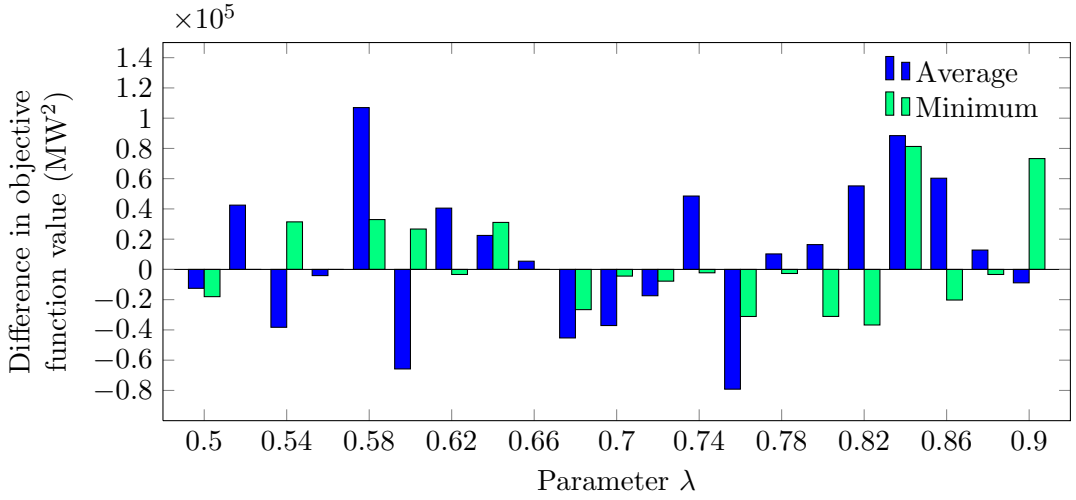


Figure 5.19: *Initial temperature analysis for the cooling schedule of Huang et al. [37] in 21-SA-E.*

The incumbent solution quality and average solution time over the different parameter values of λ are illustrated in Figure 5.20. The average solution time varies so little in maximum and minimum values (only about 3 seconds) that it may be regarded as not having an influence on the choice of parameter value. The minimum incumbent objective function value remains consistently low and level up to approximately $\lambda = 0.83$. However, the average incumbent objective function value and penalty value already start deteriorating from a λ-value of 0.7 and above. Any λ-value between 0.5 and 0.7 would be a good choice. Therefore, the value of λ was chosen as 0.6.

Figure 5.21 contains graphs for the termination criteria parameter analysis. As before, instances with a final temperature of $T_{min} = 1$ in the parameter combination are the only ones which give consistent feasible minimum incumbent solutions. This is seen in Figure 5.21(a) which contains the graph of penalty values corresponding to the minimum incumbent solutions. Therefore,

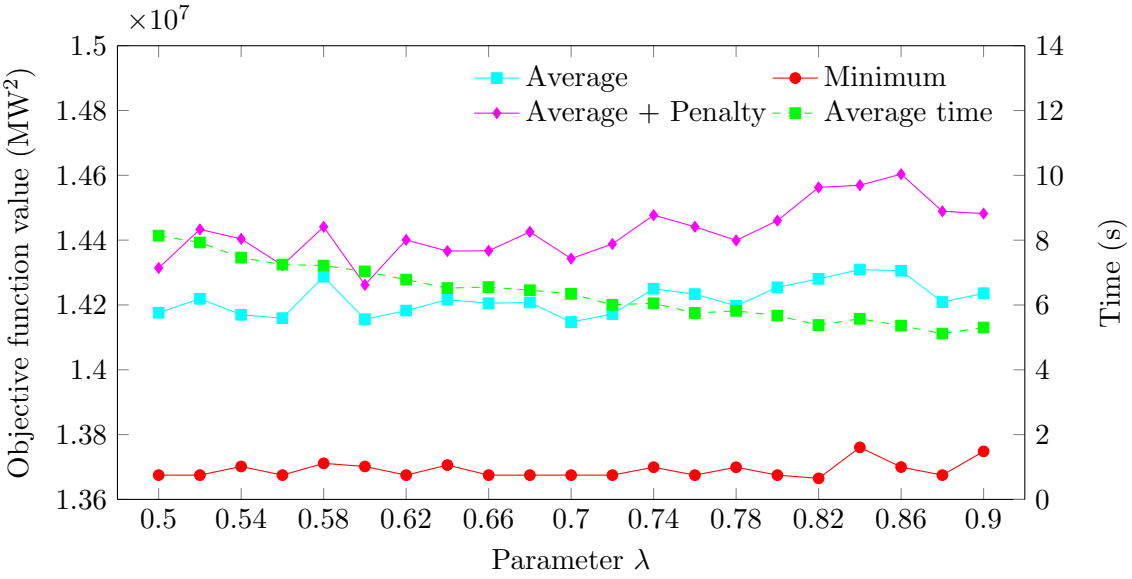
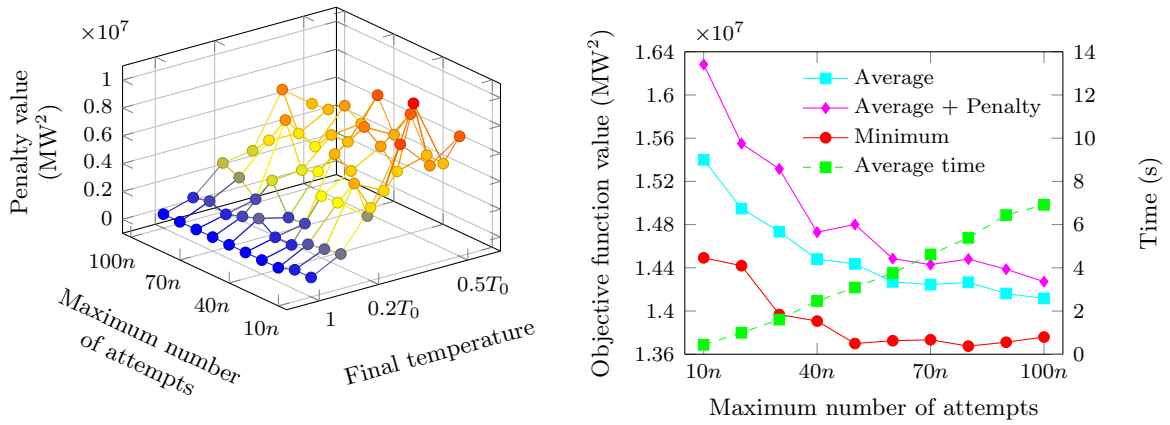


Figure 5.20: *Parameter optimisation for the cooling schedule of Huang et al. [37] in 21-SA-E.*

the final temperature was chosen to be $T_{min} = 1$. In Figure 5.21(b), the incumbent solution quality and average solution time over the different parameter values of *max_attempt*, but only at $T_{min} = 1$, are illustrated. Again, the average solution time varies insignificantly from its minimum to maximum value over the parameter range. The minimum incumbent objective function value levels off at $50n$ and above, but the average incumbent objective function value keeps on slowly declining up to the parameter value of $100n$. Since the average solution time is insignificant at its maximum (only 7 seconds), the maximum number of attempted solutions was chosen as $100n$ in order to obtain the best incumbent solution quality.



(a) Minimum incumbent solution's penalty values.

(b) Parameter analysis at $T_{min} = 1$.

Figure 5.21: *Termination criteria parameter analysis for the cooling schedule of Huang et al. [37] in 21-SA-E.*

For the cooling schedule of Van Laarhoven *et al.* [81], consider Figure 5.22 which contains a graph of the difference in incumbent objective function values (average and minimum) between those obtained using the AIM and those using the SDM. The negative bars are more prevalent in the figure and therefore, the method of determining the initial temperature was chosen as the AIM.

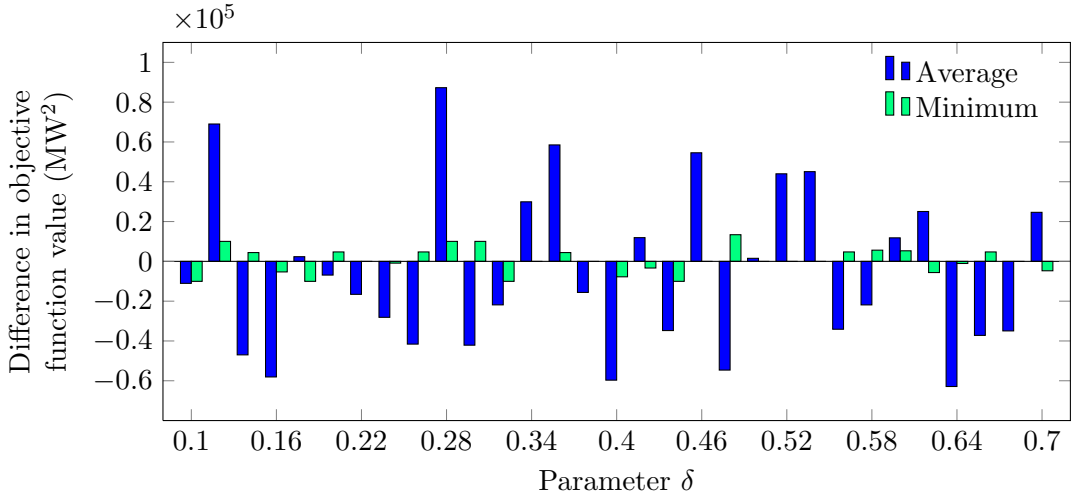


Figure 5.22: *Initial temperature analysis for the cooling schedule of Van Laarhoven et al. [81] in 21-SA-E.*

Consider next the incumbent solution quality and average solution time over the range of δ-parameter values in Figure 5.23. The average solution time decreases exponentially over the range of δ-values. As the average solution time decreases, the average incumbent objective function value climbs steadily with a deterioration in feasibility as well, indicated by the increase of the average penalty value added to the average incumbent objective function value. However, the minimum incumbent objective function value remains very consistent throughout the δ-value range. Therefore, an acceptable trade-off between average incumbent solution quality and average solution time must be determined. A value of $\lambda = 0.16$ was chosen, obtaining a very good solution quality within approximately 60 seconds of solution time.

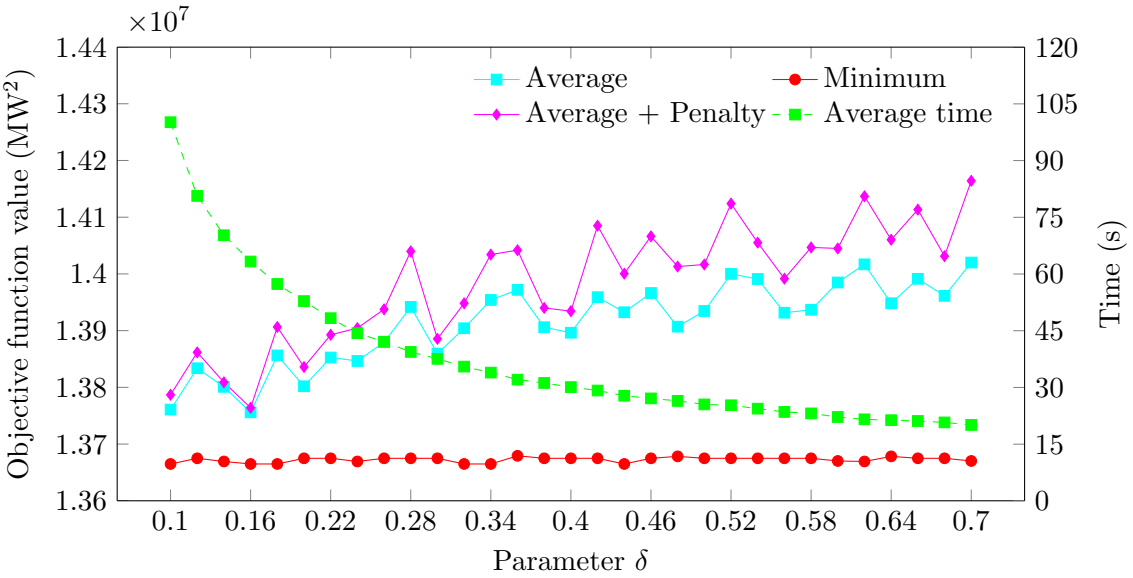
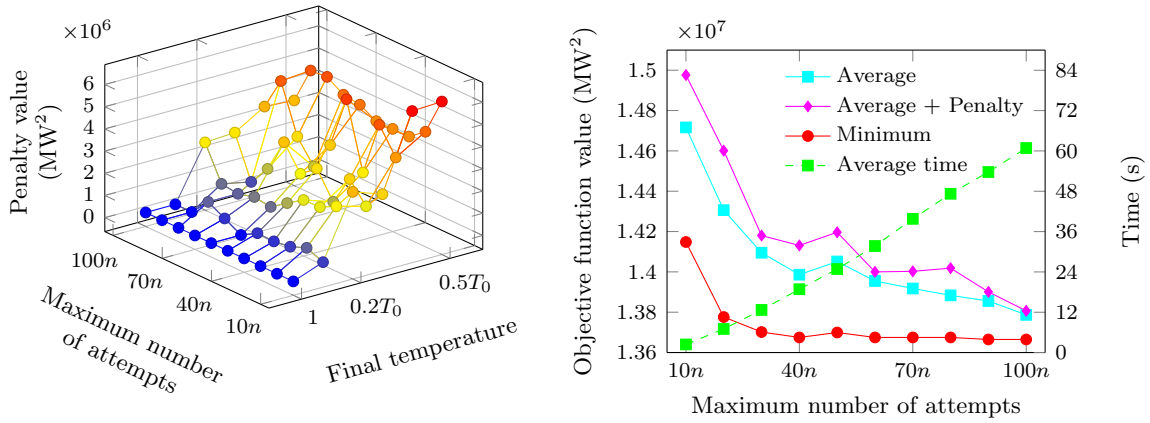


Figure 5.23: *Parameter optimisation for the cooling schedule of Van Laarhoven et al. [81] in 21-SA-E.*

The termination criteria parameters are considered for analysis in Figure 5.24. A final temperature of $T_{min} = 1$ is the only parameter value resulting in consistent feasible minimum incumbent solutions, as seen in Figure 5.24(a) which contains the penalty values corresponding to the min-

imum incumbent solutions. Therefore, the final temperature was chosen as $T_{min} = 1$. Consider now the graph in Figure 5.24(b) of the incumbent solution quality and average solution time over the parameter range of $max\_attempt$, at $T_{min} = 1$. The average solution time increases linearly over the parameter range. However, there is a sharp improvement in minimum and average incumbent solution objective function values from $10n$ to $40n$. From there upwards, the minimum incumbent objective function value remains relatively level, close to the minimum objective function value found, while the average incumbent objective function value continues to decrease (improve) but at a slower rate than before. Since the solution quality improves up to the end of the parameter range, and an average solution time of approximately 60 seconds is acceptable, the maximum number of attempted solutions was chosen to be $100n$.

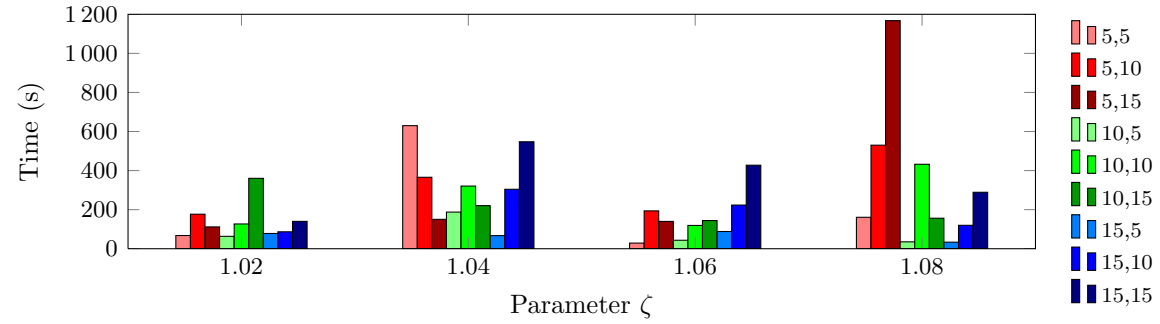


(a) Minimum incumbent solution's penalty values.

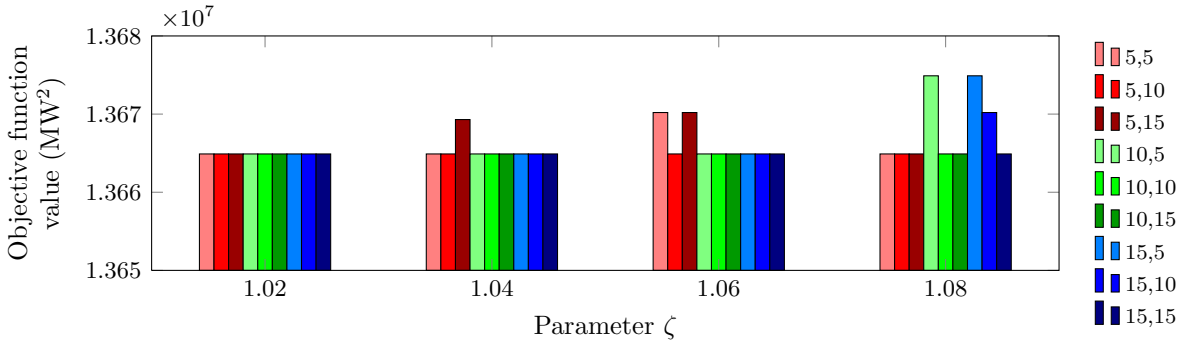
(b) Parameter analysis at $T_{min} = 1$.

Figure 5.24: *Termination criteria parameter analysis for the cooling schedule of Van Laarhoven et al. [81] in 21-SA-E.*

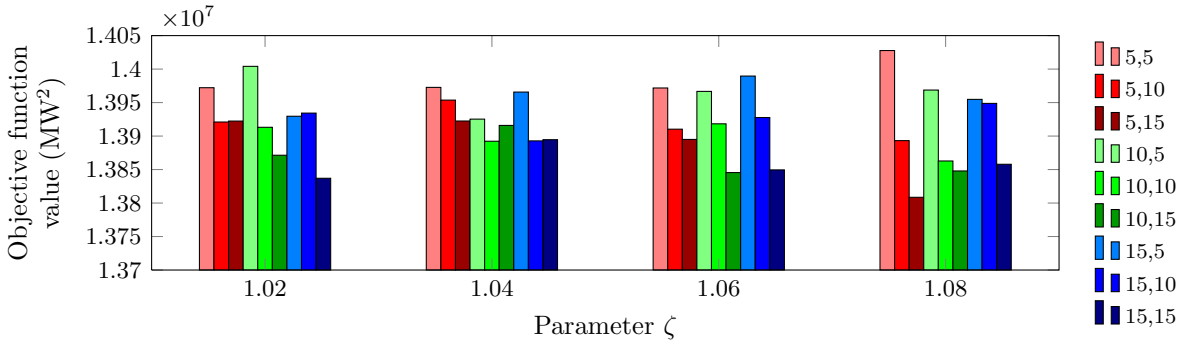
Finally, the cooling schedule proposed by Triki *et al.* [79] is considered. In contrast with the other cooling schedules which each employ only one parameter, this schedule has three perameters to be optimised concurrently. This requires a somewhat different visual representation of the analysis results than used above. Consider the parameter analysis results in Figures 5.25 and 5.26. In both figures, the parameters μ_2 and μ_1 are grouped together as a single "unit" (μ_2, μ_1) and alongside the remaining parameter ζ for analysis. In Figure 5.25, the parameters μ_2 and μ_1 are grouped together in increasing order of μ_2, while in Figure 5.26 they are grouped together in increasing order of μ_1. When considering the two average solution time graphs in Figures 5.25(a) and 5.26(a), it may be concluded that an increase in μ_1 causes the algorithm to increase significantly in solution time. This is to be expected, since μ_1 is the factor by which the expected decrease in temperature is decreased (see §4.2.5) and smaller decrements ($\Delta = \sigma/\mu_1$) mean more temperature stages. This also explains why the solution quality improves with larger values of μ_1, as observed in Figures 5.25(c) and 5.25(d), since the solution space is explored in more detail due to the increased number of temperature stages. Furthermore, an increase in ζ-value seems to increase the solution time. This may be because equilibrium is found more often between temperature stages, causing the algorithm to continue normally instead of defaulting to a (faster) greedy algorithm (see §4.2.5). A value of $\mu_2 = 10$ seems to deliver the most consistent results over all parameter combinations. Based on these observations, the parameter selection which obtains an acceptable trade-off in solution quality and time, is $\zeta = 1.02$, $\mu_2 = 10$ and $\mu_1 = 10$.



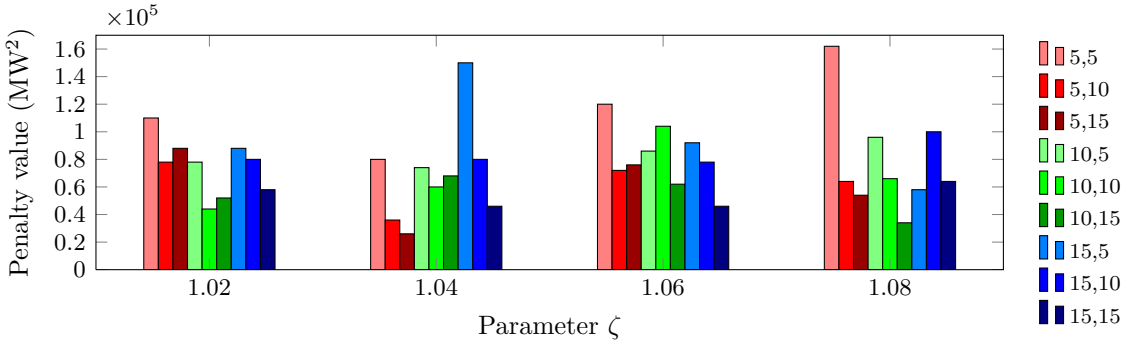
(a) Average solution time



(b) Minimum incumbent objective function value

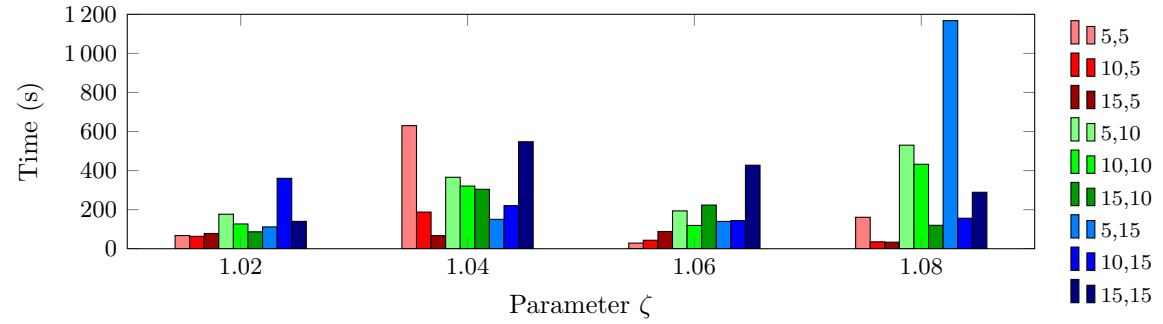


(c) Average incumbent objective function value

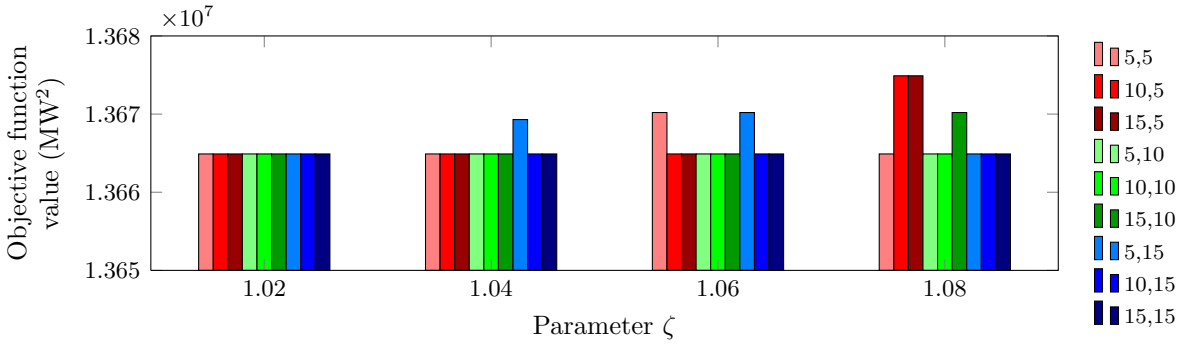


(d) Average incumbent penalty

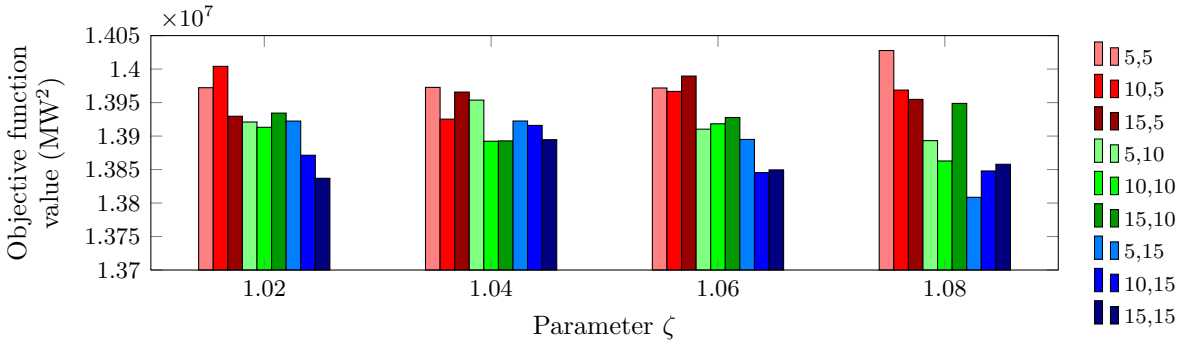
Figure 5.25: *Parameter optimisation for the cooling schedule of Triki et al. [79] in 21-SA-E. Legend entries in increasing order of the parameter μ_2.*



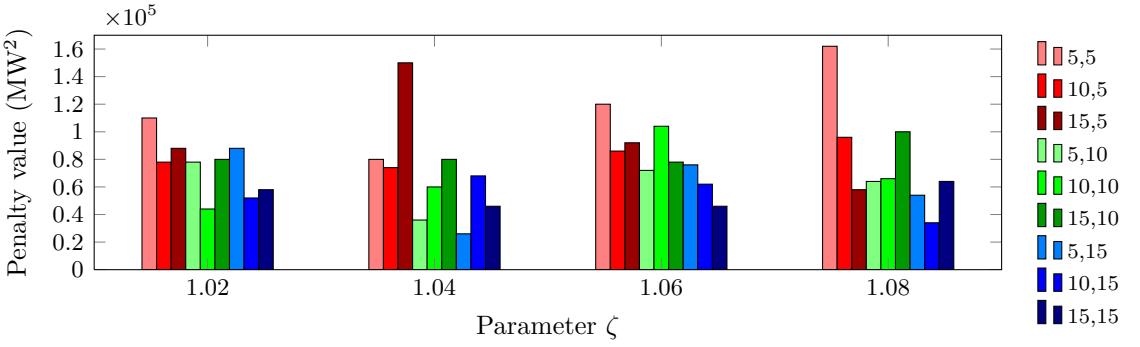
(a) Average solution time



(b) Minimum incumbent objective function value



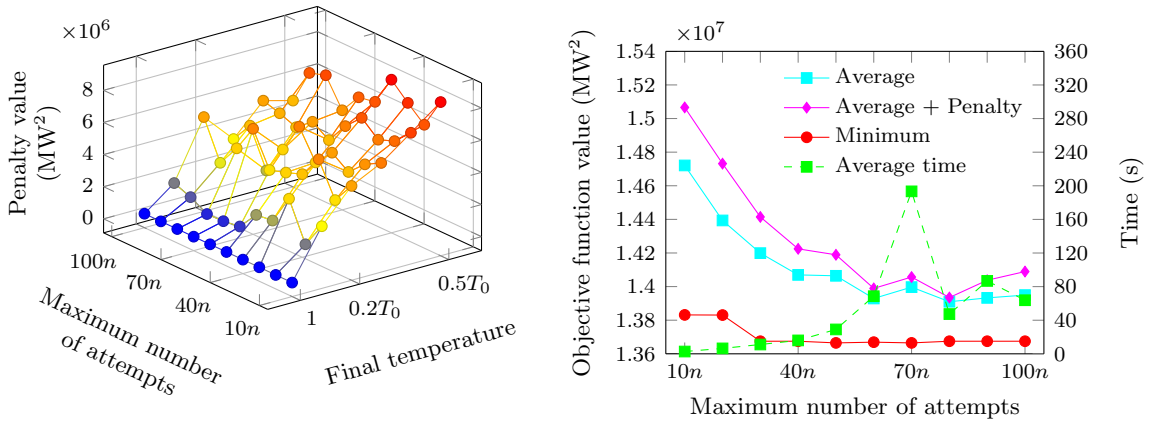
(c) Average incumbent objective function value



(d) Average incumbent penalty

Figure 5.26: *Parameter optimisation for the cooling schedule of Triki et al. [79] in 21-SA-E. Legend entries in increasing order of the parameter μ_1.*

The termination criteria parameters are considered for analysis in Figure 5.27. Again, the parameter value of $T_{min} = 1$ is the only value resulting in consistent feasible minimum incumbent solutions, as illustrated in Figure 5.27(a) which contains penalty values corresponding to the the minimum incumbent solutions. The final temperature was therefore chosen as 1. The



(a) Minimum incumbent solution's penalty values.

(b) Parameter analysis at $T_{min} = 1$.

Figure 5.27: *Termination criteria parameter analysis for the cooling schedule of Triki et al. [79] in 21-SA-E.*

graph in Figure 5.27(b) illustrates the incumbent solution quality and average solution time over the parameter values of *max_attempt*, specifically at $T_{min} = 1$. The average solution time is somewhat erratic in behaviour; however, an increase in time is nevertheless observed as the parameter values increase. The minimum incumbent objective function value remains approximately level at $30n$ and above, but the average incumbent objective function value improves rapidly up to $60n$, after which it levels off. Due to the erratic behaviour of the average solution time, the choice of *max_attempt* was based more on the average incumbent solution quality, and was subsequently therefore chosen to be $100n$ in order to obtain the best results.

The 22-unit system

The same analysis performed for parameter optimisation in the 21-unit system above, is now repeated for the 22-unit system. Before the individual cooling schedules are analysed, it is worth mentioning that the behaviour of the final temperature T_{min} in the 22-unit system is exactly the same as for the 21-unit system, *i.e.* $T_{min} = 1$ is the only value for which the minimum incumbent solutions are consistently feasible. Therefore, in all four cooling schedules, the final temperature was chosen as $T_{min} = 1$.

Starting with the geometric cooling schedule, Figure 5.28 contains the graph illustrating the difference in incumbent objective function value (average and minimum) obtained by using the AIM and the SDM to determine the initial temperature. Clearly, it may be observed that the negative bars are more prevalent. Therefore the AIM was chosen as the method for determining the initial temperature.

In Figure 5.29, the incumbent solution quality and average solution time over the range of α-values are illustrated. Unfortunately, the scaling of the penalty values are not as convenient as in the 21-unit system. However, the average incumbent objective function values still provide one with a clear picture of the solution quality — a steady improvement in quality as α increases

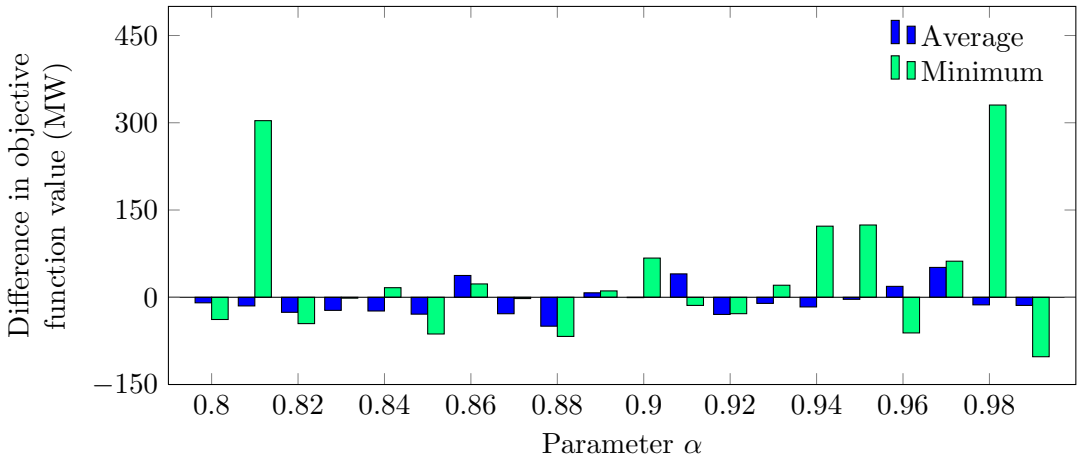


Figure 5.28: *Initial temperature analysis for the geometric cooling schedule in 22-SA-E.*

in value. The minimum incumbent objective function values follow the same downward trend, but the values are a bit erratic. Furthermore, the average solution time increases exponentially over the parameter values. In order to achieve an acceptable trade-off between solution quality and the execution time, the value of α was chosen as 0.96 in order to limit the solution time to more or less 60 seconds.

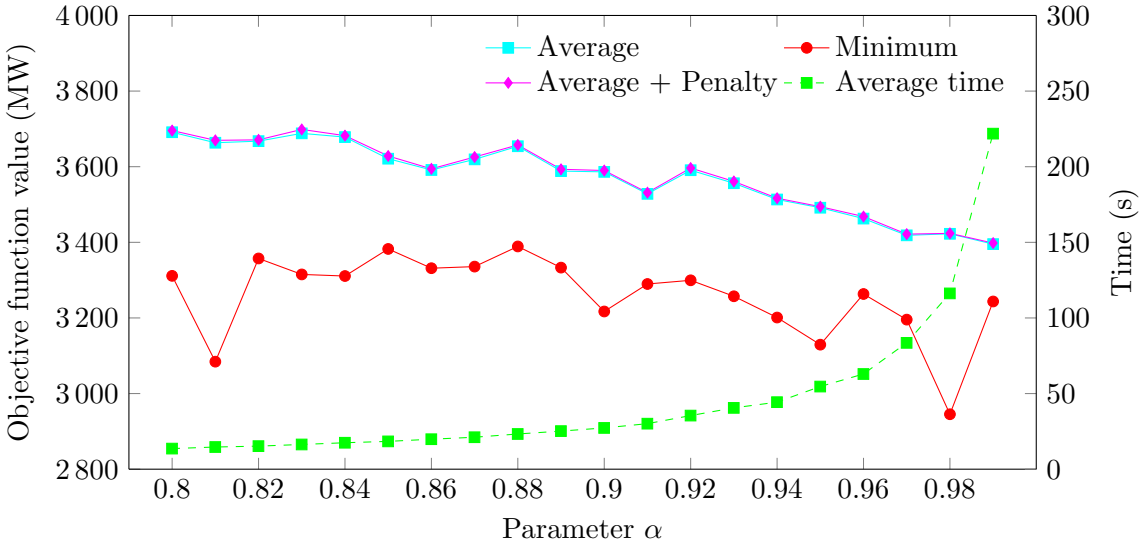


Figure 5.29: *Parameter optimisation for the geometric cooling schedule in 22-SA-E.*

For the geometric cooling schedule, the value of *max_attempt* is determined from Figure 5.30. In the graph, the incumbent solution quality and average solution time over the parameter range of *max_attempt* at $T_{min} = 1$ are considered. The maximum average solution time over the range is about 80 seconds and therefore still acceptable. The average incumbent objective function value seems to level at $70n$ and above, while the minimum incumbent objective function value is somewhat erratic. Therefore, the maximum number of attempted solutions during each temperature stage was chosen as $80n$.

The cooling schedule proposed by Huang *et al.* [37] is considered next, with the difference between objective function value (average and minimum) obtained by using the AIM and the

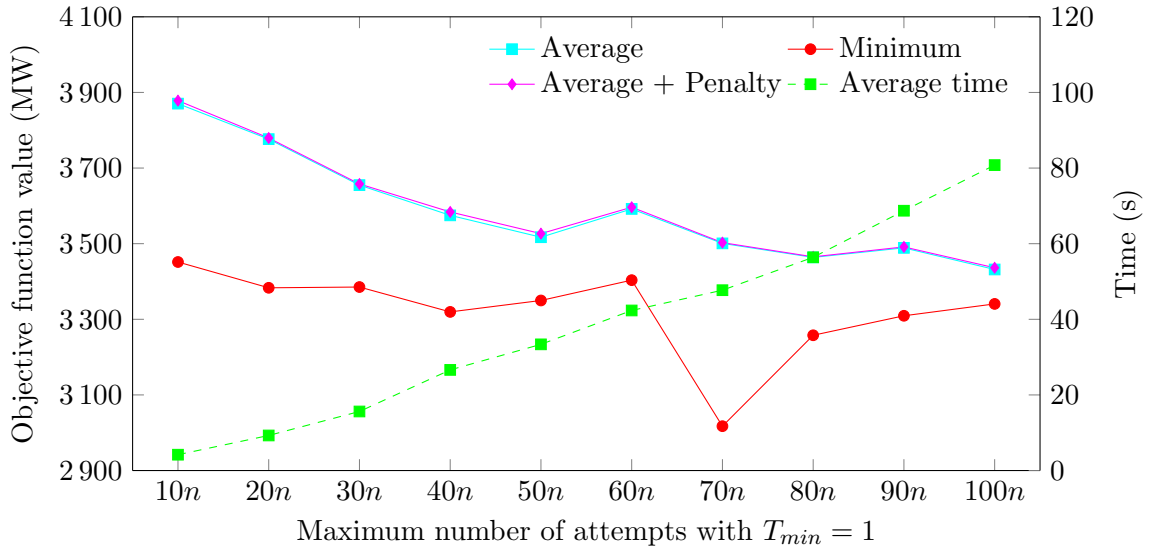


Figure 5.30: *Termination criteria parameter analysis at $T_{min} = 1$ for the geometric cooling schedule in 22-SA-E.*

SDM to determine the initial temperature as illustrated in Figure 5.31. The positive bars, indicating the superiority of the SDM, are slightly more prevalent in this case. Therefore, the method for determining the initial temperature was chosen as the SDM.

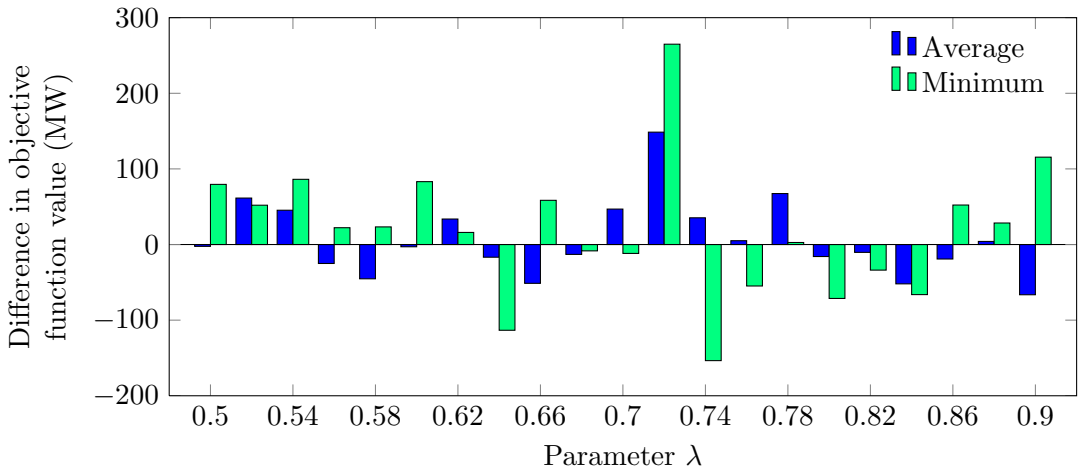


Figure 5.31: *Initial temperature analysis for the cooling schedule of Huang et al. [37] in 22-SA-E.*

The incumbent solution quality and average solution time over the parameter values of λ are shown in Figure 5.32. Since the average solution time differs by only 4 seconds between its maximum and minimum values, the solution quality does not have to be traded off against the solution time. The minimum incumbent objective function value is obtained at 0.72 but overall it had erratic behaviour. The average incumbent objective function value remains approximately level over the values of λ, obtains minimum levels at 0.7 and 0.72, and slowly increases from 0.74 onwards. Considering the above, the parameter value of λ was chosen as 0.72.

In Figure 5.33, the incumbent solution quality and average solution time over the paremeter values of *max_attempt* are presented, at a final temperature of $T_{min} = 1$. Once again, the

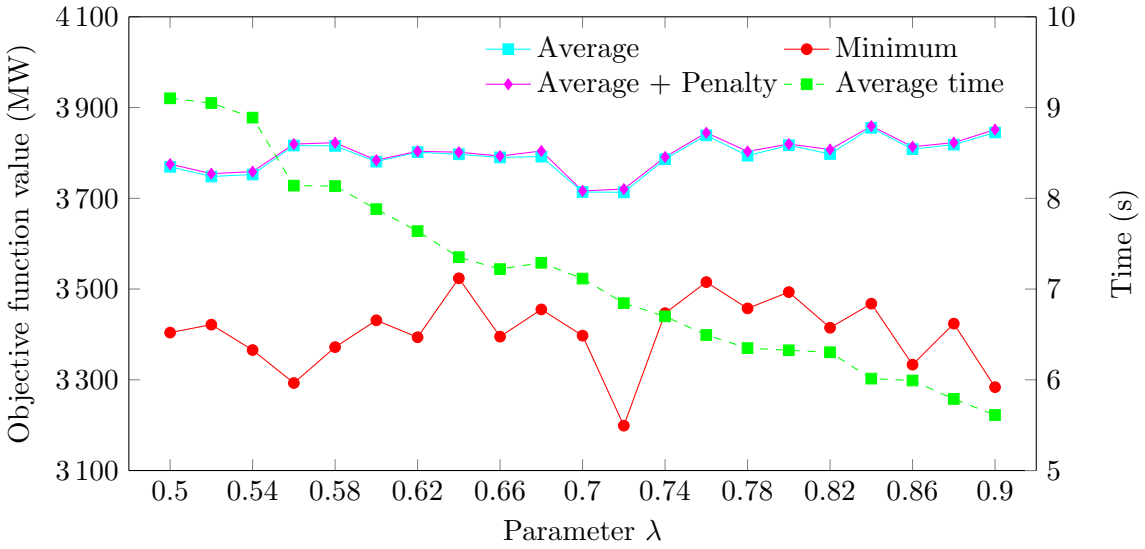


Figure 5.32: *Parameter optimisation for the cooling schedule of Huang et al. [37] in 22-SA-E.*

average solution time does not influence any trade-off arguments due to its small scale, and is accepted as a fast solution time. Both the minimum and average incumbent objective function values steadily decrease as the number of attempts increase. Thus, the maximum number of attempted solutions was chosen as $100n$ to ensure the best solution quality.

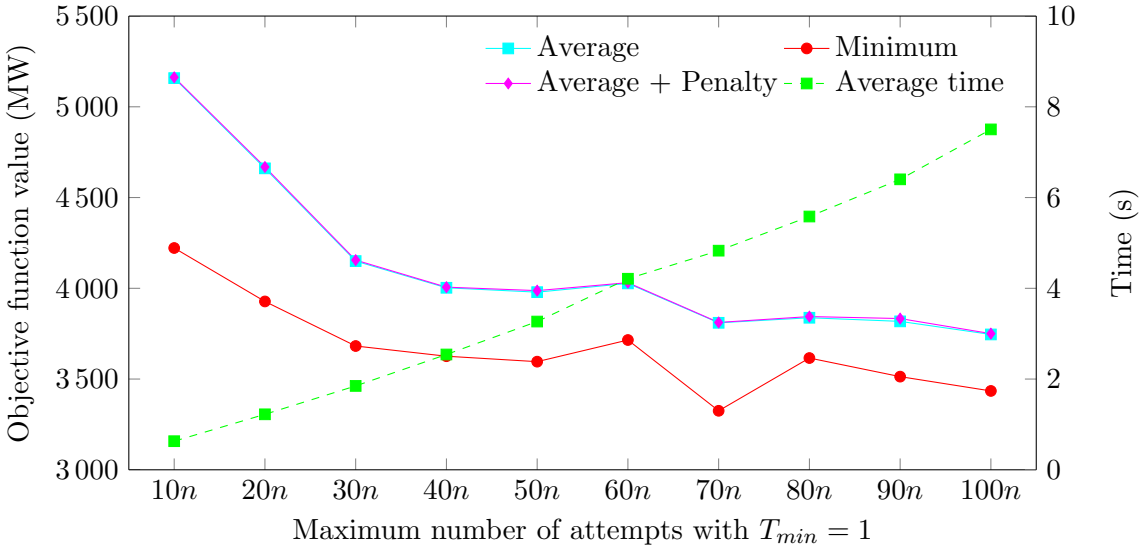


Figure 5.33: *Termination criteria parameter analysis at $T_{min} = 1$ for the cooling schedule of Huang et al. [37] in 22-SA-E.*

Within the cooling schedule proposed by Van Laarhoven *et al.* [81] the difference in objective function value (average and minimum) between using the AIM and the SDM for determining the initial temperature is presented in Figure 5.34. It may be observed that the negative bars (indicating the superiority of the AIM) are more prevalent than the positive bars. Therefore, the method for determining the initial temperature was chosen as the AIM.

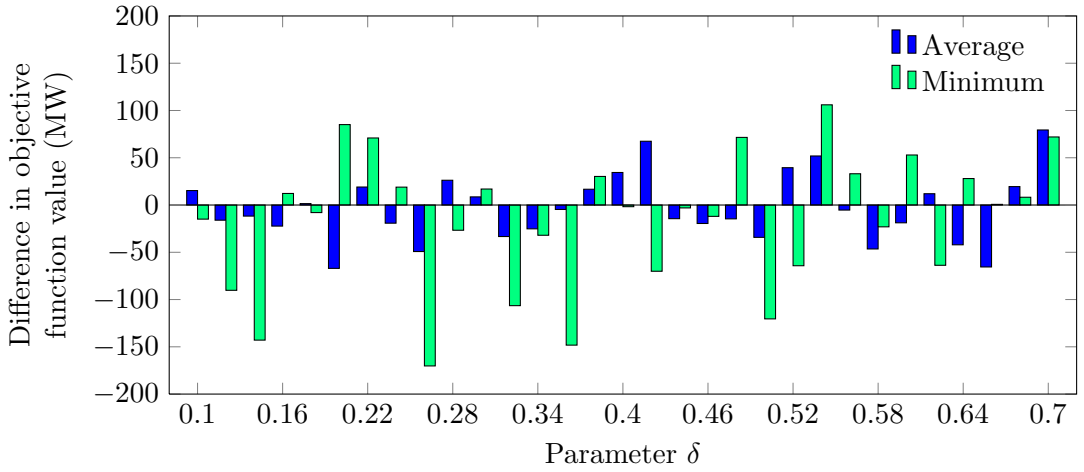


Figure 5.34: *Initial temperature analysis for the cooling schedule of Van Laarhoven et al. [81] in 22-SA-E.*

Consider the incumbent solution quality and average solution time over the range of δ-parameter values in Figure 5.35. The average solution time decreases exponentially as the value of δ increases. There is very little variation in the average incumbent objective function values as they slowly worsen over the increasing parameter values with the minimum incumbent objective function values generally following the same trend (and having slightly erratic behaviour). It seems that the highest solution quality is achieved within the range of $\delta \in [0.1,\ 0.22]$ and in order to keep the solution time acceptable, the value of δ was chosen as 0.2.

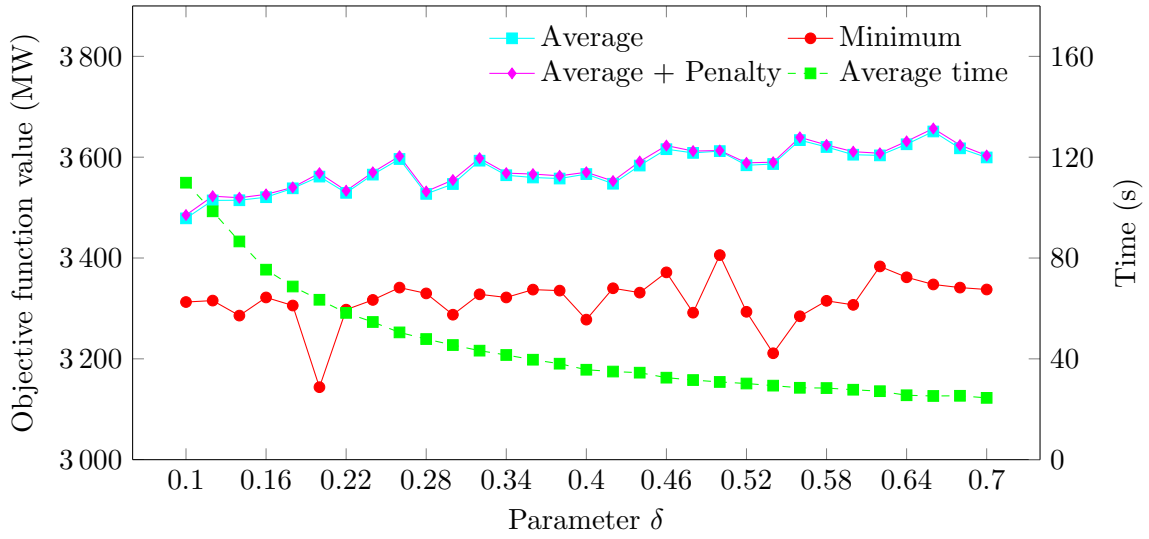


Figure 5.35: *Parameter optimisation for the cooling schedule of Van Laarhoven et al. [81] in 22-SA-E.*

The parameter analysis for the *max_attempt* parameter is illustrated in Figure 5.36. The graph contains the incumbent solution quality and average solution time over the range of parameter values fixed at $T_{min} = 1$. As the average solution time reaches an acceptable maximum of around 60 seconds, the steady decline in average and minimum incumbent objective function values are of more importance when considering a decision with respect to the parameter value. The objective function values level off more or less from a parameter value of $80n$. Therefore, the maximum number of attempted solutions was chosen as $90n$.

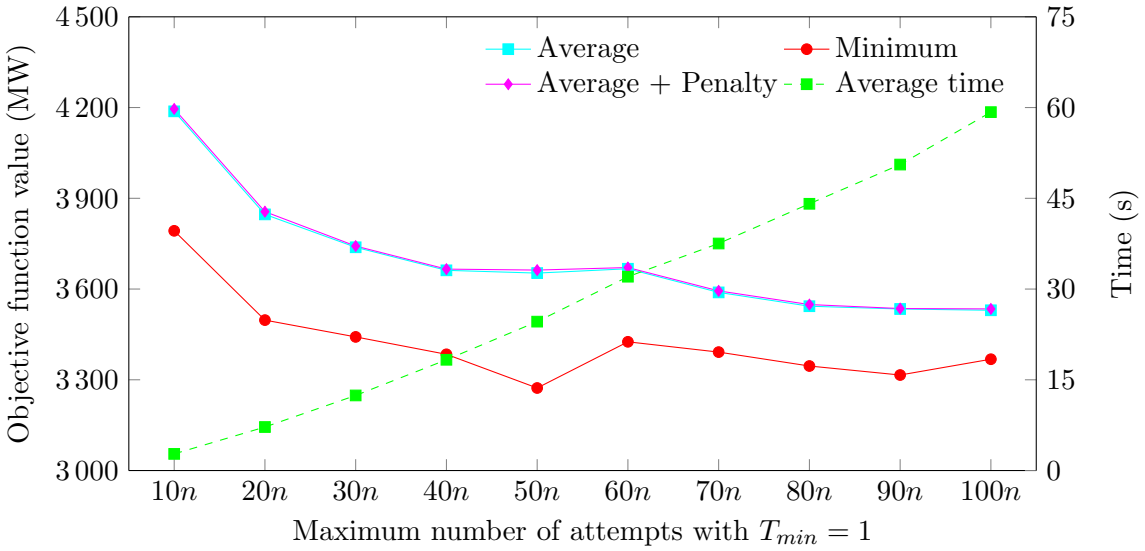


Figure 5.36: *Termination criteria parameter analysis at $T_{min} = 1$ for the cooling schedule of Van Laarhoven et al. [81] in 22-SA-E.*

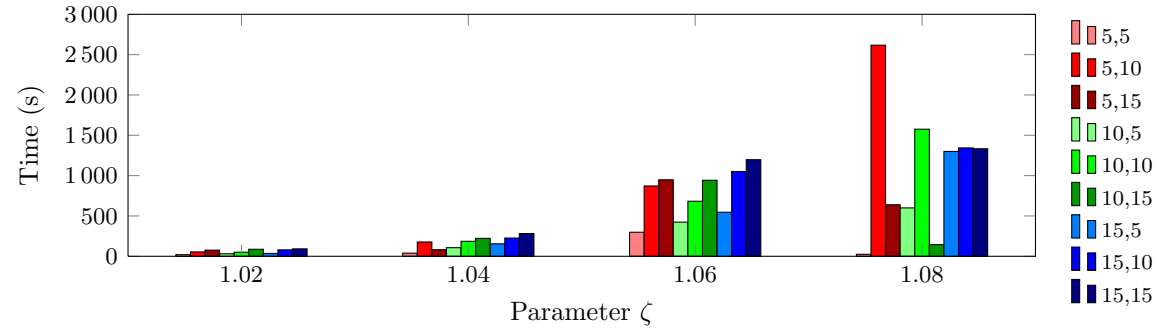
Consider Figures 5.37 and 5.38 for the parameter analysis of the schedule proposed by Triki *et al.* [79]. The graphs include results on the incumbent solution quality and average time over the different parameter combinations. As before, the parameters μ_2 and μ_1 are grouped together in single (μ_2, μ_1) "units" with those in Figure 5.37 grouped together in increasing order of μ_2 and those in Figure 5.38 in increasing order of μ_1. Consider first the average solution times in Figures 5.37(a) and 5.38(a). Irrespective of the values of μ_2 and μ_1, the choice of a ζ-value may already be restricted to 1.02 or 1.04 based on these solution times. As before, the solution quality seems generally to improve as the value of μ_1 increases, as seen in Figures 5.37(c) and 5.37(d). An increase in the value of μ_2 also seems to improve the solution quality, as seen in Figures 5.38(c) and 5.38(d). In order to obtain a good solution quality within an acceptable solution time, the parameter values for the cooling schedule proposed by Triki *et al.* [79] were chosen as $\zeta = 1.02$, $\mu_2 = 10$ and $\mu_1 = 10$.

Finally, the analysis of *max_attempt* is performed from the graph in Figure 5.39. In the figure, an illustration is given of the incumbent solution quality and average solution time over the parameter range of *max_attempt* at a final temperature of 1. The average and minimum incumbent objective function values improve steadily up to the limit of the parameter range. Since the maximum average solution time is of an acceptable magnitude of approximately 60 seconds, the maximum number of attempted solutions during each temperature stage was chosen as $100n$, in order to obtain the best solution quality.

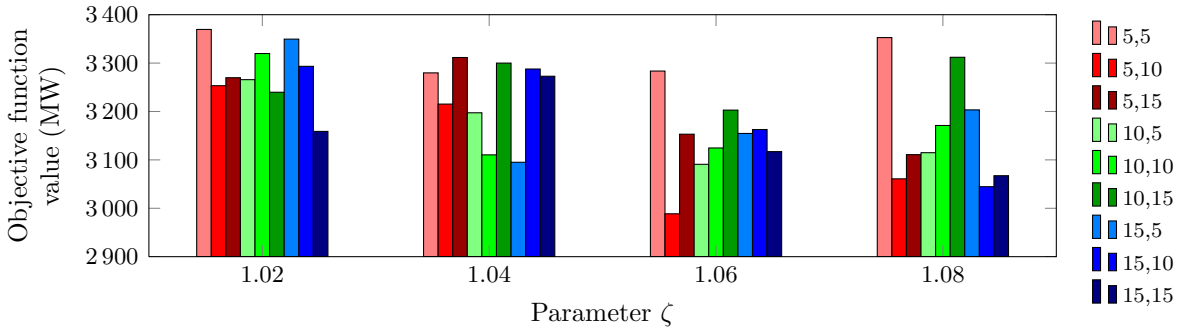
The IEEE-RTS inspired system

The analysis for the parameter optimisation in the IEEE-RTS inspired system is performed here in the same fashion as for the 21- and 22-unit systems above. Furthermore, the behaviour of the final temperature T_{min} in the IEEE-RTS inspired system is exactly the same as for the 21- and 22-unit systems, thus $T_{min} = 1$ is assumed in all four cooling schedules below.

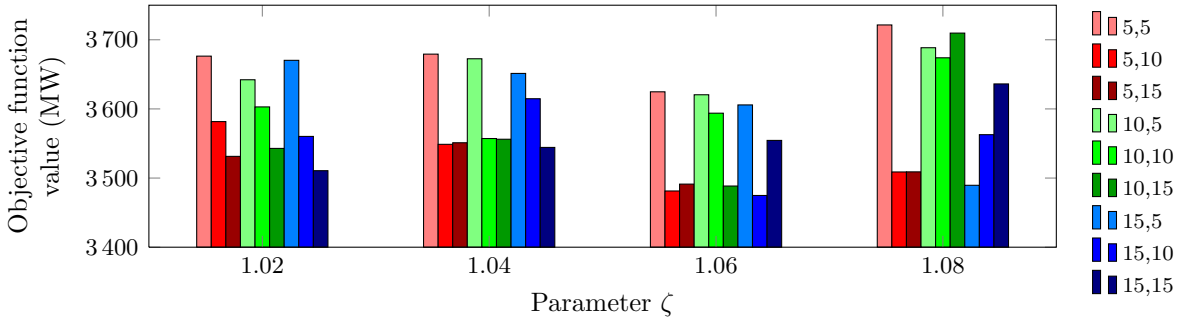
Starting with the geometric cooling schedule, an illustration of the difference in incumbent objective function values (average and minimum) between using the AIM and the SDM for



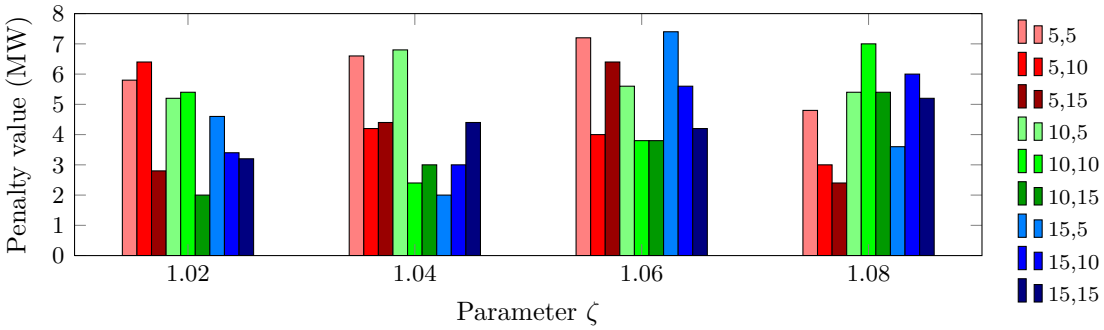
(a) Average solution time



(b) Minimum incumbent objective function value

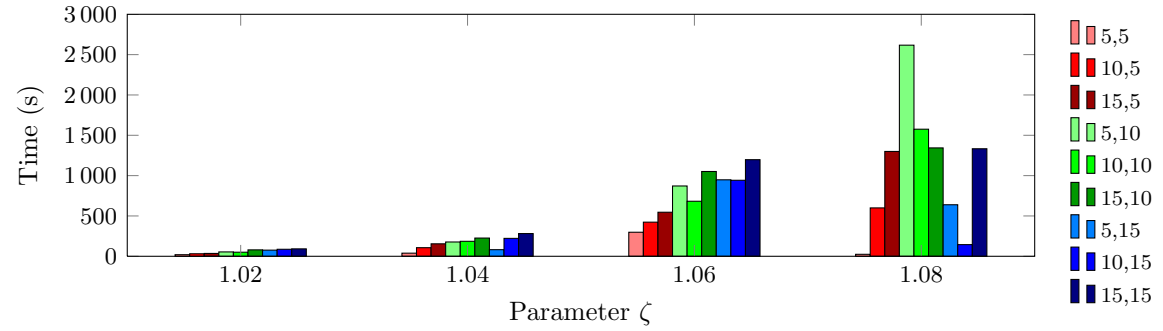


(c) Average incumbent objective function value

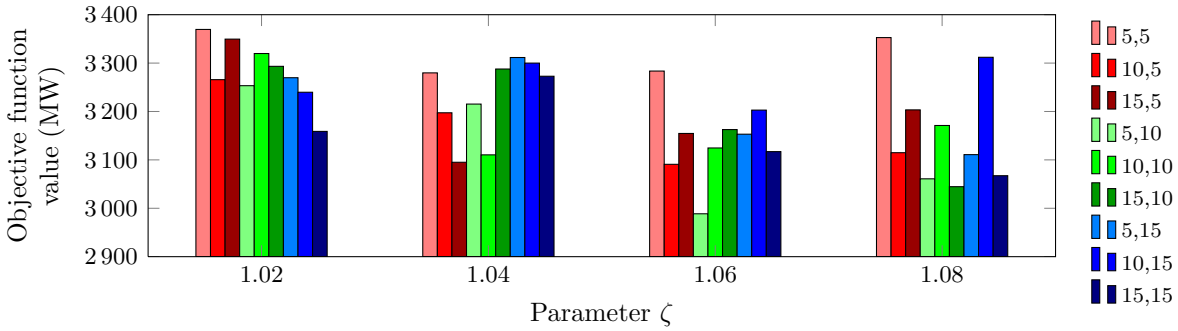


(d) Average incumbent penalty

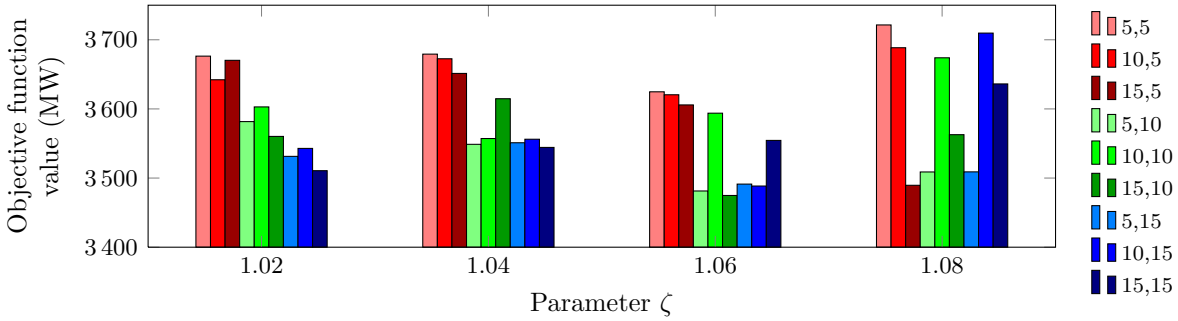
Figure 5.37: *Parameter optimisation for the cooling schedule of Triki et al. [79] in 22-SA-E. Legend entries in increasing order of the parameter μ_2.*



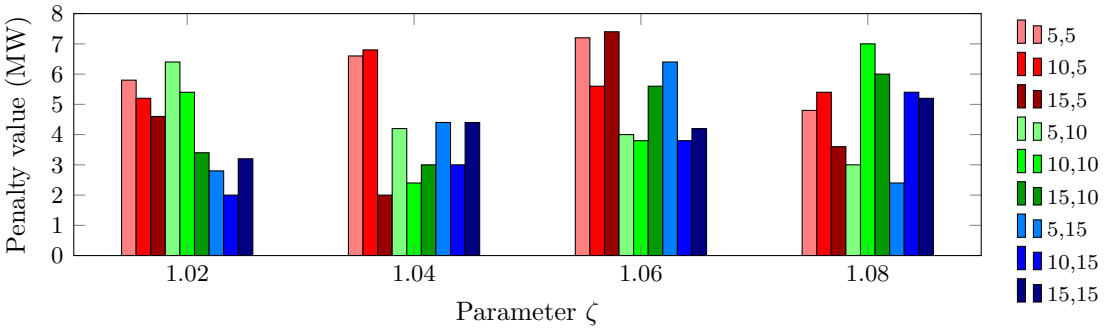
(a) Average solution time



(b) Minimum incumbent objective function value



(c) Average incumbent objective function value



(d) Average incumbent penalty

Figure 5.38: *Parameter optimisation for the cooling schedule of Triki et al. [79] in 22-SA-E. Legend entries in increasing order of the parameter μ_1.*

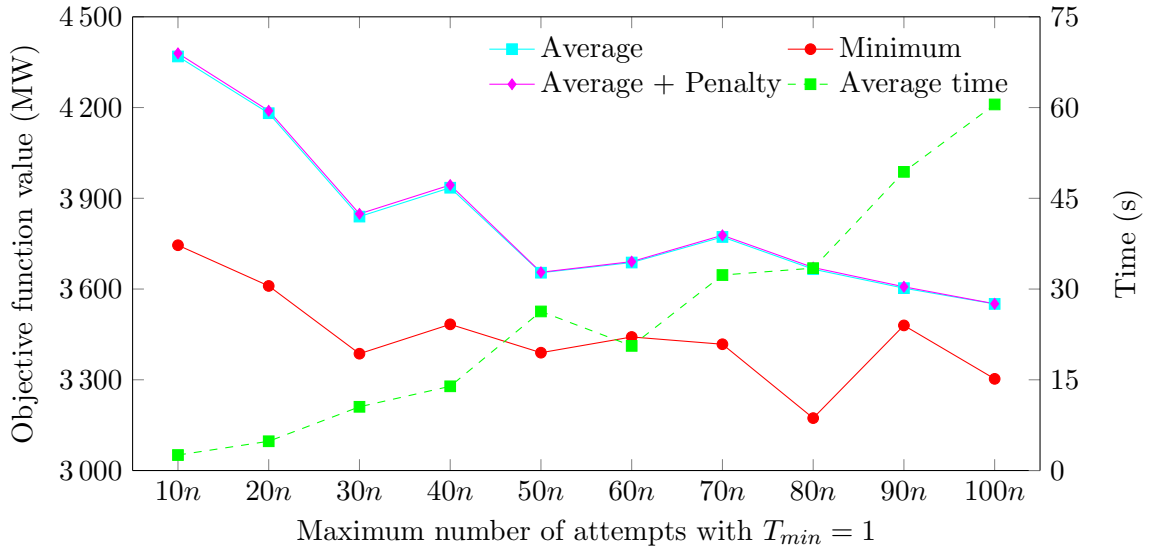


Figure 5.39: *Termination criteria parameter analysis at $T_{min} = 1$ for the cooling schedule of Triki et al. [79] in 22-SA-E.*

determining the initial temperature may be found in Figure 5.40. In the graph, the negative bars are slightly more prevalent than the positive bars, therefore the method for determining the initial temperature was chosen as the AIM.

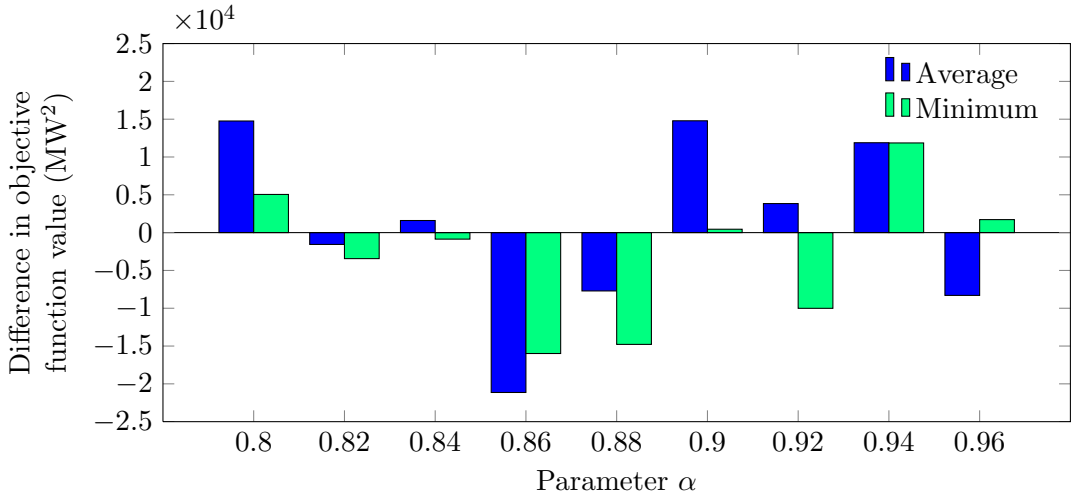


Figure 5.40: *Initial temperature analysis for the geometric cooling schedule in IEEE-SA-E.*

The variations in incumbent solution quality and average solution time over the range of α-values are shown in Figure 5.41. The average solution time increases exponentially as the parameter values increase, already reaching approximately three minutes at 0.94. With the steady decline in average incumbent objective function value over the parameter value range and the minimum incumbent objective function value remaining relatively level over the entire range, a trade-off between solution quality and execution time may be achieved by selecting the value $\alpha = 0.92$, since the solution time simply becomes too large at larger values of α.

In Figure 5.42, the incumbent solution quality and average solution time over the range of *max_attempt*, are presented. These values were obtained using a final temperature of $T_{min} = 1$.

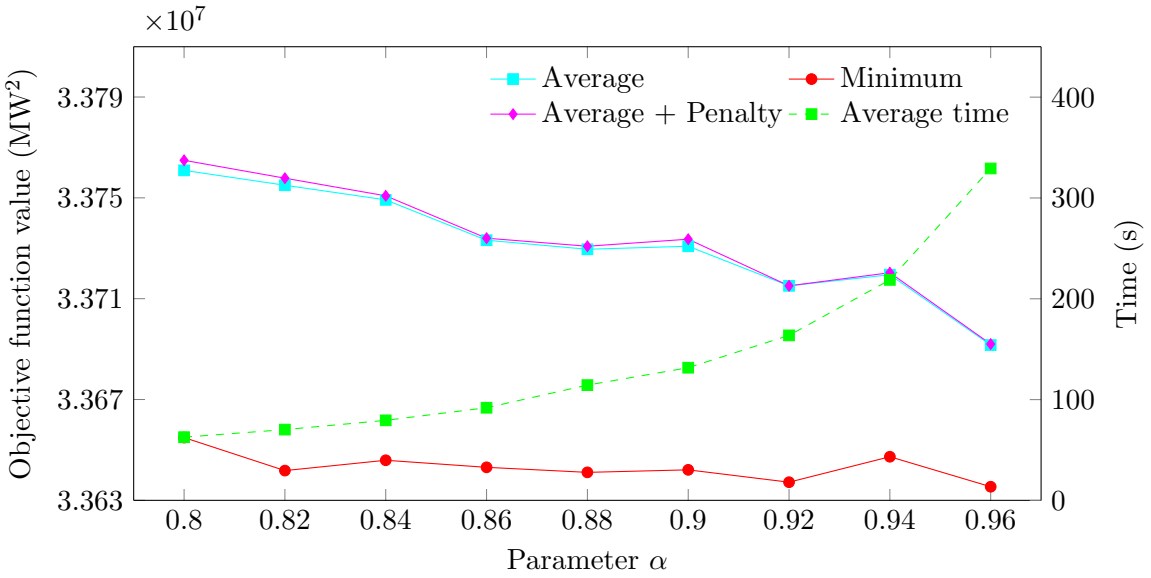


Figure 5.41: *Parameter optimisation for the geometric cooling schedule in IEEE-SA-E.*

The minimum incumbent objective function value remains more or less level between the parameter values of $50n$ and $100n$, but the average incumbent objective function values only reach a level state at $70n$ and above. By choosing $max\_attempt = 80n$, the solution quality remains very good and a decrease in solution time is achieved.

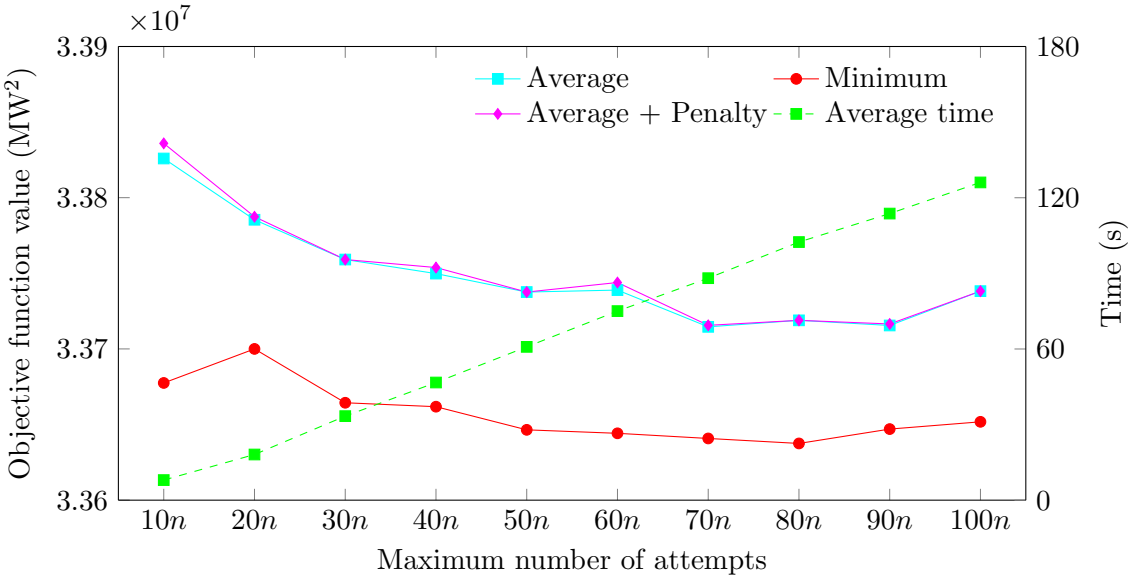


Figure 5.42: *Termination criteria parameter analysis at $T_{min} = 1$ for the geometric cooling schedule in IEEE-SA-E.*

The next cooling schedule considered is that of Huang *et al.* [37]. In Figure 5.43, the difference in incumbent objective function values (average and minimum) between solutions obtained through the use of the AIM and the SDM for calculating the initial temperature, is portrayed. Since the negative bars are more prevalent in the graph, indicating the AIM to be the superior of the two methods, the AIM was chosen for calculating the initial temperature.

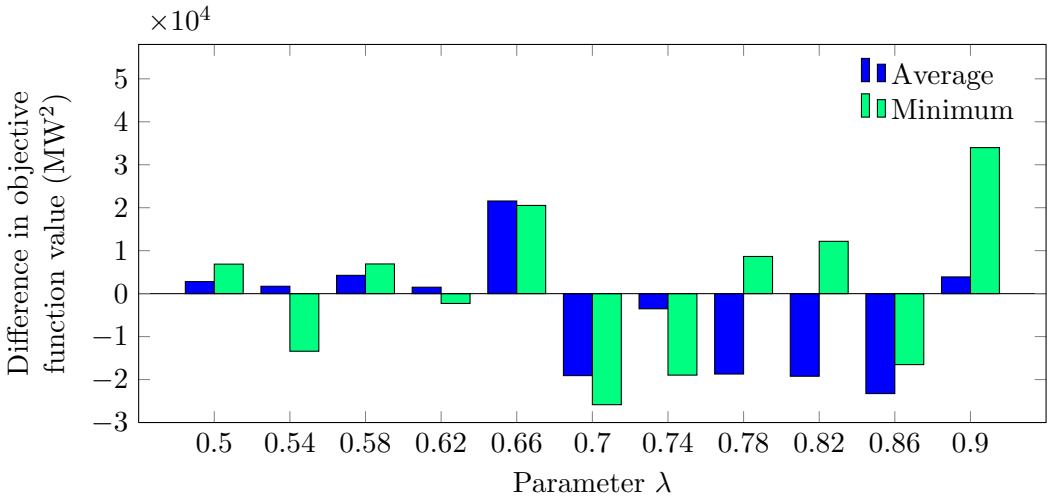


Figure 5.43: *Initial temperature analysis for the cooling schedule of Huang et al. [37] in IEEE-SA-E.*

An illustration of how the incumbent solution quality and average solution time change over the range of λ-values may be found in Figure 5.44. The average solution time decreases linearly over the parameter values; however, at its maximum value (of approximately 40 seconds) it is still of an acceptable magnitude. It may be observed that the average incumbent objective function value remains relatively level over the parameter range with a very slight increase in value, whereas the minimum incumbent objective function value is good within the subrange of $[0.5, 0.62]$. Based on these observations, the value of λ was chosen as 0.54 in order to ensure solutions of high quality.

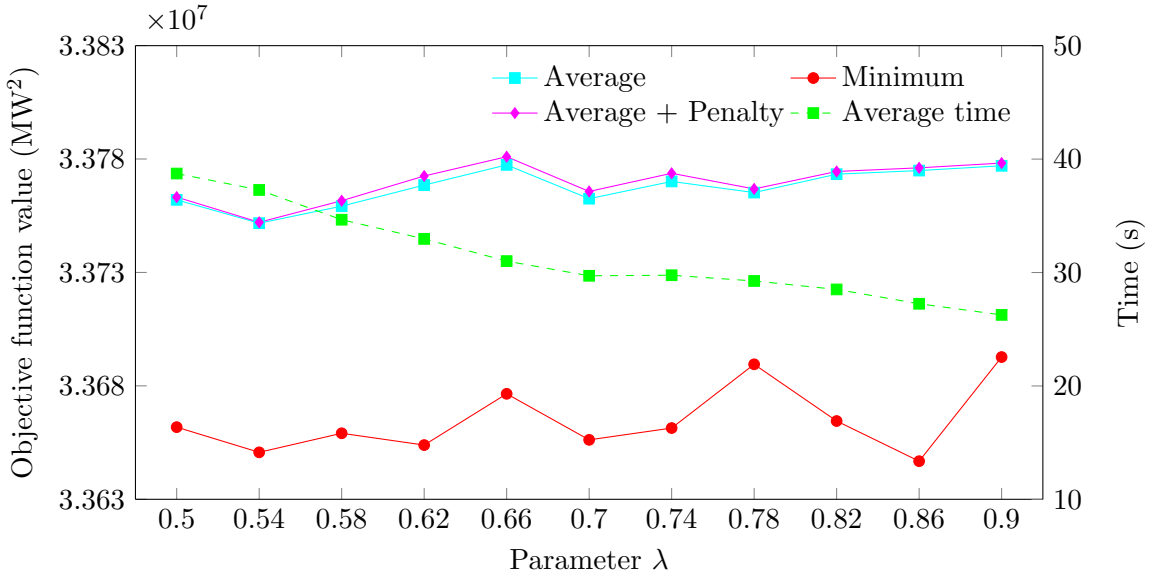


Figure 5.44: *Parameter optimisation for the cooling schedule of Huang et al. [37] in IEEE-SA-E.*

The termination criteria parameter *max_attempt* may be analised in Figure 5.45. The graph depicts the incumbent solution quality and average solution time (with $T_{min} = 1$) over the range of parameter values. With the average solution time obtaining an acceptable maximum value of less than 30 seconds at the end of the parameter range, one only has to consider the solution quality when choosing a value for *max_attempt*. The average incumbent objective function

value continues to improve up to the end of the parameter range, with the minimum incumbent objective function value also following this trend. The value of the *max_attempt* parameter was therefore chosen as $100n$.

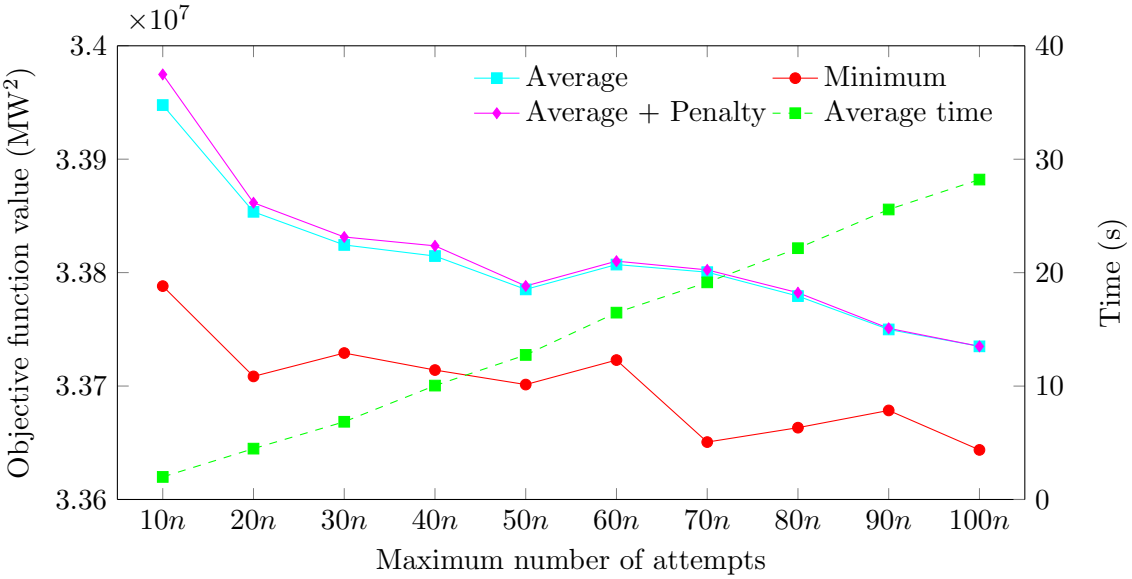


Figure 5.45: *Termination criteria parameter analysis at $T_{min} = 1$ for the cooling schedule of Huang et al. [37] in IEEE-SA-E.*

The cooling schedule proposed by Van Laarhoven *et al.* [81] is analysed next. In the graph in Figure 5.46, the difference in incumbent objective function values (average and minimum) between solutions obtained from using the AIM and the SDM for determining the initial temperature is shown. Since the graph displays a much larger prevalence in positive bars, the SDM was chosen for determining the initial temperature.

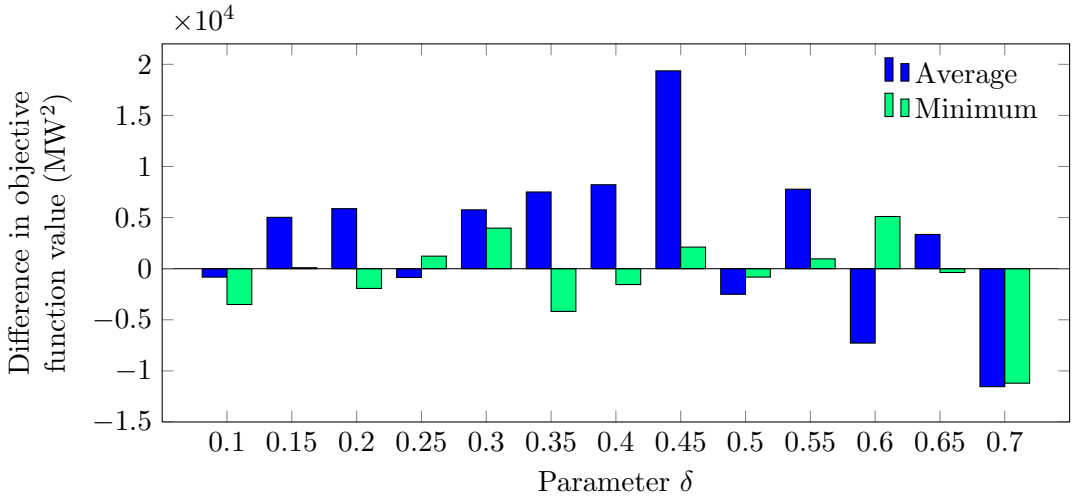


Figure 5.46: *Initial temperature analysis for the cooling schedule of Van Laarhoven et al. [81] in IEEE-SA-E.*

In Figure 5.47, the variations in incumbent solution quality and average solution time over the different values of the parameter δ are analysed. An exponential decrease in average solution

time may be observed in the graph. Unfortunately, the cooling schedule requires a considerable amount of solution time. The solution quality, however, is very good and consistent — the minimum incumbent objective function values are very close to one another over the range of values of δ and the average incumbent objective function values slowly increase, with negligible penalty values, as δ increases. In order to keep the solution time within an acceptable range, the value of δ was chosen as 0.35; however, any value within the range $[0.2, 0.4]$ will provide very good results in less than 5 minutes of solution time.

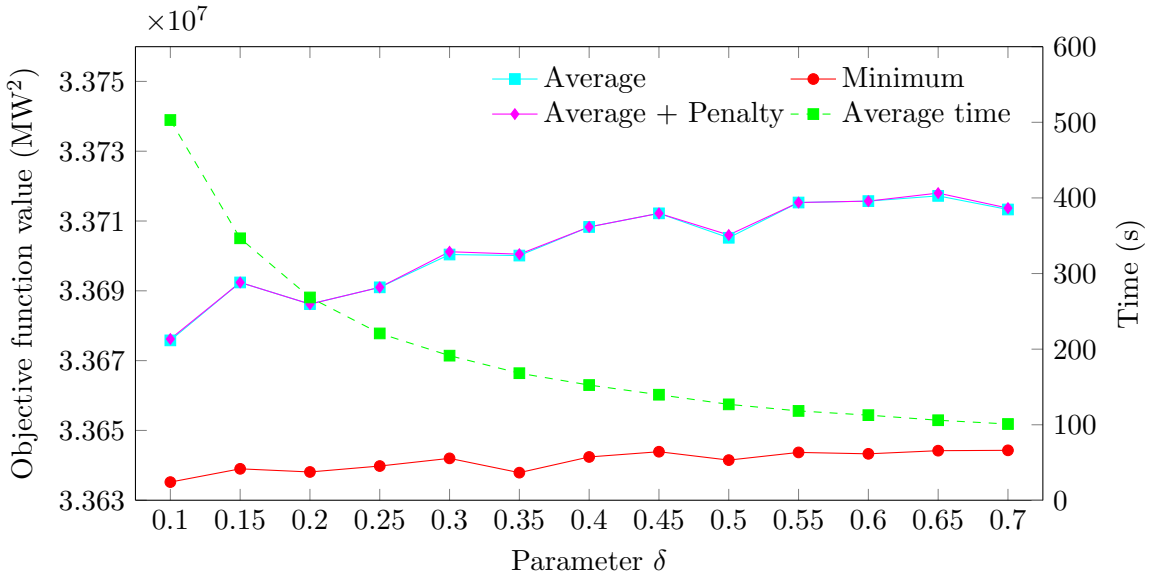


Figure 5.47: *Parameter optimisation for the cooling schedule of Van Laarhoven et al. [81] in IEEE-SA-E.*

In order to choose a value for the parameter *max_attempt*, one may consider the graph in Figure 5.48. An illustration is given of the incumbent solution quality and average solution time over the range of parameter values with the final temperature chosen as $T_{min} = 1$. Very good average incumbent objective function values are only reached near the end of the parameter range (values of $90n$ and $100n$). The minimum incumbent objective function values level off at $70n$ and above, while the average solution time increases linearly. A decrease in computational time may be achieved without affecting the solution quality by choosing $max\_attempt = 90n$.

Finally, the cooling schedule of Triki *et al.* [79] is considered for analysis. Consider the graphs in Figures 5.37 and 5.38 which contain the incumbent solution quality and average solution time over the different parameter combinations. The parameters μ_2 and μ_1 are grouped together again in single (μ_2, μ_1) "units" with those in Figure 5.49 grouped together in increasing order of μ_2 and those in Figure 5.50 in increasing order of μ_1. When considering the average solution times in Figures 5.49(a) and 5.50(a), it may be observed that the values are much more evenly spread over the different ζ-values than before. As before, an increase in μ_1 generally causes the solution time to increase as well, along with an improvement in solution quality, as observed in Figures 5.49(c) and 5.50(c). The benefit of shorter solution times, however, does not outweigh the deterioration in solution quality that a value of $\mu_1 = 5$ provides. Unfortunately, there is no visible effect in these graphs which may be attributed to the influence of μ_2. Based on the collective impression obtained from the parameter analysis, the parameter value combination was chosen as $\zeta = 1.06$, $\mu_2 = 10$ and $\mu_1 = 10$.

Finally, an analysis of the parameter *max_attempt* is performed via the graph in Figure 5.51. It provides one with the variations in incumbent solution quality and average solution time over the

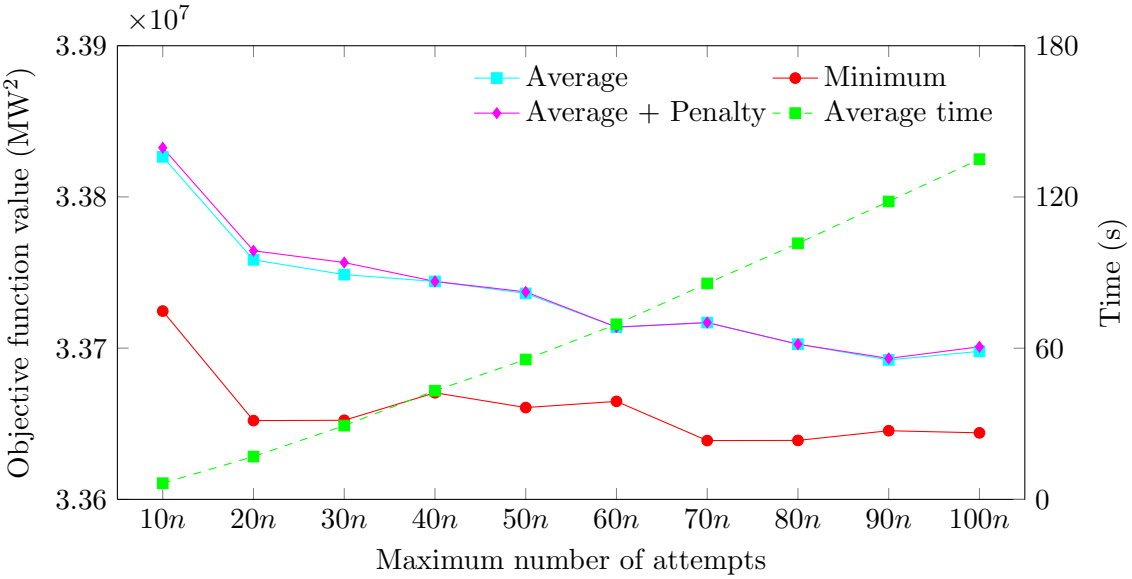


Figure 5.48: *Termination criteria parameter analysis at $T_{min} = 1$ for the cooling schedule of Van Laarhoven et al. [81] in IEEE-SA-E.*

parameter range at a final temperature of $T_{min} = 1$. A relatively sharp improvement in solution quality is obtained during the initial parameter values up to $50n$. From there upwards, both the average and minimum incumbent objective function values remain relatively level. However, the average penalty values only become acceptable at $90n$ and above. Since the average solution time increases approximately linearly, a value of $max\_attempt = 90n$ was selected.

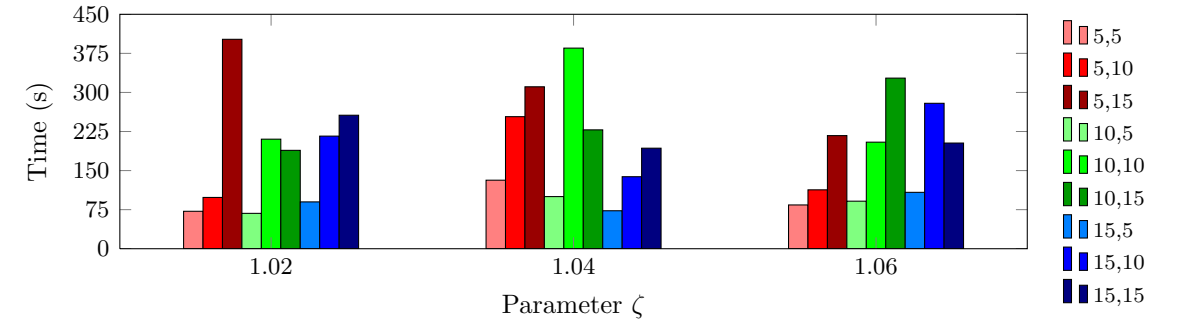
5.3.3 Summary of parameter values

A summary of the optimised parameter values for all the methods are presented in Table 5.7 for the random search heuristic and in Table 5.8 for the simulated annealing algorithm. These

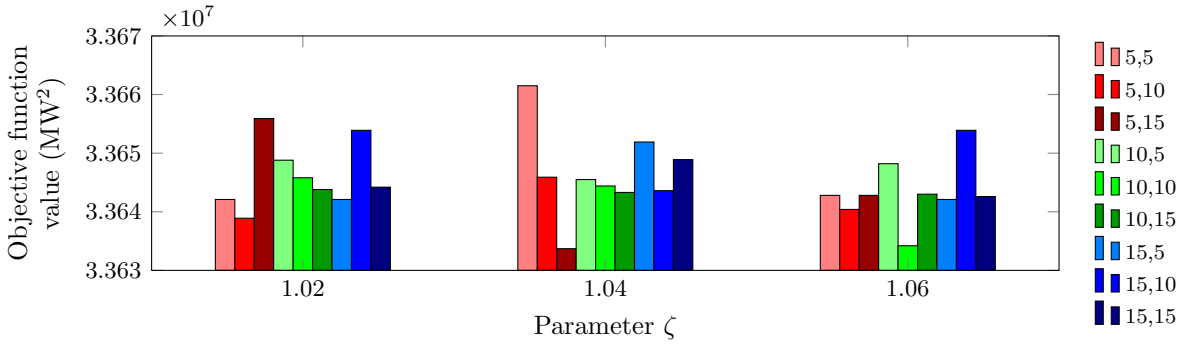
	The 21-unit system		The 22-unit system		The IEEE-RTS inspired system	
Parameter	Classical	Ejection chain	Classical	Ejection chain	Classical	Ejection chain
Neighbourhood size	m	$2n$	m	$2n$	m	n
Iterations (I)	8 000	7 000	9 000	9 000	10 000	10 000
Number of iterations without improvement	$0.8I$	$0.8I$	$0.8I$	$0.8I$	$0.9I$	$0.8I$

Table 5.7: *Optimised parameter values for the random search heuristic.*

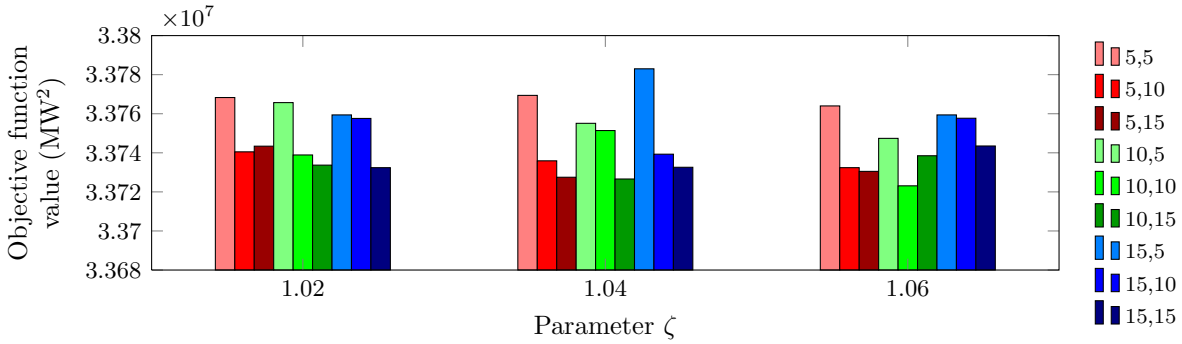
values are used for the investigation into the different cooling schedules on the GMS problem, how the new ejection chain neighbourhood operator performs against the classical operator, and the effects of the modifications to the simulated annealing algorithm.



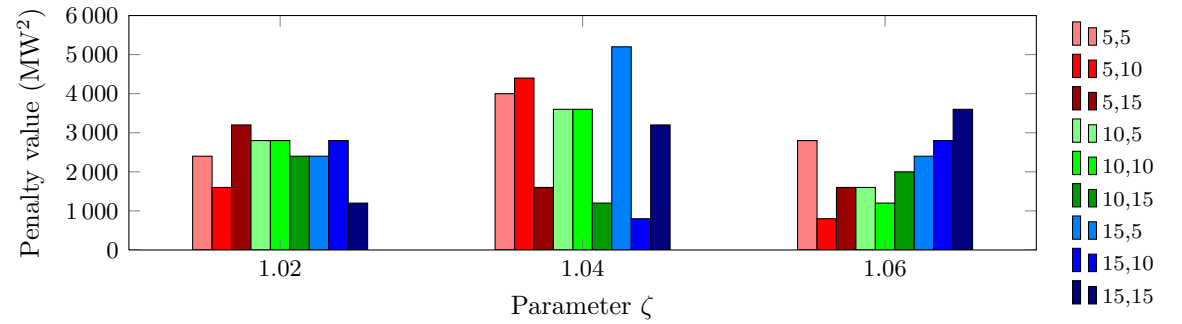
(a) Average solution time



(b) Minimum incumbent objective function value

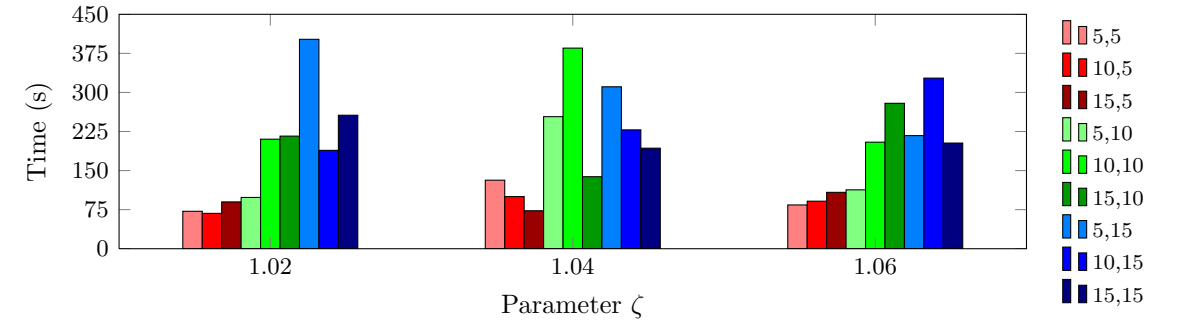


(c) Average incumbent objective function value

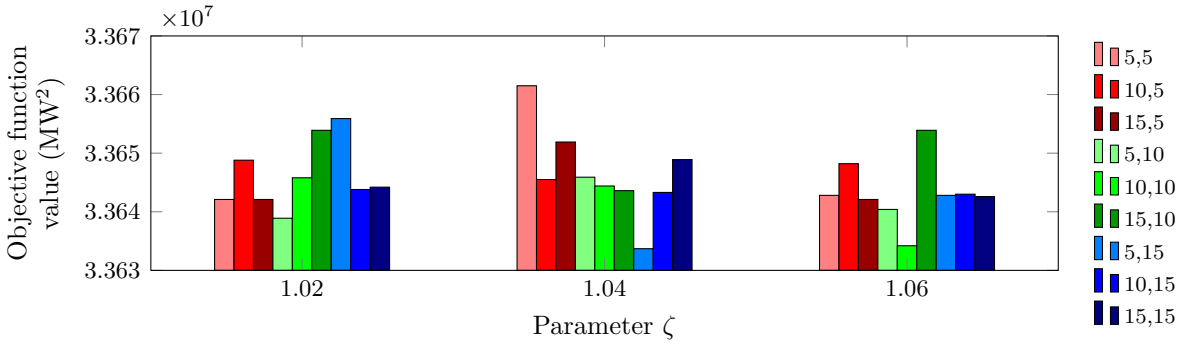


(d) Average incumbent penalty

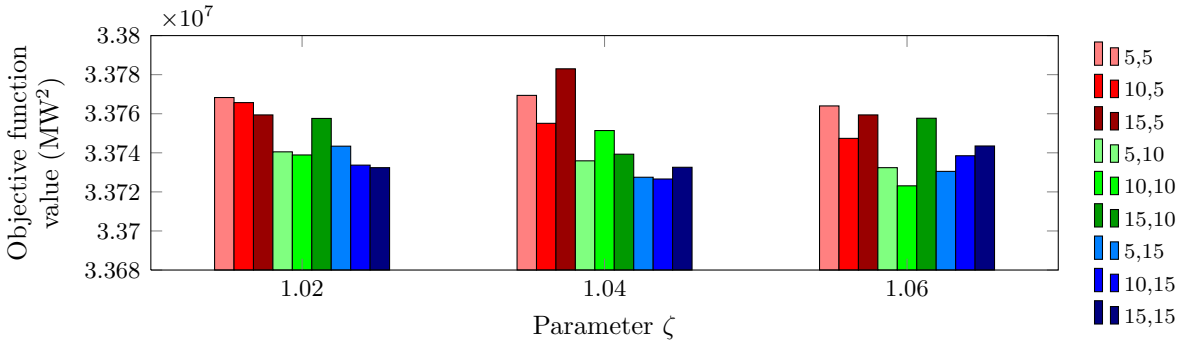
Figure 5.49: *Parameter optimisation for the cooling schedule of Triki et al. [79] in IEEE-SA-E. Legend entries in increasing order of the parameter μ_2.*



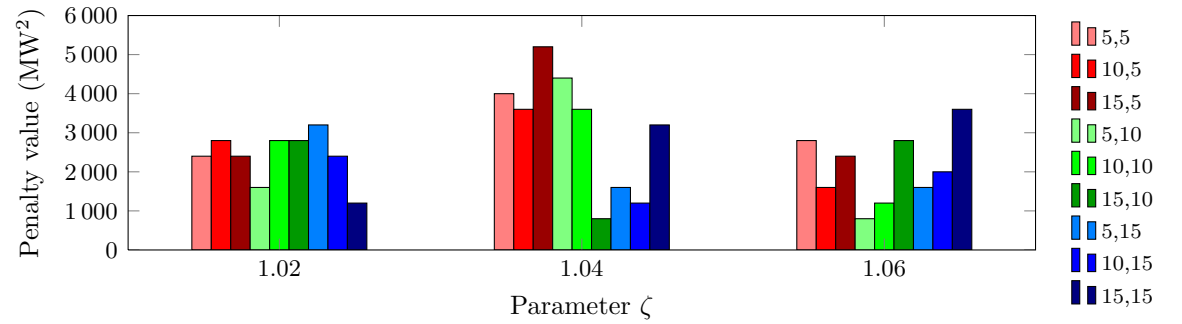
(a) Average solution time



(b) Minimum incumbent objective function value



(c) Average incumbent objective function value



(d) Average incumbent penalty

Figure 5.50: *Parameter optimisation for the cooling schedule of Triki et al. [79] in IEEE-SA-E. Legend entries in increasing order of the parameter μ_1.*

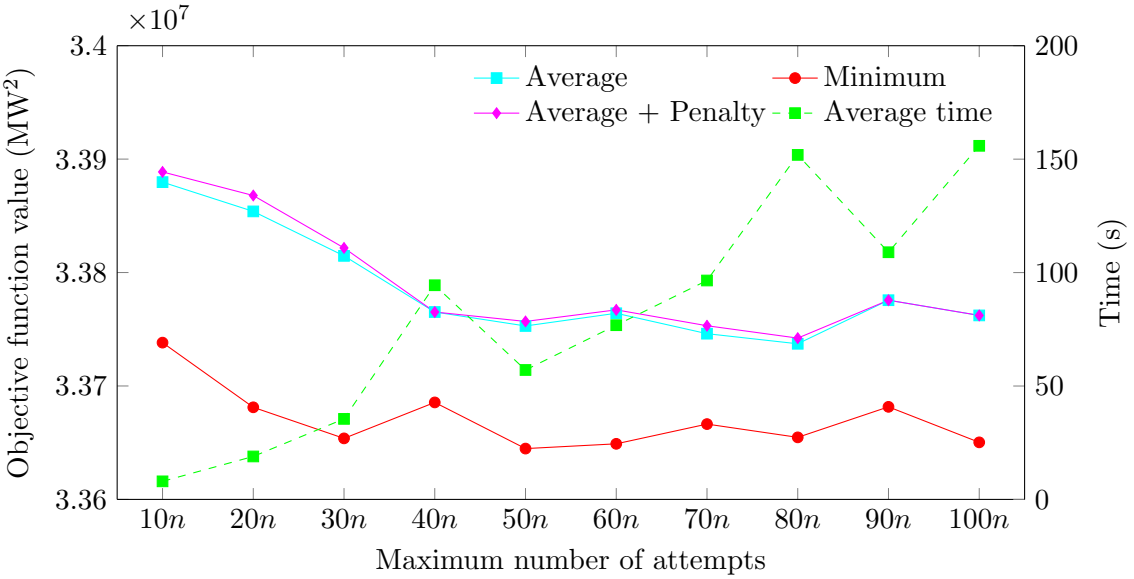


Figure 5.51: *Termination criteria parameter analysis at $T_{min} = 1$ for the cooling schedule of Triki et al. [79] in IEEE-SA-E.*

		The 21-unit system		The 22-unit system		The IEEE-RTS inspired system	
Cooling schedule	Parameter	Classical	Ejection chain	Classical	Ejection chain	Classical	Ejection chain
Geometric	Method for T_0	AIM	AIM	AIM	AIM	AIM	AIM
	α	0.95	0.92	0.97	0.96	0.95	0.92
	max_attempt	$100n$	$100n$	$80n$	$80n$	$90n$	$80n$
Huang	Method for T_0	AIM	AIM	AIM	SDM	AIM	AIM
	λ	0.6	0.6	0.62	0.72	0.54	0.54
	max_attempt	$100n$	$100n$	$100n$	$100n$	$100n$	$100n$
Van Laarhoven	Method for T_0	AIM	AIM	AIM	AIM	AIM	SDM
	δ	0.16	0.16	0.16	0.2	0.15	0.35
	max_attempt	$90n$	$100n$	$80n$	$90n$	$90n$	$90n$
Triki	Method for T_0	SDM	SDM	SDM	SDM	SDM	SDM
	ζ	1.02	1.02	1.02	1.02	1.06	1.06
	μ_1	10	10	10	10	10	10
	μ_2	10	10	5	10	10	10
	max_attempt	$100n$	$100n$	$100n$	$100n$	$100n$	$90n$

Table 5.8: *Optimised parameter values for the simulated annealing algorithm.*

5.4 Chapter summary

In this chapter, the parameter evaluation was presented with respect to the application of the approximate solution approach towards solving the GMS benchmark test systems. Three GMS benchmark test systems were described in §5.1, two being systems previously studied in the literature, and one being newly created.

Since a soft constraint approach was adopted in the approximate solution approach for the GMS problem, problem instance-dependent penalty weights had to be determined for each test system. The methodology of determining these penalty weights, and the subsequent computation thereof

for each system, were presented in §5.2.

The solution techniques of the approximate solution approach for the GMS problem require a number of parameter values to be set in order for the techniques to perform optimally. These values are also typically problem instance-dependent and the results of a detailed parameter optimisation process for each test system was performed and presented in §5.3.

CHAPTER 6

Experimental results

Contents

An analysis of the different cooling schedules, the new ejection chain neighbourhood move operator and the proposed modifications in the simulated annealing algorithm are presented in this chapter. Each variation of the solution techniques was used to solve 50 problem instances for each benchmark test system using the optimised parameter values, as determined in Chapter 5. In each case, the same set of 50 random initial solutions were used.

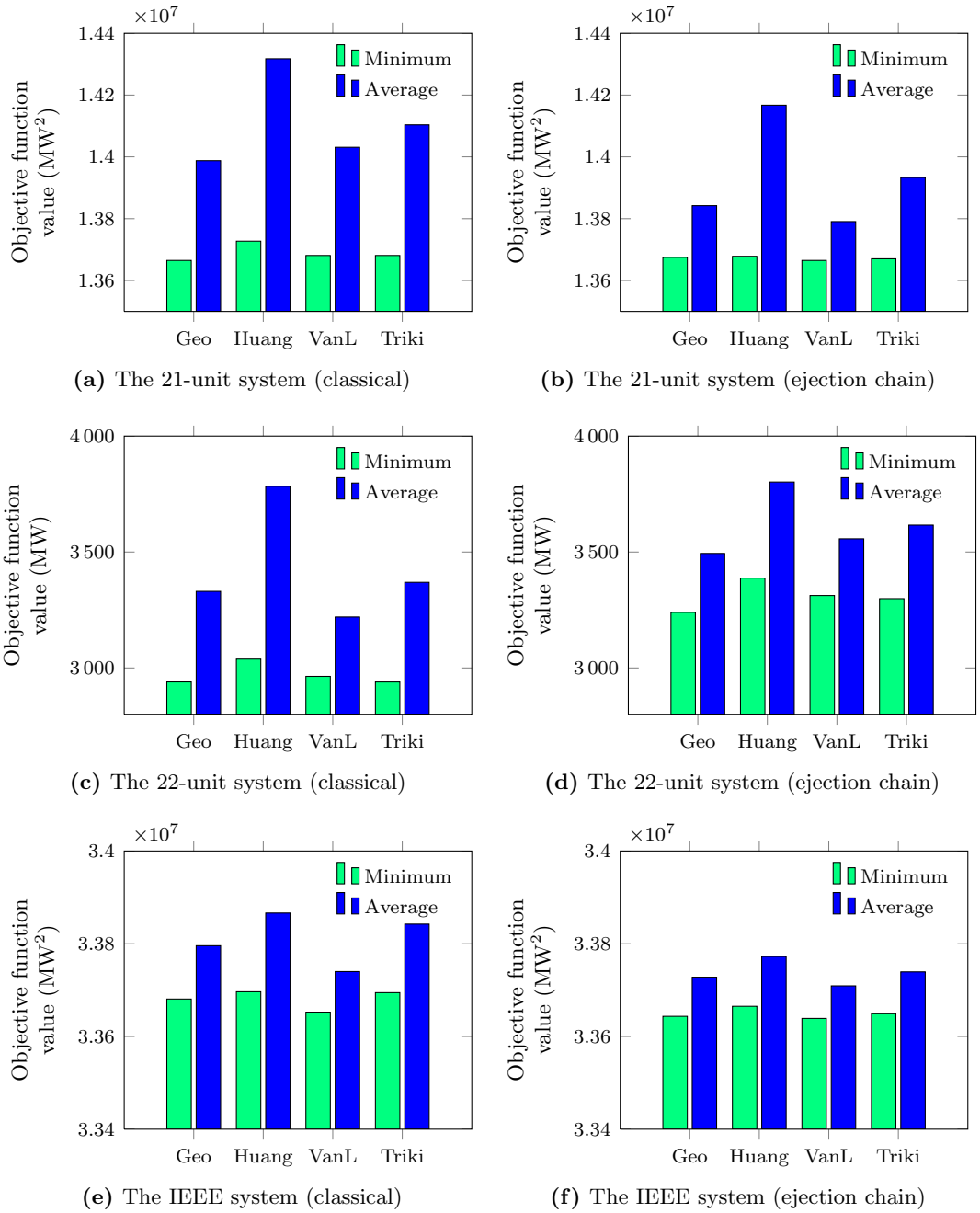
Following the performance analysis of the variations in solution technique, the results of an application of the exact solution approach described in §4.1 are presented for the three benchmark test systems introduced in §5.1. The chapter is concluded with the presentation of the best solutions obtained for each benchmark test system.

Again, this computational evaluation was performed on a personal computer with a 3.0 GHz Intel® Core™ 2 Duo E8400 processor and 3.25 GB RAM, running on Microsoft Windows XP Professional (Version 2002, Service Pack 3). The exact solution approach was implemented in the software package LINGO and the approximate solution approach was implemented in the software package MATLAB.

6.1 Performance analysis of the cooling schedules

Four SA cooling schedules were introduced in §4.2.5 of which only one had been adopted within a GMS context before, namely the geometric law of decrease. In order to establish which schedule performs the best, consider Figure 6.1. The graphs in the figure represent the minimum and

average incumbent objective function values obtained for each test system over all four cooling schedules, for each neighbourhood structure. Each horizontal pair of graphs correspond to one of the test systems and the graphs in a pair should not be compared with each other. The aim is to determine which cooling schedule performs the best, irrespective of neighbourhood structure.



(a) The 21-unit system (classical)

(b) The 21-unit system (ejection chain)

(c) The 22-unit system (classical)

(d) The 22-unit system (ejection chain)

(e) The IEEE system (classical)

(f) The IEEE system (ejection chain)

Figure 6.1: *Comparison of cooling schedules by means of incumbent objective function values. In the graphs, "Geo" refers to the geometric cooling schedule, "Huang" refers to the cooling schedule proposed by Huang et al. [37], "VanL" refers to the cooling schedule proposed by Van Laarhoven et al. [81] and "Triki" refers to the cooling schedule proposed by Triki et al. [79].*

In each test system, the cooling schedule of Huang *et al.* [37] performs the worst — at minimum level as well as on average. However, in each instance, the minimum objective function value is very close to those of the others (within 1% in the 21-unit and IEEE system and within 5% in the 22-unit system. Therefore, the cooling schedule of Huang *et al.* [37] should not be dismissed off-hand. Table 6.1 contains the average solution times (with standard deviations) corresponding to the objective function values in Figure 6.1. The schedule of Huang *et al.* [37] achieves an average solution time that is in most cases approximately ten times faster than that of the other schedules and it is very consistent (as seen by its small standard deviation) unlike that of the schedule proposed by Triki *et al.* [79]. The loss in objective function value accuracy may be considered acceptable in view of the large benefit in solution time. As mentioned above, the standard deviation in average solution time for the schedule of Triki *et al.* [79] is very high, potentially resulting in erratic solution times — a phenomenon best avoided. Furthermore, its average incumbent objective function value levels are second to worst, although its minimum objective function values are the best in some cases.

		Classical		Ejection chain	
System	Schedule	Average time (s)	Standard deviation	Average time (s)	Standard deviation
21-unit	Geo	48.35	0.19	59.1	7.13
	Huang	2.98	0.33	7.19	0.76
	VanL	23.84	1.52	66.74	4.66
	Triki	17.61	24.93	59.79	119.97
22-unit	Geo	35.67	0.98	35.76	5.49
	Huang	3.98	0.44	8.49	1.06
	VanL	43.12	2.83	65.39	3.96
	Triki	41.5	52.37	42.91	38.01
IEEE	Geo	78.95	3.49	131.9	9.45
	Huang	13.78	1.19	37.3	2.72
	VanL	119.89	6.74	151.25	7.3
	Triki	286.01	1008.9	170.2	187.35

Table 6.1: *Comparison of cooling schedules by means of average solution times. In the table, "Geo" refers to the geometric cooling schedule, "Huang" refers to the cooling schedule proposed by Huang et al. [37], "VanL" refers to the cooling schedule proposed by Van Laarhoven et al. [81] and "Triki" refers to the cooling schedule proposed by Triki et al. [79].*

Of the remaining two schedules, the schedule proposed by Van Laarhoven *et al.* [81] provides the best solution quality over all the test systems. It attains the lowest average incumbent objective function value level in four of the six cases, the lowest minimum objective function value in three of the six cases and achieves a very consistent solution time. The only drawback is its relatively long average solution time — requiring slightly more time than the geometric schedule. The geometric schedule achieves the second to best average incumbent objective function value levels and attains the lowest minimum incumbent objective function value in two of the cases. As with the schedules of Huang *et al.* [37] and Van Laarhoven *et al.* [81], the geometric schedule is very consistent in its average solution time according the standard deviations in Table 6.1.

Based on the above analysis, there is little to choose between the geometric schedule and the schedule of Van Laarhoven *et al.* [81] as both attain best minimum incumbent objective function values, with the geometric schedule requiring slightly less solution time. However, the schedule

of Van Laarhoven *et al.* [81] obtains a better average incumbent solution quality. As such, any of these two schedules may be the schedule of choice to obtain a very good solution. If the objective is to find a number of good solutions, preference should be given to the schedule of Van Laarhoven *et al.* [81]. The alternative to these two schedules, should a single, quick, good solution (not necessarily the best possible) be required, is to choose the schedule of Huang *et al.* [37] as it competes very favourably in terms of minimum incumbent objective function value, but is far superior in terms of solution time. Due to its unpredictable (and potentially long) solution times and not achieving any advantage above the other schedules with respect to average incumbent solution quality, the schedule of Triki *et al.* [79] is determined te be the least desirable cooling schedule in a GMS context.

6.2 Performance analysis of the new neighbourhood move

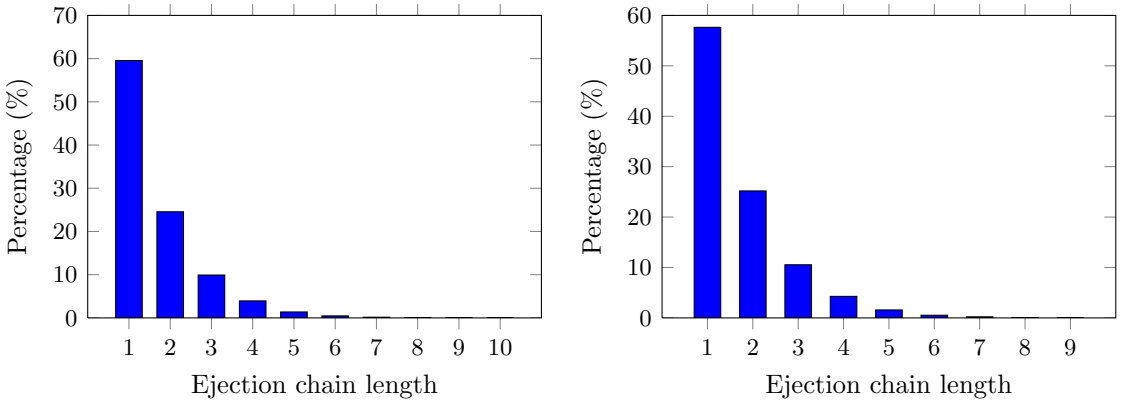
The first step in analysing the new ejection chain neighbourhood move operator is to compare it to the classical operator in the (simple) random search heuristic. The comparative results are presented in Table 6.2. It may firstly be noticed that the ejection chain operator's average solution times were nearly double that of the classical operator in each test system. This is not unexpected, as an ejection chain is a compound move, involving more than one unit's starting time to be modified, which in turn causes more time-consuming constraint evaluations. Secondly, in two of the three test systems, the classical operator achieved better minimum incumbent objective function values. However, the value of the ejection chain neighbourhood may clearly be seen in the quality of the solutions — the average incumbent objective function values obtained via the ejection chain operator are significantly better than those obtained via the classical operator, and the standard deviations are more than halved. Furthermore, the ejection chain operator achieved a considerable reduction in penalty values, suggesting that more of the incumbent solutions obtained by the random search heuristic were feasible. These preliminary findings illustrate the potential of the ejection chain operator for use in more sophisticated algorithms.

System	Neighbourhood	Average time (s)	Standard deviation	Minimum incumbent objective function value	Average incumbent objective function value	Standard deviation	Average penalty	Standard deviation
21-unit	Classical	25.66	1.61	13 679 339	14 209 078	294 150	90 000	194 044
	Ejection	80.64	9.50	13 889 975	14 126 540	139 933	30 000	119 949
22-unit	Classical	48.41	3.09	3 161.31	4 065.30	670.90	17.20	28.43
	Ejection	95.84	8.45	3 413.31	3 658.86	251.45	3.80	9.23
IEEE	Classical	51.38	3.41	33 860 134	34 261 904	241 620	50 400	51 625
	Ejection	90.70	1.58	33 653 082	33 819 405	102 335	7 200	16 542

Table 6.2: *Comparison of neighbourhood move operators within the random search heuristic.*

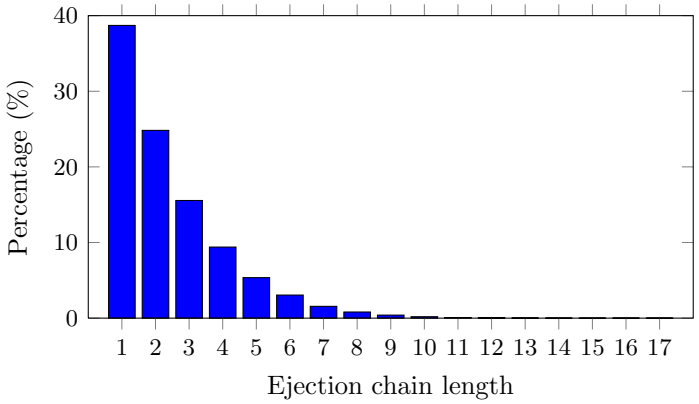
In the implementation of the ejection chain neighbourhood move operator, only one column (time period) is used when performing the vertical steps in the ejection chain. Figure 6.2 contains the typical distributions of the ejection chain lengths which were selected during the SA algorithm execution for each test system. In all three test systems, the concern regarding the generation of a significant number of non-trivial ejection chains (see §4.2.2) is not applicable, mainly due to the number of generating units not being significantly less than the number of

time periods. The graphs in Figure 6.2 illustrate that the number of non-trivial ejection chains are satisfactory.



(a) The 21-unit system

(b) The 22-unit system



(c) The IEEE system

Figure 6.2: *Typical distributions of the ejection chain lengths for each test system.*

Consider the results obtained from using the two neighbourhood structures within the simulated annealing algorithm. In Figure 6.3, the minimum incumbent objective function values obtained by means of the two neighbourhood structures are compared in the left-hand column of graphs, while the average incumbent objective function values are compared in the right-hand column of graphs. In order to grasp the scope on the neighbourhood effect, all four cooling schedules' results are provided for each test system. However, the effect of the neighbourhood structures should be determined irrespective of the cooling schedule. Figures 6.3(a) and 6.3(b) illustrate the superiority of the ejection chain operator over the classical operator in the 21-unit system (seven out of eight cases). Although the minimum incumbent objective function values remain quite close to each other (except for the schedule of Huang *et al.* [37]), the average incumbent objective function values are significantly improved by the ejection chain moves. The 22-unit system, however, shows the complete opposite picture — in both minimum and average incumbent solutions, the classical neighbourhood obtained significantly better results over all the cooling schedules. Lastly, the results from the IEEE inspired system in Figures 6.3(e) and 6.3(f) show the same behaviour as in the 21-unit system — the ejection chain neighbourhood clearly outperforms the classical neighbourhood in that system (all eight cases).

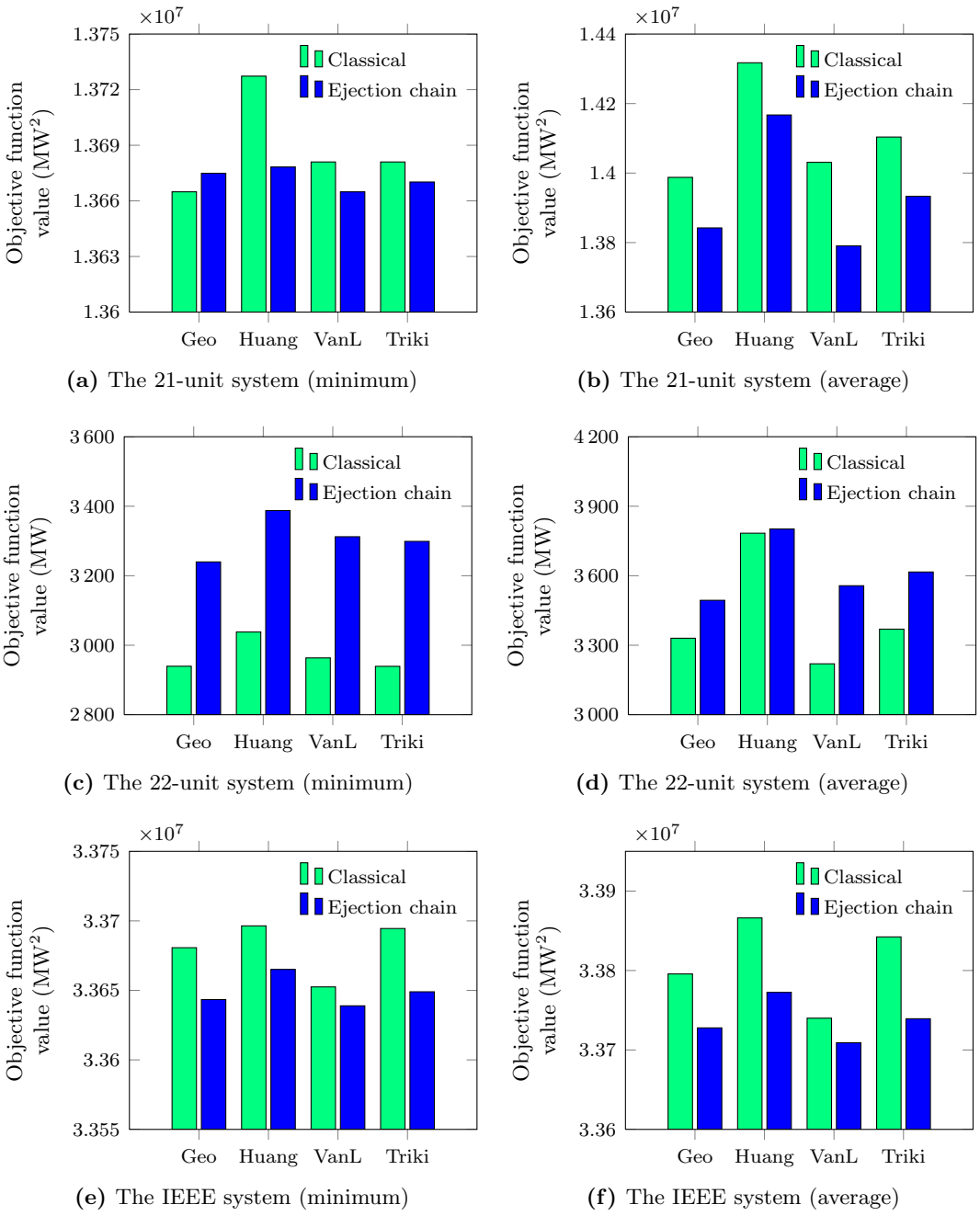


Figure 6.3: *Comparison of neighbourhood move operators by means of incumbent objective function values. In the graphs, "Geo" refers to the geometric cooling schedule, "Huang" refers to the cooling schedule proposed by Huang et al. [37], "VanL" refers to the cooling schedule proposed by Van Laarhoven et al. [81] and "Triki" refers to the cooling schedule proposed by Triki et al. [79].*

A question arises as to why the ejection chain neighbourhood did not show the same superior results in the 22-unit system as it did in the other two test systems. A brief investigation into the objective function of the test system was performed. When a sum of squares of reserve levels objective function is used instead for the 22-unit system, the differences between the objective function values obtained using the two move operators were proportionally significantly smaller than in the case above, using the sum of absolute differences objective function. However, the

classical move operator still outperformed the ejection chain move operator consistently. A possible reason for this behaviour may be the different natures of the solution spaces of the three problems. The 21-unit system is highly constrained by its maintenance crew constraint set and the IEEE system by its load demand and maintenance crew constraint sets. The 22-unit system, on the other hand, does not have a maintenance crew constraint set and furthermore is not highly constrained by the constraint sets it does have. As such, the 22-unit system has a much larger feasible solution space than the other two systems. Why the ejection chain neighbourhood should fail to find very good solutions in such a larger solution space is, however, unknown. It may be that the ejection chains execute too many "global" jumps in the solution space and never locally deepen the search to obtain those very good solutions that the classical neighbourhood explores. However, without the benefit of having more benchmark test systems, a deeper investigation into this phenomenon is not possible within the scope of this thesis.

The average solution times required when using the ejection chain neighbourhood in the SA algorithm, in contrast with the observation made in the random search heuristic results, are not nearly double those observed when using the classical neighbourhood. This may be seen from the solution times presented in Table 6.1. The average solution times of the two neighbourhood structures range from almost remaining the same (geometric schedule in the 22-unit system) to almost triplicating (schedule of Van Laarhoven *et al.* [81] in the 21-unit system). However, the ejection chain neighbourhood move operator will necessarily be the operator that requires more time.

An analysis was also conducted with respect to the average penalty values. However, these results do not provide additional insight — for the 21- and 22-unit systems, the average penalty values remained more or less the same between the two neighbourhood structures, while the IEEE system experienced a slight improvement over all four cooling schedules by using the ejection chain neighbourhood.

The performance analysis of the new ejection chain neighbourhood move operator presented above, illustrates its effectiveness and mostly superior results over that of the classical neighbourhood operator. Although it requires more solution time, the ejection chain operator may provide a valuable increase in solution quality over that of the classical operator. However, the ejection chain operator may be considered in conjunction with the classical operator, as some problems may utilise the classical operator more effectively, as seen in the 22-unit test system.

6.3 Performance analysis of the proposed modifications

The first proposed modification to the SA algorithm was the introduction of a local search heuristic, thereby hybridising the SA algorithm, as described in §4.2.6. This local search heuristic was applied in two ways — in the first hybridisation the local search heuristic was applied to every new incumbent solution encountered during the algorithm's execution. The results of this hybridisation are presented in Table 6.3. The results indicate that this hybridisation is able to achieve an effective improvement over the SA algorithm since a significant number of incumbent solutions uncovered by both neighbourhood structures in all three test systems were improved upon. In most cases over 50% of the incumbent solutions were improved upon. The average objective function value improvement percentage depends on the test system, therefore the "small" improvements should not be regarded as insignificant — the point is that many of the solutions were improved upon. Typically, these small improvements guide the search into a local optimum. The maximum improvement percentages provides one with a good impression of the potential of this hybridisation — up to a 29% improvement in one case.

System	Cooling schedule	Classical neighbourhood Percentage underwent improvement	Classical neighbourhood Avg OFV improvement	Classical neighbourhood Max OFV improvement	Ejection chain neighbourhood Percentage underwent improvement	Ejection chain neighbourhood Avg OFV improvement	Ejection chain neighbourhood Max OFV improvement
21-unit	Geo	38%	0.49%	1.27%	18%	0.50%	1.48%
	Huang	46%	1.25%	3.94%	62%	0.82%	3.49%
	VanL	54%	0.73%	2.59%	32%	0.43%	1.39%
	Triki	42%	0.80%	2.86%	44%	0.87%	2.90%
22-unit	Geo	42%	2.82%	8.74%	94%	3.00%	12.68%
	Huang	56%	3.74%	29.18%	94%	4.96%	13.85%
	VanL	50%	1.91%	8.34%	100%	4.26%	11.45%
	Triki	36%	2.94%	19.73%	96%	3.62%	14.37%
IEEE	Geo	54%	0.03%	0.22%	100%	0.02%	0.08%
	Huang	56%	0.05%	0.42%	100%	0.05%	0.34%
	VanL	82%	0.03%	0.18%	100%	0.03%	0.16%
	Triki	58%	0.09%	0.75%	98%	0.03%	0.23%

Table 6.3: *Performance analysis of the first algorithmic hybridisation. In the table, "Geo" refers to the geometric cooling schedule, "Huang" refers to the cooling schedule proposed by Huang et al. [37], "VanL" refers to the cooling schedule proposed by Van Laarhoven et al. [81] and "Triki" refers to the cooling schedule proposed by Triki et al. [79].*

Additionally, it is not suprising that the ejection chain neighbourhood has such a significant percentage of incumbent solutions that underwent an improvement, as the move operator typically functions more globally over the solution space than does the classical operator (which shares the same move operator as the local search), without necessarily intensifying the search at a certain position in the solution space. Therefore, many of the solutions found may be improved upon by the local search heuristic.

The second hybridisation, which applies the local search heuristic to the final incumbent solution obtained by the solution SA algorithm, may be analysed from the results in Table 6.4. Again, the local search heuristic obtains improvements on incumbent solutions in all the cases considered. It is noticeable that the number of incumbent solutions which underwent the second hybridisation is smaller than that in the first hybridisation, which is to be expected since the search is only a applied to single solution in each instance, whereas in the first hybridisation, it is applied to numerous solutions within each instance. Furthermore, the maximum improvement percentages are also lower due to the same reason. However, the average improvement percentages compare very favourably to those of the first hybridisation.

Based on the frequency with which incumbent solutions are improved by both the hybridisations, either one may be included in the simulated annealing algorithm as a good method to improve upon the quality of solutions. The difference in solution times do not vary significantly between the two hybridisations. As such, one may consider the first hybridisation to be the superior of the two. Note, however, that the first hybridisation does not necessarily reduce to the second hybridisation in a worst case scenario and should therefore not be considered as a generalisation of the second hybridisation.

Finally, the proposed modification of introducing a "good" random initial solution is analysed. Figures 6.4 and 6.5 contain graphs comparing the minimum and average incumbent objective function values obtained by using the original initial random solutions and the modified good initial solutions for each test system. Each horizontal pair of graphs correspond to one of the test systems.

System	Cooling schedule	Classical neighbourhood Percentage underwent improvement	Classical neighbourhood Avg OFV improvement	Classical neighbourhood Max OFV improvement	Ejection chain neighbourhood Percentage underwent improvement	Ejection chain neighbourhood Avg OFV improvement	Ejection chain neighbourhood Max OFV improvement
21 unit	Geo	16%	0.59%	1.12%	14%	0.73%	1.48%
	Huang	10%	1.33%	2.56%	50%	0.59%	1.75%
	VanL	24%	0.49%	1.04%	20%	0.35%	1.08%
	Triki	26%	0.89%	2.86%	38%	0.77%	2.90%
22 unit	Geo	22%	0.92%	1.88%	80%	3.97%	13.33%
	Huang	16%	1.36%	4.40%	82%	4.74%	10.45%
	VanL	24%	1.14%	4.37%	96%	3.53%	15.87%
	Triki	16%	2.21%	7.12%	90%	4.20%	17.51%
IEEE	Geo	30%	0.01%	0.03%	100%	0.03%	0.08%
	Huang	18%	0.00%	0.01%	100%	0.04%	0.10%
	VanL	48%	0.02%	0.18%	100%	0.03%	0.13%
	Triki	18%	0.02%	0.11%	96%	0.03%	0.16%

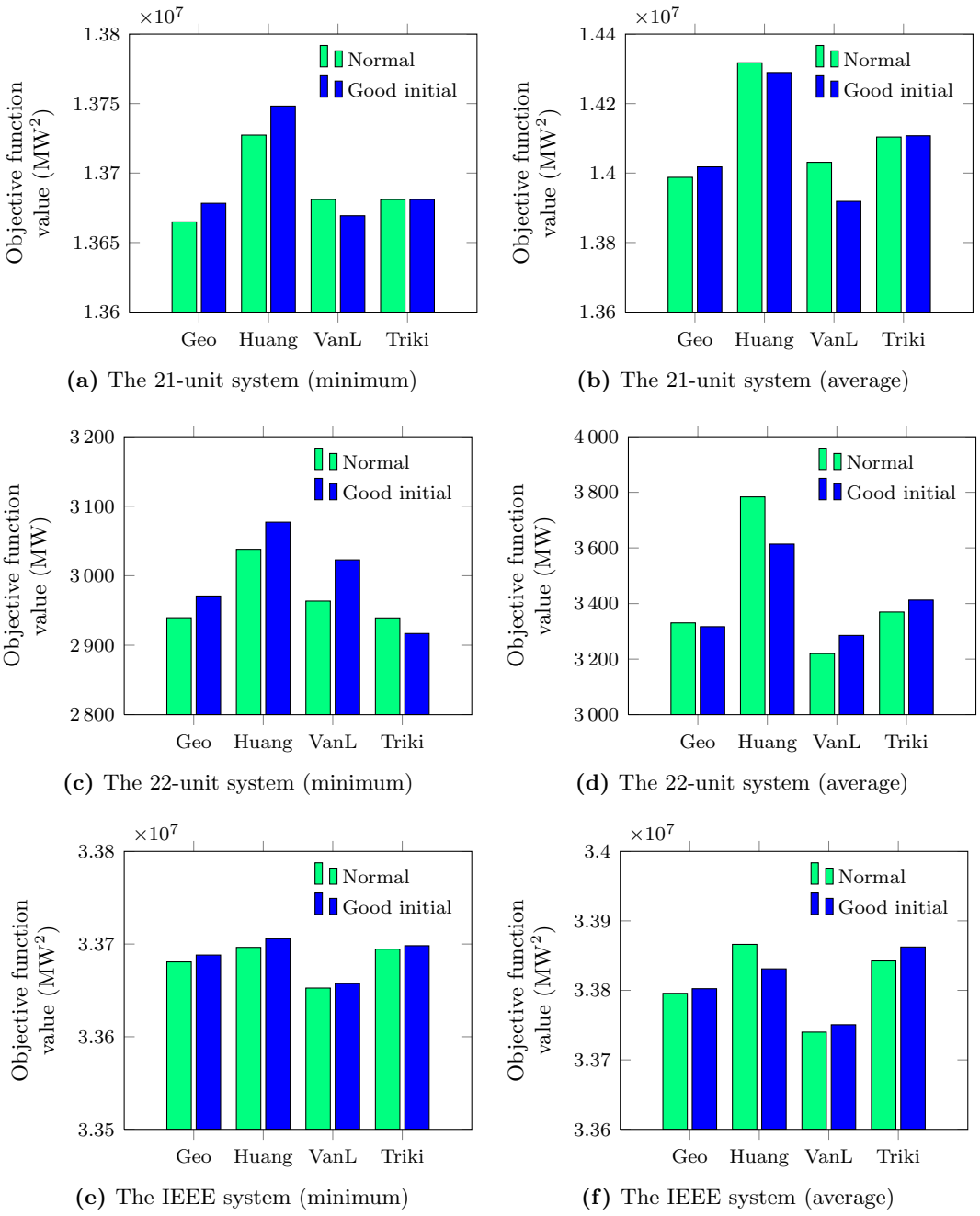
Table 6.4: *Performance analysis of the second algorithmic hybridisation. In the table, "Geo" refers to the geometric cooling schedule, "Huang" refers to the cooling schedule proposed by Huang et al. [37], "VanL" refers to the cooling schedule proposed by Van Laarhoven et al. [81] and "Triki" refers to the cooling schedule proposed by Triki et al. [79].*

In Figure 6.4, the results of using the classical neighbourhood indicate that using good initial solutions has a minimal impact, with only 7 of the 24 cases showing improved results. For the other cases, the solution qualities are very close to each other. However, in Figures 6.4(a) and 6.4(c) the minimum incumbent solution qualities are actually significantly worsened by the good initial solutions.

The results, when using the ejection chain neighbourhood, are presented in the graphs of Figure 6.5. Now, the impact of using a good initial solution is much more prevalent. In the 21-unit system, Figure 6.5(b) shows that the good initial solutions significantly worsened the average incumbent solution quality. The same can be said for the IEEE system, where both the minimum and average incumbent solutions' qualities are significantly worsened, as illustrated in Figures 6.5(e) and 6.5(f). However, the 22-unit system consistently yielded contradicting behaviour, but in this case for the better. The good initial solutions resulted in a considerable improvement in incumbent solution quality, making the results comparable to those via the classical neighbourhood, which obtained far superior results to those of the ejection chain neighbourhood for the 22-unit system, as mentioned in §6.2.

The reason for the contrasting behaviour of the good initial solutions in the 22-unit system with ejection chain neighbourhood is unknown, as the worsening behaviour may be attributed to the solution algorithm converging prematurely from the good initial solution. The solution times were almost identical, when measured from the step in the algorithm following the initial solution declaration. Since the solution time required to obtain a good initial solution is not necessarily insignificant, a larger amount of time is actually spent solving the problem without the benefit of improved solution quality.

Since the negative contradicting behaviour of the ejection chain neighbourhood move operator in the 22-unit test system may be negated to a certain extent by the positive contradicting behaviour of having a good initial solution, one may consider using the two methods in conjunction with each other. By doing so, a general solution methodology towards GMS problems only has to include the ejection chain neighbourhood move operator. Unfortunately, due to a shortage of benchmark test systems, it is not possible, within the scope of this thesis, to investigate



(a) The 21-unit system (minimum)

(b) The 21-unit system (average)

(c) The 22-unit system (minimum)

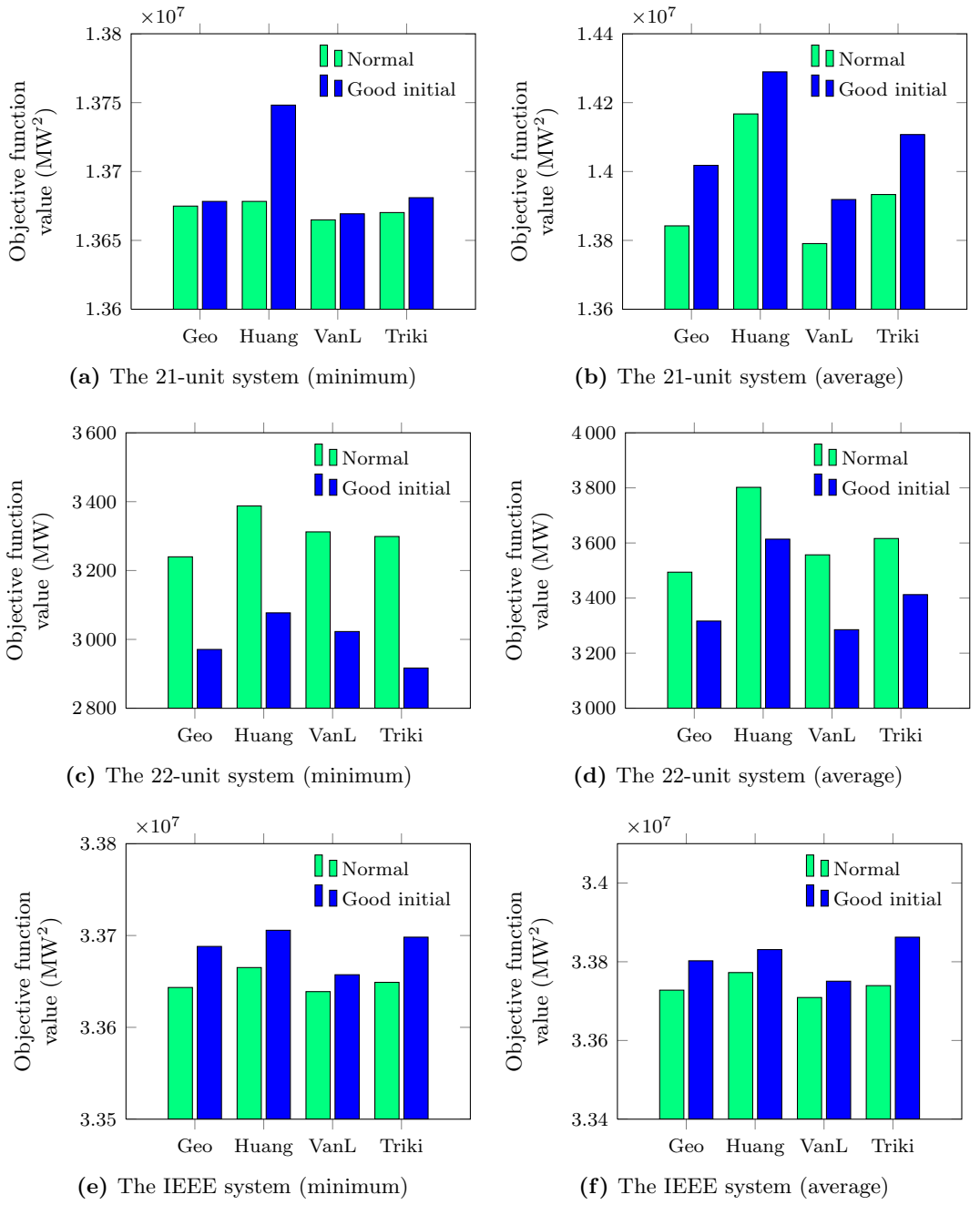
(d) The 22-unit system (average)

(e) The IEEE system (minimum)

(f) The IEEE system (average)

Figure 6.4: *Comparison of random versus good random initial solutions when using the classical neighbourhood. In the graphs, "Geo" refers to the geometric cooling schedule, "Huang" refers to the cooling schedule proposed by Huang et al. [37], "VanL" refers to the cooling schedule proposed by Van Laarhoven et al. [81] and "Triki" refers to the cooling schedule proposed by Triki et al. [79].*

whether this negating behaviour will hold in general for systems that are not highly constrained (of which the 22-unit system is an example). On the other hand, it may be concluded that good initial solutions should not be used with the SA algorithm on highly constrained GMS problems as it typically results in premature convergence and potentially worsening incumbent solutions.



(a) The 21-unit system (minimum)

(b) The 21-unit system (average)

(c) The 22-unit system (minimum)

(d) The 22-unit system (average)

(e) The IEEE system (minimum)

(f) The IEEE system (average)

Figure 6.5: *Comparison of random versus good random initial solutions when using the ejection chain neighbourhood. In the graphs, "Geo" refers to the geometric cooling schedule, "Huang" refers to the cooling schedule proposed by Huang et al. [37], "VanL" refers to the cooling schedule proposed by Van Laarhoven et al. [81] and "Triki" refers to the cooling schedule proposed by Triki et al. [79].*

6.4 Benchmark system solutions

In this section, the best solutions obtained during the course of work towards this thesis are presented for each benchmark test system. The results of the exact solution approach are presented firstly, followed by the results of the approximate solution approach, having obtained the best solutions overall.

6.4.1 The exact solution approach results

As stated in §4.1, the LINGO software package employs different solvers to solve a variety of problems and therefore each benchmark test system was solved separately by employing the appropriate solvers. Furthermore, the computational time restriction adopted for obtaining a solution was set at 12 hours. The results are presented in Table 6.5, which contains the objective function values of the best solutions found via LINGO after 12 hours of solution time. Additionally, the entries in the final column are lower bounds on the objective function values obtained by LINGO. The theoretical bound on the 21-unit system, as calculated in §5.1.1, was improved upon, the 22-unit system now has a lower bound and the theoretical bound on the IEEE system, as calculated in §5.1.3, was also improved upon. Entries in the table marked with an asterisk (*) are locally optimal solutions.

	Solver				
System	Linear	Nonlinear	Global	Quadratic	Lower bound
21-unit (MW^2)	n/a	14 800 440	13 884 370*	13 973 150	11 977 600
22-unit (MW)	3 578.24	6 645.28	4 153.04	n/a	2 007.76
IEEE (MW^2)	n/a	35 175 960	35 462 640*	33 904 230	33 479 440

Table 6.5: *The benchmark test system solutions obtained from an exact solution approach.*

The best exact approach solution obtained for the 21-unit system is 16.67% away from the lower bound. For the 22-unit system, the best solution obtained is 78.22% away from the lower bound, and finally for the IEEE system, the best solution obtained is 1.27% away from the lower bound.

6.4.2 The 21-unit system

The best solution obtained for the 21-unit test system has an objective function value of 13 664 879 MW^2, which is 14.09% away from the lower bound — an improvement from the 16.67% obtained by LINGO. This solution matches the best solution found for the 21-unit system in the literature (13.665×10^6 MW^2), as reported in [29] using ant colony optimisation algorithms to solve the problem.

In a number of instances alternative incumbent solutions were found, also attaining this best objective function value. Having alternative good solutions is always beneficial to decision makers, as it allows greater flexibility should conflicting opinions arise with respect to a given solution. The solution vector of the best solution presented in Figure 6.6 is $[6, 37, 20, 2, 47, 3, 24, 30, 13, 14, 1, 27, 12, 16, 9, 22, 52, 35, 42, 43, 6]$. A list containing all the alternative best solution vectors obtained in this study may be found in Appendix C. Figure 6.6 is a visual representation of the maintenance schedule over the planning year.

In Figure 6.7, the corresponding load demand for the planning year is depicted, along with the available capacity during each time period when the best solution (the schedule in Figure 6.6) is followed. Finally, in Figure 6.8, the corresponding reserve levels during each time period are shown.

6.4.3 The 22-unit system

The best solution obtained for the 22-unit test system has an objective function value of 2 899.08 MW, which is 44.39% away from the lower bound — significantly better than the 78.22%

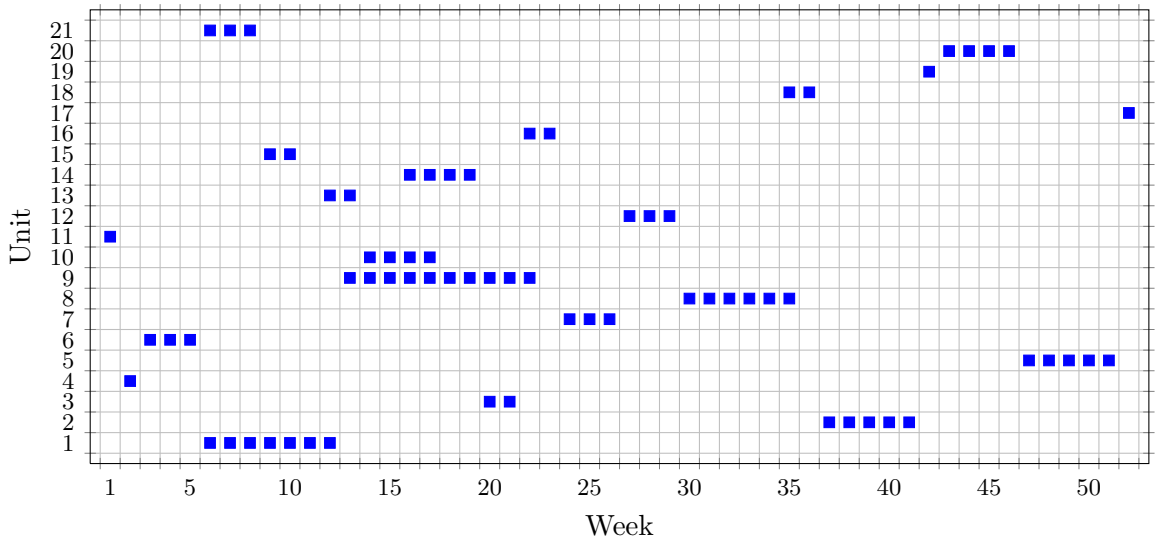


Figure 6.6: *The best maintenance schedule found for the 21-unit test system.*

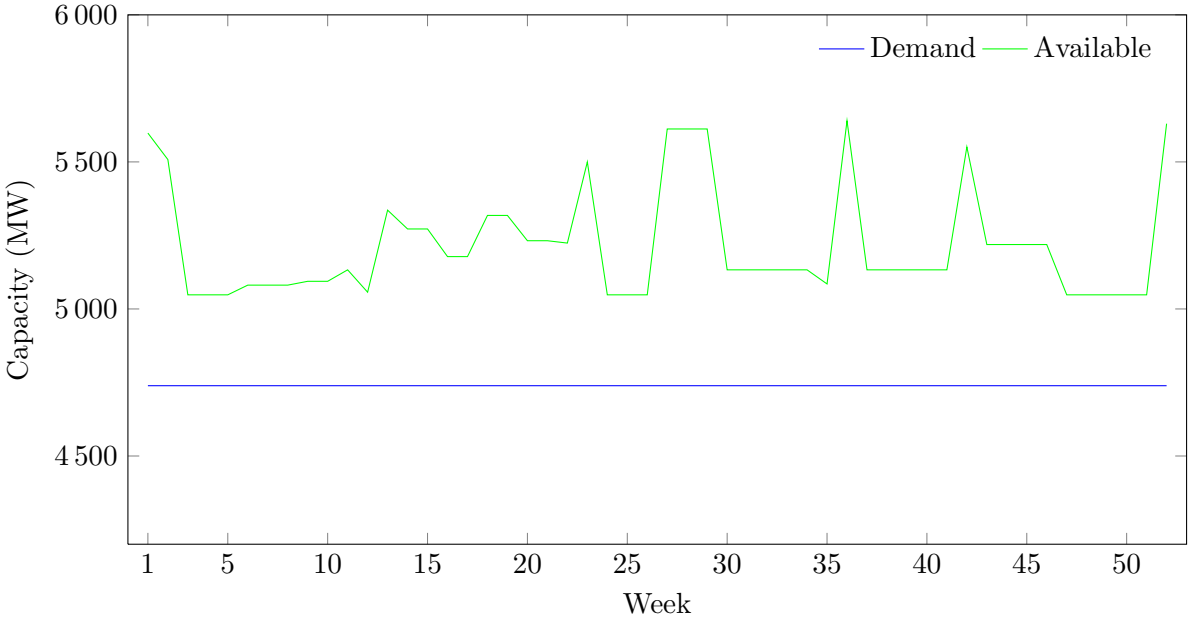


Figure 6.7: *The available capacities for the best solution depicted in Figure 6.6 for the 21-unit test system.*

obtained by LINGO. This best solution vector is [17, 14, 27, 26, 22, 33, 26, 12, 41, 6, 30, 30, 18, 18, 1, 16, 46, 30, 22, 38, 46, 1]. Unfortunately, due to a confliction in the literature [23, 29] and the author's unsuccessful attempts at establishing a correspondence with the these authors, the value of the best solution found in the literature for the 22-unit system is unknown. Unlike the 21-unit system, no alternative best solutions were obtained for this test system. Figure 6.9 is a visual representation of the maintenance schedule corresponding to the above solution vector over the planning year.

The corresponding load demand and safety margin for the year are depicted in Figure 6.10, along with the available capacity during each time period. Finally, the corresponding reserve levels during each time period are illustrated in Figure 6.11.

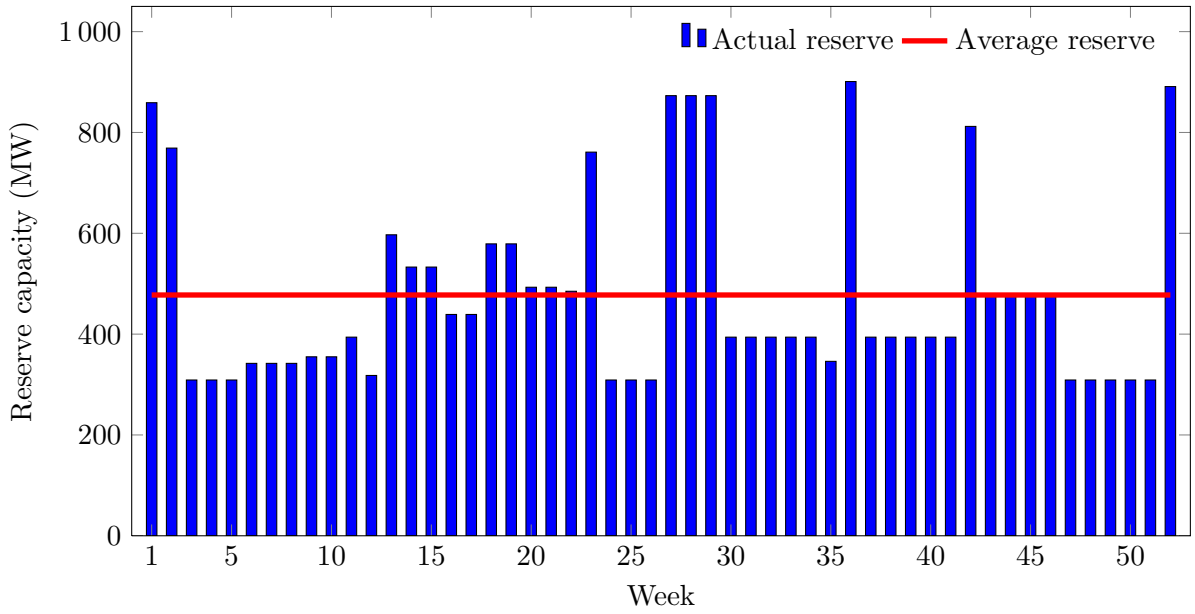


Figure 6.8: *The reserve levels for the best solution depicted in Figure 6.6 for the 21-unit test system.*

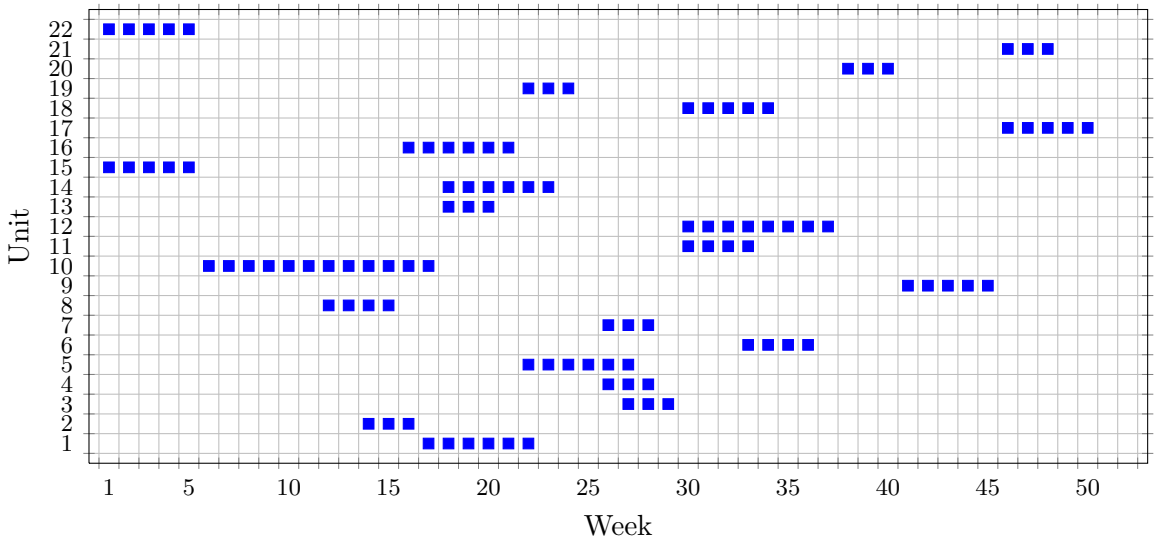


Figure 6.9: *The best maintenance schedule found for the 22-unit test system.*

6.4.4 The IEEE-RTS inspired system

The best solution obtained for the IEEE-RTS inpired test system has an objective function value of 33 627 292 MW^2, which is 0.44% away from the lower bound — an improvement from the 1.27% obtained by LINGO. The corresponding solution vector is [6, 25, 1, 44, 3, 31, 22, 36, 41, 27, 9, 14, 4, 37, 43, 26, 34, 34, 25, 15, 27, 8, 31, 34, 42, 26, 17, 13, 36, 12, 19, 38]. This solution is therefore very close to being optimal. No alternative best solutions were obtained, but two "near-best-found" solutions with objective function values of 33 627 838 MW^2 and 33 628 094 MW^2, respectively, were found. Since the IEEE-RTS inspired test system was newly created for the purposes of this thesis, the best solution above is currently the best solution known for the test system. Figure 6.12 is a visual representation of the corresponding maintenance schedule over the planning year.

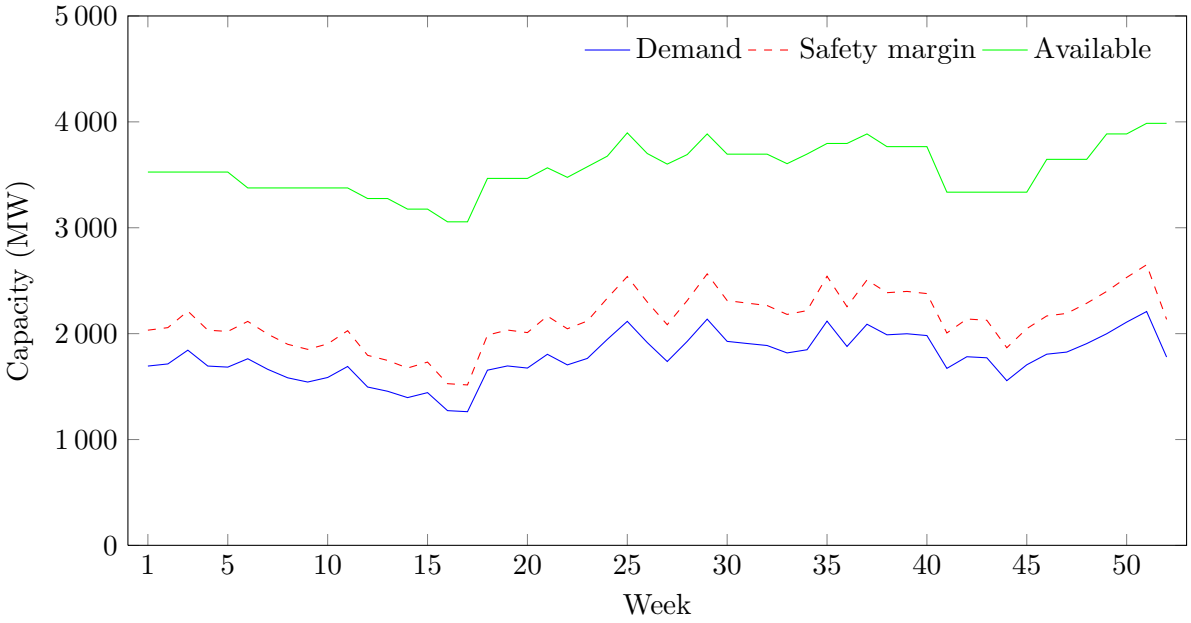


Figure 6.10: *The available capacities for the best solution depicted in Figure 6.9 for the 22-unit test system.*

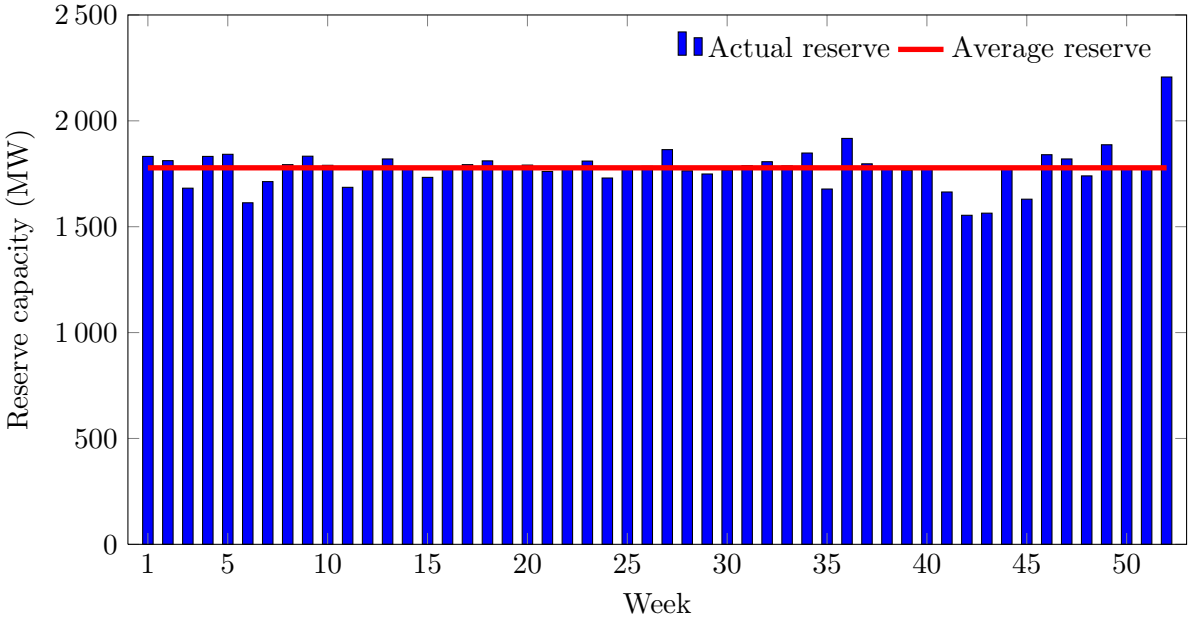


Figure 6.11: *The reserve levels for the best solution depicted in Figure 6.9 for the 22-unit test system.*

The corresponding load demand and safety margin for the planning year are depicted in Figure 6.13, along with the available capacity during each time period. Finally, the corresponding reserve levels during each time period are shown in Figure 6.14.

6.5 Chapter summary

In this chapter, the results of performance analyses into different variations of the proposed approximate solution techniques for solving the GMS problem were presented.

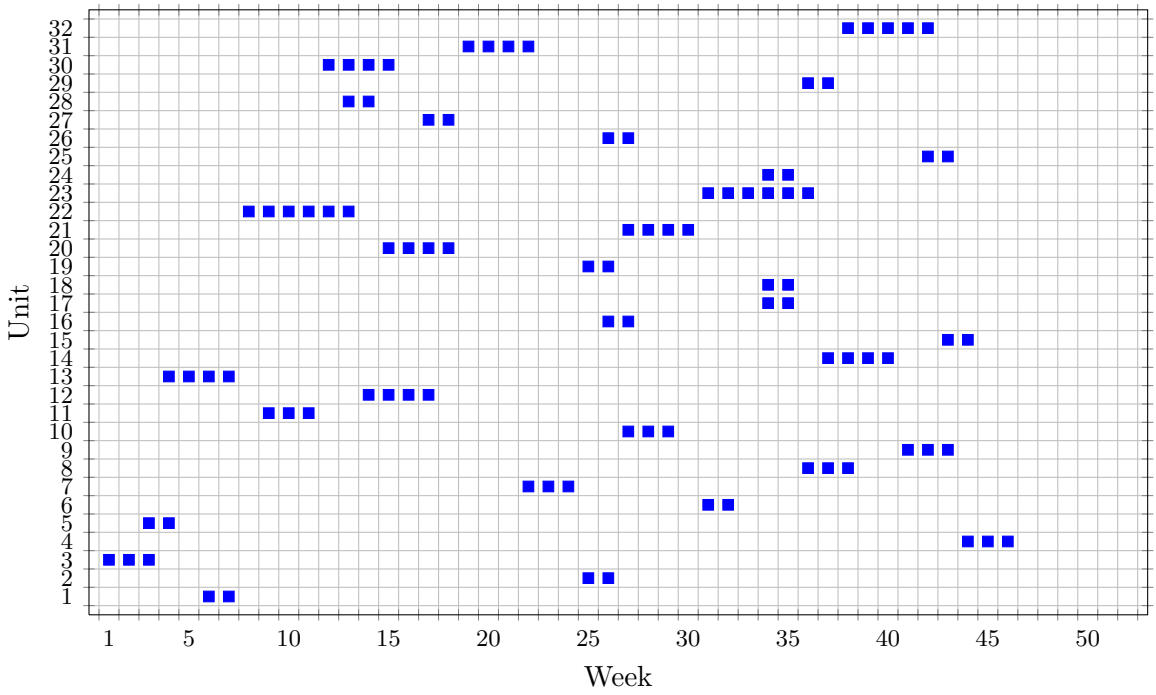


Figure 6.12: *The best maintenance schedule found for the IEEE-RTS inspired test system.*

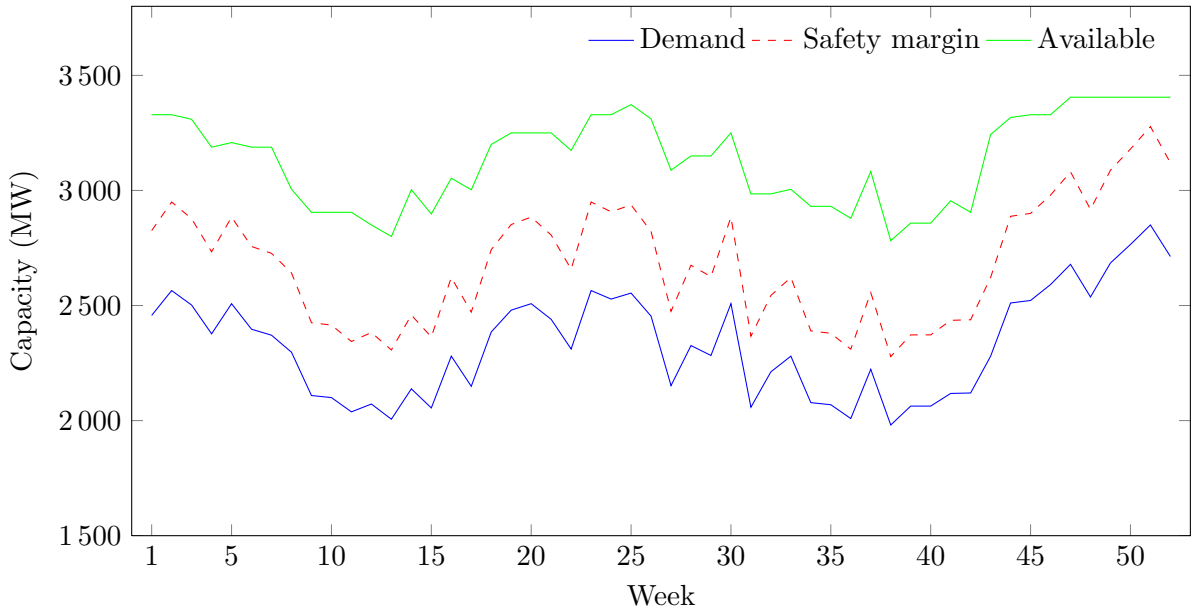


Figure 6.13: *The available capacities for the best solution depicted in Figure 6.12 for the IEEE-RTS inspired test system.*

The results of a performance analysis of the various cooling schedules were presented in §6.1. The section concluded with recommendations on a choice of cooling schedule within the context of GMS.

In §6.2, comparative results obtained via the new ejection chain and the classical neighbourhood move operators were presented, illustrating that the ejection chain operator is highly effective and in some instances significantly superior to the classical operator.

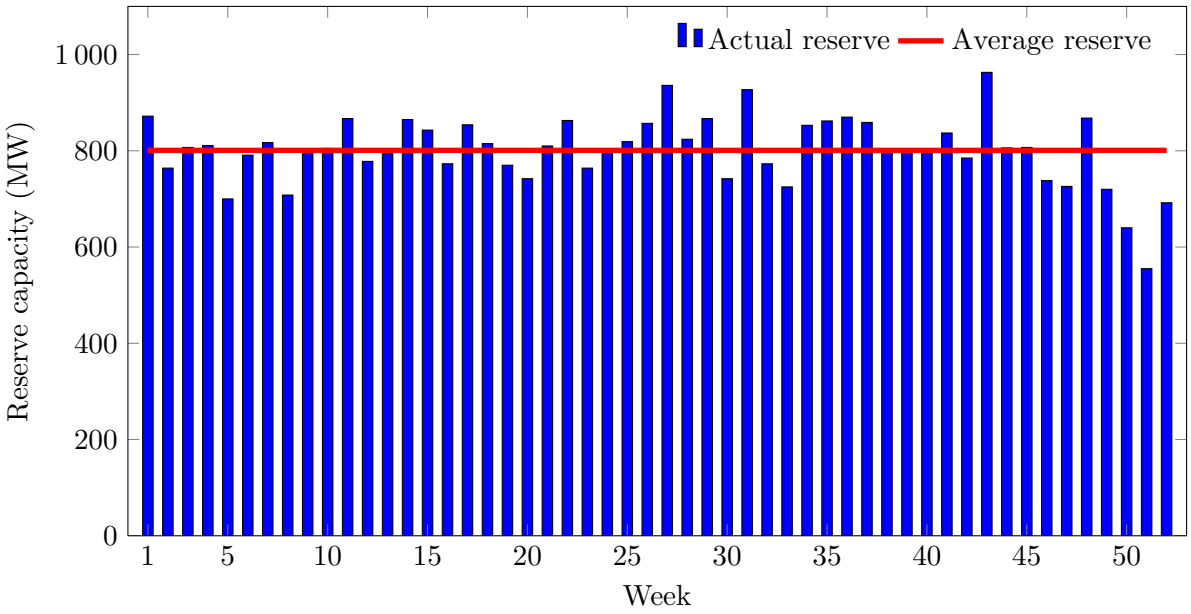


Figure 6.14: *The reserve levels for the best solution depicted in Figure 6.12 for the IEEE-RTS inspired test system.*

The performance analysis of the proposed modifications to the basic simulated annealing algorithm was presented in §6.3. These modifications consisted of the introduction of a local search heuristic and an improved initial solution.

Finally, in §6.4, the results obtained by the exact solution approach for each benchmark test system was presented. This was followed by the presentation of best solutions obtained during the course of the thesis for each GMS benchmark test system.

CHAPTER 7

The decision support system

Contents

In this chapter, a computerised *decision support system* (DSS) is presented which may be used to solve a GMS problem instance, having the generalised form of (A.1) or (A.2), in any power system. The difficulties encountered in designing such a decision support tool within a GMS context are touched upon, as well as how they were partially overcome, in order to create the DSS. This is followed by a description of the DSS implementation on a personal computer and its working.

Finally, a case study of a realistic GMS scenario in the context of the South African national power system, provided by the South African electricity utility Eskom [67], is solved via the DSS and the associated results are presented.

7.1 General framework

The focus in this chapter is turned away from solving GMS benchmark test systems for solution technique validation purposes towards the implemention of a general, computerised GMS solution platform for use in any power system. As indicated in Chapter 5, one of the difficulties inherent to a GMS problem instance is the unknown, unique penalty weight values for its different constraint sets and, unfortunately, a general rule-of-thumb does not exist to determine these values. Furthermore, different GMS problems instances perform differently for certain

parameter values when employing the techniques described in Chapter 4, thereby increasing the difficulty in designing a general GMS solution platform which will perform well across the board.

To that extent, the results obtained in Chapters 5 and 6 are utilised in order to derive a solution methodology and parameter setting which should provide acceptable to good solutions to any GMS problem instance.

7.1.1 The penalty weights

The first problem regarding the set of penalty weights of a GMS problem instance, is the scale of this set. Depending on whether the objective function is chosen from a sum of squares approach (*i.e.* employing the objective function (3.10)) or selecting a sum of absolute differences approach (*i.e.* employing the objective function (3.13)), the penalty weights may differ by orders of magnitude. Consider, as an example, the penalty weights associated with the maintenance window constraint set of the 21-unit and 22-unit systems in §5.2. Due to a difference in the nature of the objective function, the same constraint set has a penalty weight of 500 000 in the one problem instance and 10 in the other problem instance.

Moreover, the scale of penalty weights may differ by orders of magnitude even if the same objective function is used. In such cases, the level of constriction in different GMS problem instances cause the difference in scale. An example of such an occurence may be seen in the penalty weights associated with the maintenance window constraint set of the 21-unit and IEEE-RTS inpired systems in §5.2. With the same objective function in both problem instances, the same constraint set has a penalty value of 500 000 in the one instance and 40 000 in the other instance, due to a difference in problem constriction.

Ultimately, these results indicate that a penalty weight analysis is unavoidable before embarking on an attempt to solve a GMS problem instance. However, performing such an analysis is very time consuming as suitable penalty weight ranges have to be explored in order to find the correct magnitudes and intervals. Such experiments typically consist of repeatedly solving a GMS problem instance and evaluating the solution quality, as described in §5.2. Therefore, the smaller the range interval in the experiments, the more accurate the penalty will be, but the longer the required computional time will be. An acceptable trade-off between penalty accuracy and computational time should therefore be implemented.

The adopted trade-off, implemented in the DSS presented in this chapter, functions as follows. The penalty weight of a constraint set is set as 10^k, with k initially equal to zero. Ten problem instances are solved using an unmodified version of the simulated annealing algorithm with the classical neighbourhood move operator, utilising the geometric cooling schedule with $\alpha = 0.7$. If seven or more of the problem instances obtain feasible incumbent solutions, the penalty weight is chosen at its current value, otherwise the value of k is incremented by one and the process is repeated until a value for the penalty weight is chosen which results in seven or more feasible incumbent solutions, or when $k = 10$. The assumption is that no constraint set in any GMS problem instance will reach a penalty weight of 10^{10}. Due to the exponential increase in weight, the nature of the classical neighbourhood structure and the small α-value in the geometric cooling schedule, this trade-off attempts to keep the computational time for determining the penalty weights as short as possible.

7.1.2 The solution method

In order to select the most appropriate solution method (*i.e.* variation of the simulated annealing algorithm) for a given GMS problem instance, the results of Chapter 6 are considered.

It was concluded in §6.2 that the ejection chain neighbourhood move operator is superior over the classical operator in highly constrained GMS problem instances. Due to the ejection chain operator's contradictory behaviour in the 22-unit system, it was deemed prudent to consider use of the operator in conjunction with the classical operator in order to accommodate systems with similar behaviour. Unfortunately, unless both operators are applied to a problem instance and their solutions compared, one will not be able to tell whether the ejection chain performed in a superior fashion or not. Instead of implementing both operators in the DSS, which adds unnecessary complications for a user, only the ejection chain operator was implemented, based on the assumption that most GMS problems will be highly constrained. However, in order to accommodate systems where the ejection chain operator is not superior, the conclusion in §6.3 with respect to the negating effect of a "good" initial solution on such systems is heeded. An optional choice for the user on whether to use a good initial solution in the SA algorithm or not is implemented in the DSS, thereby applying the local search heuristic to the random initial solution, if chosen, to obtain a good initial solution.

Two cooling schedules were implemented in the DSS in order to add functionality to user. These two schedules are represented in the DSS by a choice between a "quick" or "standard" solution method. In §6.1, it was concluded that the cooling schedule proposed by Huang *et al.* [37] may be used to obtain a relatively good solution quickly. The reason was due to its average computational time being far less than those associated with the other schedules, while still providing good incumbent solutions in comparison. If one considers the schedule's optimised parameter values in Table 5.8, a good choice of generic values may be the AIM for calculating the initial temperature, $\lambda = 0.6$ and $max\_attempt = 100n$.

However, the preferred cooling schedule in terms of solution quality was found to be either the geometric schedule or the schedule proposed by Van Laarhoven *et al.* [81]. As the schedule proposed by Huang *et al.* [37] provides an option for fast results, the schedule of Van Laarhoven *et al.* [81] with its superior solution quality was chosen above the geometric schedule with its slightly shorter solution time. Considering again the optimised parameter values in Table 5.8, a good choice of generic values may be the AIM for calculating the initial temperature and $max\_attempt = 90n$. However, a good generic choice of value for the parameter δ, when using the ejection chain neighbourhood move operator, is more difficult. The value of δ is directly linked to the solution time required by the SA algorithm. Unfortunately, a value resulting in acceptable solution times for a problem instance having only maintenance window and load demand constraint sets (for example), may cause unacceptably long solution times in a problem instance employing additional maintenance crew and/or exclusions constraints. Therefore, the optimised values for each test system in Table 5.8 are the only guidelines to go by. As such, those values are assumed to be representative for each respective problem type. In the DSS implementation, a value of $\delta = 0.15$ was chosen if maintenance window, load demand and safety margin constraints are present. The value is kept the same if maintenance crew constraints are added, while $\delta = 0.25$ if exclusion constraints are added. Finally, $\delta = 0.35$ if both of these constraint sets are added.

In §6.3, it was concluded that the introduction of a local search heuristic into the SA algorithm improved the quality of incumbent solutions. Therefore, since the first hybridisation is considered to be the superior of the two hybridisations, it was implemented in the DSS.

7.2 The implementation of the decision support system

In this section, the appearance and functionality of the DSS implementation are elucidated. The correct procedures to follow at each step towards solving a GMS problem instance via the DSS are described, along with the errors that may occur when using the DSS.

The DSS implementation is a collection of MATLAB scripts[1] with a corresponding *graphical user interface* (GUI). Furthermore, the input and output files are Microsoft Excel workbooks. As such, the DSS requires an installation of MATLAB and Microsoft Excel 2007 (or later) on a personal computer in order to function.

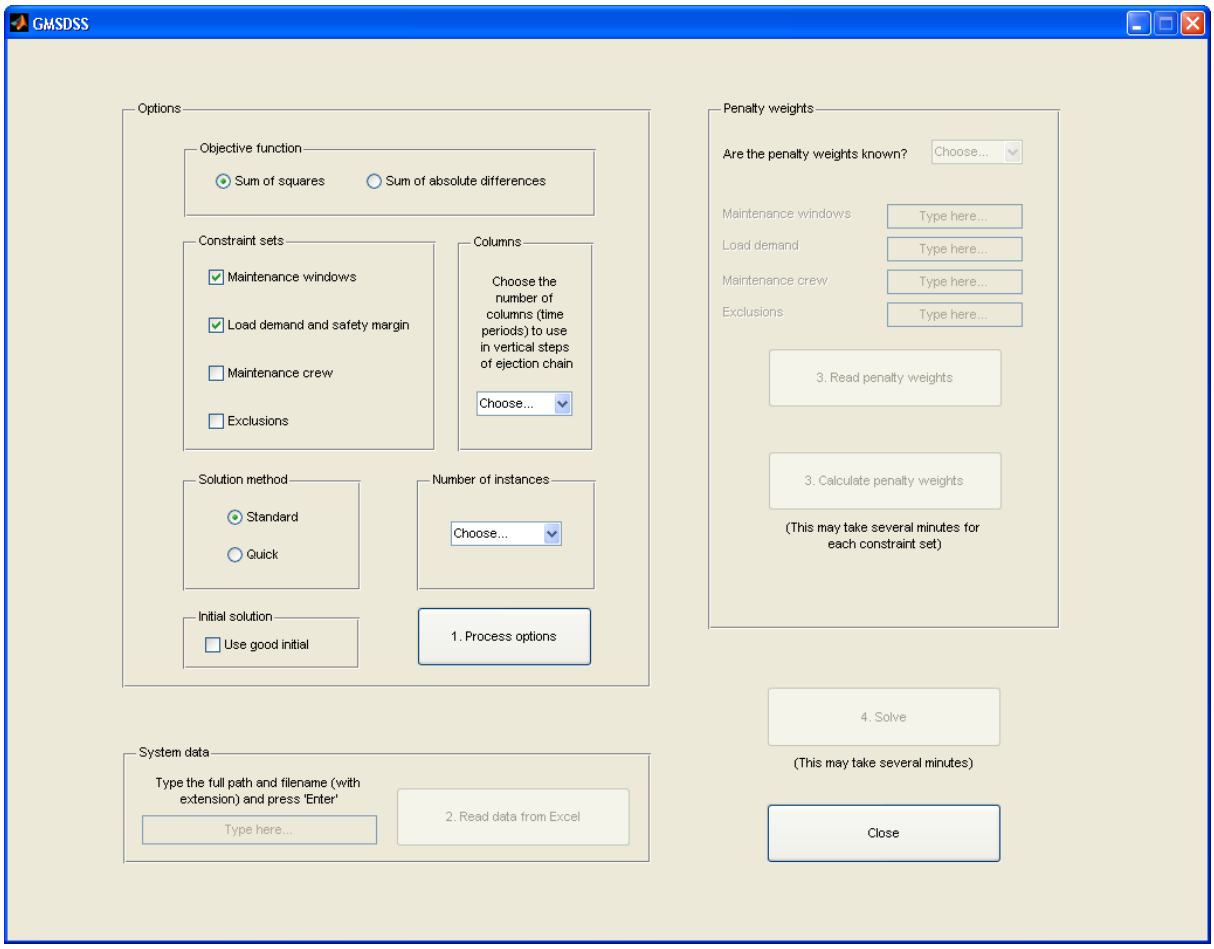


Figure 7.1: *Screenshot of the graphical user interface of the DSS upon opening.*

Upon executing the DSS, the GUI, illustrated by the screenshot in Figure 7.1, appears on the computer screen. The GUI contains three panels (each with specific options) that have to be addressed sequentially by the user in order to finally solve the GMS problem instance. Upon conclusion of each panel's option selection, the following panel is activated, culminating in the activation of the "4. Solve" button. The relevant procedures to follow within the GUI's respective panels are described in the sections that follow.

[1] A script file is an external file that contains a sequence of MATLAB statements.

7.2.1 The "Options" panel

The first panel in the DSS GUI contains characteristics of the GMS problem instance which are required for the data importing and solution method. Initially, it is the only active panel.

- The "Objective function" sub-panel contains a choice (implemented as radio buttons) between the two objective functions that may be used. The default selection is the sum of squares of the reserve levels objective function.
- The "Constraint sets" sub-panel contains a check-box listing of the possible GMS constraint sets. Depending on the problem to be considered, the user ticks the relevant boxes. This selection directly affects the importing of the GMS problem instance data. The default setting ticks the maintenance windows, and load demand and safety margin constraint sets, which is the minimum requirement for any GMS problem instance. If the user unticks one or both of these two boxes, an error message will be displayed upon left-clicking the "1. Process options" button.
- The "Columns" sub-panel contains a drop-down menu from which the user selects the number of columns to be used in the vertical steps of the ejection chain calculation in the solution algorithm. This option is included, based on the concern with respect to non-trivial ejection chains expressed in §4.2.2. It is recommended that the user only selects "2" or "3" columns for problem instances where the number of time periods is much larger than the number of generating units. An error message will be displayed upon left-clicking the "1. Process options" button if no selection is made.
- The "Solution method" sub-panel contains a choice (implemented as radio buttons) between the two solution methods that may be used. These two methods, standard and quick, utilise the cooling schedules proposed by Van Laarhoven *et al.* [81] and Huang *et al.* [37], repectively, as described above. The default selection is the standard method.
- The "Number of instances" sub-panel contains a drop-down menu from which the user selects the number of problem replications that should be solved. The options are 1, 5, 10, 20, 30, 40 and 50. Each replication starts with a different random initial solution in the solution algorithm and the best incumbent solution over all instances is returned by the DSS. Furthermore, if more than one replication is selected, the three best incumbent solutions are returned. An error message will be displayed upon left-clicking the "1. Process options" button if no selection is made.
- The "Initial solution" sub-panel contains a check-box for selecting whether a good initial solution should be used in the solution algorithm or not, as discussed above. It has a default setting of being unticked.
- After all the options described above have been addressed, the user left-clicks the "1. Process options" button. If all the procedures in the "Options" panel have been followed correctly, the relevant variables are initialised according to the selected options, the text on the button changes to "Process options... Done" and the next panel, "System data", is activated.

7.2.2 The "System data" panel

In the second panel of the DSS GUI, the user may read the power system data of the GMS problem instance into the DSS. The data is required to be in a very specific format, contained in

a Microsoft Excel workbook. This format is demonstrated in Appendix D. Both the old ".xls", and the new ".xlsx" file formats, are accepted by the DSS.

- The user is required to type the name of a file, with its extension (*i.e.* filename.xls *or* filename.xlsx), into the edit box and then to press 'Enter' on the keyboard. The full pathname is required, unless the file is contained in the same working directory on the computer as the DSS. If an incorrect file extension has been entered, an error message will appear, otherwise the "2. Read data from Excel" button is activated.

- When the user left-clicks the "2. Read data from Excel" button, the DSS opens the specified Microsoft Excel workbook and reads the data, as specified by the check-boxes in the "Constraint sets" sub-panel of the "Options" panel, into the DSS. As this data transferral process between Excel and MATLAB may take several seconds, a message box containing a progress bar is displayed. After the data have been read in successfully, the text on the button changes to "Read data from Excel... Done" and the next panel, "Penalty weights", is activated. However, if the specified file does not exist, an error message will appear and the "2. Read data from Excel" button is deactivated.

7.2.3 The "Penalty weights" panel

The third and final panel in the DSS GUI addresses the penalty weights of the GMS problem instance. As the penalty weights need only be calculated once for a specific problem instance, the user is presented with the option of entering suitable penalty weights, if known or guestimated, otherwise these weights have to be calculated. Typically, the calculation process is rather time consuming, taking several minutes for each constraint.

- Initially, the only active item in the panel is a drop-down menu from which the user selects "YES" if the penalty weights of the current GMS problem instance are known or "NO" if they are not and need to be calculated.

- Upon selection of "YES" in the drop-down menu, the edit boxes of the constraint sets corresponding to ticked check-boxes in the "Constraint sets" sub-panel of the "Options" panel are activated, as well as the "3. Read penalty weights" button. The user is required to type the known penalty weight values of the constraint sets into the correct edit boxes. When the user left-clicks on the "3. Read penalty weights" button, the weight values are read into the DSS. If a non-numeric value was entered into any of the edit boxes, an error message will appear, otherwise the text in the button change to "Read penalty weights... Done" and the "4. Solve" button in the GUI is activated.

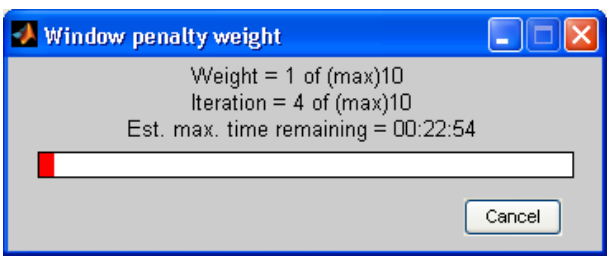


Figure 7.2: *Screenshot of the progress bar displayed during the calculation of the penalty weights.*

- Upon selection of "NO" in the drop-down menu, the "3. Calculate penalty weights" button is activated. When the user left-clicks on the button, the procedure for calculating the penalty weight values of each constraint set, as presented in §7.1.1, is executed. As this procedure is also rather time-consuming, a message box containing a progress bar is displayed, as illustrated by the screenshot in Figure 7.2. A new progress bar is displayed when the following constraint set is reached within the procedure. Additional information is provided in the message box, namely the current weight, current iteration and estimated maximum time remaining. The value displayed as the current weight, corresponds to the current weight iteration (*i.e.* from §7.1.1, the first iteration is when $k = 0$ up to the tenth and final iteration when $k = 9$). The value displayed as the current iteration, corresponds to the current problem replication being solved — ten for each penalty weight value. Lastly, the estimated maximum time remaining provides a worst case amount of time remaining, *i.e.* if the procedure continues up to the tenth weight iteration for the current constraint set. However, this is very unlikely to happen and the procedure typically moves on to the following constraint set when a smaller, suitable weight-value has been reached.

  If the user left-clicks the "Cancel" button in the message box, the procedure terminates at the end of the current iteration, after which an error message is displayed, stating that the calculation of the penalty weight values have not been completed. Otherwise, when the procedure completes the calculation of the penalty weight values, the text on the button in the panel changes to "Calculate penalty weights... Done" and the "4. Solve" button in the GUI is activated.

7.2.4 Solving a problem instance

At this stage, if all the procedures at each panel have been followed correctly, the "4. Solve" button is activated and the GMS problem instance is ready to be solved. When the user left-clicks the button, the DSS executes the solution algorithm presented in §7.1.2 according to the specified settings in the "Options" panel. When the problem has been solved, the DSS generates a number of figures and saves the results to an Excel workbook.

A message box, containing a progress bar, is displayed upon executing the solution algorithm. Figure 7.3 contains a screenshot of such a box. Progress on the number of instances that have been solved, is displayed in the message box. Therefore, the progress bar will only be meaningful

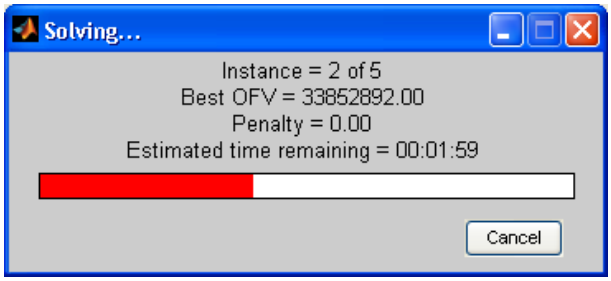
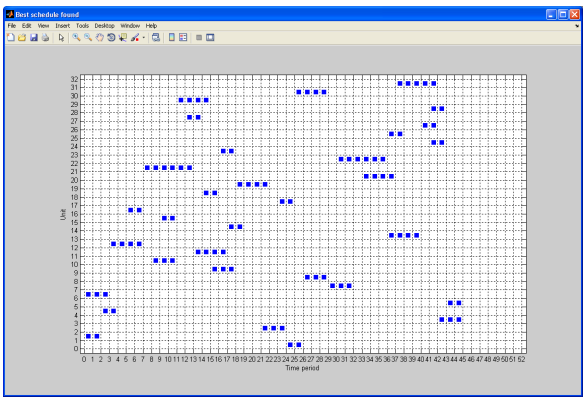


Figure 7.3: *Screenshot of the progress bar displayed during execution of the solution algorithm.*

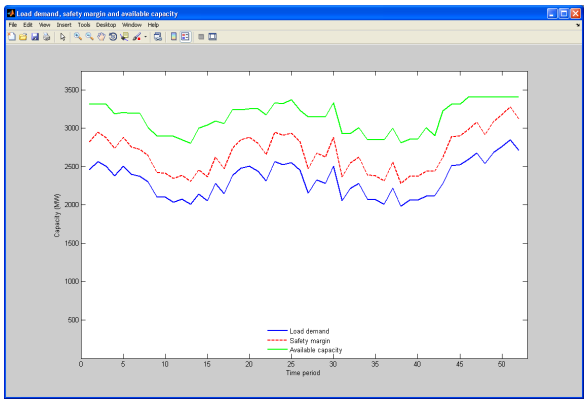
when more than one instance has been selected in the "Options" panel. The additional information contained in the message box consists of the current instance replication being solved, the best incumbent objective function value obtained up to the current stage, its corresponding penalty value and the estimated solution time remaining. If the user left-clicks the "Cancel"

button in the message box, the solution algorithm terminates at the end of the current instance replication and user-control is returned to the GUI.

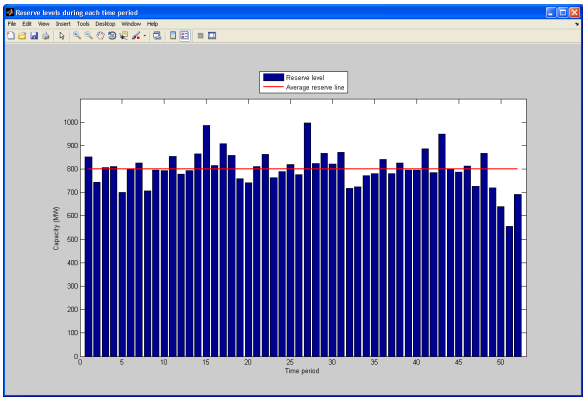
Upon completion of the solution algorithm over the specified number of instances, the DSS generates a number of MATLAB figures containing results from the best solution found. Examples of these figures are presented in Figure 7.4 in the order in which they are generated. These figures may be manipulated within the figure-environment of MATLAB and saved for future use by the user, either as MATLAB ".fig" files or exported as graphics files. The first



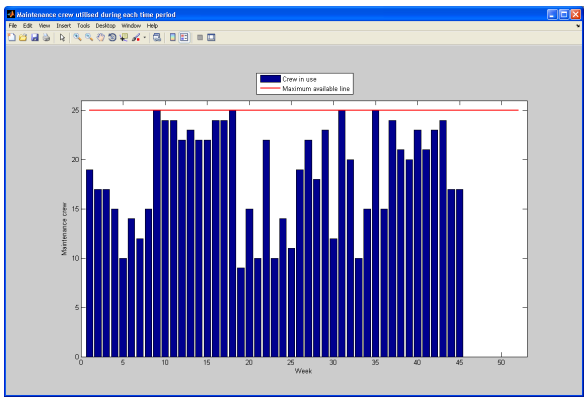
(a) Best schedule found



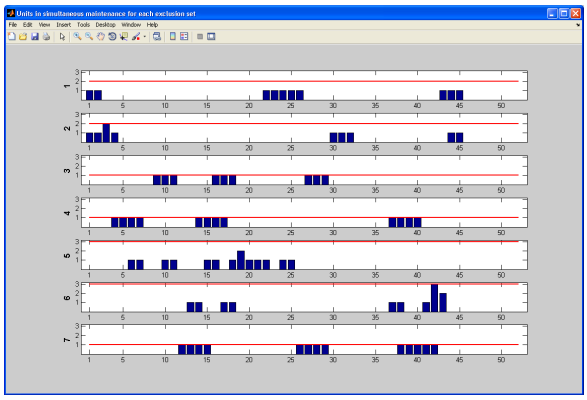
(b) Demand and available capacity levels



(c) Reserve levels



(d) Maintenance crew utilised



(e) Number of units in simultaneous maintenance

Figure 7.4: *Examples of the output figures generated by the DSS.*

figure, illustrated in Figure 7.4(a) is a visual depiction of the best schedule found. The second figure, illustrated in Figure 7.4(b) is a graph of the load demand, safety margin and available generating capacity under the best schedule. In the third figure, the reserve levels during each time period under the best schedule is graphed, as illustrated in Figure 7.4(c). These three generated figures correspond directly to the three figures presented in §6.4 for each benchmark test system. The additional two figures (fourth and fifth) provide visual representations of the maintenance crew and exclusion constraint sets, respectively, under the best schedule. A graph of the amount of utilised manpower during each time period is shown in Figure 7.4(d), while the number of units within each exclusion set that are in a state of simultaneous maintenance is depicted in the graphs of Figure 7.4(e).

As stated in §7.2.1, if more than one instance is selected in the "Options" panel, the DSS returns the best three solutions found. Therefore, when this is the case, two additional figures are generated by the DSS in the same fashion as the graph in Figure 7.4(a), depicting the second and third best schedules found.

After the figures have been generated, the DSS saves the results of the solution to a Microsoft Excel workbook (".xlsx" file format). As the data transferral between MATLAB and Excel may take several seconds, a message box is displayed which, only upon the completion of the transferral, closes automatically. The DSS creates a new workbook for the results and it receives a unique filename of the format *"Solution_yyyymmdd_HHMMSS.xlsx"* which corresponds to the date-time-stamp when the problem was solved. As an example, a file named "Solution_20110825_062641.xlsx" contains the results of a problem solved by the DSS on 2011/08/25 at 06:26:41.

Within the workbook, at least two worksheets are present, irrespective of the selected constraint sets. These worksheets are titled "Schedule" and "Capacities." If the GMS problem instance employs a maintenance crew constraint set, an additional worksheet titled "Crew" is present and if the problem includes an exclusion constraint set, an additional worksheet titled "Exclusions" is present.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1																	
2		Date solved			2011/08/25 06:26												
3																	
4		System data			sampleData.xlsx												
5		Objective function			squares												
6		Solution method			quick		Good initial solution?				No						
7		Number of instances			5		Total solution time (all instances) (sec)				219.8401						
8																	
9		Penalty weights:		Maintenance windows				100000									
10				Load demand and safety margin				1									
11				Maintenance crew				100000									
12				Exclusions				100000									
13																	
14																	
15		Best solution					Second best solution						Third best solution				
16																	
17		Objective function value			33668206		Objective function value			33702530			Objective function value			33745322	
18		Penalty value			0		Penalty value			0			Penalty value			0	
19																	
20		Schedule					Schedule						Schedule				
21		Unit	Start	End			Unit	Start	End				Unit	Start	End		
22		1	25	26			1	24	25				1	4	5		
23		2	1	2			2	23	24				2	5	6		
24		3	22	24			3	8	10				3	22	24		
25		4	43	45			4	30	32				4	29	31		
26		5	3	4			5	24	25				5	24	25		
27		6	44	45			6	43	44				6	38	39		
28		7	1	3			7	20	22				7	7	9		
29		8	30	32			8	27	29				8	30	32		
30		9	27	29			9	43	45				9	35	37		
31		10	16	18			10	37	39				10	16	18		

Figure 7.5: *Screenshot of the "Schedule" worksheet in the DSS results.*

Figure 7.5 contains a screenshot of an example of the "Schedule" worksheet. The top part of the sheet contains the settings of the DSS that were chosen by the user, as well as the penalty values for each constraint set. Additionally, the date and time when the problem was solved are included, along with the total solution time required. The bottom part of the sheet contains the actual schedule (best solution) found by the DSS (and second and third best schedules when more than one replication was solved). The corresponding objective function and penalty values are listed, followed by the commencement and completion time periods of each generating unit's maintenance period over the planning horizon.

	A	B	C	D	E	F	G	H	I	J
1										
2		Time	=	Time period						
3		CapOnM	=	Capacity on maintenance (unavailable)						
4		AvailCap	=	Available capacity (Installed capacity minus CapOnM)						
5		Demand	=	Load demand						
6		Reserve	=	Capacity in reserve						
7										
8										
9		Time	CapOnM	AvailCap	Demand	Reserve				
10		1	96	3309	2457	852				
11		2	96	3309	2565	744				
12		3	96	3309	2502	807				
13		4	217	3188	2377	811				
14		5	197	3208	2508	700				
15		6	209	3196	2397	799				
16		7	209	3196	2371	825				
17		8	400	3005	2297	708				
18		9	500	2905	2109	796				
19		10	512	2893	2100	793				

Figure 7.6: *Screenshot of the "Capacities" worksheet in the DSS results.*

An example of the second worksheet, "Capacities", is provided in Figure 7.6. Shortened column-headers are used, the definitions of which are listed at the top of the sheet. The results consist of the system capacity out on maintenance, the resulting available system capacity, the system load demand and the corresponding reserve capacity levels during each time period, under the best solution found. From these data, the graphs in Figures 7.4(b) and 7.4(c) may be reproduced.

	A	B	C	D	E	F	G	H	I
1									
2		Time	=	Time period					
3		UsedCrw	=	Crew used for maintenance					
4		AvailCrw	=	Available (remaining) crew for maintenance					
5									
6									
7		Time	UsedCrw	AvailCrw					
8		1	19	6					
9		2	17	8					
10		3	17	8					
11		4	15	10					
12		5	10	15					
13		6	14	11					
14		7	12	13					
15		8	15	10					
16		9	25	0					
17		10	24	1					

Figure 7.7: *Screenshot of the "Crew" worksheet in the DSS results.*

The additional "Crew" worksheet is illustrated by means of an example screenshot in Figure 7.7. Again, shortened column-headers are used and their definitions are listed at the top of the sheet. The number of utilised maintenance crews and corresponding maintenance crews still available during each time period under the best solution found, are listed in this sheet. These data may be used to reproduce the graph in Figure 7.4(d).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1																	
2		Time	=	Time period													
3		NumInM	=	Number of units in simultaneous maintenance													
4		NumA	=	Number of units still allowed to be in simultaneous maintenance													
5																	
6																	
7			Set 1		Set 2		Set 3		Set 4		Set 5		Set 6		Set 7		
8		Time	NumInM	NumA	NumInM	NumA	NumInM	NumA	NumInM	NumA	NumInM	NumA	NumInM	NumA	NumInM	NumA	
9		1	1	1	1	1	0	1	0	1	0	3	0	3	0	1	
10		2	1	1	1	1	0	1	0	1	0	3	0	3	0	1	
11		3	0	2	2	0	0	1	0	1	0	3	0	3	0	1	
12		4	0	2	1	1	0	1	1	0	0	3	0	3	0	1	
13		5	0	2	0	2	0	1	1	0	0	3	0	3	0	1	
14		6	0	2	0	2	0	1	1	0	1	2	0	3	0	1	
15		7	0	2	0	2	0	1	1	0	1	2	0	3	0	1	
16		8	0	2	0	2	0	1	0	1	0	3	0	3	0	1	
17		9	0	2	0	2	1	0	0	1	0	3	0	3	0	1	
18		10	0	2	0	2	1	0	0	1	1	2	0	3	0	1	

Figure 7.8: *Screenshot of the "Exclusions" worksheet in the DSS results.*

Finally, the additional worksheet entitled "Exclusions" is illustrated by means of an example screenshot in Figure 7.8. The results contained in this worksheet consist of the number of units that are in simultaneous maintenance within each exclusion subset, and number of units that are additionally allowed to be in simultaneous maintenance within each exclusion subset, during each time period under the best solution found. Once more, the shortened column-headers used in the sheet are defined at the top. The data of each exclusion subset in the sheet may be used to produce each subgraph in Figure 7.4(e).

7.3 A real case study

The DSS was used to solve a real case study, provided by Eskom [67], within the context of the South African national power generation system. Due to confidentiality concerns, the data do not represent an exact description of the current Eskom generation system, but the case study does represent a realistic GMS scenario. Constraints in the problem instance are restricted to the adherence to maintenance windows, the system meeting the load demand together with a safety margin, and respecting exclusion constraints. The system data are presented in Appendix E due to its large dimensions affecting readability.

7.3.1 The nature of the problem instance

The generation system consists of 105 power generating units with a total installed capacity of 39 949 MW. A number of units must undergo multiple maintenance outages over the planning period of 365 days, while some units require no maintenance outages. In order to solve the GMS problem, the DSS requires each unit to have a single maintenance outage — therefore, dummy units were added to the system for each additional maintenance occurence of a unit, increasing the number of units in the system to 157. The resulting additional capacity that these dummy units provide to the system was subtracted from the new total capacity in order to render the system unaffected. Furthermore, units not requiring maintenance over the planning period, receive earliest and latest starting times of zero. Table E.1 contains the generation system specifications and maintenance requirements.

The exclusion constraints of the problem restrict a unit with multiple maintenance outages, whose maintenance windows overlap, such that its different maintenance outages do not occur simultaneously. The exclusion sets are presented in Table E.3.

Table E.4 contains the daily peak load demand of the power system. A minimum safety margin capacity of 2 000 MW was translated into a safety margin of 8% of the peak load demand and has to be maintained throughout the planning period.

Finally, a theoretical lower bound on the objective function value (if the sum of squares of the reserve levels is used as the objective function) has been determined from the average daily reserve level of 5 364 MW. This lower bound is 10 501 819 298 MW^2.

7.3.2 Results achieved

A number of solution settings were chosen in the DSS to solve the case study. As no preference was given to which objective function should be used, both were selected. All four combinations of solution method and good initial solution were attempted over a selection of 1, 10 and 50 replications. Only one column was selected for the vertical steps in the ejection chains.

The best solution obtained by the DSS using the sum of squares objective function, attained an objective function value of 11 101 712 702 MW^2 with zero penalty. This objective function value lies 5.7% away from the theoretical lower bound. Although the DSS obtained this best objective function value twice, the solution vectors were identical.

In the case of the sum of absolute differences objective function, the DSS obtained a best solution with an objective function value of 368 546.58 MW and zero penalty. Again, the DSS obtained this best objective function value twice, but the solution vectors were identical.

A summary of the DSS results on the Eskom data set is provided in Table 7.1. Instead of listing the objective function values (the largely scaled values affect readibility), the percentage by which each objective function value lies away from that of the best solution found is listed. Considering that the average solution time (Van Laarhoven *et al.* [81] solution method) for the IEEE-RTS inpired test system (dimension of 32 units by 52 weeks) was two and a half minutes, the DSS performs quite well timewise in solving the Eskom system, since its dimension is almost 35 times larger, but is solved in a time only 23 times greater.

Solution method	Instances	Sum of squares			Sum of absolute differences		
		Time (minutes)	Percentage from best	Feasible	Time (minutes)	Percentage from best	Feasible
Quick	1	8.49	0.07%	No	6.18	1.80%	No
	10	105.06	0.04%	Yes	80.34	0.86%	No
	50	517.54	0.02%	Yes	396.65	0.04%	Yes
Quick	1	8.58	0.01%	Yes	5.50	1.04%	Yes
(Good Initial)	10	100.57	0.03%	Yes	80.87	0.45%	No
	50	516.37	0.02%	Yes	395.96	0.04%	Yes
Standard	1	56.40	0.10%	No	43.52	0.76%	No
	10	551.98	0.01%	Yes	411.62	0.18%	No
	50	2756.82	0%	Yes	2047.87	0%	Yes
Standard	1	56.66	0.10%	No	40.56	0.26%	Yes
(Good Initial)	10	424.88	0.01%	Yes	407.73	0.18%	No
	50	2762.33	0%	Yes	2028.06	0%	Yes

Table 7.1: *Results obtained by the DSS on the Eskom case study.*

The solution vectors of the best solutions obtained from using both objective function values are listed next to each other in Table 7.2, in order to display the difference caused by the objective

function. In Figures 7.9 and 7.10, visual representations of the corresponding maintenance schedules are given. Finally, Figure 7.11 contains graphs of the load demand and the available generating capacities for the two best solutions.

7.4 Chapter summary

A computerised decision support system for solving generator maintenance scheduling problems was presented in this chapter. The DSS was implemented in the software package, MATLAB, as a graphical user interface along with a collection of scripts, while the input and output files containing the system data and results were created in Microsoft Excel.

In §7.1, a generic penalty weight procedure, parameter setting and solution methodology for solving a general GMS problem were derived from the problem-specific GMS instances considered in Chapters 5 and 6. These generic methods were required before an attempt could be made to solve a general GMS problem instance satisfactorily, without performing problem-specific solution technique optimisation.

The computer implementation of the DSS and its working were described in §7.2 by means of screenshots and bulleted procedures to follow during each step of the solution process. Examples of the output produced by the DSS were also presented while the correct input format for the DSS, was described in Appendix D.

Finally, the DSS was used in §7.3 to solve a case study containing a realistic GMS scenario within the context of the South African national power generation system. This system was considerably larger than the test systems considered in Chapters 5 and 6. The results obtained are very promising, as the DSS computed a best solution with an objective function value only 5.7% away from the theoretical lower bound associated with the problem instance, when using the sum of squares objective function. As a lower bound for the sum of absolute differences objective function was unknown, the quality of the best solution obtained by the DSS could not be determined in a similar fashion. Furthermore, the solution time of the larger system increased at a lower rate than the increase in problem size, indicating the effectiveness of the solution methods implemented.

Unit	Squares	Absolute differences	Unit	Squares	Absolute differences	Unit	Squares	Absolute differences	Unit	Squares	Absolute differences	Unit	Squares	Absolute differences
1	25	27	33	0	0	65	111	79	97	0	0	129	66	59
2	19	15	34	0	0	66	211	229	98	85	55	130	153	151
3	173	171	35	1	1	67	0	0	99	0	0	131	237	237
4	285	292	36	122	121	68	344	320	100	112	110	132	54	60
5	25	15	37	54	19	69	0	0	101	29	29	133	123	141
6	27	25	38	139	145	70	0	0	102	99	99	134	180	186
7	173	180	39	286	286	71	0	0	103	338	338	135	299	299
8	285	292	40	326	334	72	13	22	104	23	23	136	78	64
9	25	15	41	19	47	73	0	0	105	257	257	137	68	61
10	32	30	42	335	335	74	98	120	106	6	7	138	187	183
11	180	184	43	32	33	75	258	204	107	122	122	139	306	306
12	297	295	44	293	293	76	0	0	108	270	264	140	66	57
13	0	0	45	13	11	77	122	122	109	120	120	141	187	194
14	0	0	46	60	53	78	8	8	110	92	92	142	306	312
15	0	0	47	328	328	79	287	270	111	330	346	143	87	95
16	0	0	48	0	0	80	287	289	112	358	356	144	306	276
17	46	47	49	111	111	81	250	255	113	1	1	145	115	101
18	234	235	50	239	239	82	338	319	114	323	308	146	354	353
19	53	53	51	0	0	83	341	341	115	76	85	147	73	73
20	256	262	52	1	1	84	22	22	116	331	331	148	241	241
21	47	46	53	263	263	85	320	327	117	71	76	149	20	2
22	284	284	54	95	94	86	333	325	118	126	124	150	262	256
23	75	82	55	246	246	87	13	12	119	190	193	151	81	81
24	312	305	56	25	26	88	86	86	120	230	231	152	123	123
25	22	22	57	244	246	89	236	282	121	0	0	153	325	333
26	269	260	58	59	66	90	0	0	122	0	0	154	22	51
27	110	116	59	297	296	91	15	16	123	1	1	155	151	155
28	277	277	60	5	4	92	297	298	124	106	106	156	275	269
29	0	0	61	126	125	93	1	9	125	268	240	157	95	93
30	0	0	62	0	0	94	345	345	126	24	24			
31	61	71	63	0	0	95	0	0	127	132	139			
32	259	234	64	0	0	96	0	0	128	251	244			

Table 7.2: *The best solutions obtained by the DSS for the Eskom case study using both objective functions. "Squares" indicates the solution corresponding to the sum of squares objective function, while "Absolute differences" indicates the solution corresponding to the sum of absolute differences objective function.*

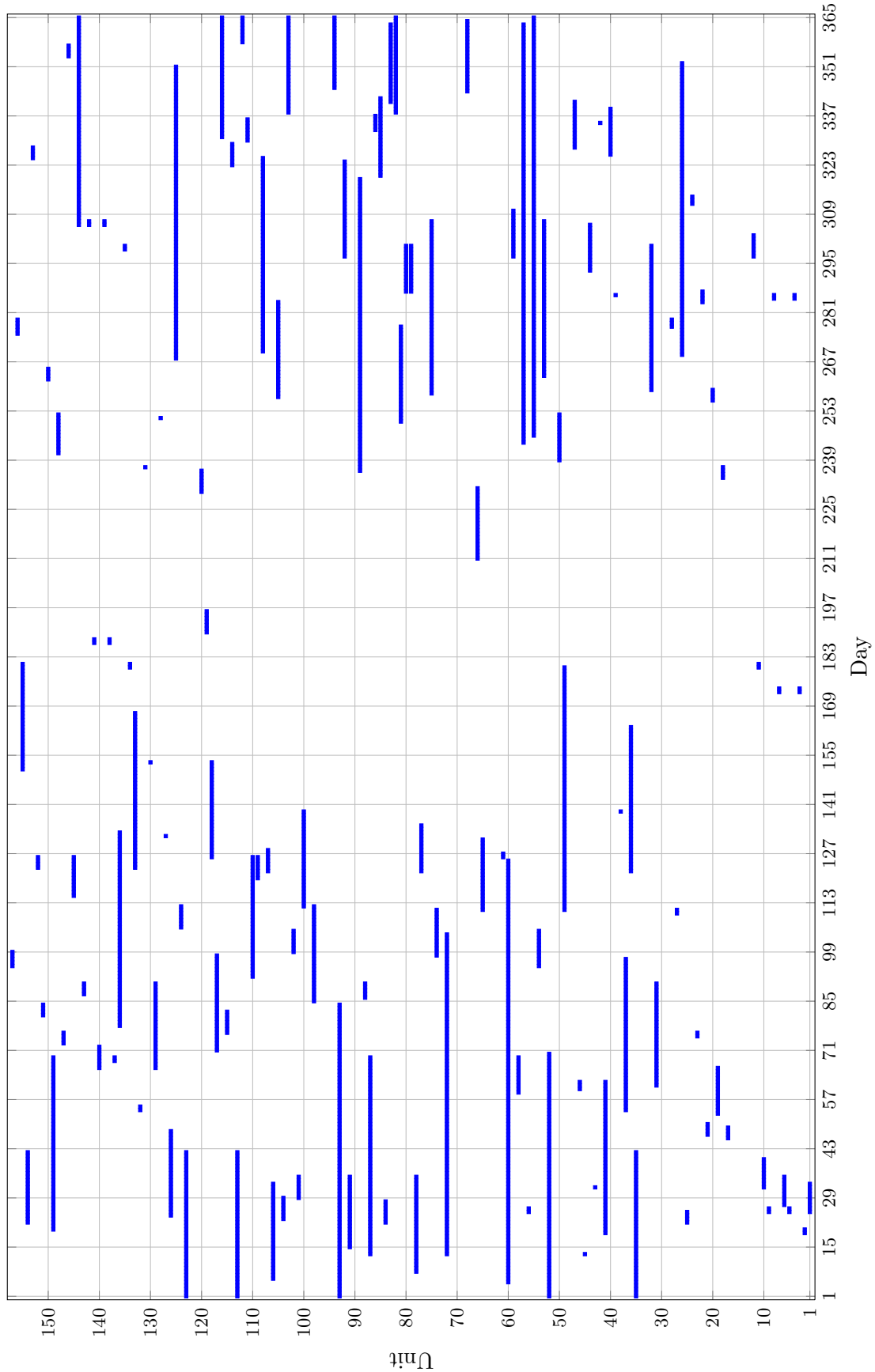


Figure 7.9: *The best maintenance schedule found using the sum of squares objective function for the Eskom case study.*

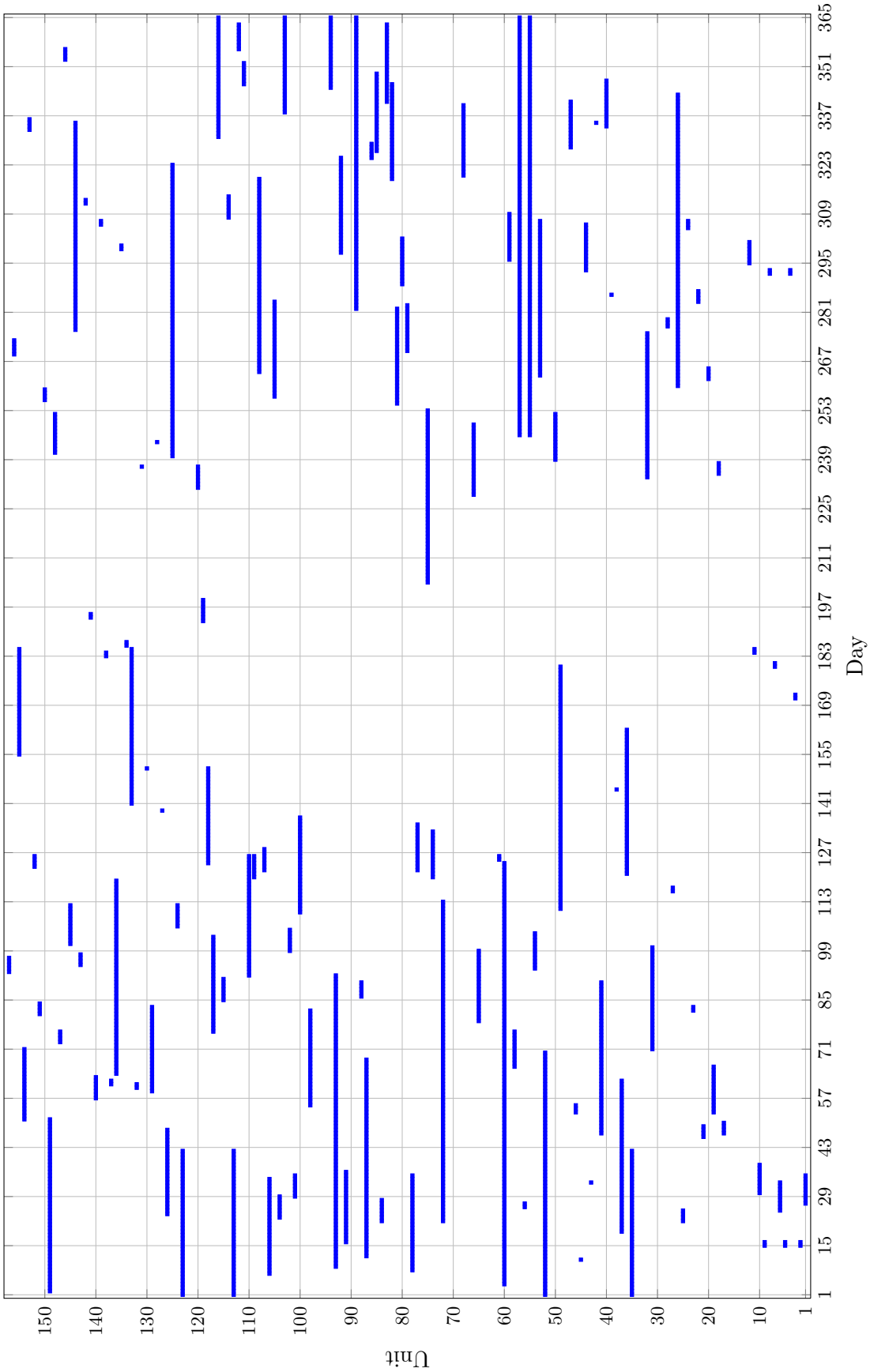
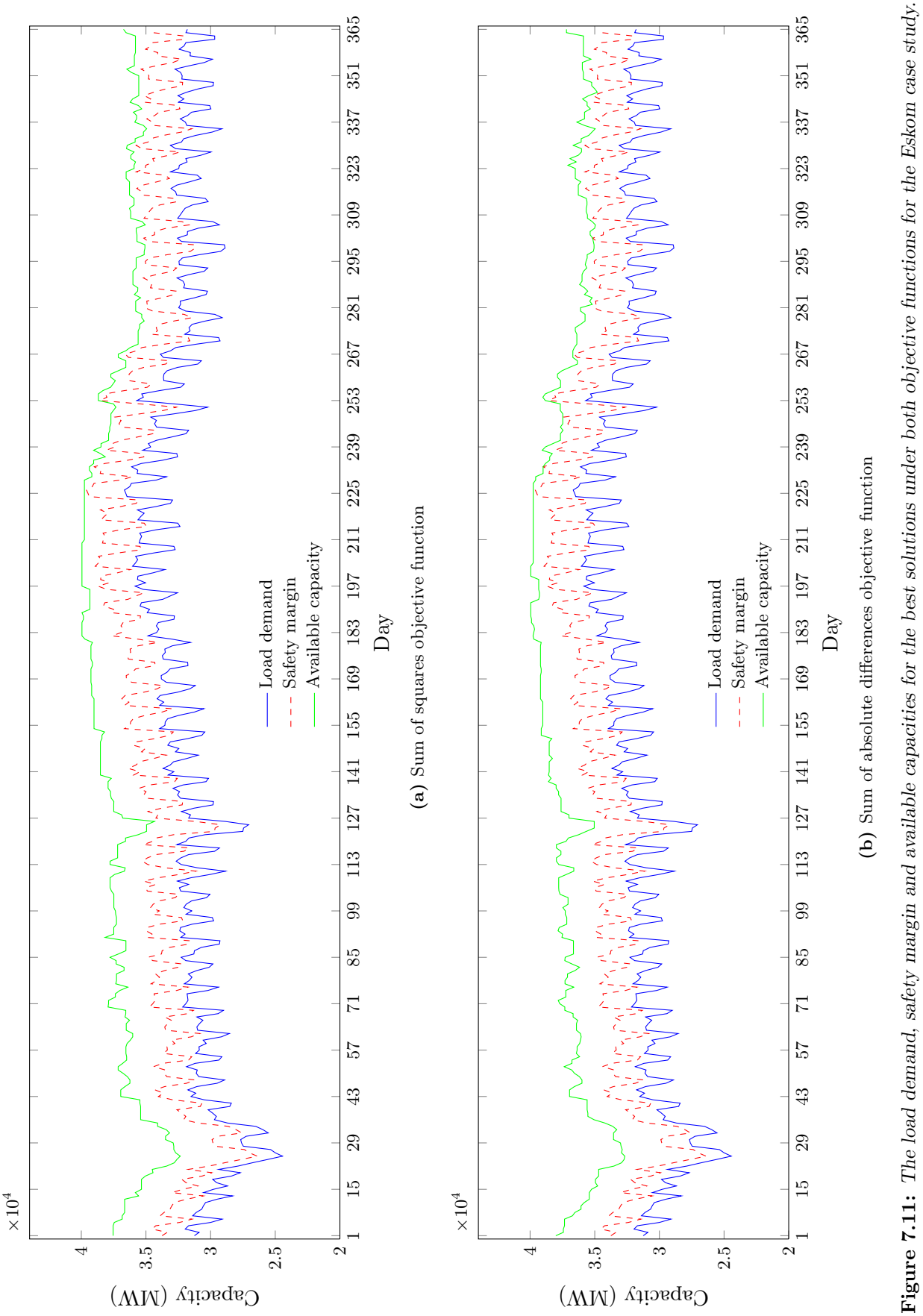


Figure 7.10: *The best maintenance schedule found using the sum of absolute differences objective function for the Eskom case study.*



(a) Sum of squares objective function

(b) Sum of absolute differences objective function

Figure 7.11: *The load demand, safety margin and available capacities for the best solutions under both objective functions for the Eskom case study.*

CHAPTER 8

Conclusion

Contents

This chapter concludes the thesis with a summary of the work contained therein, a list of the contributions of the thesis and finally, proposals for future work to further this study.

8.1 Thesis summary

In the introductory chapter to this thesis, a brief history of the generation and supply of electricity was presented, including the so-called *War of Currents* between Thomas Edison's DC power supply and Nikola Tesla's AC power supply. An informal problem description of the scheduling problem considered in this thesis, called the *generator maintenance scheduling* (GMS) problem in the literature, was presented in the second section of the chapter. This was followed by the objectives that were persued during the work towards this thesis.

A comprehensive literature review on the GMS problem was presented in Chapter 2. The chapter contained a discussion on the general modelling considerations that have to be addressed before attempting to formulate a GMS model (in fulfilment of Thesis Objective I, as stated in §1.3), followed by previously considered mathematical programming formulations of the various constraint sets and objective functions typically associated with the GMS problem (in partial fulfilment of Thesis Objective II). Formulations other than that of a mathematical programming nature were briefly described, as well as extensions to the GMS problem which may be considered (in final fulfilment of Thesis Objective II). The various solution techniques that had previously been applied to the GMS problem were presented (in fulfilment of Thesis Objective III) which included both exact and approximate solution approaches.

In Chapter 3, the GMS problem was placed in its proper context within the broader scope of power system operations scheduling. The necessary problem assumptions were discussed and motivated in order to develop a suitable GMS model within the scope of this thesis (in partial fulfilment of Thesis Objective IV). Two models were derived, based on the constraint sets of the

GMS problem — a simple model which included only the essential constraint sets required for a general GMS problem formulation, as well as a more advanced model which included additional constraint sets. For each model, three mixed-integer mathematical programming formulations were presented (in final fulfilment of Thesis Objective IV) which differed from one another in their objective functions. Having the same goal, the objective functions were quadratic, nonlinear and linear in nature.

The solution methodology adopted in this thesis was presented in Chapter 4. An exact solution approach utilising a commercial off-the-shelf software package, was one of the adopted solution methods and, hence the various algorithms employed by this software package were described. The other solution method was approximate in nature and consisted of a random search heuristic (functioning as a baseline method) and a simulated annealing algorithm (in partial fulfilment of Thesis Objective V). Within the approximate approach towards solving the GMS problem, a soft constraint approach was adopted, which meant that constraints were allowed to be violated, subject to some corresponding penalty incursion. The algorithmic implementation designs of the two approximate solution techniques were presented in the chapter, which included a description on a so-called ejection chain neighbourhood move operator, newly proposed in the context of GMS problems (in partial fulfilment of Thesis Objective VI). Additionally, two modifications to the simulated annealing algorithm were proposed (in partial fulfilment of Thesis Objective V) which revolved around the introduction of a local search heuristic applied to incumbent solutions uncovered by the simulated annealing algorithm.

Three GMS benchmark test systems were presented in Chapter 5. Two of these systems had previously been studied in the literature, while the third system was newly created. As a result of the soft constraint approach adopted in the approximate solution approach, described in Chapter 4, penalty weight values had to be obtained for the constraint sets in each of the benchmark test systems. The methodology for determining these penalty weights was presented and the subsequent values of the penalty weights were determined for each test system. The approximate solution techniques also contain several parameters which have to be fixed at appropriate values in order for the techniques to perform optimally. However, these values are typically problem instance-dependent. As such, the results of a comprehensive parameter optimisation process was also presented for each test system.

Some of the main results of this thesis were presented in Chapter 6, where the solution methods presented in Chapter 4 were applied to the benchmark test systems of Chapter 5 in order to investigate the effectiveness of the variations in the approximate solution techniques, and to solve the GMS benchmark test systems approximately. A performance analysis of the new ejection chain neighbourhood move operator was also presented (in partial fulfilment of Thesis Objective VI) and it was concluded that the operator was effective and yielded results that were mostly superior to those obtained by using the classical neighbourhood move operator that had previously been utilised in the literature. In addition, a performance analysis was also performed with respect to the use of different cooling schedules within the simulated annealing algorithm and in conjunction the proposed modifications to the simulated annealing algorithm (in final fulfilment of Thesis Objective VI). A presentation of the best solutions found during work towards this thesis for each benchmark test system concluded the chapter (in final fulfilment of Thesis Objective V), indicating that the approximate solution approach was superior to the exact solution approach.

In Chapter 7, a computerised decision support system for solving GMS problem instances in any power system conforming to the models in Chapter 3, was introduced. The results in Chapters 5 and 6 were used to derive the penalty weight calculation procedure and the GMS solution

methodology adopted in the DSS (in fulfilment of Thesis Objective VII). The appearance and functionality of the DSS (as implemented on a personal computer) were described by means of screenshots and bulleted procedural lists containing the steps to be followed by a user during the solution process (in partial fulfilment of Thesis Objective VIII). Finally, the solutions obtained by the DSS, when applied to a realistic case study provided by Eskom within the context of the South African national power generation system, were presented (in final fulfilment of Thesis Objective VIII) along with a brief discussion and interpretation of the DSS results. The DSS computed a best solution with an objective function value only 5.7% away from the theoretical lower bound associated with the problem instance, when using the sum of squares objective function.

8.2 Thesis contributions

The main contributions of this thesis are outlined in this section.

Contribution 1: *A modular constraint and objective function overview of GMS problem formulations in the literature in §2.2.1 and §2.2.2.*

A difficulty experienced by the author during his research towards this thesis, was that almost all of the GMS models in the literature share some portion of formulation, but ultimately differ from one another because the nature of the specific power systems under consideration, and because the priorities of the electricity utilities differ. Therefore, different constraint sets under different objective functions in a number of combinations are scattered throughout the literature, making it a tedious and difficult task to investigate GMS models from one's own perspective. The literature review in this thesis is modular with respect to constraint sets and objective functions instead of presenting entire GMS models as is typically done in the different literature. Future researchers may benefit from this easy access to different formulations of a specific constraint set or objective function.

Contribution 2: *A mathematical programming formulation for load constraints in §3.3.1 and crew constraints in §3.4.1 which requires binary starting decision variables and do not require any sets to be defined.*

The author could not find the constraint set formulations (3.6) or (3.26) in GMS literature and therefore these are assumed to be newly proposed in this thesis. The first advantage of these formulations lies in the fact that the decision variables $x_{i,j}$ are the only variables required. As a result, the auxiliary variables $y_{i,j}$ need not be introduced into a model formulation which would otherwise have doubled in number of variables. As illustrated §2.2.1, other formulations using only the decision variables $x_{i,j}$ do exist for these constraint sets. However, (2.13) and (2.17) require the definition of three groups of sets (the total number of sets being $n + nm + m$) in order to achieve the same result as (3.6) and (3.26). Herein lies the second advantage of the formulations in this thesis — the parameters $g'_{p,i,j}$ and $m'_{p,i,j}$ replace the need for numerous sets to be defined.

Contribution 3: *A hybridised simulated annealing algorithm utilising a new neighbourhood move operator in the context of generator maintenance scheduling in §4.2.*

The standard simulated annealing algorithm, previously used in the literature for solving a GMS problem, was improved upon by the utilisation of an ejection chain neighbourhood move operator and a hybridisation with a local search heuristic, utilising the classical neighbourhood move operator. In the performance analysis in §6.2, it was concluded that the ejection chain

neighbourhood move operator performed superior to the classical neighbourhood move operator in highly constrained systems. The hybridisation consistently improved upon most incumbent solutions obtained by the standard SA algorithm, as presented in §6.3.

Contribution 4: *A new GMS benchmark test system in §5.1.3.*

The literature on generator maintenance scheduling contains very few benchmark test systems, probably due to the fact that the majority of research is conducted with respect to specific (different) power system problem instances. A new benchmark test system was introduced in this thesis with constraints for maintenance windows, meeting of load demand, adherence to a safety margin, the availability of maintenance crew (or general resource) and general exclusion constraints. This test system may be used in future GMS research.

Contribution 5: *An investigation into the effectiveness of different cooling schedules in a simulated annealing algorithm within the context of solving GMS problems in §6.1.*

The author could find no reference in the GMS literature of simulated annealing algorithms utilising any cooling schedule other than the geometric schedule, or any indication that experimentation with various schedules had been performed, finding that the geometric schedule was the superior schedule. Therefore, this investigation into four different cooling schedules within a GMS context was the first of its kind and it revealed very useful results, as concluded in §6.1.

Contribution 6: *A computerised GMS decision support tool in Chapter 7.*

The main contribution of this thesis is a computerised decision support system capable of solving GMS problem instances for any power system conforming to the general model formulation (A.1) or (A.2). A power system operations scheduler tasked with GMS may utilise this DSS to assist him/her by suggesting good maintenance schedules.

8.3 Future work

A number of suggestions for possible future work with respect to the GMS problem, which emerged during the compilation of this thesis, are presented in this section. Two categories of suggestions arose: the first category is concerned with the modelling and formulation of the GMS problem itself, while the second category is concerned with the solution techniques for the GMS problem.

8.3.1 Suggestions on modelling and formulating the GMS problem

The GMS problem encompasses such a large array of dynamics within a power system that its modelling considerations are rather extensive. A number of suggestions towards future possibilities to extend and/or improve upon the GMS model and formulation adopted in this thesis are presented below.

Suggestion 1: *Consider incorporating unplanned maintenance allowances.*

One of the reasons why a power system retains a specific reserve safety margin throughout the year, is to ensure it can accommodate unforeseen reductions in power supply. Extraneous events may cause a generating unit to break down and require reactive maintenance. Such maintenance is referred to as *unplanned maintenance* and it is an unavoidable part of the functioning of any power system. As a result, electricity utilities typically employ predictive models so as to ensure a certain level of preparedness. The results of these predictive models may

be incorporated into the GMS model to enhance the power system's robustness with respect to such events. Incorporating an unplanned maintenance allowance may take the form of a stochastic variable modelling different properties for different generating units, which will result in dimished capacity.

Suggestion 2: *Consider incorporating the effect that deferred maintenance has on the performance of a power generating unit.*

The efficiency and reliability of a generating unit is directly related to its regular preventative maintenance. As units grow older or run at very high output levels for a long period of time, these preventative maintenance occurrences become vital to the effective functioning of the unit. Therefore, should the planned maintenance of a unit be deferred for some reason, a negative effect is incurred on that unit. This negative effect is twofold: firstly, the power generating effiency of the unit may deminish to a level below its expected contributing capacity, and secondly, the risk of an unforeseen break-down increases and the predictive models relating to unplanned maintenance for the unit are rendered deficient. In order to incorporate these deferred maintenance effects into the GMS model, models which provide the correlation between the actual service time beyond maintenance date and power output efficiency, as well as between the actual service time beyond maintenance date and the failure rate of a unit have to be developed. These models may then be included in the GMS model.

Suggestion 3: *Consider a frequency-based maintenance outage formulation.*

Current formulations of GMS problems in the literature explicitly specify a time window during which a unit is required to undergo maintenance. However, these window specifications are calculated from each generating unit's maintenance frequency (typically specified by the manufacturer), when it completed its previous maintenance outage and some subjective range of acceptable time periods that may pass before the unit is expressly required to undergo maintenance. The subjective element in this process may be circumvented if the frequency of maintenance is modelled instead of specified windows. In such a model, the formulation would require a minimum number of time periods to pass between maintenance outages of a unit; the maximum number of time periods to pass between maintenance outages is addressed by the frequency of maintenance itself. Solving the problem would then result in a schedule where the maintenance outages are automatically spread out correctly, not being subjected to the possibility of ill-defined and hampering windows.

Suggestion 4: *Consider a multiobjective modelling approach.*

During the initial stages of research conducted towards this thesis, interviews with an Eskom employee [61] revealed their need for the assurance of reliability in the South African power system during the planned maintenance outages of generating units. Economic considerations were deemed to be subservient to their reliability concerns. It was also evident from correspondence with this employee that it would be very challenging to obtain the necessary economic data required for an economic optimality criterion. Therefore, the decision was made to employ a single objective reliability-orientated modelling approach to the GMS problem, since the final decision support system was to be beneficial from a South African viewpoint. Much later (very near the completion date of this thesis), however, correspondence with another Eskom employee [67] revealed that the minimisation of production costs should also be a priority. At this stage there was not sufficient time remaining to incorporate changes to the general modelling approach. However, a multiobjective modelling approach, considering both reliability and production costs as objectives, would be of great interest and functionality as a possible future enhancement to the GMS model adopted in this thesis. Success in this respect has also been

achieved in the literature in terms of modelling multiobjective goals by means of fuzzy sets within a GMS context.

Suggestion 5: *Consider deriving an integrated model containing components of the power system operations scheduling problem other than just the GMS subproblem.*

In the literature the GMS problem has typically been considered as a segregated problem within the context of power system operations scheduling, mainly because it lies much higher in the temporal hierarchy of operations scheduling than do UC or ED problems. However, an integrated model would represent the more realistic modelling approach as there is unquestionably feedback between the various problems contained in operations scheduling. As computing power increases over time, the drawback of increased problem dimensions when considering an integrated approach is expected to become less of a constraining factor. Furthermore, since transmission constraints have been included in recent GMS models, a limited form of the UC problem already had to be considered in such models. One may as well include the entire scope of the UC problem in those cases. An integrated model would provide interesting results, as it would increase the accuracy of the GMS and transmission maintenance solutions, as well as provide a good baseline solution for the UC problem. Unfortunately, the UC problem would need additional attention as the level of detail provided by an integrated model would not necessarily be good enough for the UC problem. The ED problem requires an even greater level of detail, specifically in the temporal domain. Therefore, the computing power would have to be extremely powerful in order to incorporate ED into the model as well.

8.3.2 Suggestions regarding the solution techniques of the GMS problem

The realm of combinatorial solution techniques is vast and there surely exist methods which may obtain better solutions than the method adopted in this thesis. Two suggestions for possible future study in terms of solution methodology for the GMS problem are presented in this section.

Suggestion 6: *Consider the use of Monte Carlo simulation for modelling uncertainty in capacity and/or demand.*

A concern in any GMS model is the uncertainty present in much of the data, such as uncertain demand and variable generating capacity levels. Deterministic formulations can only attempt to accommodate these uncertainties by including as many parameter variations into the formulation as possible (*e.g.* instead of using a generating capacity g_i for unit i, one may consider using a generating capacity $g_{i,j}$ for unit i during time period j). Both instances are deterministic, but the latter contains the option of different values for the same unit during different times over the planning period. The appeal of deterministic formulations lies in the fact that they are easier to formulate and to solve, since deterministic solution methods may be utilised.

In order to preserve the advantage of the deterministic formulation and solution technique, but still introduce a deeper level of uncertainty into the problem, one may consider using Monte Carlo simulation. The generating capacities and/or load demand of the power system may be replaced by randomly drawn values (according to some probability distribution to be determined) and may be solved deterministically, thereby creating one step in a Monte Carlo simulation. Such a Monte Carlo method approach will lend more realism to the model while retaining the advantages of a deterministic formulation and solution method. However, the drawback in such an approach is the potential that it may become very time consuming. This depends on the problem dimensions and the efficiency of the solution algorithm, since many problem instances need to be solved for a Monte Carlo simulation in order to provide adequate results.

Suggestion 7: *Consider the use of hyperheuristic solution methods.*

It has been well established in the literature that metaheuristic methods perform very well in solving instances of the GMS problem. The simulated annealing algorithm used in this thesis, further demonstrates this observation. Furthermore, this good level of performance is not limited to one or two methods, but also apply to genetic algorithms, tabu searches, and ant colony optimisation. A number of hybridisations have also performed very successfully. A natural next step would be to consider a heuristic approach at a higher level than than that of metaheuristics in order to solve the GMS problem. According to [94], a hyperheuristic is a heuristic search method that seeks to automate the process of selecting, combining, generating or adapting several simpler heuristic methods (or components of heuristic methods) so as to efficiently solve computational search problems.

Considering the effectiveness of the various metaheuristic methods in solving the GMS problem, these methods may be employed as the "simpler heuristics" contained in a hyperheuristic. The fact that GMS models vary in almost every new study from those previously considered, reinforces the suggestion of adopting a hyperheuristic solution framework for the GMS problem, since the aim of hyperheuristics is to search for a "generally applicable methodology [94]" which is able to accommodate classes of problems (*e.g.* the GMS class of problems) rather than being capable of solving just one problem instance. Such a methodology, in all likelihood, would perform very well in a GMS decision support system.

Bibliography

[1] Ahmad A & Kothari DP, 1998, *A review of recent advances in generator maintenance scheduling*, Electric Power Components and Systems, **26(4)**, pp. 373–387.

[2] Ahmad A & Kothari DP, 2000, *A practical model for generator maintenance scheduling with transmission constraints*, Electric Power Components and Systems, **28(6)**, pp. 501–513.

[3] Alardhi M & Labib AW, 2008, *Preventive maintenance scheduling of multi-cogeneration plants using integer programming*, Journal of the Operational Research Society, **59(4)**, pp. 503–509.

[4] Albrecht PF, Bhavaraju MP, Biggerstaff BE, Billington R, Jorgensen GE, Reppen ND & Shortley PB, 1979, *IEEE Reliability test system*, IEEE Transactions on Power Apparatus and Systems, **PAS-98(6)**, pp. 2047–2054.

[5] Allan RN, Billinton R & Abdel-Gawad NMK, 1986, *The IEEE Reliability test system — Extensions to and evaluation of the generating system*, IEEE Transactions on Power Systems, **PWRS-1(4)**, pp. 1–7.

[6] Arueti S & Okrent D, 1990, *A knowledge-based prototype for optimization of preventive maintenance scheduling*, Reliability Engineering and System Safety, **30**, pp. 93–114.

[7] Bergesen C, 2011, *The world's largest power plants*, [Online], [Cited August 30th, 2011], Available from `http://www.industcards.com/top-100-pt-1.htm`.

[8] Black PE, 2009, *Dictionary of algorithms and data structures*, [Online], [Cited September 10th, 2010], Available from `http://www.itl.nist.gov/div897/sqg/dads/HTML/metaheuristic.html`.

[9] Burke EK, Clark JA & Smith AJ, 1998, *Four methods for maintenance scheduling*, Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, Springer, New York (NY), pp. 264–269.

[10] Burke EK & Smith AJ, 2000, *Hybrid evolutionary techniques for the maintenance scheduling problem*, IEEE Transactions on Power Systems, **15(1)**, pp. 122–128.

[11] Canto SP, 2008, *Application of Benders' decomposition to power plant preventive maintenance scheduling*, European Journal of Operational Research, **184**, pp. 759–777.

[12] china guide — about beautiful country, 2011, *The Three Gorges Dam*, [Online], [Cited August 30th, 2011], Available from `http://www.china-consulates.com/gallery/the-three-gorges-dam-nsGI8.html`.

[13] Contaxis GC, Kavatza SD & Vournas CD, 1989, *An interactive package for risk evaluation and maintenance scheduling*, IEEE Transactions on Power Systems, **4(2)**, pp. 389–395.

[14] Dahal KP, Aldridge CJ & McDonald JR, 1999, *Generator maintenance scheduling using a genetic algorithm with a fuzzy evaluation function*, Fuzzy Sets and Systems, **102**, pp. 21–29.

[15] Dahal KP & Chakpitak N, 2007, *Generator maintenance scheduling in power systems using metaheuristic-based hybrid approaches*, Electric Power Systems Research, **77**, pp. 771–779.

[16] Dahal KP & McDonald JR, 1997, *Generational and steady state genetic algorithms for generator maintenance scheduling problems*, Paper presented at the International Conference on Artificial Neural Networks and Genetic Algorithms, Norwich.

[17] Dahal KP & McDonald JR, 1997, *A review of generator maintenance scheduling using artificial intelligence techniques*, Paper presented at the 32nd Universities Power Engineering Conference (UPEC '97), University of Manchester, Manchester.

[18] Dahal KP, McDonald JR & Burt GM, 2000, *Modern heuristic techniques for scheduling generator maintenance in power systems*, Transactions of the Institute of Measurement and Control, **22(2)**, pp. 179–194.

[19] Digalakis JG & Margaritis KG, 2002, *A multipopulation cultural algorithm for the electrical generator scheduling problem*, Mathematics and Computers in Simulation, **60**, pp. 293–301.

[20] Dréo J, Pétrowski A, Siarry P & Taillard E, 2006, *Metaheuristics for hard optimization — Methods and case studies*, Springer-Verlag, Berlin.

[21] Edwin KW & Curtius F, 1990, *New maintenance-scheduling method with production cost minimization via integer linear programming*, Electrical Power & Energy Systems, **12(3)**, pp. 165–170.

[22] Eglese RW, 1990, *Simulated annealing: A tool for operational research*, European Journal of Operational Research, **46**, pp. 271–281.

[23] El-Amin I, Duffuaa S & Abbas M, 2000, *A tabu search algorithm for maintenance scheduling of generating units*, Electric Power Systems Research, **54**, pp. 91–99.

[24] El-Sharkh MY & El-Keib AA, 2003, *An evolutionary programming-based solution methodology for power generation and transmission maintenance scheduling*, Electric Power Systems Research, **65**, pp. 35–40.

[25] El-Sharkh MY, El-Keib AA & Chen H, 2003, *A fuzzy evolutionary programming-based solution methodology for security-constrained generation maintenance scheduling*, Electric Power Systems Research, **67**, pp. 67–72.

[26] Escudero LF, Horton JW & Scheiderich JF, 1980, *On maintenance scheduling for energy generators*, Proceedings of the IEEE-PES Winter Meeting, New York (NY), (IEEE Catalog 80 CH-1523-0 PWR, paper A-90-11-7).

[27] Eskom Holdings Limited, 2008, *Eskom Holdings Limited: Annual report 2008*, [Online], [Cited August 31st, 2011], Available from `http://www.financialresults.co.za/eskom_ar2008/ar_2008/downloads.htm`.

[28] Eskom Holdings Limited, 2010, *Eskom Holdings Limited: Integrated Report 2010*, [Online], [Cited September 3rd, 2010], Available from `http://www.eskom.co.za/annreport10/`.

[29] Foong WK, 2007, *Ant colony optimisation for power plant maintenance scheduling*, Doctoral Dissertation, The University of Adelaide, Adelaide.

[30] Foong WK, Maier HR & Simpson AR, 2005, *Ant colony optimization for power plant maintenance scheduling optimization*, Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO '05), ACM, New York (NY), pp. 249–256.

[31] Foong WK, Maier HR & Simpson AR, 2008, *Power plant maintenance scheduling using ant colony optimization: An improved formulation*, Engineering Optimization, **40(4)**, pp. 309–329.

[32] Foong WK, Simpson AR, Maier HR & Stolp S, 2008, *Ant colony optimization for power plant maintenance scheduling optimization — A five-station hydropower system*, Annals of Operations Research, **159**, pp. 433–450.

[33] Frost D & Dechter R, 1998, *Optimizing with constraints: A case study in scheduling maintenance of electric power units*, Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming, Springer-Verlag, London, pp. 469–488.

[34] Geetha T & Swarup KS, 2009, *Coordinated preventive maintenance scheduling of GENCO and TRANSCO in restructured power systems*, Electrical Power & Energy Systems, **31**, pp. 626–638.

[35] Grigg C, Wong P, Albrecht P, Allan R, Bhavaraju M, Billinton R, Chen Q, Fong C, Haddad S, Kuruganty S, Li W, Mukerji R, Patton D, Rau N, Reppen D, Schneider A, Shahidehpour M & Singh C, 1999, *The IEEE Reliability test system — 1996*, IEEE Transactions on Power Systems, **14(3)**, pp. 1010–1020.

[36] Huang CJ, Lin CE & Huang CL, 1992, *Fuzzy approach for generator maintenance scheduling*, Electric Power Systems Research, **24**, pp. 31–38.

[37] Huang MD, Romeo F & Sangiovanni-Vincentelli AL, 1986, *An efficient general cooling schedule for simulated annealing*, Proceedings of the IEEE International Conference on Computer-Aided Design, IEEE, Santa Clara (CA), pp. 381–384.

[38] Huang SJ, 1997, *Generator maintenance scheduling: A fuzzy system approach with genetic enhancement*, Electric Power Systems Research, **41**, pp. 233–239.

[39] Huang SJ, 1998, *A genetic-evolved fuzzy system for maintenance scheduling of generating units*, Electrical Power & Energy Systems, **20(3)**, pp. 191–195.

[40] Ignizio JP, 1982, *Linear programming in single- & multiple-objective systems*, Prentice-Hall International Series in Industrial and Systems Engineering, Prentice-Hall, Englewood Cliffs (NJ).

[41] Kim H, Hayashi Y & Nara K, 1997, *An algorithm for thermal unit maintenance scheduling through combined use of GA, SA and TS*, IEEE Transactions on Power Systems, **12(1)**, pp. 329–335.

[42] Kim JH, Park JB, Park JK & Chun YH, 2005, *Generating unit maintenance scheduling under competitive market environments*, Electrical Power & Energy Systems, **27**, pp. 189–194.

[43] Klee V & Minty GJ, 1972, *How good is the simplex algorithm?*, Proceedings of the Third Symposium on Inequalities (Inequalities III), Academic Press, New York (NY), pp. 159–175.

[44] Kolb J, 2004, *ESCOM 1923–1929: The early years of establishment*, [Online], [Cited August 31st, 2011], Available from `http://heritage.eskom.co.za/heritage/escom_1923.htm`.

[45] Kolb J, 2009, *Eskom 2000–2008: Our recent past*, [Online], [Cited August 31st, 2011], Available from `http://heritage.eskom.co.za/heritage/eskom_2000.htm`.

[46] Kralj BL & Petrović R, 1988, *Optimal preventive maintenance scheduling of thermal generating units in power systems — A survey of problem formulations and solution methods*, European Journal of Operational Research, **35**, pp. 1–15.

[47] Kralj BL & Petrović R, 1995, *A multiobjective optimization approach to thermal generating units maintenance scheduling*, European Journal of Operational Research, **84**, pp. 481–493.

[48] Kralj BL & Rajaković N, 1994, *Multiobjective programming in power system optimization: New approach to generator maintenance scheduling*, Electrical Power & Energy Systems, **16(4)**, pp. 211–220.

[49] Kuzle I, Pandzic H & Brezovec M, 2007, *Implementation of the benders decomposition in hydro generating units maintenance scheduling*, Paper presented at the Hydro 2007 Conference: New Approaches for a New Era, Cairns.

[50] Land AH & Doig AG, 1960, *An automatic method of solving discrete programming problems*, Econometrica, **28(3)**, pp. 497–520.

[51] Leou RC, 2006, *A new method for unit maintenance scheduling considering reliability and operation expense*, Electrical Power & Energy Systems, **28**, pp. 471–481.

[52] Lin CE, Huang CJ, Huang CL, Liang CC & Lee SY, 1992, *An expert system for generator maintenance scheduling using operation index*, IEEE Transactions on Power Systems, **7(3)**, pp. 1141–1148.

[53] LINDO Systems Inc, 2005, *LINGO 9.0 — Optimisation modelling tool for linear, non-linear, and integer modelling*, [Online], [Cited September 9th, 2010], Available from `http://www.lindo.com`.

[54] LINDO Systems Inc, 2008, *LINGO user's guide*, LINDO Systems Inc, Chicago (IL).

[55] LINDO Systems Inc, 2009, *LINGO 11.0.1.3 — Optimisation modelling tool for linear, non-linear, and integer modelling*, [Online], [Cited September 9th, 2010], Available from `http://www.lindo.com`.

[56] LINDO Systems Inc, 2011, *Powerful LINGO solvers*, [Online], [Cited June 25th, 2011], Available from `http://www.lindo.com/index.php?view=article&catid=4%3Alingo&id=13%3Apowerful-lingo-solvers&option=com_content&Itemid=3`.

[57] Lootsma FA, 1997, *Fuzzy logic for planning and decision making*, Volume 8 of *Applied Optimization*, Kluwer Academic Publishers, Dordrecht.

[58] Luenberger DG & Ye Y, 2008, *Linear and nonlinear programming*, 3rd Edition, Springer, New York (NY).

[59] Marwali MKC & Shahidehpour SM, 1998, *A deterministic approach to generation and transmission maintenance scheduling with network constraints*, Electric Power Systems Research, **47**, pp. 101–113.

[60] Marwali MKC & Shahidehpour SM, 1999, *A probabilistic approach to generation maintenance scheduler with network constraints*, Electrical Power & Energy Systems, **21**, pp. 533–545.

[61] Micali V, Corporate Consultant (Business Sciences) at *Eskom*, [Personal Communication], Contactable at `Vince.Micali@eskom.co.za`.

[62] Mohanta DK, Sadhu PK & Chakrabarti R, 2004, *Fuzzy reliability evaluation of power plant maintenance scheduling incorporating uncertain forced outage rate and load representation*, Electric Power Systems Research, **72**, pp. 73–84.

[63] Mohanta DK, Sadhu PK & Chakrabarti R, 2007, *Deterministic and stochastic approach for safety and reliability optimization of captive power plant maintenance scheduling using GA/SA-based hybrid techniques: A comparison of results*, Reliability Engineering and System Safety, **92**, pp. 187–199.

[64] Moro LM & Ramos A, 1999, *Goal programming approach to maintenance scheduling of generating units in large scale power systems*, IEEE Transactions on Power Systems, **14(3)**, pp. 1021–1028.

[65] Mromlinski LR, 1985, *Transportation problem as a model for optimal schedule of maintenance outages in power systems*, Electrical Power & Energy Systems, **7(3)**, pp. 161–164.

[66] Mytakidis T & Vlachos A, 2008, *Maintenance scheduling by using the bi-criterion algorithm of preferential anti-pheromones*, Leonardo Journal of Sciences, **12**, pp. 143–164.

[67] Narotam R, Senior Advisor (Generation Division) at *Eskom*, [Personal Communication], Contactable at `NarotaR@eskom.co.za`.

[68] Negnevitsky M & Kelareva G, 1999, *Genetic algorithms for maintenance scheduling in power systems*, Proceedings of the Australasian Universities Power Engineering Conference and IEAust Electric Energy Conference, Northern Territory University, Darwin, pp. 184–189.

[69] Oxford Dictionaries Online, 2011, *electricity*, [Online], [Cited August 29th, 2011], Available from `http://oxforddictionaries.com/definition/electricity`.

[70] Pagès A, Nabona N & Ferrer A, 2007, *Joint solution to the long-term power generation planning and maintenance scheduling*, Paper presented at the 23rd International Federation for Information Processing TC7 Conference, Krakow.

[71] Pretorius V, 2010, *What is system management*, [Online], [Cited September 3rd, 2010], Available from `http://www.eskom.co.za/live/content.php?Category_ID=759`.

[72] Saraiva JT, Pereira ML, Mendes VT & Sousa JC, 2011, *A simulated annealing based approach to solve the generator maintenance scheduling problem*, Electric Power Systems Research, **Article in Press**.

[73] Satoh T & Nara K, 1991, *Maintenance scheduling by using simulated annealing method*, IEEE Transactions on Power Systems, **6**, pp. 850–857.

[74] Sen S & Kothari DP, 1998, *Optimal thermal generating unit commitment: A review*, Electrical Power & Energy Systems, **20(7)**, pp. 443–451.

[75] Sergaki A & Kalaitzakis K, 2002, *A fuzzy knowledge based method for maintenance planning in a power system*, Reliability Engineering and System Safety, **77**, pp. 19–30.

[76] Taşkin ZC, 2010, *Benders decomposition*, in Cochran JJ (Ed), *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Malden (MA).

[77] The MathWorks Inc, 2009, *MATLAB R2009a — The language of technical computing*, [Online], [Cited September 9th, 2010], Available from `http://www.mathworks.com`.

[78] Toth P & Vigo D, 2002, *The Vehicle Routing Problem*, SIAM, Philadelphia (PA).

[79] Triki E, Collette Y & Siarry P, 2005, *A theoretical study on the behaviour of simulated annealing leading to a new cooling schedule*, European Journal of Operational Research, **166**, pp. 77–92.

[80] USA Today, 2007, *China's Three Gorges Dam to require more moves*, [Online], [Cited August 30th, 2011], Available from `http://www.usatoday.com/news/world/2007-10-12-china-dam_N.htm`.

[81] Van Laarhoven PJM & Aarts EHL, 1987, *Simulated annealing: Theory and applications*, Reidel, Dordrecht.

[82] Volkanovski A, Mavko B, Bošević T, Čauševski A & Čepin M, 2008, *Genetic algorithm optimisation of the maintenance scheduling of generating units in a power system*, Reliability Engineering and System Safety, **93**, pp. 757–767.

[83] Wang Y & Handschin E, 2000, *A new genetic algorithm for preventive unit maintenance scheduling of power systems*, Electrical Power & Energy Systems, **22**, pp. 343–348.

[84] Wikipedia The Free Encyclopedia, 2010, *Genotype-phenotype distinction*, [Online], [Cited September 10th, 2010], Available from `http://en.wikipedia.org/wiki/Genotype-phenotype_distinction`.

[85] Wikipedia The Free Encyclopedia, 2011, *Almarian Decker*, [Online], [Cited August 30th, 2011], Available from `http://en.wikipedia.org/wiki/Almarian_Decker`.

[86] Wikipedia The Free Encyclopedia, 2011, *Alternating current*, [Online], [Cited August 30th, 2011], Available from `http://en.wikipedia.org/wiki/Alternating_current`.

[87] Wikipedia The Free Encyclopedia, 2011, *Assignment problem*, [Online], [Cited January 18th, 2011], Available from `http://en.wikipedia.org/wiki/Assignment_problem`.

[88] Wikipedia The Free Encyclopedia, 2011, *Electric generator*, [Online], [Cited August 29th, 2011], Available from `http://en.wikipedia.org/wiki/Electrical_generator`.

[89] Wikipedia The Free Encyclopedia, 2011, *Electric power industry*, [Online], [Cited August 29th, 2011], Available from `http://en.wikipedia.org/wiki/Electric_power_industry`.

[90] Wikipedia The Free Encyclopedia, 2011, *Electricity*, [Online], [Cited August 29th, 2011], Available from `http://en.wikipedia.org/wiki/Electricity`.

[91] Wikipedia The Free Encyclopedia, 2011, *Electrolytic cell*, [Online], [Cited August 29th, 2011], Available from `http://en.wikipedia.org/wiki/Electrolytic_cell`.

[92] Wikipedia The Free Encyclopedia, 2011, *Etymology of electricity*, [Online], [Cited August 29th, 2011], Available from `http://en.wikipedia.org/wiki/Etymology_of_electricity`.

[93] Wikipedia The Free Encyclopedia, 2011, *Grand Inga Dam*, [Online], [Cited August 30th, 2011], Available from `http://en.wikipedia.org/wiki/Grand_Inga_Dam`.

[94] Wikipedia The Free Encyclopedia, 2011, *Hyper-heuristic*, [Online], [Cited September 1st, 2011], Available from `http://en.wikipedia.org/wiki/Hyper-heuristic`.

[95] Wikipedia The Free Encyclopedia, 2011, *Linderhof Palace*, [Online], [Cited August 30th, 2011], Available from `http://en.wikipedia.org/wiki/Linderhof_Palace`.

[96] Wikipedia The Free Encyclopedia, 2011, *Power station*, [Online], [Cited August 29th, 2011], Available from `http://en.wikipedia.org/wiki/Power_station`.

[97] Wikipedia The Free Encyclopedia, 2011, *Successive linear programming*, [Online], [Cited July 28th, 2011], Available from `http://en.wikipedia.org/wiki/Successive_linear_programming`.

[98] Wikipedia The Free Encyclopedia, 2011, *Thomas Edison*, [Online], [Cited August 30th, 2011], Available from `http://en.wikipedia.org/wiki/Thomas_Edison`.

[99] Wikipedia The Free Encyclopedia, 2011, *Travelling salesman problem*, [Online], [Cited January 18th, 2011], Available from `http://en.wikipedia.org/wiki/Travelling_salesman_problem`.

[100] Wikipedia The Free Encyclopedia, 2011, *Voltaic pile*, [Online], [Cited August 29th, 2011], Available from `http://en.wikipedia.org/wiki/Voltaic_pile`.

[101] Wikipedia The Free Encyclopedia, 2011, *War of Currents*, [Online], [Cited August 30th, 2011], Available from `http://en.wikipedia.org/wiki/War_of_Currents`.

[102] Winston WL, 2004, *Operations research: Applications and algorithms*, 4th Edition, Brooks/Cole, Belmont (CA).

[103] WolframAlpha computational knowledge engine, 2011, *Africa electricity consumption*, [Online], [Cited August 30th, 2011], Available from `http://www.wolframalpha.com/input/?i=Africa+electricity+consumption`.

[104] Yamayee Z, Sidenblad K & Yoshimura M, 1983, *A computationally efficient optimal maintenance scheduling method*, IEEE Transaction on Power Apparatus and Systems, **PAS-102(2)**, pp. 330–338.

[105] Yamayee ZA, 1982, *Maintenance scheduling: Description, literature survey, and interface with overall operations scheduling*, IEEE Transaction on Power Apparatus and Systems, **PAS-101(8)**, pp. 2770–2779.

[106] Zhao Y, Volovoi V, Waters M & Mavris D, 2006, *A sequential approach for gas turbine power plant preventative maintenance scheduling*, Journal of Engineering for Gas Turbines and Power, **128**, pp. 796–805.

[107] Zurn HH & Quintana VH, 1975, *Generator maintenance scheduling via successive approximations dynamic programming*, IEEE Transaction on Power Apparatus and Systems, **PAS-94(2)**, pp. 665–671.

APPENDIX A

Advanced problem formulations

This appendix contains the three problem formulations for the more advanced GMS model presented in §3.4.

A.1 Mixed-integer quadratic program

The fourth GMS problem formulation in §3.4.2 extends the problem formulation (3.12) and is the MIQP in which the objective is to

$$\text{minimise} \qquad \sum_{j=1}^{m}(D_j S + r_j)^2 \tag{A.1}$$

subject to the constraints

$$\begin{aligned}
\sum_{j=e_i}^{\ell_i} x_{i,j} &= 1, && i \in \mathcal{I} \\
x_{i,j} &= 0, && j < e_i \text{ or } j > \ell_i,\ i \in \mathcal{I} \\
y_{i,j} &= 0, && j < e_i \text{ or } j > \ell_i + d_i - 1,\ i \in \mathcal{I} \\
\sum_{j=e_i}^{\ell_i+d_i-1} y_{i,j} &= d_i, && i \in \mathcal{I} \\
y_{i,j} - y_{i,j-1} &\leq x_{i,j}, && i \in \mathcal{I},\ j \in \mathcal{J}\backslash\{1\} \\
y_{i,1} &\leq x_{i,1}, && i \in \mathcal{I} \\
\sum_{i=1}^{n} g_{i,j}(1 - y_{i,j}) &= D_j(1+S) + r_j, && j \in \mathcal{J} \\
\sum_{i=1}^{n}\sum_{p=1}^{j} m'_{p,i,j} x_{i,p} &\leq M_j, && j \in \mathcal{J} \\
\sum_{i \in \mathcal{I}_k} y_{i,j} &\leq K_k, && j \in \mathcal{J},\ k \in \mathcal{K} \\
x_{i,j},\ y_{i,j} &\in \{0,1\}, && i \in \mathcal{I},\ j \in \mathcal{J} \\
r_j &\geq 0, && j \in \mathcal{J}.
\end{aligned}$$

A.2 Mixed-integer nonlinear program

The fifth GMS problem formulation in §3.4.2 extends the problem formulation (3.15) to the MINP in which the objective is to

$$\text{minimise} \qquad \sum_{j=1}^{m} |D_j S + r_j - \overline{r}| \tag{A.2}$$

subject to the constraints

$$\begin{aligned}
\overline{r} &= \frac{1}{m}\sum_{j=1}^{m} D_j S + r_j \\
\sum_{j=e_i}^{\ell_i} x_{i,j} &= 1, && i \in \mathcal{I} \\
x_{i,j} &= 0, && j < e_i \text{ or } j > \ell_i,\ i \in \mathcal{I} \\
y_{i,j} &= 0, && j < e_i \text{ or } j > \ell_i + d_i - 1,\ i \in \mathcal{I} \\
\sum_{j=e_i}^{\ell_i+d_i-1} y_{i,j} &= d_i, && i \in \mathcal{I} \\
y_{i,j} - y_{i,j-1} &\leq x_{i,j}, && i \in \mathcal{I},\ j \in \mathcal{J}\backslash\{1\} \\
y_{i,1} &\leq x_{i,1}, && i \in \mathcal{I} \\
\sum_{i=1}^{n} g_{i,j}(1 - y_{i,j}) &= D_j(1+S) + r_j, && j \in \mathcal{J} \\
\sum_{i=1}^{n}\sum_{p=1}^{j} m'_{p,i,j} x_{i,p} &\leq M_j, && j \in \mathcal{J} \\
\sum_{i \in \mathcal{I}_k} y_{i,j} &\leq K_k, && j \in \mathcal{J},\ k \in \mathcal{K} \\
x_{i,j},\ y_{i,j} &\in \{0,1\}, && i \in \mathcal{I},\ j \in \mathcal{J} \\
r_j &\geq 0, && j \in \mathcal{J}.
\end{aligned}$$

A.3 Mixed-integer linear program

The sixth and final GMS problem formulation in §3.4.2 extends the problem formulation (3.19) to the MILP in which the objective is to

$$\text{minimise} \qquad \sum_{j=1}^{m} o_j + u_j \tag{A.3}$$

subject to the constraints

$$\begin{aligned}
\overline{r} &= \frac{1}{m}\sum_{j=1}^{m} D_j S + r_j \\
\sum_{j=e_i}^{\ell_i} x_{i,j} &= 1, && i \in \mathcal{I} \\
x_{i,j} &= 0, && j < e_i \text{ or } j > \ell_i,\ i \in \mathcal{I} \\
y_{i,j} &= 0, && j < e_i \text{ or } j > \ell_i + d_i - 1,\ i \in \mathcal{I} \\
\sum_{j=e_i}^{\ell_i+d_i-1} y_{i,j} &= d_i, && i \in \mathcal{I} \\
y_{i,j} - y_{i,j-1} &\leq x_{i,j}, && i \in \mathcal{I},\ j \in \mathcal{J}\backslash\{1\} \\
y_{i,1} &\leq x_{i,1}, && i \in \mathcal{I} \\
\sum_{i=1}^{n} g_{i,j}(1 - y_{i,j}) &= D_j(1+S) + r_j, && j \in \mathcal{J} \\
\sum_{i=1}^{n}\sum_{p=1}^{j} m'_{p,i,j} x_{i,p} &\leq M_j, && j \in \mathcal{J} \\
\sum_{i \in \mathcal{I}_k} y_{i,j} &\leq K_k, && j \in \mathcal{J},\ k \in \mathcal{K} \\
\overline{r} &= (D_j S + r_j) + u_j - o_j, && j \in \mathcal{J} \\
x_{i,j},\ y_{i,j} &\in \{0,1\}, && i \in \mathcal{I},\ j \in \mathcal{J} \\
r_j, o_j, u_j &\geq 0, && j \in \mathcal{J}.
\end{aligned}$$

Note that in all three problem formulations presented in this appendix, the crew constraint set (3.26) is used and may be substituted with the alternative crew constraint set of (3.24), namely

$$\sum_{i=1}^{n} m_{i,j} y_{i,j} \leq M_j, \qquad j \in \mathcal{J},$$

depending on the requirements of the problem.

APPENDIX B

Pseudo-code listings

This appendix contains the pseudo-code listings of two algorithms that were touched upon in Chapter 4, but which were not presented in the main text so as to improve for readability.

In Algorithm B.1, a pseudo-code listing of the GMS random search heuristic with the incorporation of the classical neighbourhood move operator is presented. The random search method's implementation is presented in §4.2.4 and Algorithm B.1 differs from Algorithm 4.5 only in its neighbourhood structure.

A pseudo-code listing of the algorithmic implementation adopted in this thesis of the cooling schedule proposed by Triki *et al.* [79], is presented in Algorithm B.2. It is a slight modification of the original algorithm presented in [79]. The updating rule of the cooling schedule in question, is the rule (4.11), as presented in §4.2.5.

Algorithm B.1: The GMS random search heuristic with classical neighbourhood

Input: A power system scenario for which to solve the generator maintenance scheduling problem

Output: The best maintenance schedule found

```
1  dataset ← declareSystemData()
2  [current, currentObj] ← generateRandomSolution(dataset)
3  [incumbent, incumbentObj] ← [current, currentObj]
4  iterationCounter ← 0
5  nonImproveCounter ← 0
6  while (iterationCounter < maxIteration) and (nonImproveCounter < maxNonImprove)
   do
7      bestNeighbour ← ∅
8      bestNeighbourObj ← some very large number
9      moves ← createClassicalNeighbourhoodList (n, e, ℓ, W_ext)
10     moves ← randShuffle(moves)
11     for neighbourCounter ← 1 to neighbourhoodSize do
12         neighbour ← current
13         Apply moves(neighbourCounter) on neighbour to create new neighbour
14         P ← checkFeasibilityAndCalculatePenalty(neighbour, dataset)
15         Calculate neighbourObj
16         neighbourObj ← neighbourObj + P
17         if neighbourObj < bestNeighbourObj then
18             [bestNeighbour, bestNeighbourObj] ← [neighbour, neighbourObj]
19         end
20     end
21     if bestNeighbourObj < incumbentObj then
22         [incumbent, incumbentObj] ← [bestNeighbour, bestNeighbourObj]
23         nonImproveCounter ← 0
24     else
25         nonImproveCounter ← nonImproveCounter + 1
26     end
27     [current, currentObj] ← [bestNeighbour, bestNeighbourObj]
28     iterationCounter ← iterationCounter + 1
29 end
```

Algorithm B.2: Simulated annealing with targeted average decrease in cost

```
1  dataset ← declareSystemData()
2  [current, currentObj ] ← generateRandomSolution(dataset)
3  [avgT_0, stdT_0] ← initialTemperature(current, currentObj, dataset)
4  T ← stdT_0
5  σ ← stdT_0
6  Δ ← σ/μ_2
7  negativeTemperature ← 0
8  reinitialising ← true
9  doGeometric ← false
10 while system not frozen do
11     Do the inner loop of SA and return the standard deviation      // Metropolis loop
12     σ ← standard deviation
13     if doGeometric = false then
14         if reinitialising = false then
15             if currentAverageCost/(previousAverageCost − Δ) > ζ then
16                 equilibriumNotReached ← equilibriumNotReached + 1
17             else
18                 equilibriumNotReached ← 0
19             end
20         end
21         if equilibriumNotReached > K_1 then
22             reinitialising ← true
23             equilibriumNotReached ← 0
24             T ← geometricCooling(λ_1, T)                    // reheating (λ_1 > 1)
25             Δ ← σ/μ_1
26         else if TΔ/σ^2 > 1 then
27             negativeTemperature ← negativeTemperature + 1
28             reinitialising ← true
29             if negativeTemperature < K_2 then
30                 T ← geometricCooling(λ_1, T)                // reheating (λ_1 > 1)
31                 Δ ← σ/μ_1
32             else
33                 doGeometric ← true
34             end
35         else
36             reinitialising ← false
37             previousAverageCost ← currentAverageCost
38             T ← TrikiCooling(Δ, σ, T)
39         end
40     else
41         T ← geometricCooling(λ_2, T)
42     end
43 end
```

APPENDIX C

Alternative best solutions for the 21-unit test system

In this appendix, a listing of all the alternative best solutions obtained in this study for the 21-unit test system, as stated in §6.4.2, is presented. The solutions are listed in Table C.1 in column vector format.

Unit	01	02	03	04	05	06	07	08	09	10	11	12	13	14
1	1	5	5	9	6	1	5	3	8	4	2	6	1	2
2	42	42	39	47	47	42	40	47	27	27	45	27	33	30
3	15	13	13	17	14	15	13	11	16	12	16	14	15	10
4	26	26	23	1	5	19	4	1	4	25	1	1	22	1
5	28	27	31	34	27	47	35	33	32	42	38	48	27	48
6	19	23	24	6	2	20	24	24	1	22	20	3	23	20
7	22	2	2	3	24	23	1	21	5	1	23	24	19	23
8	36	35	46	27	40	35	46	27	44	47	27	40	46	42
9	8	12	12	16	13	8	12	10	15	11	9	13	8	9
10	9	15	15	19	16	9	15	13	18	14	10	16	9	12
11	25	1	1	2	1	26	23	2	26	26	26	2	26	26
12	33	32	36	39	37	28	32	43	41	38	50	33	38	27
13	7	11	11	15	12	7	11	9	14	10	8	12	7	8
14	11	17	17	21	18	11	17	15	20	16	12	18	11	14
15	4	9	9	13	10	5	9	7	12	8	6	10	5	6
16	17	21	21	25	22	17	21	19	24	20	18	22	17	18
17	48	48	45	33	36	52	45	46	51	41	44	32	45	41
18	46	46	51	51	51	40	51	51	49	31	32	45	51	34
19	27	41	44	42	46	27	31	42	52	33	43	47	32	40
20	49	49	27	43	32	31	27	38	37	34	34	36	41	36
21	1	6	6	10	7	2	6	4	9	5	3	7	2	3

Table C.1: *List of alternative best solution vectors for the 21-unit system.*

APPENDIX D

Input format for the DSS

In this appendix, the specific format of the input data for the DSS, referred to in §7.2.2, is illustrated in order to ensure that the DSS can read and solve a problem instance properly. As stated in §7.2, the data input file is required to be a Microsoft Excel workbook. Four mandatory worksheets are required in the workbook, having the specific sheetnames of: "System", "Capacity", "Demand" and "Windows." Two optional worksheets may be appended, having the specific sheetnames of "Crew" and "Exclusions," should those constraint sets be present in the problem being considered.

Figure D.1 contains a screenshot example of the "System" sheet and it contains the number of units (n), the number of time periods during the planning horizon (m) and the maintenance duration of each generating unit (d_i).

	A	B	C	D	E	F	G	H
1								
2								
3			Number of units				32	
4			Number of time periods in planning horizon				52	
5								
6								
7			Unit	Maintenance duration				
8			1	2				
9			2	2				
10			3	3				
11			4	3				
12			5	2				
13			6	2				
14			7	3				
15			8	3				
16			9	3				
17			10	3				

Figure D.1: *Screenshot of the "System" worksheet in the DSS input file.*

In Figure D.2, a screenshot example of the "Capacity" worksheet is presented. It contains the capacity of each unit during each time period ($g_{i,j}$) as well as the total additional capacity (*i.e.* the repeated capacity), in the event of dummy units being present in the system.

The safety margin (S) and the peak load demand during each time period over the planning horizon (D_j) are contained within the "Demand" worksheet. A screenshot example may be found in Figure D.3.

	A	B	C	D	E	F	G	H	I	J	K
1											
2	**Repeated**	0		**Time periods**							
3				**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
4		**Units**	**1**	20	20	20	20	20	20	20	20
5			**2**	20	20	20	20	20	20	20	20
6			**3**	76	76	76	76	76	76	76	76
7			**4**	76	76	76	76	76	76	76	76
8			**5**	20	20	20	20	20	20	20	20
9			**6**	20	20	20	20	20	20	20	20
10			**7**	76	76	76	76	76	76	76	76
11			**8**	76	76	76	76	76	76	76	76
12			**9**	100	100	100	100	100	100	100	100
13			**10**	100	100	100	100	100	100	100	100

Figure D.2: *Screenshot of the "Capacity" worksheet in the DSS input file.*

	A	B	C	D	E	F
1						
2						
3			**Safety margin**	0.15		
4						
5						
6			**Time period**	**Load demand**		
7			**1**	2457		
8			**2**	2565		
9			**3**	2502		
10			**4**	2377		
11			**5**	2508		
12			**6**	2397		
13			**7**	2371		
14			**8**	2297		
15			**9**	2109		
16			**10**	2100		

Figure D.3: *Screenshot of the "Demand" worksheet in the DSS input file.*

An example of the final mandatory worksheet entitled "Windows," is illustrated in Figure D.4. It contains the number of time periods by which window constraints may be violated (W_{ext}) as well as the earliest (e_i) and latest (ℓ_i) maintenance starting times for each unit.

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3			**Number of time periods by which windows may be violated**						3		
4											
5											
6			**Unit**	**Earliest**	**Latest**						
7			**1**	1	25						
8			**2**	1	25						
9			**3**	1	24						
10			**4**	27	50						
11			**5**	1	25						
12			**6**	27	51						
13			**7**	1	24						
14			**8**	27	50						
15			**9**	1	50						
16			**10**	1	50						

Figure D.4: *Screenshot of the "Windows" worksheet in the DSS input file.*

An example of the optional worksheet entitled "Crew," is shown in Figure D.5. The maximum available manpower during each time period (M_j) and the required manpower for maintenance of each unit in its k-th week of maintenance (m_i^k) are contained in the worksheet.

Finally, Figure D.6 contains an example screenshot of the optional "Exclusions" worksheet. The number of exclusion subsets (K), the subsets themselves ($\mathcal{I}_k$), the maximum number of units

	Time period	Maximum available crew
	1	25
	2	25
	3	25
	4	25
	5	25
	6	25
	7	25
	8	25
	9	25
	10	25

Unit	Crew requirements for each time period of maintenance		
1	7	7	
2	7	7	
3	12	10	10
4	12	10	10
5	7	7	
6	7	7	
7	12	10	10
8	12	10	10
9	10	10	15
10	10	10	15

Figure D.5: *Screenshot of the "Crew" worksheet in the DSS input file.*

within each subset that are allowed to be in simultaneous maintenance during any time period (K_k) and a listing of which subset each unit belongs to are all contained in the worksheet.

Number of subsets of units: 7

Unit	Subset unit belongs to
1	1
2	1
3	1
4	1
5	2
6	2
7	2
8	2
9	3
10	3

Subset	Maximum number of units simultaneous
1	2
2	2
3	1
4	1
5	3
6	3
7	1

Subset	Units included in each subset					
1	1	2	3	4		
2	5	6	7	8		
3	9	10	11			
4	12	13	14			
5	15	16	17	18	19	20
6	24	25	26	27	28	29
7	30	31	32			

Figure D.6: *Screenshot of the "Exclusions" worksheet in the DSS input file.*

These screenshots all illustrate the specific format of the input data, but more importantly, they illustrate the specific cell locations in the Excel workbook (worksheet name and cell address) where each data value has to lie or data range has to start. Failure to have the file in this format may not necessarily result in an immediate error. However, when the penalty procedure or solution algorithm commences, an incorrect data input will cause errors in the DSS.

APPENDIX E

System specifications of the case study

This appendix contains the data of the Eskom case study presented in §7.3. In Table E.1, the generation system and maintenance specification are presented. The power system consists of 105 power generating units, as seen in the "Unit name" column of Table E.1. However, as stated in §7.3, dummy units are added to the system for each additional maintenance outage of the same unit. As a result, the data contains 157 units. The additional capacity of these dummy units is 12 422 MW. Table E.3 contains the exclusion sets, while Table E.4 contains the daily peak load demand of the power system.

Unit	Unit name	Capacity (MW)	Earliest starting time (day)	Latest starting time (day)	Duration (days)
1	A1	57	8	29	9
2	A1	57	15	29	2
3	A1	57	169	183	2
4	A1	57	281	295	2
5	A2	57	15	29	2
6	A2	57	15	36	9
7	A2	57	169	183	2
8	A2	57	281	295	2
9	A3	57	15	29	2
10	A3	57	29	50	9
11	A3	57	176	190	2
12	A3	57	288	309	7
13	B1	148	0	0	0
14	B2	148	0	0	0
15	B3	148	0	0	0
16	B4	148	0	0	0
17	C1	330	36	50	4
18	C1	330	225	239	4
19	C2	350	29	57	14
20	C2	350	253	267	4
21	C3	380	36	50	4
22	C3	380	281	295	4
23	C4	350	71	85	2
24	C4	350	302	316	3
25	C5	350	8	22	4
26	C5	350	239	282	84
27	C6	350	106	120	2

Table E.1: *Data for the Eskom case study.*

Unit	Unit name	Capacity (MW)	Earliest starting time (day)	Latest starting time (day)	Duration (days)
28	C6	350	274	288	3
29	D1	190	0	0	0
30	D2	190	0	0	0
31	D3	185	50	92	30
32	D4	180	218	267	42
33	D5	180	0	0	0
34	D6	160	0	0	0
35	D7	170	1	50	42
36	D8	180	85	134	42
37	E1	250	8	64	44
38	E1	250	134	148	1
39	E1	250	281	295	1
40	E1	250	316	344	14
41	E2	250	8	64	44
42	E2	250	330	344	1
43	E3	250	29	43	1
44	E3	250	267	295	14
45	E4	250	8	22	1
46	E4	250	50	64	3
47	E4	250	309	337	14
48	F1	575	0	0	0
49	F2	575	71	148	70
50	F3	575	211	239	14
51	F4	575	0	0	0
52	F5	575	1	78	70
53	F6	575	232	288	45
54	G1	90	71	99	11
55	G1	90	246	246	120
56	G2	90	15	29	2
57	G2	90	232	246	120
58	G3	90	43	71	11
59	G3	90	295	323	14
60	G4	90	1	85	121
61	G4	90	113	127	2
62	H1	148	0	0	0
63	H2	148	0	0	0
64	H3	148	0	0	0
65	I1	190	78	113	21
66	I2	190	211	246	21
67	I3	190	0	0	0
68	I4	190	309	344	21
69	I5	190	0	0	0
70	I6	190	0	0	0
71	J1	190	0	0	0
72	J2	185	1	78	92
73	J3	190	0	0	0
74	J4	190	92	120	14
75	J5	190	204	267	50
76	J6	190	0	0	0
77	J7	190	113	141	14
78	J8	190	1	43	28
79	J8	190	260	288	14
80	J9	190	281	309	14
81	J10	190	218	260	28

Table E.1: *(continued) Data for the Eskom case study.*

Unit	Unit name	Capacity (MW)	Earliest starting time (day)	Latest starting time (day)	Duration (days)
82	J10	190	302	338	28
83	K1	640	330	343	23
84	K2	640	1	22	7
85	K3	640	295	330	23
86	K4	640	323	344	5
87	K5	640	1	71	57
88	K6	640	85	106	5
89	L1	900	225	282	84
90	L2	900	0	0	0
91	M1	475	1	36	21
92	M1	475	288	330	28
93	M2	475	1	78	84
94	M3	475	323	345	21
95	M4	475	0	0	0
96	M5	475	0	0	0
97	M6	475	0	0	0
98	N1	593	43	85	28
99	N2	593	0	0	0
100	N3	593	99	141	28
101	N4	593	29	50	7
102	N5	593	78	99	7
103	N5	593	316	338	28
104	N6	593	15	36	7
105	N6	593	218	260	28
106	O1	612	1	43	28
107	O2	612	106	127	7
108	O2	612	260	310	56
109	O3	612	99	120	7
110	O4	669	64	113	35
111	O5	669	330	351	7
112	O6	669	351	358	8
113	P1	615	1	50	42
114	P1	615	302	323	7
115	P2	615	71	92	7
116	P3	615	323	331	35
117	P4	615	43	85	28
118	P5	615	85	127	28
119	P6	615	190	211	7
120	Q1	575	218	239	7
121	Q2	575	0	0	0
122	Q3	575	0	0	0
123	Q4	575	1	50	42
124	Q5	575	85	106	7
125	Q6	575	239	282	84
126	R1	200	22	57	25
127	R1	200	127	141	1
128	R1	200	239	253	1
129	R2	200	36	71	25
130	R2	200	141	155	1
131	R2	200	232	246	1
132	S1	57	50	64	2
133	S1	57	85	141	45
134	S1	57	176	190	2
135	S1	57	295	309	2
136	S2	57	15	85	56

Table E.1: *(continued) Data for the Eskom case study.*

Unit	Unit name	Capacity (MW)	Earliest starting time (day)	Latest starting time (day)	Duration (days)
137	S2	57	57	71	2
138	S2	57	183	197	2
139	S2	57	302	316	2
140	S3	57	57	78	7
141	S3	57	183	197	2
142	S3	57	302	316	2
143	T1	585	85	99	4
144	T1	585	274	306	60
145	T2	585	92	120	12
146	T2	585	344	358	4
147	T3	585	64	78	4
148	T3	585	218	246	12
149	T4	585	1	64	50
150	T4	585	253	267	4
151	T5	585	71	85	4
152	T6	585	120	134	4
153	T6	585	323	337	4
154	U1	120	22	57	21
155	U1	120	120	162	31
156	U1	120	267	288	5
157	U2	120	85	106	5

Table E.2: *(continued) Data for the Eskom case study.*

Exclusion set	Units	Maximum
1	1, 2	1
2	5, 6	1
3	9, 10	1
4	60, 61	1
5	133, 134	1
6	136, 137	1

Table E.3: *Exclusion data for the Eskom case study.*

Day	Demand (MW)	Day	Demand (MW)	Day	Demand (MW)	Day	Demand (MW)	Day	Demand (MW)	Day	Demand (MW)
1	31 252	62	28 546	123	27 505	184	34 008	245	34 179	306	29 321
2	30 890	63	31 000	124	27 454	185	34 197	246	34 333	307	29 839
3	31 962	64	31 062	125	27 056	186	34 043	247	34 187	308	32 542
4	31 704	65	30 857	126	28 862	187	32 829	248	34 625	309	32 344
5	29 997	66	30 594	127	31 694	188	32 798	249	32 802	310	32 173
6	29 114	67	31 061	128	31 554	189	34 902	250	31 153	311	31 974
7	31 389	68	29 416	129	32 251	190	34 722	251	30 232	312	31 424
8	31 116	69	29 007	130	31 283	191	35 453	252	32 849	313	30 186
9	30 684	70	32 158	131	29 811	192	35 352	253	35 701	314	30 414
10	30 558	71	31 917	132	29 779	193	33 711	254	35 199	315	32 626
11	30 390	72	31 747	133	32 324	194	33 140	255	35 326	316	32 706
12	29 515	73	31 780	134	32 555	195	32 558	256	33 948	317	33 061
13	28 311	74	31 820	135	32 404	196	35 203	257	32 314	318	32 610
14	30 548	75	30 198	136	32 907	197	35 841	258	32 033	319	31 614
15	29 778	76	29 399	137	32 138	198	35 034	259	33 618	320	30 730
16	28 690	77	32 039	138	30 351	199	35 654	260	33 651	321	31 031
17	29 537	78	31 650	139	30 164	200	35 621	261	33 605	322	33 101
18	29 835	79	31 440	140	33 279	201	34 464	262	33 359	323	32 462
19	28 380	80	31 722	141	32 842	202	33 522	263	32 557	324	32 416
20	27 703	81	31 073	142	33 674	203	36 036	264	30 996	325	31 818
21	29 374	82	30 108	143	33 133	204	35 806	265	30 715	326	31 919
22	27 893	83	29 823	144	32 864	205	35 636	266	33 649	327	30 761
23	26 775	84	31 917	145	31 830	206	35 799	267	33 884	328	29 970
24	25 588	85	32 275	146	30 963	207	34 406	268	33 102	329	32 701
25	24 438	86	31 295	147	32 722	208	32 743	269	32 686	330	33 105
26	24 992	87	31 249	148	32 455	209	32 834	270	31 114	331	32 253
27	25 364	88	30 709	149	33 312	210	35 522	271	29 267	332	32 076
28	27 468	89	29 318	150	33 114	211	35 457	272	29 380	333	31 155
29	27 609	90	29 275	151	31 759	212	35 334	273	31 994	334	29 880
30	27 674	91	32 305	152	30 672	213	35 478	274	31 586	335	29 126
31	27 249	92	31 975	153	30 451	214	34 220	275	31 670	336	31 840
32	25 544	93	32 390	154	33 942	215	32 366	276	31 427	337	31 946
33	25 955	94	32 321	155	33 531	216	32 535	277	30 396	338	32 344
34	26 510	95	31 425	156	33 509	217	35 619	278	29 107	339	31 781
35	28 693	96	29 767	157	33 316	218	35 400	279	29 673	340	31 472
36	29 705	97	29 864	158	32 459	219	35 136	280	31 929	341	29 975
37	29 552	98	31 895	159	31 256	220	35 659	281	32 468	342	30 062
38	29 846	99	31 456	160	30 527	221	34 236	282	32 321	343	32 479
39	30 191	100	32 071	161	33 938	222	33 097	283	32 348	344	32 070
40	28 515	101	31 894	162	33 627	223	32 942	284	31 243	345	32 521
41	28 397	102	31 483	163	34 135	224	36 463	285	30 344	346	31 878
42	30 494	103	30 314	164	33 903	225	36 559	286	30 245	347	30 996
43	31 515	104	30 051	165	32 549	226	36 664	287	32 083	348	29 800
44	31 149	105	32 312	166	31 609	227	36 256	288	32 003	349	29 781
45	31 757	106	32 413	167	31 188	228	36 127	289	32 129	350	32 225
46	31 064	107	31 702	168	33 431	229	34 199	290	32 594	351	32 198
47	29 494	108	32 563	169	33 806	230	33 408	291	31 676	352	32 321
48	28 916	109	31 525	170	33 822	231	35 680	292	30 495	353	32 751
49	30 910	110	30 029	171	33 373	232	35 632	293	30 223	354	31 669
50	31 111	111	28 841	172	33 151	233	36 104	294	32 420	355	30 478
51	31 030	112	31 007	173	31 752	234	35 364	295	32 446	356	29 857
52	31 862	113	32 273	174	31 774	235	33 695	296	32 189	357	31 587
53	31 204	114	32 663	175	33 665	236	32 559	297	30 878	358	31 564
54	29 352	115	32 301	176	33 731	237	32 611	298	29 227	359	32 430
55	29 132	116	31 699	177	33 504	238	35 252	299	28 879	360	32 055
56	31 156	117	29 685	178	33 543	239	34 638	300	28 932	361	31 912
57	30 873	118	29 318	179	32 954	240	34 811	301	32 250	362	29 682
58	30 899	119	32 605	180	31 560	241	34 183	302	32 562	363	29 684
59	30 338	120	31 914	181	32 974	242	33 005	303	31 798	364	31 908
60	30 577	121	31 675	182	34 836	243	31 906	304	31 785	365	31 798
61	29 170	122	30 420	183	34 173	244	31 701	305	30 952		

Table E.4: *The daily peak load demands for the Eskom case study.*

APPENDIX F

Contents of the accompanying compact disc

This appendix contains a brief description of the contents of the compact disc included with this thesis. The compact disc contains the personal computer implementation of the DSS presented in Chapter 7, Excel workbooks populated with input data, the unabridged Eskom case study results, and an electronic version of the thesis in ".pdf" file format. The DSS was created in MATLAB version R2009a and may be executed in this version or later. A version of Microsoft Excel 2007 or later is required by the DSS in order to create the output file. There are four directories on the compact disc and their contents are described here by their directory names.

Thesis. This directory contains an electronic copy of the thesis in ".pdf" format.

Data sets. This directory contains four Excel workbooks populated with the system data for the Eskom case study, the 21-unit system, the 22-unit system and the IEEE-RTS inspired system in the correct input format for the DSS. Note that, since the DSS does not consider precedence constraints (which are present in the 22-unit system), any 22-unit system results obtained by the DSS should not be compared to the results obtained in §6.4.3.

Decision support system. This directory contains the DSS implementation as MATLAB script files (".m" format). These files are unprotected and may be opened in the MATLAB editor for modification. The DSS GUI is a MATLAB ".fig" file which may be opened in the MATLAB GUI design environment (GUIDE). In order to execute the DSS, the user is required to copy this directory onto his/her own computer, because the DSS will write the output file to this directory, and the compact disc is read-only. After the directory has been copied onto the computer, the current directory in MATLAB should be changed to that directory and "RunGMSDSS" should be entered into the MATLAB command line in order to execute the DSS. The procedures presented in §7.2 may be followed in order to solve a GMS problem instance by means of the DSS. Should the user require the DSS to terminate immediately, he/she may enter the MATLAB interrupt command keystroke "Ctrl + c" into the command line. However, it will cause any visible message box to be unclosable until MATLAB itself has been terminated.

Case study results. This directory contains the unabridged Eskom case study results obtained by the DSS. An abridged version of these results were presented in §7.3. There are 24 subdirectories in this directory, corresponding to each solution instance that was considered, respectively. The subdirectories contain the Excel output file and MATLAB figures generated by the DSS for each solution instance.